UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA

TESIS DOCTORAL

Architecture, Techniques and Models for enabling Data Science in the Gaia Mission Archive

Arquitectura, Técnicas y Modelos para posibilitar la Ciencia de Datos en el Archivo de la Misión Gaia

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Daniel Tapiador de Pedro

DIRECTORES

Eduardo Huedo Cuesta
Luis Manuel Sarro Baro

Madrid, 2018

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Architecture, Techniques and Models for Enabling Data Science in the Gaia Mission Archive

(Arquitectura, Técnicas y Modelos para Posibilitar la Ciencia de Datos en el Archivo de la Misión Gaia)

Daniel Tapiador de Pedro

Directores:

Eduardo Huedo Cuesta

Luis Manuel Sarro Baro

Marzo 2017

Architecture, Techniques and Models for Enabling Data Science in the Gaia Mission Archive

(Arquitectura, Técnicas y Modelos para Posibilitar la Ciencia de Datos en el Archivo de la Misión Gaia)

TESIS DOCTORAL

Daniel Tapiador de Pedro

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

Marzo 2017

Architecture, Techniques and Models for Enabling Data Science in the Gaia Mission Archive

Author: **Daniel Tapiador de Pedro**

Thesis submitted to the Complutense University of Madrid in partial fulfillment of the requirements for the degree of

Ph.D. in Computer Science

Supervisors:

Eduardo Huedo Cuesta

Luis Manuel Sarro Baro

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática
Universidad Complutense de Madrid

March 2017

A Nuria e Iván,
por su apoyo incondicional,
y por dar sentido a este esfuerzo.

Acknowledgements

I would like to express my gratitude to the countless people that have directly or indirectly supported this thesis throughout the last ten years, helping me research in this passionate field of distributed systems applied to astronomy and astrophysics.

In the first place, I would like to thank my supervisors, Eduardo Huedo Cuesta and Luis Manuel Sarro Baro for their guidance and contributions in making the thesis and all related publications a thorough and impactful work within the discipline. I also want to thank William O'Mullane for the opportunity to work in one of the most exciting European Space Agency (ESA) scientific missions, i.e. Gaia, and for his support and ambition to make its goals a success. In the same way, members of the Data Processing and Analysis Consortium (DPAC) like Xavier Luri, Anthony Brown, Cesc Julbe and Francesca Figueras deserve my thankfulness.

Special recognition should be allocated to the European Space Astronomy Centre (ESAC), as a world-class reference for space and planetary astronomy. I have had the pleasure to work there for eight years and to develop and validate part of the research comprised in this thesis. There are many brilliant professionals who I have learned a lot from, and who have become great friends along the way. It would be endless to name them all, but I would like to at least mention Christophe Arviset, Fernando Félix, Rubén Álvarez and Aitor Ibarra.

Furthermore, I would like to thank Ángel Berihuete for the leadership he has shown in the last (and toughest) part of this journey. He deserves a special mention as an unofficial supervisor of this thesis, and for spending the necessary time (often his personal one) on the topics that I needed to learn and work on for the research being put forward today. This commitment and perseverance is crucial for the success of any research, and for the growth of PhD candidates working on their theses.

In addition, I have the pleasure to have worked in this thesis within one of the leading research and innovation groups for distributed systems, Cloud computing and other cutting-edge technological advances, i.e. DSA-Research. Ignacio Martín Llorente and Rubén Santiago Montero have certainly helped me spark my curiosity and enthusiasm for applied research and innovation in distributed systems.

Moreover, the many knowledgeable colleagues that I have had the opportunity to work with, in leading companies like Telefonica, Etisalat and the Emirates Integrated Telecommunications Company (du), as well as the cross-fertilization of some of the ideas learned there, have proved to be very useful both for this research and for my professional

endeavours.

Last but not least, the support and push that I have received from my family and friends have been paramount for the accomplishment of this research. They have given me the inspiration and strength from the beginning, all the way down to these words of acknowledgement.

Contents

List of Figures

List of Tables

Acronyms

2MASS Two Micron All Sky Survey. 94, 95

ADQL Astronomical Data Query Language. 28, 93

AGIS Astrometric Global Iterative Solution. 62

AIO Archive Inter-Operability. 30, 32

AO Announcement of Opportunity. 62

API Application Programming Interface. 4, 45, 46, 57, 58, 122, 124

AWS Amazon Web Services. 6, 67, 81, 82, 132

CANFAR Canadian Advanced Network for Astronomical Research. 37

CAPEX Capital Expenditure. 6

CAS Catalog Archive Server. 35

CCD Charge Coupled Device. 61

CD-ROM Compact Disc-Read Only Memory. 23

CDS Centre de Données astronomiques de Strasbourg. 96

CPU Central Processing Unit. 6, 43, 45, 50, 54, 60, 133

CRM Customer Relationship Management. 10

CSV Comma Separated Values. 69

CU Coordination Unit. 20, 62, 64, 68, 98

DaaS Data as a Service. 58, 60

DAG Directed Acyclic Graph. 54, 88

DAO Data Access Object. 31

DAS Direct-Attached Storage. 42, 67

DBA Database Administrator. 45

DBMS Database Management System. XI, 43, 45, 46, 50, 52, 67, 68, 72, 76, 88, 133

DPAC Data Processing and Analysis Consortium. VII, XII, 62, 98

DPC Data Processing Centre. 62

DSL Domain Specific Language. 59

DSS Digitized Sky Survey. 94

EBS Amazon Elastic Block Store. 67

EC2 Amazon Elastic Compute Cloud. 67, 81

EMR Amazon Elastic MapReduce. 81, 82, 85

ESA European Space Agency. VII, 16, 23, 25, 39, 61, 62, 131

ESAC European Space Astronomy Centre. VII, 19

ESDC ESAC Science Data Centre. XI, 15, 30, 31

ETL Extract, Transform and Load. XI, 25, 39, 46, 86

FLAME Final Luminosity, Age and Mass Estimation. 98, 99, 108, 109, 114, 117, 119, 134

FTP File Transfer Protocol. 31

GALEX Galaxy Evolution Explorer. 94

GAP Gaia Archive Preparation. 62

GB Gigabyte. 54, 67, 75, 124

GFS Google File System. 33

GP Gaussian Process. 116, 117

GPL GNU General Public License. 39

GPU Graphics Processing Unit. 134

GUI Graphical User Interface. 30

GUMS Gaia Universe Model Snapshot. XI, XII, 17, 51, 52, 70, 72, 74, 75, 82, 84, 87, 91

HBM Hierarchical Bayesian Modeling. XXIV, 3, 14, 98, 99, 100, 108, 117, 119

HDFS Hadoop Distributed File System. XI, XII, 17, 20, 42, 45, 48, 50, 69, 82, 83, 84, 85, 95, 124

HEALPix Hierarchical Equal Area Iso Latitude Pixelation of a sphere. XII, 18, 66, 68, 69, 76, 82, 86

HPC High Performance Computing. 4, 5, 33

HQL Hive Query Language. 57

HRPS Hybrid Range Partitioning Strategy. 67

HSO Herschel Space Observatory. 12

HST Hubble Space Telescope. 61

HTTP HyperText Transfer Protocol. 28, 30, 31

I/O Input/Output. 6, 17, 43, 45, 48, 50, 54, 87, 88, 124

IaaS Infrastructure as a Service. 6

ICD Interface Control Document. 27

ICT Information and Communications Technology. 10, 30

IMF Initial Mass Function. XXIV, 3, 98, 100, 103

IOPS Input/Output Operations Per Second. 6

IoT Internet of Things. 44

IVOA International Virtual Observatory Alliance. 12, 13, 27

JAR Java Archive. 91

JVM Java Virtual Machine. 87

KDE Kernel Density Estimation. 119

LAN Local Area Network. 4

LHC Large Hadron Collider. 6, 41

LRMS Local Resource Management System. 5, 37

LSST Large Synoptic Survey Telescope. XI, 13, 24, 29, 95

MAGIC Multiattribute Grid Declustering. 67

MB Megabyte. 33, 54, 124

MCMC Markov chain Monte Carlo. 3, 18, 58, 95, 98, 99, 100, 101, 102, 108, 117, 122, 123, 133

MDS Monitoring and Discovery System. 37

MOC Mission Operations Centre. 39

MPI Message Passing Interface. 5, 6, 76, 134

MPP Massively Parallel Processing. XV, 14, 18, 42, 43, 44, 47, 48, 51, 55, 62, 64, 67, 68, 69, 70

NoSQL Not Only SQL. 44

NUTS No-U-Turn Sampler. 134

OAIS Open Archival Information System. XI, 12, 25, 27, 28, 61

OLAP On Line Analytical Processing. 43

OLTP On Line Transaction Processing. 43, 51, 70, 133

OPEX Operational Expenditure. 6

ORC Optimized Row Columnar. XI, 50, 51, 54, 57, 86, 94

PaaS Platform as a Service. 6, 38, 58, 133

PB Petabyte. 46, 61, 62

PDAD Present-Day Age Distribution. XIII, XIV, XXIV, 3, 14, 20, 98, 99, 100, 102, 112, 113, 114, 115, 117, 119, 122, 123, 124, 126, 128, 131, 134

PDF Probability Density Function. XIII, XIV, 98, 100, 103, 108, 109, 112, 113, 114, 117, 119, 134

PDMF Present-Day Mass Function. XIII, XIV, XXIV, 3, 14, 20, 98, 99, 100, 102, 103, 108, 109, 112, 109, 122, 123, 124, 126, 128, 131, 134

PDS Planetary Data System. 12

PVM Parallel Virtual Machine. 5

Q3C Quad Tree Cube. XII, 18, 66, 69, 70, 71, 72, 74

RAID Redundant Array of Independent Disks. 67

RDBMS Relational Database Management System. XV, 41, 42, 44, 48, 64, 71, 94, 95

RDD Resilient Distributed Dataset. 56, 57, 58, 122, 123, 133

REST Representational State Transfer. 28

RPC Remote Procedure Call. 30

RVS Radial Velocity Spectrometer. 68

S3 Amazon Simple Storage Service. 50, 81

SaaS Software as a Service. 6

SAMP Simple Application Messaging Protocol. 29

SDSS Sloan Digital Sky Survey. 35, 76, 94

SFH Star Formation History. 98, 113

SFR Star Formation Rate. XXIV, 3, 98, 100, 113

SGE Sun Grid Engine. 5, 37

SKA Square Kilometer Array. XI, 13, 24, 37, 44

SLA Service-Level Agreement. 6

SMP Shared-memory MultiProcessing. 4, 41, 42

SOA Service-Oriented Architecture. 28

SOAP Simple Object Access Protocol. 28

SOC Science Operations Centre. XI, 16, 17, 23, 25, 39, 57, 58, 62, 82, 92, 131

SPASE Space Physics Archive Search and Extract. 12

SPMD Single-Program Multiple-Data. 32

SQL Structured Query Language. XII, 14, 28, 35, 43, 44, 46, 48, 55, 57, 58, 64, 70, 76, 86, 88, 92, 95

SSAP Simple Spectral Access Protocol. 93

TAP Table Access Protocol. 28, 57, 92, 93

TB Terabyte. 35, 43, 54, 67, 124

UDF User Defined Function. 14, 31, 44, 45, 76, 90

URL Uniform Resource Locator. 31

VO Virtual Observatory. XI, 13, 19, 27, 28, 29, 30, 38, 57, 58, 61, 93

VWS Virtual Workspace Service. 37

WSDL Web Service Definition Language. 28

XML Extensible Markup Language. 28, 31

XSA XMM-Newton Science Archive. 32

YARN Yet Another Resource Negotiator. 95

Abstract

The massive amounts of data that the world produces every day pose new challenges to modern societies in terms of how to leverage their inherent value. Social networks, instant messaging, video, smart devices and scientific missions are just mere examples of the vast number of sources generating data every second. As the world becomes more and more digitalized, new needs arise for organizing, archiving, sharing, analyzing, visualizing and protecting the ever-increasing data sets, so that we can truly develop into a data-driven economy that reduces inefficiencies and increases sustainability, creating new business opportunities on the way.

Traditional approaches for harnessing data are not suitable any more as they lack the means for scaling to the larger volumes in a timely and cost efficient manner. This has somehow changed with the advent of Internet companies like Google and Facebook, which have devised new ways of tackling this issue. However, the variety and complexity of the value chains in the private sector as well as the increasing demands and constraints in which the public one operates, needs an ongoing research that can yield newer strategies for dealing with data, facilitate the integration of providers and consumers of information, and guarantee a smooth and prompt transition when adopting these cutting-edge technological advances.

This thesis aims at providing novel architectures and techniques that will help perform this transition towards Big Data in massive scientific archives. It highlights the common pitfalls that must be faced when embracing it and how to overcome them, especially when the data sets, their transformation pipelines and the tools used for the analysis are already present in the organizations. Furthermore, a new perspective for facilitating a smoother transition is laid out. It involves the usage of higher-level and use case specific frameworks and models, which will naturally bridge the gap between the technological and scientific domains. This alternative will effectively widen the possibilities of scientific archives and therefore will contribute to the reduction of the time to science.

The research will be applied to the European Space Agency cornerstone mission Gaia, whose final data archive will represent a tremendous discovery potential. It will create the largest and most precise three dimensional chart of our galaxy (the Milky Way), providing unprecedented position, parallax and proper motion measurements for about one billion stars. The successful exploitation of this data archive will depend to a large degree on the ability to offer the proper architecture, i.e. infrastructure and middleware, upon which scientists will be able to do exploration and modeling with this huge data

set. In consequence, the approach taken needs to enable data fusion with other scientific archives, as this will produce the synergies leading to an increment in scientific outcome, both in volume and in quality. The set of novel techniques and frameworks presented in this work addresses these issues by contextualizing them with the data products that will be generated in the Gaia mission. All these considerations have led to the foundations of the architecture that will be leveraged by the *Science Enabling Applications Work Package*.

Last but not least, the effectiveness of the proposed solution will be demonstrated through the implementation of some ambitious statistical problems that will require significant computational capabilities, and which will use Gaia-like simulated data (the first Gaia data release has recently taken place on September 14$^{\text{th}}$, 2016). These ambitious problems will be referred to as the Grand Challenge, a somewhat grandiloquent name that consists in inferring a set of parameters from a probabilistic point of view for the Initial Mass Function (IMF) and Star Formation Rate (SFR) of a given set of stars (with a huge sample size), from noisy estimates of their masses and ages respectively. This will be achieved by using Hierarchical Bayesian Modeling (HBM). In principle, the HBM can incorporate stellar evolution models to infer the IMF and SFR directly, but in this first step presented in this thesis, we will start with a somewhat less ambitious goal: inferring the PDMF and PDAD. Moreover, the performance and scalability analyses carried out will also prove the suitability of the models for the large amounts of data that will be available in the Gaia data archive.

Resumen

Las grandes cantidades de datos que se producen en el mundo diariamente plantean nuevos retos a la sociedad en términos de cómo extraer su valor inherente. Las redes sociales, mensajería instantánea, los dispositivos inteligentes y las misiones científicas son meros ejemplos del gran número de fuentes generando datos en cada momento. Al mismo tiempo que el mundo se digitaliza cada vez más, aparecen nuevas necesidades para organizar, archivar, compartir, analizar, visualizar y proteger la creciente cantidad de datos, para que podamos desarrollar economías basadas en datos e información que sean capaces de reducir las ineficiencias e incrementar la sostenibilidad, creando nuevas oportunidades de negocio por el camino.

La forma en la que se han manejado los datos tradicionalmente no es la adecuada hoy en día, ya que carece de los medios para escalar a los volúmenes más grandes de datos de una forma oportuna y eficiente. Esto ha cambiado de alguna manera con la llegada de compañías que operan en Internet como Google o Facebook, ya que han concebido nuevas aproximaciones para abordar el problema. Sin embargo, la variedad y complejidad de las cadenas de valor en el sector privado y las crecientes demandas y limitaciones en las que el sector público opera, necesitan una investigación continua en la materia que pueda proporcionar nuevas estrategias para procesar las enormes cantidades de datos, facilitar la integración de productores y consumidores de información, y garantizar una transición rápida y fluida a la hora de adoptar estos avances tecnológicos innovadores.

Esta tesis tiene como objetivo proporcionar nuevas arquitecturas y técnicas que ayudarán a realizar esta transición hacia *Big Data* en archivos científicos masivos. La investigación destaca los escollos principales a encarar cuando se adoptan estas nuevas tecnologías y cómo afrontarlos, principalmente cuando los datos y las herramientas de transformación utilizadas en el análisis existen en la organización. Además, se exponen nuevas medidas para facilitar una transición más fluida. Éstas incluyen la utilización de software de alto nivel y específico al caso de uso en cuestión, que haga de puente entre el dominio científico y tecnológico. Esta alternativa ampliará de una forma efectiva las posibilidades de los archivos científicos y por tanto contribuirá a la reducción del tiempo necesario para generar resultados científicos a partir de los datos recogidos en las misiones de astronomía espacial y planetaria.

La investigación se aplicará a la misión de la Agencia Espacial Europea (ESA) Gaia, cuyo archivo final de datos presentará un gran potencial para el descubrimiento y hallazgo desde el punto de vista científico. La misión creará el catálogo en tres dimensiones

más grande y preciso de nuestra galaxia (la Vía Láctea), proporcionando medidas sin precedente acerca del posicionamiento, paralaje y movimiento propio de alrededor de mil millones de estrellas. Las oportunidades para la explotación exitosa de este archivo de datos dependerán en gran medida de la capacidad de ofrecer la arquitectura adecuada, es decir infraestructura y servicios, sobre la cual los científicos puedan realizar la exploración y modelado con esta inmensa cantidad de datos. Por tanto, la estrategia a realizar debe ser capaz de combinar los datos con otros archivos científicos, ya que esto producirá sinergias que contribuirán a un incremento en la ciencia producida, tanto en volumen como en calidad de la misma. El conjunto de técnicas e infraestructuras innovadoras presentadas en este trabajo aborda estos problemas, contextualizándolos con los productos de datos que se generarán en la misión Gaia. Todas estas consideraciones han conducido a los fundamentos de la arquitectura que se utilizará en el paquete de trabajo de aplicaciones que posibilitarán la ciencia en el archivo de la misión Gaia (*Science Enabling Applications*).

Por último, la eficacia de la solución propuesta se demostrará a través de la implementación de dos problemas estadísticos que requerirán cantidades significativas de cómputo, y que usarán datos simulados en el mismo formato en el que se producirán en el archivo de la misión Gaia (la primera versión de datos recogidos por la misión está disponible desde el día 14 de Septiembre de 2016). Estos ambiciosos problemas representan el Gran Reto (*Grand Challenge*), un nombre grandilocuente que consiste en inferir una serie de parámetros desde un punto de vista probabilístico para la función de masa inicial (*Initial Mass Function*) y la tasa de formación estelar (*Star Formation Rate*) dado un conjunto de estrellas (con una muestra grande), desde estimaciones con ruido de sus masas y edades respectivamente. Esto se abordará utilizando modelos jerárquicos bayesianos (*Hierarchical Bayesian Modeling*). En principio, los modelos propuestos pueden incorporar otros modelos de evolución estelar para inferir directamente la función de masa inicial y la tasa de formación estelar, pero en este primer paso presentado en esta tesis, empezaremos con un objetivo algo menos ambicioso: la inferencia de la función de masa y distribución de edades actual (*Present-Day Mass Function* y *Present-Day Age Distribution* respectivamente). Además, se llevará a cabo el análisis de rendimiento y escalabilidad para probar la idoneidad de la implementación de dichos modelos dadas las enormes cantidades de datos que estarán disponibles en el archivo de la misión Gaia.

Chapter 1

Introduction

In summary, all great work is the fruit of patience and perseverance, combined with tenacious concentration on a subject over a period of months or years.

Reglas y Consejos sobre Investigación Científica: Los tónicos de la voluntad.
Santiago Ramón y Cajal.

1.1 Overview

The idea of data creating value is not new, however, the effective use of data is becoming the basis of competition in the private sector, and the enabler of wider and deeper knowledge in the scientific one. Internet companies like *Google*, *Facebook* or *Amazon* are good examples of how data can be the key to innovation, and how its successful processing and exploitation can disrupt entire industries and become a new source of growth. These companies leading the change are now including *Big Data* from both within and outside the enterprise, including structured and unstructured data, machine data, as well as online and mobile data, which supplement their organizational data and provide the basis for historical and forward-looking views (descriptive, predictive and prescriptive). *Big Data* is thus fundamentally changing the way businesses compete and operate. Data is the new currency. It holds the secrets to maintaining customer loyalty, shaping profitable initiatives, minimizing risks and boosting innovation and knowledge.

All this has led to a significant growth of *Big Data* platforms, tools and techniques that are making their way towards a variety of industries and scientific fields. These recent advances in large scale computing paradigms and environments enable new opportunities to extract the inherent value out of the vast amounts of data being currently generated. Nevertheless, their successful adoption is not straightforward in certain complex areas like science, as there are still some barriers that need to be overcome. Those comprise:

- The existence of models and legacy code that need to be adapted to the new distributed environments.

- The lack of high-level and use case specific frameworks, tools and models that facilitate a smoother transition.
- The scarcity of profiles with the balanced skill sets that can bridge the gap between the technological and scientific domains.

The European Space Agency's Gaia mission (Gaia Collaboration et al., 2016b; Mignard, 2005) is a good example of a scientific venture whose results will likely be the astronomical data resource for decades thereafter, representing a tremendous discovery potential. It will create the largest and most precise three dimensional chart of our galaxy (the Milky Way), by surveying more than one billion stars. Furthermore, it will also provide unprecedented position, parallax and proper motion measurements as per the mission science performance constraints (de Bruijne, 2012).

The resulting catalog, along with the raw data collected by the mission's instruments, will be made available to the scientific community, and will be analyzed in many different ways for many different purposes. In this scenario, the identification and availability of the proper analytical tools and frameworks will become crucial to ease the process taken by scientists when building models or conducting any other research.

To this extent, the *Gaia Science Enabling Applications* work package in general, and the *Data Mining* sub-work package in particular, have been established. They aim at a wide variety of requirements and functionality that will be provided to the community. These include:

- The provision of enriched and high level data access tools.
- Capabilities for cross-matching the final catalogue with other scientific surveys.
- A framework to perform complex queries (normally with some analytical workload).
- A set of models that can be utilized in a straightforward and seamless way, allowing composability for more complex use cases and configurability for different approaches.
- The means for managing the variety of workloads coming through (and their security constraints).
- Visualization of complex relations and high dimensional data sets.

Other concrete challenges (Brown, 2012) that are being tackled comprise the ability to re-analyze the raw data and do hypothesis testing against the raw image pixels (as well as model comparisons), the capacity to shift scientific workloads towards the data centre (bring the software and models to the data and not the other way around), the means to publish a *living data archive* (that can get updated with new results), the exposure of a window to science, and the seamless combination with other surveys or data sets (multi-mission archive).

Many of these topics, at least those not too bound to the scientific domain, are being addressed within the novel disciplines of *Big Data* and *Data Science*. The former suggests (among other things) a cost-efficient, highly scalable and multi-tenant frame architecture that gravitates around the so-called *data lake* (O'Leary, 2014), which is roughly no more than a scalable data repository that can securely hold and govern different types of data, sometimes in their native format, allowing its exploitation in different domains or use cases. The latter basically involves the skill set, mindset, techniques, etc. required to perform the analyses and tasks that transform the data into value for the specific area of expertise.

The application of these new disciplines to science is often cumbersome because of the complexity of any scientific research in the first place, and more concretely due to the non-negligible gap between the scientific and technological domains (as both usually demand a high level of specialization on their own). Some concrete examples in the Gaia mission scientific realm involve answers to questions like, how can we validate and compare already existing models of the Galaxy with the data that will be generated by the mission? Or, how can we effectively and efficiently build new models that help us explain what the observations are showing? Or, given the huge amount of data that will be produced, how can we learn and build non-parametric models that have no previous assumptions or restrictions of what the outcome might look like? In that case, would these non-parametric or strictly empirical approaches be good enough to model complex systems like our Galaxy (Bar-Yam, 2013)?

In this thesis, we aim at providing a qualitative and quantitative overview to these challenges from the Gaia mission perspective, even though much of the reasoning and the proposed approaches are directly applicable to other fields and industries. Furthermore, we will show how the evolution of technology in general and distributed systems in particular have driven the way scientific archives are designed and implemented. We will also address some of the key technological trends and advances that have lately appeared, i.e. *Big Data* and *Data Science*, and how they will help unleash the huge potential of the Gaia mission's archive.

Last but not least, we will lay out two ambitious statistical problems that require significant computational capabilities, and which will use Gaia-like data in the same format as the mission will produce (only simulations available so far). We will refer to this ambitious problem as the *Grand Challenge*, a somewhat grandiloquent name that will serve as a demonstration of the effectiveness of the approach taken. The *Grand Challenge* consists in inferring a set of parameters from a probabilistic point of view for the Initial Mass Function (IMF) (Chabrier, 2005) and Star Formation Rate (SFR) (Elmegreen and Scalo, 2006) of a given set of stars with a huge sample size, from noisy estimates of their masses and ages respectively. This will be achieved by leveraging the proposed architecture and techniques, and by using Markov chain Monte Carlo (MCMC), Gaussian processes and Hierarchical Bayesian Modeling (HBM). In principle, the HBM can incorporate stellar evolution models to infer the IMF and SFR directly, but in this first step presented in this work, we will start with a somewhat less ambitious goal: inferring the Present-Day Mass Function (PDMF) and Present-Day Age Distribution

(PDAD). The main difference stems from the fact that we will not be including in our analysis the mass that would now be encountered in the latest stages of stellar evolution (white dwarfs, neutron stars or black holes).

1.2 Evolution of Distributed Systems

Computing capabilities have evolved dramatically over the past decades, especially since the arrival of the Web. Moore's law (Moore, 2000; Brock and Moore, 2007) predicted that the number of transistors per chip would double every two years and, in a broader sense, the possibilities of integrated circuits would enable new fields of opportunity for personal computing, automatic controls and communications equipment. Similarly, Gordon Bell formulated his law of computer classes (Bell, 2008; Bell et al., 1972), which describes how types of computing systems form, evolve and may eventually die out. The classification ranges from mainframes in the 1960s (shared time systems with centralized resources), minicomputers in the 1970s, personal computers and workstations in the 1980s for applications executed locally and Local Area Network (LAN) networks, all the way down to client/server architectures (web-based applications) in the 1990s and new computing paradigms in the 2000s, like *Grid computing*, peer-to-peer and ubiquitous computing (network-based applications and mobile devices).

As technology matured and microprocessors caught up with supercomputers, parallel processing and the architectures and networks supporting them have also evolved considerably. Specifically, High Performance Computing (HPC) has undergone major architectural changes over the last decades, like moving from vector to parallel (towards multi-core and many-core architectures), from customized to standardized processors, and from single systems to clusters of commodity hardware (Resch and Küster, 2008). This last part has given rise to novel approaches like shared-nothing architectures (Stonebraker, 1986), whose main goal is to increase the scalability of the system by adding pieces of inexpensive hardware (usually without any interruption of the service), as well as shared-something (typically disk) architectures (Taniar et al., 2008), which have also been in the limelight when a compromise is sought in the extensibility limitation of the shared-memory approach and the load balancing problem of the shared-nothing one.

The most popular standard Application Programming Interface (API) for Shared-memory MultiProcessing (SMP) is *OpenMP* (Dagum and Menon, 1998). This API basically consists of a set of compiler directives and callable runtime library routines that influence run-time behaviour. Program execution begins as a single process, and executes serially until encountering a parallel construct, where the runtime forms a team of one or more processes and creates the data environment for each team member. Each of these processes are then executed in a different core. The programming model is simple yet powerful, and there are many compilers of *OpenMP* available in Fortran, C and C++ programming languages. However, scalability is obviously restricted to the number of cores that can be allocated in a shared-memory architecture, which, in a sense, is alleviated due to the latest trends in microprocessor design (Sutter, 2005), that are capping the clock speed and significantly increasing the number of cores per chip. One

could also argue that this trend was indeed the originator of such an API for concurrent programming.

Parallelism in HPC environments is achieved through message passing. Many different implementations like Parallel Virtual Machine (PVM) were developed before it was clear that the efforts were being duplicated and a standard would be more effective. That is how a broadly based committee of vendors, implementers and users proposed Message Passing Interface (MPI), a library specification for message passing that gives full control to the developer for implementing the best parallel algorithm (e.g. when to send what information to whom and how to synchronize the different nodes collaborating in the work). This approach proved to be very suitable for the scientific domain as shown in Foster (1995), and was a step forward towards the universal access to parallel and distributed computation. Once there was a standard which different institutions and private companies could rely on for parallel processing, new challenges for resource sharing arose, which in turn made way for the emergence of the *Grid*.

Grid computing enables the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations (Foster et al., 2001, 2002; Foster and Kesselman, 1999). One of its goals is to provide the necessary tools and services for efficiently managing the inherent complexity of multidisciplinary ventures (usually in the research field) by enabling the so-called *Virtual Organizations* (see Figure 1.1 for an example).

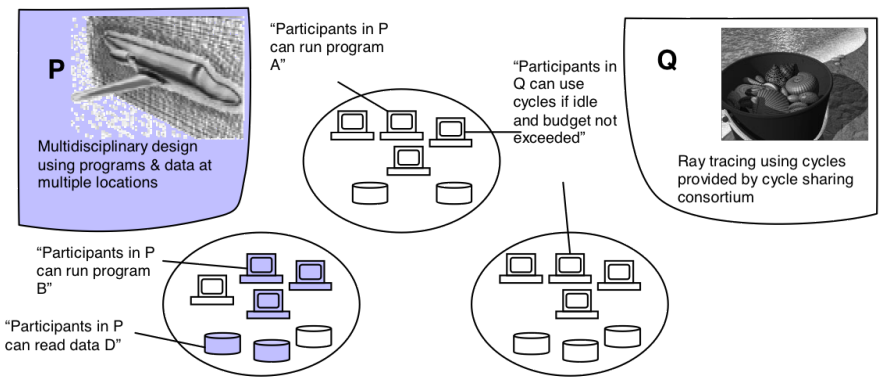


Figure 1.1: *Virtual Organizations* in *Grid computing*.

The de facto implementation for *Grid computing* is the *Globus Toolkit* (Foster, 2005). It consists of services for:

- Resource allocation, that can submit jobs to a cluster managed by a Local Resource Management System (LRMS), e.g. Sun Grid Engine (SGE) (Gentzsch, 2001).
- Information discovery and monitoring of status.
- Data management, allowing third-party transfers of huge data sets in a secure way.

- Meta-scheduling capabilities on top of the middleware for harnessing the federated infrastructure (Huedo et al., 2004).

Even if there are quite remarkable deployments of *Grid computing* technology for well renowned projects like the Large Hadron Collider (LHC) (Geddes, 2012), it did not fully engage the private sector due to its limitations in the definition of Service-Level Agreement (SLA) and usage metering, the lack of a sensible approach for load balancing, as well as concerns for resource sharing without strong enforcements. Furthermore, it did not generate a prosperous business landscape and did not offer new alternatives for parallel processing, leaving just the option of MPI, where the developer has to take care of the intrinsic problems of the distributed system.

Cloud computing is seen by many as the successor of *Grid computing* due to its suitability for resource sharing (with proper SLA enforcements), its capabilities for load balancing, and its on-demand configuration alternatives. There have been attempts such as Rodríguez et al. (2008) to integrate both technologies, where the *Grid* infrastructure is enlarged or shrinked by adding or removing virtual nodes, with the goal of adapting the available resources depending on the workload (computational needs). However, even if the *Grid* helped create a certain technology reality, which made *Clouds* possible, it is likely that *Grids* will be re-branded or merged into *Cloud computing* (Joszczuk-Januszewska, 2010).

The definition of *Cloud computing* presented in Mell and Grance (2011), states that it is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (like networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This *Cloud* model is composed of five essential characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service; three service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS); and four deployment models: private cloud, community cloud, public cloud and hybrid cloud.

Cloud computing has been possible due to advances in virtualization technology, i.e. reducing more and more the overhead of running a virtualized guest on a hosting server. Once the overhead of Central Processing Unit (CPU) bound workflows was decreased to negligible levels, Input/Output (I/O) became the main bottleneck as shown in Ghoshal et al. (2011) and Shafer (2010). Nowadays, it is rare to find a microprocessor that does not provide extensions for virtualization, thus making any overhead negligible. Furthermore, commercial *Cloud* services like Amazon Web Services (AWS)[1] offer the possibility of guaranteed Input/Output Operations Per Second (IOPS), ensuring all potential workloads can be met.

Even though server consolidation (and the implied reductions in capital expenditures and power consumption) is a use case worth remarking, the main driver for the adoption of the *Cloud* seems to be more related to the changes and the impact it produces in how businesses and institutions look at computing power. Companies can now move away

[1] https://aws.amazon.com/

from a traditional Capital Expenditure (CAPEX) model, where they buy a dedicated hardware and depreciate it over time (normally overdimensioned for covering peaks in demand), towards an Operational Expenditure (OPEX) model where they pay as they go, typically more than what a similar capacity would cost when operated in-premise, but avoiding the fixed costs of an underutilized infrastructure, and with full tightness to the company's business volumes at all times. This elasticity (Herbst et al., 2013), now possible in the *Cloud*, was one of the main goals when the *Grid* originated (the idea of computing becoming a utility, like the power grid), which turned out to be its greatest failure, even though, as stated above, it largely contributed to create momentum for such a disruption. All in all, we can observe an explosion of the number of Internet-based start-ups, thanks to the new reality in computing power supply. It has now become possible for an entrepreneur to fully operate a company with just a laptop in a coffee house, provisioning the rest of the company's computing (IaaS), platform (PaaS) and service (*SaaS*) needs from the *Cloud*.

Nevertheless, *Cloud computing* does not truly offer new paradigms for tackling and harnessing the amount of data being generated in the digital economy. It is more of a complementary technology which new parallel processing paradigms and disciplines (i.e. *Big Data* and *Data Science*) have mutually favoured from.

1.3 Big Data and Data Science

It is difficult to find a definition for *Big Data*, as the term has quickly spreaded across multiple industries and domains due to its generality and applicability to all fields and disciplines, including business, science, social, health, etc. One broadly accepted approximation for this buzzword might be the so-called three *Vs*, i.e. data that is too large in Volume, moves too fast and needs a prompt response (Velocity), and varies a lot in format, not fitting into any existing processing tool or data management system (Variety).

However, as sharply pointed out in Sicular (2013), these three *Vs* (originating from Gartner) just represent one of the parts of their definition. Quoting this article, "*Big Data* is high-volume, -velocity and -variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making". It is then clear that the value of *Big Data* in the general sense is not intrinsic to the characteristics of the data, but to the (cost-effective) innovations for information processing, and especially to the insights and decision making that will be enabled, being this last part one of the goals of this research.

The three *Vs* (see Figure 1.2) are a priori easier to understand though, not only because the idea of extracting value out of data is not new and somehow is widely accepted, but also because stressing too much the value of the innovations for information processing might seem biased towards technological companies (and their proprietary solutions), in an effort to force their customers to move to *Big Data* and capitalize on the latest trend. Furthermore, because of the probably badly chosen adjective (i.e. *Big*), many still think that volume is the most important dimension. This perception does

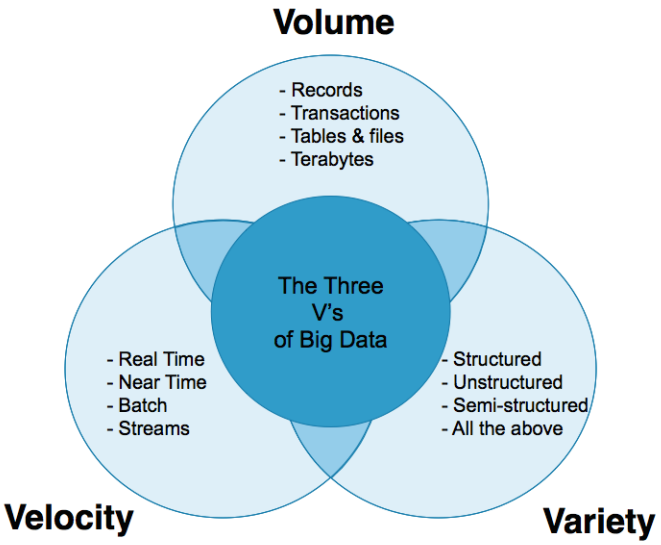


Figure 1.2: The three *Vs* in *Big Data*.

not logically fit in the current connected world, where the real value resides in tapping into the relevant sources of data (variety) and mashing them up for producing a real and lasting competitive advantage. In addition, velocity is often equated with near-real time analytics (as remarked again in Sicular (2013)), but it also deals with linking data sets coming at different speeds or adapting for changes in the temporal relationships between them.

One way or another, the availability of an heterogeneous but well-integrated ecosystem for *Big Data* is now a reality. There are open source and commercial distributions available for institutions to roll out, and the only pitfall for its successful adoption seems to be the lack of both strategy and knowledge about what to do with these humongous amounts of data. This gets accentuated by the hype around *Big Data*, as well as the fact that the data revolution is happening alongside business as usual, which can be extremely overwhelming (Marr, 2015).

The lack of expertise in the field originates in the complexity of the tasks involved when extracting value from data, i.e. how to consolidate and govern, how to efficiently process and transform into insights, and how to report it in a way that all stakeholders get what they need (nothing less but nothing more). The diversity of the roles needed for such an endeavor makes it even more frustrating, often because those roles are not used to collaborating with each other in many (typical) traditional institutions. Examples for this might be business people and executives being unable and/or unwilling to understand what technology may do for them and what it is all about (even at a high level), or developers just focusing on their daily work, without the courage or strength to look beyond into the larger picture.

Data Science has emerged to specifically designate a new profession that is expected to fill this gap, and therefore make sense of the vast amounts of data that we find in every discipline and industry. Its definition is again a bit fuzzy, but everybody seems to agree on the fact that (i) it is something that has appeared for a reason (even before the term *Big Data*), (ii) needs to somehow evolve and further define its attributions, and (iii) should be taught to the next generations because the data amounts they will face will keep on soaring (Cleveland, 2001; Graham, 2012; Borne et al., 2009). Figure 1.3 shows the three fields which any *Data Scientist* should have enough expertise in. A deep mastery is not required in all those three pillars as it is obviously impractical in the least, but all those three skills must be present to some extent in any *Data Scientist* role. These are in more detail:

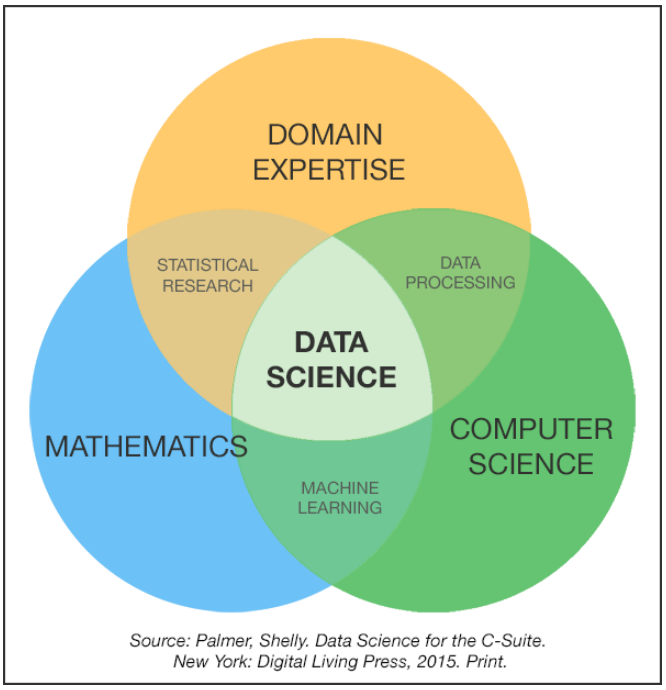


Figure 1.3: Skill sets for a *Data Scientist*.

- Mathematics in general and descriptive/predictive statistics in particular. *Data Scientists* can explore data sets and identify patterns in data. They are acquainted in statistical modeling, probability models, pattern recognition, predictive analytics, sampling techniques, graph theory, etc. They can apply unsupervised learning methods and techniques (e.g. clustering, principal components analysis, graphical models and so on), supervised learning ones (e.g. classification, regression, decision trees, feature selection, cross-validation, etc), and most importantly, they should actually know each technique strengths and shortcomings and thus when to use each of them, when to combine several and so forth.

- Computer Science. *Data Science* happens inside computers and the data sets involved are normally high-volume, -velocity and -variety (*Big Data*). Therefore, models must be ready to be scaled to the sizes/volumes/variety of data sets found in real deployments. The ability to scale an algorithm requires expertise in *Big Data* techniques and distributed processing. Very much like statistical methods, one size will not fit all problems (Stonebraker and Cetintemel, 2005). Then, there is always a trade-off in the data processing techniques and frameworks to use in each situation. Coding skills are also mandatory and some authors even talk about hacking skills, deepening into the fact that this discipline is about creativity and perseverance, i.e. keep on trying until a reasonable solution to the problem is found.

- Domain expertise. A *Data Scientist* represents an evolution from the business or data analyst role. The formal training is similar as remarked in the other two skill sets, but what draws the *Data Scientist* apart is strong business acumen or field knowledge, coupled with the ability to communicate findings (or tell stories) to both technical and non-technical audiences in a way that can influence how an organization approaches a business challenge. Good *Data Scientists* will not just address business problems, they will pick the right problems that have the most value to the organization. Whereas a traditional data analyst may look only at data from a single source. e.g. a Customer Relationship Management (CRM) system, a *Data Scientist* will most likely explore and examine data from multiple disparate sources. They will sift through all incoming data with the goal of discovering a previously hidden insight, which in turn can provide a competitive advantage or address a pressing business problem.

Therefore, *Data Science* is not simply about collecting and reporting on data, but also on looking at it from many angles, determining what it means and then recommending ways to apply the data. It is a discipline that requires an inquisitive mindset: exploring, asking questions, doing “what if" analysis and questioning existing assumptions and processes. It also includes the communication of informed conclusions and recommendations across an organization’s leadership structure.

The need for multidisciplinary profiles is not something new. If we look at astronomy and astrophysics, we can identify two new interdisciplinary areas that have lately appeared. Those are astrostatistics (Sarro et al., 2014) and astroinformatics (Accomazzi et al., 2013). Astrostatistics is a discipline that spans astrophysics, statistical analysis and data mining. It could be framed into the statistical research area of Figure 1.3. Furthermore, astroinformatics, which involves the combination of astronomy and Information and Communications Technology (ICT), might be represented in the data processing overlap in Figure 1.3.

By defining *Data Science* as being the intersection of such three complex subjects, i.e. computer science (*Big Data*), mathematics and the particular scientific or business domain, the expectations of finding talent with a suitable mix of these skill sets decrease substantially. One way of mitigating this scarcity is by setting up teams whose members span all three disciplines (when combined in a group), and use appropriate methodologies

and techniques to reduce the frictions that may arise in such diverse configurations. In addition, the exchange of ideas and knowledge sharing should be fostered so that a learning-by-doing culture can be established. Team members must be responsible and accountable for the overall team goals, as otherwise this recipe might not succeed.

1.4 Scientific Archives

The archival of scientific data and supporting documentation is crucial to research and innovation. The scientific reasons for preserving data derive from the fact that observations, knowledge and understanding are cumulative. We then believe that the more complete the record, the more we can extract from it. Observed data provide a baseline for determining rates of change and for computing the frequency of occurrence of unusual events. The longer the record, the greater our confidence in the conclusions we draw from it. Our traditional observational records have portrayed frozen instants of reality. If preserved, they will continue to provide insights, but if neglected, they will melt away (Council, 1995).

There are thus strong motivations for preserving these scientific data sets:

- Many observations about the natural world are a record of events that will never be repeated exactly. Examples comprise observations of an atmospheric storm, a deep ocean current, a volcanic eruption, and the energy emitted by a supernova. Once lost, such records can never be replaced.
- Observed data provide a baseline for determining rates of change and for computing the frequency of occurrence of unusual events. They specify the observed envelop of variability. The longer the record, the greater our confidence in the conclusions we draw from it.
- A data record may have more than one life. As scientific ideas advance, new concepts may emerge (in the same or entirely different disciplines) from study of observations that led earlier to different kinds of insights. New computing technologies for storing and analyzing data enhance the possibilities for finding or verifying new perspectives through reanalysis of existing data records.
- The substantial investments made to acquire data records justify their preservation. The cost of preservation will almost always be smaller than the cost of observation. Because we cannot predict which data will yield the most scientific benefit in years ahead, the data that is discarded today may be the data that would have been invaluable tomorrow. This includes the raw data before any treatment is applied to it, because new ways of processing, calibrating and analysing will certainly produce different outcomes, usually with more quality and accuracy.

Scientific archives in the fields of astronomy and astrophysics follow these patterns and reasoning. Any astronomical and astrophysical research is nowadays likely to include data coming from a variety of archives, typically collected by sky surveys operating at

different wavelengths. This reinforces the idea that the knowledge acquired by each and every experiment is cumulative, effectively leading to more robust results. Furthermore, the cost of storing, preserving and making this data available to the scientific community is way lower than the cost of its acquisition. This is mainly due to:

- High assembly and operational costs of ground based telescopes. These telescopes are normally installed at high altitude places, avoiding a significant portion of the Earth's atmosphere, to diminish the effects of the weather conditions, turbulences, absortion of infrared and submilimiter wavelenths by water vapor, etc.
- Very expensive costs to assemble, launch and operate a space satellite. Astronomical space missions are commonly required when we need to perform high accurate measurements or observe far distant objects. The Gaia mission (Gaia Collaboration et al., 2016b; Mignard, 2005) or the Herschel Space Observatory (HSO) (Pilbratt, 2008) are good examples of such space missions.
- There has been a significant boost of open source software initiatives (Laurent, 2004). This holds true for many of the engines and tools in the Big Data and Data Science fields, reducing entry barriers for adoption and lowering operational costs.
- The commoditization of storage and data processing hardware together with the fact that these new open source tools and engines are designed to run on this cheaper hardware, leads to much lower costs.

However, the complexity of the different data archives pose a lot of issues on its own for effectively and efficiently squeezing out their inherent value. The architecture of a scientific archive is thus something difficult to generalize. The Open Archival Information System (OAIS), in its reference model (Lavoie, 2004), identifies and describes six services or functional components that any scientific archive should implement. Those refer to *Ingest*, *Archival Storage*, *Data Management*, *Preservation Planning*, *Access* and *Administration*. These six building blocks ensure respectively that the data can be taken to the archive, digitally stored, its metadata is made available, there is a preservation strategy in place, data can be accessed by relevant parties and its day-to-day operations and activites are defined.

Furthermore, the availability of more and more observed data, with different degrees of overlap, raises the need to combine it in a meaningful way, producing synergies along the way. We refer to this concept as data fusion, which is no more than the integration of multiple data sets and knowledge about the same real-world object or phenomena into a consistent, accurate, and useful representation. This is crucial for scientific research as it provides different observations and perspectives about the same reality. As an example in the field of astronomy, data fusion facilitates (among other things) the task of cross-matching objects from different catalogues surveyed at different wavelengths, enabling both a richer exploration of the galaxy and ways of cross-validating hypotheses.

To address this challenge, virtual observatories are being established in a wide range a disciplines, supported by a variety of agencies. Groups such as the International Virtual

Observatory Alliance (IVOA), Planetary Data System (PDS) and the Space Physics Archive Search and Extract (SPASE) consortium are defining metadata standards to aid in archiving and sharing of information resources. The role of the virtual observatories is to locate available resources and help users find what they need and then gain access to it. There are many different existing resource providers from which virtual observatories must collect descriptions for. These resource providers may have associations with other providers so the topology of information exchange can often become complicated (King et al., 2008).

In Astronomy, the Virtual Observatory (VO) was born to address interoperability and integration of both tools and data sets, utilizing the Internet to form a scientific research environment in which scientific research programs can be conducted. Its main goal is to allow transparent and distributed access to data available worldwide. This naturally enables scientists to discover, access, analyze, and combine heterogeneous data collections in a user-friendly manner. VO standards are being driven and agreed within the IVOA. These standards focus on information registries, query languages, data models, semantics, data access, protocols and visualization.

1.5 Motivation and Main Objectives

Scientific data output is currently increasing at 30% every year (Pryor, 2012). Some studies (Vines et al., 2014) conclude that the usage of existing scientific data sets decline 17% per year, with 80% of them being simply unavailable after 20 years. Given that a lot of research endeavours are nowadays publicly funded, more and more presure is being allocated to them in order to get an optimum return on investment, not only from the specific project outcome perspective, but also from the potential synergies produced when leveraging the results of other existing undertakings (and those to come).

This is particularly the case in astronomy and astrophysics, where the data being (or to be) collected by both ground and space based instruments and satellites is growing exponentially. Some examples of experiments and missions producing more and more data include the Gaia mission itself (Gaia Collaboration et al., 2016b; Mignard, 2005), Euclid (Laureijs et al., 2011), the Large Synoptic Survey Telescope (LSST) (Ivezic and Tyson, 2008) or the Square Kilometer Array (SKA) (Dewdney et al., 2009). They will produce data sets ranging from a petabyte for the entire mission in the case of Gaia to 10 petabytes of reduced data (output data after initial processing) per day in the SKA.

This data deluge in astronomy has fostered the appearance of two new interdisciplinary areas, the aforementioned astrostatistics (Sarro et al., 2014) and astroinformatics (Accomazzi et al., 2013), as a way of seamlessly integrating the different skillsets found in disciplines like astronomy/astrophysics, computer science and statistics. These new disciplines will efficiently combine the ever-increasing sources of data in the field. The succesful integration of the architecture, data sets and skills will be the key to increase research output both in volume and in quality. This will certainly allow scientists to do exploration and modeling in a smooth, integrated and effective way.

Furthermore, raw data (re-)analysis is becoming an asset for scientific research as it

opens up new possibilities to scientists that may lead to more accurate results, enlarging the scientific return of every mission. In order to cope with the large amount of data, the approach to take has to be different from the traditional one in which the data is requested and afterwards analyzed (even remotely). One option is to move the workloads to Cloud environments where one can upload the data analysis work flows so that they run in a low latency environment and can access every single bit of information.

Quite a lot of research has been going on to address these challenges and new computing paradigms have lately appeared such as NoSQL databases (that relax transactional constraints), or other *shared-nothing* architectures like Massively Parallel Processing (MPP), MapReduce (Dean and Ghemawat, 2008) and higher level generic distributed engines like Apache Spark (Zaharia et al., 2012). These new architectures emphasize the scalability and availability of the system over the structure of the information and the savings in storage hardware this may produce. In this way, the scale-up of problems is kept reasonably close to the theoretical linear case, allowing us to tackle more complex problems by investing more money in hardware instead of making new software developments which are always far more expensive.

An interesting feature of this new type of data management systems is that they do not impose just a declarative language like Structured Query Language (SQL), but they also allow users to plug in their algorithms no matter the programming language they are written in and let them run and visit every single record of the data sets. Then, users can leverage the best of all worlds depending on the workload, i.e. declarative through SQL and/or functional programming, but also imperative in a more procedural way. This may also be accomplished to some extent in traditional SQL databases through User Defined Function (UDF), although code porting is always an issue as it depends a lot on the peculiarities of the database, and debugging is not straightforward (Pavlo et al., 2009).

Therefore, scientists and many application developers are often more experienced at, or may feel more comfortable with, embedding their algorithms in a piece of software (i.e. a framework) that sits on top of the distributed system, while not caring much about what is going on behind the scenes or about the details of the underlying system. This will effectively bring the scientists to the data centre, will enable them to work close to the data archive, and perform tasks such as exploration, discovery, modeling or visualisation in a low latency environment that allows both declarative and programmatic workloads.

This new perspective reduces the limitations in scalability found in current approaches where the data is first downloaded from the archive and then analysed locally, often on the scientist workstation. The new ideas to be taken to the Gaia mission archive must allow scientists to devise and run their models on the archive infrastructure, which is scalable by design, in order to find the best model of our galaxy (the Milky Way), and unravel its structure and formation history, which is the main science driver for Gaia. Such a model should also explain the stellar populations in the Galaxy and thus make predictions for their distributions in age, luminosity, metallicity and chemical abundance patterns (Brown, 2012).

Constructing these models is a non-trivial task. In this thesis, we aim at providing the bases of how a model dealing with the Gaia billion-source catalogue can be implemented

in a distributed and scalable environment. This refers to the previously introduced Grand Challenge, which focuses on an ambitious statistical problem that will require significant computational capabilities for inferring a set of parameters (from a probabilistic point of view) for the PDMF and PDAD. It will be achieved by using HBM.

The models are validated with simulated data, although the simulations are in the same format as they will be made available in the Gaia archive. This will empower a direct usage of the models with real observed data once it is available. Moreover, we benchmark the execution times in different scenarios and with different configurations, and demonstrate that they scale to bigger data sets and that performance is improved when adding more resources, proving the suitability of the approach taken. In addition, constraints in the number of parameters used, optimizations done to boost performance and other decisions made along the way are also analyzed and discussed. These will guide scientists in this or other fields of research to:

1. Identify the areas to focus on, and how to work around the potential issues that may appear, leading to shorter times to science in massive data archives.

2. Evolve them to more complex use cases considering other attributes that will be observed and collected throughout the Gaia mission.

3. Serve as a demonstration of the effectiveness of the approach taken, and learn, by example, how to successfully leverage new distributed processing paradigms when adapting other already existing models and pipelines.

The ultimate goal of the architecture depicted in the Grand Challenge is not only to provide the building blocks for data mining within the Gaia mission realm, but also to serve as a frame reference for enabling the interoperability of those models with other data archives available at the ESAC Science Data Centre (ESDC) or elsewhere. Furthermore, the concepts of a collaborative or "living" archive (Brown, 2012) where algorithms, models and the resulting data can be made available to the scientific community, or where an *application store* can enable the community to participate in the development of the future archive *apps*, are getting traction in new generation archives (O'Mullane, 2011).

Moreover, the architecture for these new generation scientific archives will also need to empower a *window to science* and the relevant back end mechanisms for multi-mission interfaces like Merín et al. (2015), perhaps to also allow data mining and other resource intensive processing activities in addition to data exploration and retrieval. This may be the next focus of virtual observatories, which will certainly benefit from scalable and multi-purpose architectures and techniques that can even integrate backwards with the pipelines generating the data to be exposed to the community in the same infrastructure.

The research presented in this thesis does not intend to provide a comprehensive guide of techniques for large-scale data analysis but just some important topics to account for when tapping into these disciplines from the perspective of the future *collaborative* science archives. These new generation data archives will need to successfully make use of all available technologies like Cloud computing, column orientation, or new general purpose large scale data processing engines and programming paradigms, to offer a more efficient

and integrated environment for science. The best way to accomplish this is by shedding some light and contributing to the trends that need to be adopted, and by presenting some real (and ambitious) scientific use cases and examples.

1.6 Main Contributions and Roadmap of the Thesis

The research presented in this document encompasses an extensive study of several technological disruptions in the distributed systems field (for around ten years), sometimes contributing to its conceptualization. Furthermore, it includes the comparative assessment of the suitability and applicability of some aspects of those disruptions to massive astronomical archives, contextualizing and exemplifying with the data that will be produced in the European Space Agency (ESA) Gaia mission. The validation of all these considerations is eventually performed through the implementation of two ambitious statistical problems that are key for the scientific results of Gaia (i.e. the Grand Challenge). The models built require significant computational capabilities given the large amount of observations that will be made available, and thus a scalable approach is proposed and assessed.

The main contributions and achievements of this thesis can be summarized as follows:

- An extensive study of how innovative computing architectures can shift the way scientists access data archives from the old approach of querying their metadata and then download the relevant data products, to a more modern way of taking the software and analyses to the data centre where the scientific archives are stored. Scientists are then provided with far more computing power and we avoid any latencies incurred when moving data around (sometimes through the Internet). The first way proposed in this thesis consists of using Grid computing innovations to let users submit their processing jobs to the data centre where the data is placed. This proves to be a step forward but has some limitations like the fact that the data is still downloaded to the worker nodes (even if they are in the same data centre), or that different tasks cannot cooperate to accomplish a shared goal (e.g. exchanging data among them during the process). Research leveraging Cloud computing reduces costs by adapting the resources devoted to a task depending on the workload, but the paradigm stays the same. A proper usage of the so-called Big Data technologies, MapReduce in the case of the framework for hypercubes, and Apache Spark for the Grand Challenge, enables the implementation of this new approach that can handle larger data sets, and which offers wider possibilities to combine different observations of the same reality. A set of guidelines are also summarised on how to take advantage of new multi-purpose distributed processing engines for transitioning towards more collaborative archives. These archives can be queried and analysed with both declarative and programming languages, they are interoperable and interactive, and they can also be consolidated with the Science Operations Centre (SOC). This last bit facilitates code and infrastructure reuse, enabling re-processing of the archive and reproducibility of the pipelines.

- Implementation of an innovative framework on top of MapReduce for creating ad-hoc and generic summary statistics (applied to the Gaia catalogue). The framework is meant to generate several hypercubes at the same scan of data (which can be filtered differently for each of them). The cardinality of the key space may also change independently per hypercube and so can the number of dimensions. The way the framework is built empowers scientists to plug in their own (new or already existing) code/analyses without dealing with any specific interface to the scalable distributed system below (MapReduce). This is the main challenge for the adoption of cutting-edge technologies in science. They require a high degree of specialization that many scientists lack of (for obvious reasons). Valuable feedback and user experience is collected when exposing PhD students and senior scientists to this higher level framework that helps them deal with the massive amounts of data they will face. This validates one of the main contributions made in this thesis: bridge or reduce the gap between scientists and the latest technological advances for data processing. This approach, taken to the extreme, could certainly let scientists focus on more ambitious use cases that would eventually increase the scientific return of expensive missions like Gaia.

- Development of ad-hoc capabilities for column orientation on top of Hadoop Distributed File System (HDFS). At the time the implementation was performed, it did not exist any other alternative that could store data in a columnar format on top of HDFS. The rapidly evolving Big Data ecosystem and the slow peer review processes in journals forces the final version of the publication (in its latest iteration) to include references to other more comprehensive implementations. Nevertheless, at the time the implementation took place it was a clear innovative contribution that has been validated by the quick adoption from the community these days. Furthermore, different data storage model configurations are assessed for a Gaia-like simulated catalogue, e.g. Gaia Universe Model Snapshot (GUMS) version 10. Row or column oriented formats, different compression algorithms and presence or absence of locality are benchmarked among themselves to show what needs to be considered when porting a scientific data set into HDFS. These include the optimizations to be done for I/O bound workflows: column-orientation, perform several analyses with one scan of data, light (fast) compression techniques, and not pay much attention to the serialization scheme as the time to (de)serialize is always negligible compared to the (de)compression one. In the concrete case studied, the serialization technique was operationally pre-defined by the Gaia SOC.

- A thorough analysis, including the innovative approach taken in the framework, of the different algorithms and tools available for performing data aggregations in Hadoop. The scalability concerns of the solutions and the features to consider when picking either of them are also assessed. This proves that in certain cases it makes more sense to develop and ad-hoc application instead of using a generic (existing) tool (like Apache Hive or Pig). The contribution is validated throughout the benchmark by showing a lower execution time (or equal at most) even if the

selected strategy for aggregation (i.e. MergeSort) is less suitable than the one configured with the generic tool (Apache Hive). The possibility to perform several aggregations at the same data scan makes it more adequate than other declarative counterparts that would require several data scans given the limitations of their declarative languages available to end users.

- Applied techniques for partitioning astronomical catalogues in MPP through the usage of Hierarchical Equal Area Iso Latitude Pixelation of a sphere (HEALPix) and Quad Tree Cube (Q3C). This enables faster geometrical queries and the ability to collocate different catalogues so that we can perform more efficient geometrical operations like cross-matches.

- Implementation of two probabilistic models focusing on one particular astrophysical problem: the inference of a full spatio-temporal Milky Way galaxy model from Gaia observations of billions of stars. This is the ultimate goal of the Gaia mission and, in its most complex version, is certainly beyond present-day computing facilities. The models presented as part of this research are a first, simplified (yet very ambitious) step towards that ultimate goal. In the implementation, an already existing MCMC sampler is leveraged, a situation that occurs very often in science where existing software needs to be reused (not ported or reimplemented). The models built serve the purpose of exemplification, and offer ways for learning by comparison, which will facilitate potential replications to other problems. They scale to bigger data sets and performance is improved when adding more resources, proving the suitability of the approach taken. The aim of these models is also to provide the reference of how a model dealing with the Gaia billion-source catalogue can be implemented in a distributed and scalable environment.

One of the key results of this research has been the definition of a data mining architecture for Astrostatistics. It will be used by scientists around the world when dealing with the rich data sets that will be produced in the Gaia mission during the next few years.

The most important contributions of this thesis to scientific literature comprise the following peer-reviewed publications enumerated below. Some other co-authored publications have not been formally presented as part of this work because they were either published in non-indexed conferences, or the contribution was not so extensive to be considered as such. However, their content is still relevant (and impactful), and it has affected in one way or another the research presented in this document. They will be mentioned and/or cited within the roadmap below and/or in every chapter where appropriate.

Publications in international journals:

- D. Tapiador, A. Berihuete, L.M. Sarro, F. Julbe, E. Huedo, *Enabling data science in the gaia mission archive: The present-day mass function and age distribution*, **Astronomy and Computing**, 19 (2017) 1 - 15.
- D. Tapiador, W. O'Mullane, A. Brown, X. Luri, E. Huedo, P. Osuna, *A framework for building hypercubes using mapreduce*, **Computer Physics Communications**, 185 (5) (2014) 1429 - 1438.
- A. Ibarra, D. Tapiador, E. Huedo, R. S. Montero, C. Gabriel, C. Arviset, I. M. Llorente, *On-the-fly xmm-newton spacecraft data reduction on the grid*, **Scientific Programming**, 14 (2) (2006) 141-150.

Publications in international conferences:

- M. Rodríguez, D. Tapiador, J. Fontán, E. Huedo, R. S. Montero, I. Llorente, *Dynamic provisioning of virtual clusters for grid computing*, in: **Euro-Par 2008 Workshops - Parallel Processing**, Vol. 5415 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 23-32.

Roadmap As previously stated, the research presented in this thesis does not intend to provide a comprehensive guide of techniques for large-scale data analysis but some important contributions that will drive the adoption of new processing techniques and environments, as well as the foundations of one particular astrophysical problem, which in its most complex version, represents the ultimate goal of the Gaia mission.

The new generation data archives will need to successfully make use of several technology advances like Cloud computing, column orientation, or new general-purpose large scale data processing engines (and programming paradigms), to offer a more efficient and integrated environment for science. The best way to accomplish this is by presenting some real and ambitious scientific use cases and examples, where technological innovations play a crucial role in their implementations. This is indeed the approach that has been taken for this work.

The rest of the document is structured as follows. The order of the chapters somehow resembles the chronological order of the research put forward in the thesis:

- Chapter 2 will give an overview of the components of a scientific archive, the main goals and services of the VO and an overview of the different generations of archives in Astronomy. It will also discuss the contributions made to leverage key technologies like Grid and Cloud computing in the technological shift towards bringing the analysis workloads to the data centres (and not the data to the consumer's environment). Furthermore, some of the ideas for the next generation archives (more collaborative and integrated) will also be highlighted as a later checklist for the innovations presented afterwards. It is important to remark that the contributions presented in this chapter were deployed at European Space Astronomy

Centre (ESAC), successfully running the pipelines for the Herschel mission[2], providing on-the-fly re-processing capabilities for XMM-Newton[3] data products, and also for mosaic construction from observations coming through Integral[4]. Some co-authored and related publications not formally presented as part of this thesis include Osuna et al. (2010); Fajersztejn et al. (2011); Fernandez et al. (2010); Leon et al. (2010); Ibarra et al. (2006); Tapiador et al. (2007); Ceballos et al. (2010); Vazquez et al. (2010); Gabriel et al. (2008); Arviset et al. (2008); Ibarra et al. (2007); Planck Collaboration et al. (2011).

- In Chapter 3, we cover the latest technology advances in *Big Data* and *Data Science*, and how they can be effectively applied for enabling the large scale data mining capabilities that are needed for the successful exploitation of the Gaia archive. It sets the technological foundations upon which the research taking place in the following chapters is based on. The topics included are the emerging concept of the data lake, in-memory capabilities for iterative processing (useful for data mining and machine learning algorithms), and most importantly, the reasoning behind the transition from monolithic relational architectures to distributed and scalable environments where the data is stored in a denormalized and schema-less way. The publications originally describing the need for a change are Tapiador et al. (2014, 2017).

- We will deep dive in some of the main challenges of the Gaia mission archive in Chapter 4. The techniques applied to partition astronomical catalogues as well as the innovative framework for ad-hoc and generic descriptive statistics have been key for setting up the Gaia Coordination Unit (CU) 9, which is responsible for building the Gaia mission archive. Furthermore, other contributions in how to leverage scalable distributed architectures (for scientific workflows), column-oriented innovations on top of HDFS, user experiences with the framework, and the benchmark of different storage model optimizations and techniques for data aggregation are discussed in depth. The chapter contributions are mainly supported by Tapiador et al. (2014); Gaia Collaboration et al. (2016b,a, 2017); Tapiador (2012, 2011).

- Chapter 5 describes the Grand Challenge. It contains a detailed explanation of the PDMF and PDAD models implemented, along with their mathematical and scientific justification. A variety of experiments are discussed for proving that the speed-up and scale-up of the solution suits the needs of the enormous catalogue that Gaia will generate. Furthermore, the models are ready to consume Gaia data in the same format it will be produced, being then ready for producing science. Moreover, they will be evolved to tackle one of the main goals of the Gaia mission, i.e. the inference of a full spatio-temporal Milky Way galaxy model from Gaia observations of billions of stars. Last, the implementation will also serve as an example of

[2]http://www.cosmos.esa.int/web/herschel
[3]http://www.cosmos.esa.int/web/xmm-newton
[4]http://www.cosmos.esa.int/web/integral

how to build sophisticated analyses on top of cutting-edge distributed processing engines, reusing existing software packages when appropriate. In consequence, many scientists will use it as a reference in order to shorten the time taken for modeling with such a vast data set. The chapter contributions are backed by Tapiador et al. (2017).

- Chapter 6 draws the conclusions of this research and the future areas of work enabled in this thesis, which comprise the scientific use and evolution of the models and frameworks implemented, as well as the advances in large scale data processing applied to astronomical archives.

Chapter 2

Evolution of Scientific Archives

By denying scientific principles,
one may maintain any paradox.

Galileo Galilei.

Scientific archives host the main deliverable of missions and experiments in astronomy. The way they are built and the way scientists interact with them determine to a large degree how successful the scientific return on the investment will be. This chapter presents the evolution of the implementation approaches that have taken place to cope with the more demanding scenarios of new missions. This offers new opportunities to scientists for data analysis, archive re-processing reusing existing pipelines, sharing of new data items derived from the archive contents, and some other ideas related to more consolidated approaches where the same infrastructure can be leveraged for the entire lifecycle of the data products (i.e. their reduction, archiving and exploitation).

The original contributions made in the chapter include the utilization of Grid and Cloud computing technological advances for performing the first step in bringing the software to the data centre, instead of the more limited approach of downloading archived data into the scientist's local desktop for analysis. As previously discussed, the research has been leveraged to build the infrastructure that has eventually processed the SOC pipelines for the Herschel mission, providing on-the-fly re-processing capabilities for XMM-Newton data products, and also for mosaic construction from observations coming through the Integral satellite.

The publications endorsing these contributions include Ibarra et al. (2006); Rodríguez et al. (2008); Osuna et al. (2010); Fajersztejn et al. (2011); Fernandez et al. (2010); Leon et al. (2010); Ibarra et al. (2006); Tapiador et al. (2007); Ceballos et al. (2010); Vazquez et al. (2010); Gabriel et al. (2008); Arviset et al. (2008); Ibarra et al. (2007); Planck Collaboration et al. (2011).

2.1 Scientific Archives in Astronomy and Astrophysics

Scientific archives in astronomy have evolved much over the last two decades. Looking back in the 1990s, and more concretely to the ESA Hipparcos mission (the predecesor of Gaia), and how its high-precision catalogue (Perryman et al., 1997) and the lower precision *Tycho* one (Høg et al., 2000) were released, we observe a rapid evolution from printed paper and Compact Disc-Read Only Memory (CD-ROM) (Perryman et al., 1997) to the round the clock online content, exploratory tools and interoperable services that we find in current astronomical archives.

This transition has been accelerated by both the increase in experiments being undertaken in the field, and mostly by the enormous amounts of data being collected from their instruments, which grow exponentially in all scientific disciplines, including astronomy (see Figure 2.1). The availability of more and more data, along with the newest statistical techniques, challenges the scientific method itself (Anderson, 2008) (which was based on hypothesize, model and test), as it may oversimplify the reality. Another approach would suggest analyzing the data without hypotheses about what it might show. This would then allow to find hidden patterns which the experiments were not planned for or that were simply unexpected in the first place.

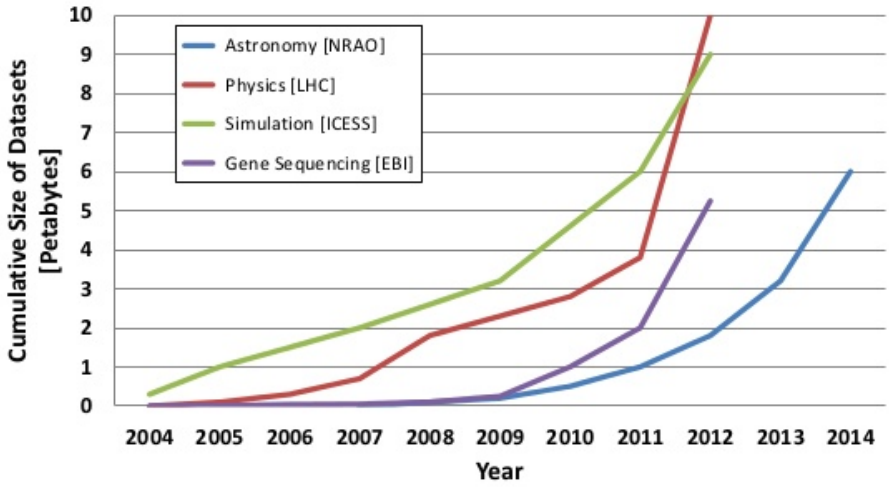


Figure 2.1: Illustration of the scientific data exponential growth. The volume for astronomical data does not include missions like Gaia or Euclid or telescopes like the LSST or the SKA, which would make the curve much steeper.

Moreover, institutional and cultural pressures lead scientists to avoid risk-taking and choose inefficient research strategies (Rzhetsky et al., 2015; Foster et al., 2013). Qualitative research suggests that this choice is shaped by a tension between the professional demand for productivity and a conflicting drive towards risky innovation. Research following a risky strategy is more likely to be ignored but also more likely to achieve high

impact and recognition. While the outcome of a risky strategy has a higher expected reward than the outcome of a conservative strategy, the additional reward is insufficient to compensate for the additional risk. The empirical demonstration suggests policy interventions that may foster more innovative research. However, there are other factors that will also contribute to embrace riskier strategies, like the availability of the proper tools that can enable innovative ways of processing the data, applying new algorithms or inferring new empirical models in a straightforward way, hiding the computational complexity and reducing the entry barriers, thus letting scientists be more productive.

The evolution taken by astronomical archives has somehow reflected these new possibilities, making both data exploration and extraction easier and more convenient across the different archives. Figures 2.2 and 2.3 show the wide variety of ESA missions for which a data archive is available or being developed. The goal of these archives is to maximize the scientific exploitation of the collected data. Multi-mission, multi-instrument and multi-wavelength, as well as interoperability and science support are some of the key ingredients of an astronomical data archive.

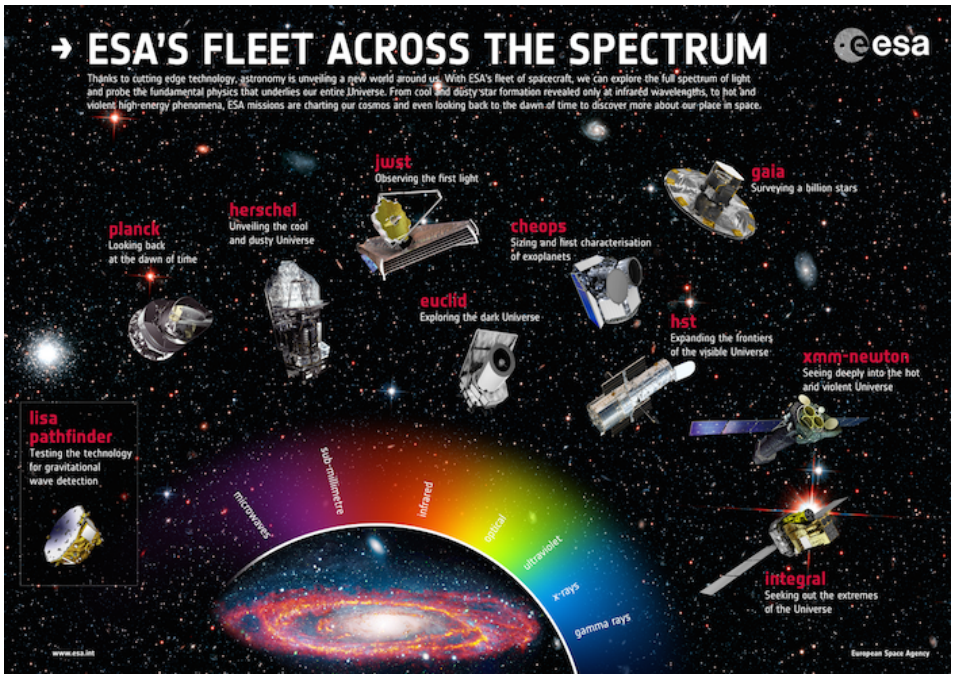


Figure 2.2: ESA's fleet across the spectrum.

One of the commonly accepted high-level functional architectures for implementing archival information systems is the OAIS reference (see Figure 2.4). It contains several functional components:

- *Ingest* refers to the set of processes responsible for accepting information submitted by the mission or telescope data processing pipeline, typically the SOC. It prepares

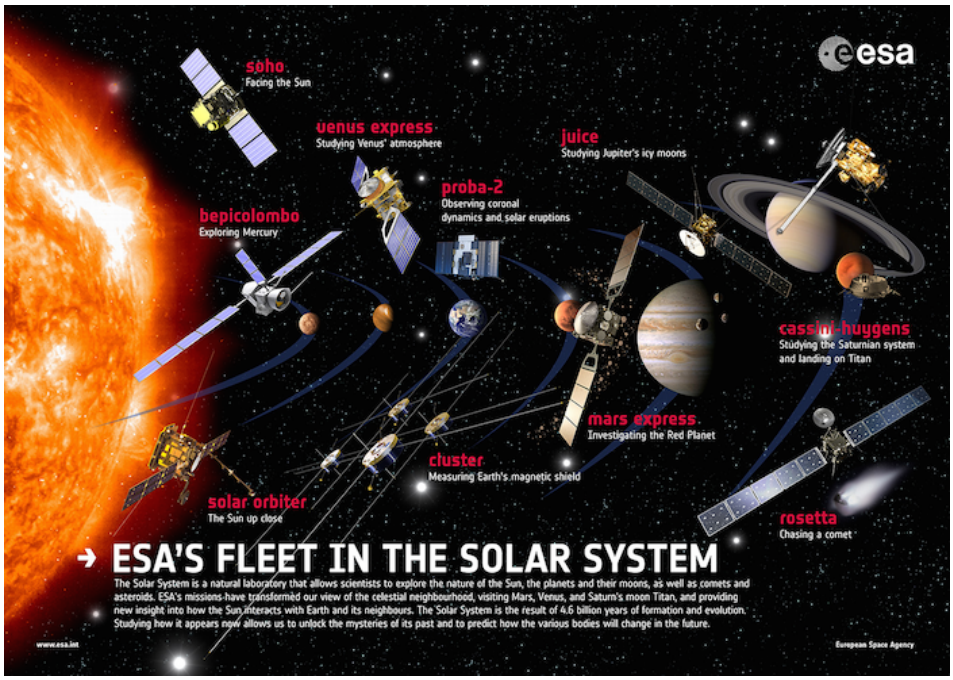


Figure 2.3: ESA's fleet in the Solar System.

the information for inclusion in the archival storage. It corresponds to the widely known Extract, Transform and Load (ETL) processes and also extracts relevant descriptive metadata to allow for exploration of the archive contents.

- *Archival Storage* is the part of the archive that manages the long-term storage and maintenance of the data. It corresponds to the database (or data lake) where the data products are stored. It also includes procedures and policies for preservation and disaster recovery.
- *Data Management* maintains databases of descriptive metadata identifying and describing the archived information. It supports the search and retrieval of the archived content, as well as the administration of the internal operations.
- *Preservation Planning* is a service responsible for mapping out the preservation strategy, as well as recommending appropriate revisions to this strategy in response to evolving conditions in the environment. It monitors the external environment for changes that could impact the ability to preserve and maintain access to the information, such as innovations in storage and access technologies, or shifts in the scope or expectations of the community. This component represents the safeguard against a constantly evolving user and technology environment.
- *Access* manages the processes and services by which information consumers locate, request, and receive delivery of items residing in the *Archival Storage*. Its primary

goal is thus to make the archived data available to the user community. In addition, security and access control is also taken care of by this component.

- The *Administration* function manages the day-to-day operations of the archive system, coordinating the activities of the other high-level OAIS services. Other responsibilities include interacting with both producers and consumers of information, e.g. to negotiate Interface Control Document (ICD) for data ingestion and to provide customer service support respectively.

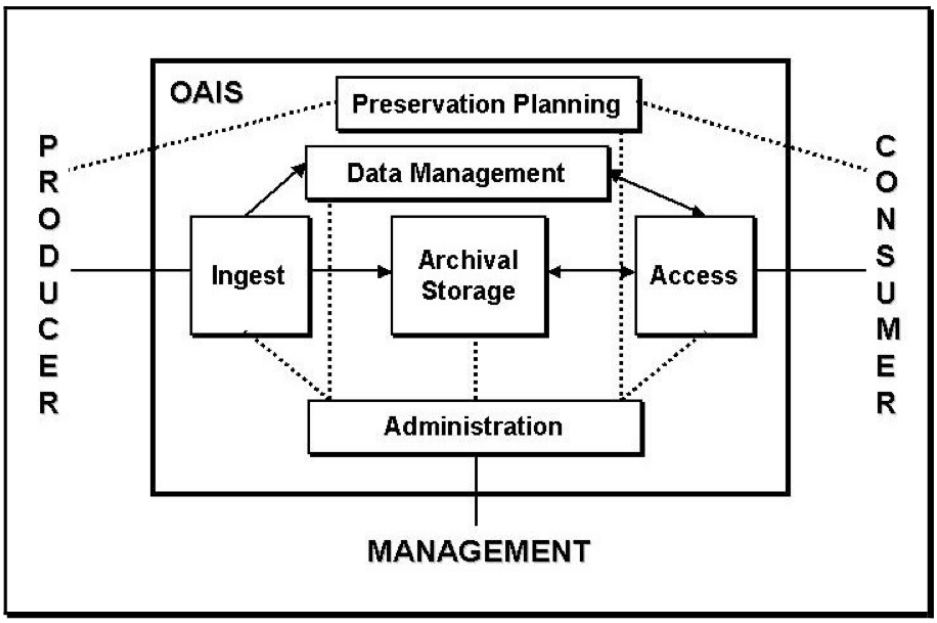


Figure 2.4: OAIS Functional model.

The OAIS architecture is a good reference for any scientific archive as it identifies the main components to be embedded in such a system. However, its outline somehow biases the potential implementations towards a specific client-server model, obviously because that was the prevalent approach when the reference model was issued. Furthermore, it does not explicitly tackle questions like the ones enumerated before, i.e. how to integrate and interoperate with other data archives, and how to effectively bring the scientists' work to the data centre where the data is located. This last challenge is indeed one of the reasons for specialized data centres to exist, as unifying forces to tackle the diversity found in the data sets. They offer a consolidated environment where research can be boosted, effectively becoming the place where it is conducted.

2.2 The Virtual Observatory

The VO is an ecosystem of mutually compatible data sets, resources, services, and software tools which use a common set of technologies and a common set of standards. It is the vision that they should work as a seamless whole. The IVOA[1] is the organization that debates and issues the technical standards that are needed to make the VO a reality. It originated as a way of making all components interoperable among them.

The VO architecture is depicted in Figure 2.5. It contains many of the concepts in the OAIS architecture previously discussed in Section 2.1, and defines the necessary *middle layer* framework which connects the resource layer to the user layer in a seamless and transparent manner. It enables the astronomical scientific community to access astronomical resources wherever and however they are stored by the astronomical data and services providers.

On one hand, the resource layer comprises any astronomical data collections like images, spectra, catalogues, time series, theoretical models, etc. with their associated descriptive metadata and access services. It also refers to storage and computing services for processing this data. On the other hand, the user layer represents the consumers of these data and computing services, be it individual researchers, teams of them, or computer systems. The interactions can be through browser based applications, standalone desktop applications, or scriptable applications that can run in automatic and/or batch modes.

The technical architecture of the VO was conceived from a Service-Oriented Architecture (SOA) pattern point of view. It was built on top of Internet standards, especially HyperText Transfer Protocol (HTTP) and Extensible Markup Language (XML), and either Simple Object Access Protocol (SOAP) with Web Service Definition Language (WSDL) or Representational State Transfer (REST) for description of web services. It contains functionality that, when taken as a whole, enables a more collaborative way of exposing and consuming astronomical resources by the research community. In particular, the definition of common formats and data models, as well as the usage of common semantics is required to have a uniform and common description of astronomical data sets. This way, they become interoperable and queryable through standard query languages to enable cross analysis amongst various data sets. There are several categories of functionality and standards in the VO (Hanisch, 2015):

- Registries as the *yellow pages* of the VO, collecting metadata about data resources and information services into a queryable database. Like the VO resources and services themselves, the registry is also distributed, with replicas around the network.

- Access to data and metadata collection is done through data access protocols, which specify a uniform way of getting data and metadata from various different providers. The typical example for flexible data access include Table Access Protocol (TAP) (Dowler et al., 2010) with the Astronomical Data Query Language

[1] http://ivoa.net/

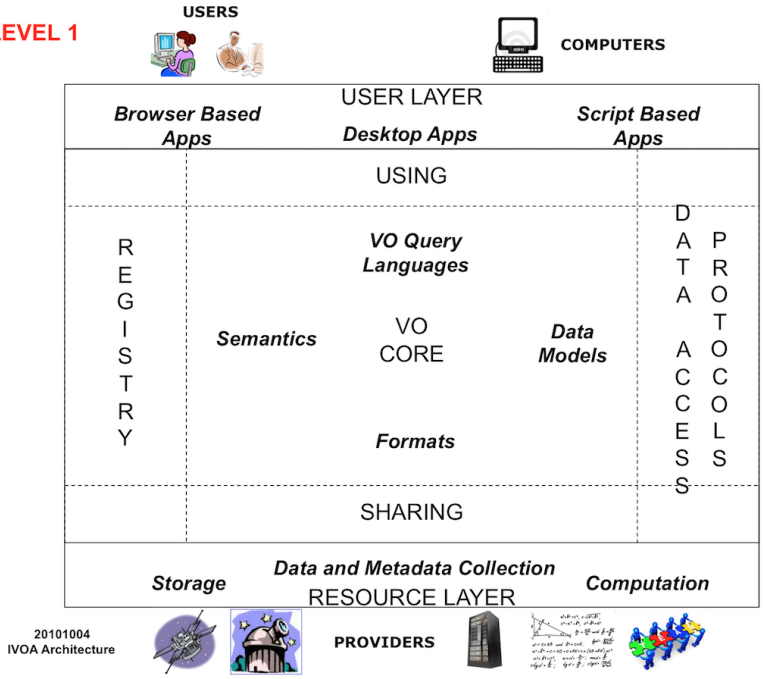


Figure 2.5: VO architecture.

(ADQL) (Osuna et al., 2008), which is a standard based on SQL, but contextualized for the astronomical domain (including ways to specify searching within a circular region in the sky and so on). Results are returned in a standardized data format for tables, i.e. VOTable (Ochsenbein and Williams, 2009), which contains the data itself as well as some metadata, for example the units in which the data is retrieved. A VOTable can also be hierarchical, allowing for more complex data structures.

- Data models to properly standarize the semantics of different astronomical data types. Examples might be *Phot DM* for photometry data, or *Obs core DM* for databases describing collections of observational data.

- Sharing layer for single sign-on, or for enabling an ecosystem of applications that can share data among each other through Simple Application Messaging Protocol (SAMP) (Taylor et al., 2015). This way, instead of having heavier applications that implement a lot of functionality, lighter (and more specialized) ones can be built and interoperate among themselves.

Nowadays, the VO is striving for adapting the defined standards (or defining new ones) to be able to cope with the new reality of massive and complex data sets coming from the new missions and telescopes like Gaia, Euclid or the LSST. Furthermore,

the challenges being faced by data centres, which create and host the majority of the astronomical data sets, constitute an ongoing line of work. This research will help to contribute to both efforts.

2.3 Evolution of Implementation Approaches

Technology and networks have driven the way scientific archives have been implemented. The approaches taken have evolved much since the ability to digitalize content, and more dramatically since the appearance of the Internet. The solutions taken resemble the state-of-the-art in ICT. They range from pre-Internet simple printed archives (i.e. book collections) and digital content distributed in CD-ROM, to interconnected client-server architectures or latest trends in distributed computing including Grid and Cloud computing and Big Data technologies.

2.3.1 Query Metadata and Download Data

This approach has been prevalent since the automation of scientific archives. It follows a client-server approach in which the metadata is queried from graphical user interfaces (or through interoperable ones), and the selected data products are later on downloaded to the client environment, be it a scientist's workstation or a remote node of a data centre where the data will be procesed further.

There have been many ways in which solutions falling in this category have been implemented. Looking at the transformation of the scientific archives hosted at the ESDC (Osuna et al., 2010; Fajersztejn et al., 2011), we observe an evolution from:

- Client-server communications with a home made Remote Procedure Call (RPC) through high ports (normally blocked by firewalls), to higher level and standard protocols like HTTP that support tunneling, encryption, timeouts handling, request compression, etc.
- Complex and ad-hoc persistence layer difficult to maintain and expand, to exisiting open source frameworks (Hibernate) and databases (PostgreSQL).
- High development and maintenance costs without much separation of concerns, to three-tier and reusable architectures that boost development of new archives by reusing an entire framework and many generic components.

Figure 2.6 shows the three-tier architecture of such archives. The client layer (Fernandez et al., 2010) encapsulates all the interfaces for accessing the system from the external world. Two different entry types are normally offered. The first one is a Graphical User Interface (GUI), intended for human access. It contains search panels that allow the user to select and filter items to be downloaded. This layer can be implemented both as a desktop application, or preferrably as a thinner web layer. The second one is the Archive Inter-Operability (AIO). This interface allows for a lower-level machine interface,

facilitating those automatic and normally batch use cases to interact with the archive through scripting or similar ways. The AIO is also the layer that allows for VO access to the archives.

The server layer (Leon et al., 2010) contains all the back-end machinery internal to the system. The heart of this layer is the request manager subsystem that deals with the complexities of the filtering requests coming from the client layer. The data layer (Leon et al., 2010) represents the data storage and data access subsystems. It is worth remarking that the storage part can be exchanged without affecting any other subsystem.

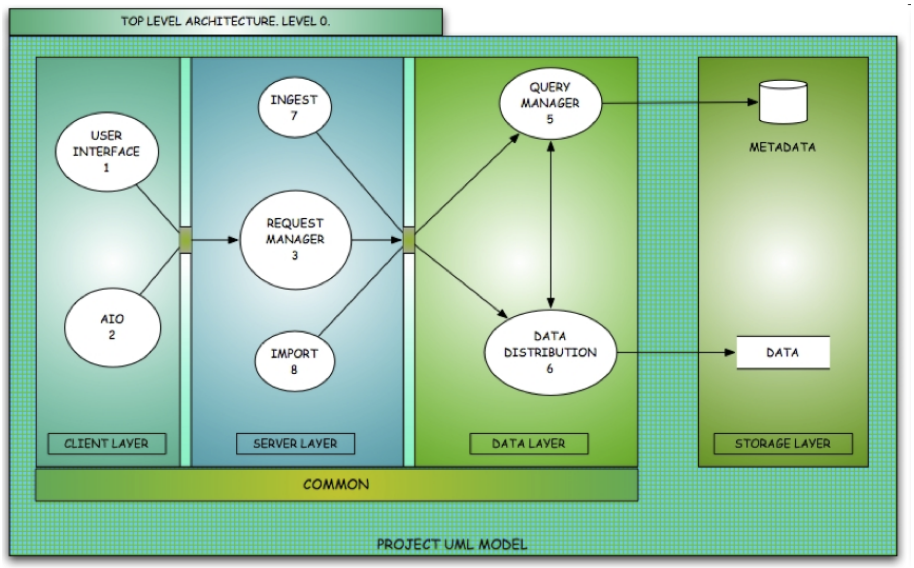


Figure 2.6: ESDC scientific archives architecture.

Drilling down through the data layer, we identify two main components:

- The query manager. It handles all requests related to the metadata. It basically maps the requests coming from the client layer to the corresponding Data Access Object (DAO) that is later on mapped to the database tables containing the relevant data through configurable XML files. When several entities are queried altogether, the query builder performs the relevant joins following the shortest path from the tables containing the requested information. Geometrical queries like cone searches are mapped to the appropriate UDF in the database (typically PgSphere PostgreSQL module). Once the query is executed against the metadata, the results are returned as objects from the project entity model (a kind of entity-relationship diagram).

- The data distribution module processes all requests related to repository files. The data can be requested in different granularity levels. Once the module finds the data requested, a HTTP Uniform Resource Locator (URL) will be returned to the user

for download. This URL is internally redirected to a File Transfer Protocol (FTP) server. This module commonly offers authenticated synchronous and asynchronous data retrieval. The tarballs are internally made of links to the requested data files. This avoids the duplication of the data products.

This common architecture shared among different archives allow to build them in an easier and faster ways, reuse knowledge and tools to maximize resources, generate a reusable set of generic libraries and components and reduce development and maintenance costs.

2.3.2 Bring the Software to the Data

Many scientific archives are then based upon the idea of data being retrieved from the data centre and then analysed somewhere else (often locally on the scientist's desktop). The exponential increase in data volumes for scientific archives challenges this approach, as the data transferred throughout the network becomes a bottleneck (not to account for the latencies incurred), and its further processing cannot be performed locally, given the limitations of computing power of the end user laptops and desktops.

Therefore, it is better to keep the data in the data centre or in the Cloud and process and analyse it there. This does not only reduce latencies as it is not required that the data moves around, but also permits more scalability in the resources that can be utilized in the exploration and analysis, enabling an eventual specialization of the software and hardware components leveraged for it, as well as a sharing of those resources and the potential outcomes generated with them.

There have been numerous attempts to implement new approaches in which the software (low volume and easy to deploy) is somehow taken to the data (high volumes and very costly to move around). Early experiments (Ibarra et al., 2006; Ibarra et al., 2006) followed a distributed Single-Program Multiple-Data (SPMD) approach on a Grid infrastructure (with no inter-process communication or synchronization), where several data centres with high bandwidth interconnection networks performed the XMM-Newton Science Archive (XSA) EPIC-pn instrument data reduction on-the-fly.

Figures 2.7 and 2.8 show the architecture and workflow implemented in this experiment. The steps taken by the processing were as follows:

- The scientist connects to the gateway from which the Grid jobs would be started. Several jobs were launched and each of them would process a different part of the relevant XSA observation on a different node allocated by the scheduler.
- Each process, spawned on a different resource, would then download the appropriate data from the archive, using the previously mentioned AIO interface.
- The actual processing of the data takes place (each job with a different part of the data but running the same routine).
- The results are taken back to the gateway that was used to submit the jobs. These copies are made through the Internet.

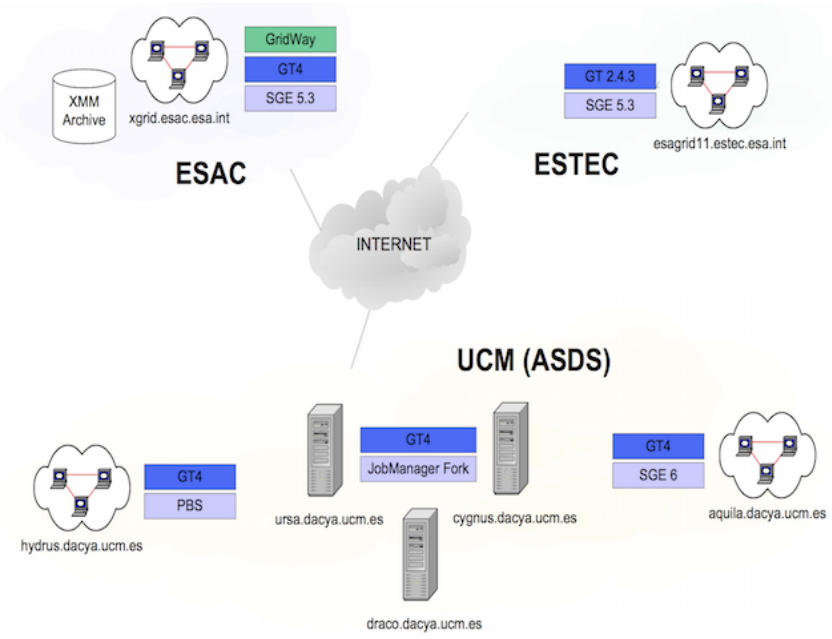


Figure 2.7: Experiment architecture for XMM-Newton EPIC-pn instrument on-the-fly data reduction.

This was one of the first attempts to leverage the existing technologies and implement an architecture that could somehow allow scientists to bring their software and scripts to the data centre, and process as much data as required (in a SPMD way) without having to open remote sessions to those machines.

Even though this architecture was a step forward, it fell short in several areas:

- The data is effectively being moved around, and even when the data is processed in the same data centre, there is no data locality (we find this feature in one way or another in most of the recent distributed processing engines found nowadays[2]).
- Results are transferred back to the system that submitted the jobs in the first place. There is thus a clear limitation as those (common) workloads generating a big amount of data could not be implemented in this way.

[2]Traditional HPC architectures separate compute nodes and storage nodes, which are interconnected with high speed links to satisfy data access requirement in multi-user environments. However, the capacity of those high speed links is still much less than the aggregate bandwidth of all compute nodes. In parallel systems such as Google File System (GFS) (Ghemawat et al., 2003) or MapReduce (Dean and Ghemawat, 2008), clusters are built with commodity hardware and each node takes the roles of both computation and storage, which makes it possible to bring compute to data. Data locality is a significant advantage of parallel systems over traditional HPC systems. Good data locality reduces cross-switch network traffic, one of the bottlenecks in data-intensive computing (Guo et al., 2012).

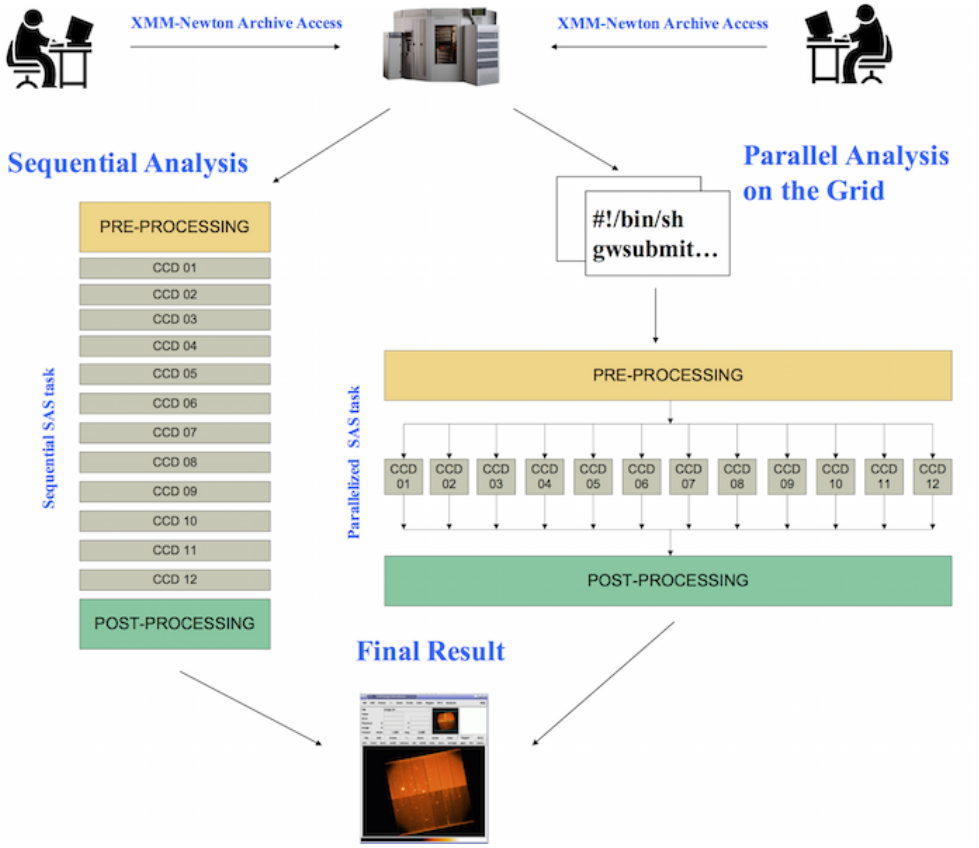


Figure 2.8: Experiment workflow for XMM-Newton EPIC-pn instrument on-the-fly data reduction.

- In scientific data reduction workflows, there is normally a set of auxiliary files used for calibration. These files are needed for the processing and then they need to be distributed to all data centres (and again once they are updated). This generates some maintenance costs. The same applies to the scientific routines to be applied to the data. For a mission like XMM-Newton, its size can reach a few hundred Megabyte (MB). Furthermore, all these files will be placed in different areas depending on the data centre, increasing again maintenance costs for the solution.

The Sloan Digital Sky Survey (SDSS) science database, with a size of 1.6 Terabyte (TB), goes a bit further with the Catalog Archive Server (CAS) (Li and Thakar, 2008; O'Mullane et al., 2005). It provides several levels of query interface to the SDSS data via the SkyServer[3] website. It allows users to execute queries to the archive in two different queues. These queries might be some sort of *software* in the *bring the software to the data* approach. One of the queues is high priority for short queries (seconds or minutes), which are the most common ones and for which the system is preconfigured to execute in a fast way with the help of indexes and similar tools. The other queue is for much longer queries, lasting typically hours or even some days.

Furthermore, the transfer of very large result sets from queries over the network is something to avoid as remarked above (statistics suggest that much of this data transfer is unnecessary). Therefore, the archive allows users to store results locally in order to allow further processing (in the form of joins, filtering, etc), also enabling external data upload for usage in the analyses and/or sharing among collaborators (without any download).

This solution, when compared to the one referred above, reduces the amount of data transferred through the network, and provides a convenient way to work on the data centre. However, it does not expose a generic interface where any generic processing can be done, i.e. it is limited to the SQL language and its extensions (which vary depending on the specific instance chosen). Moreover, the scalability of the approach did not scale as much as the one leveraging the Grid infrastructure for obvious reasons.

2.3.3 Cloud Computing as an Enabler

Cloud computing is a game changer for data centres. It lowers costs by consolidating servers that are not running at full capacity for instance, but also speeds up project deployment, permitting the infrastructure to scale up or down depending on the computation needs. This last idea was successfully leveraged in a prototype (Rodríguez et al., 2008) in which resources were dynamically adapted to the workload.

The prototype was based on the fact that virtual machines can greatly simplify Grid computing by providing an isolated, well-known environment that increases security. This is of paramount importance to data centres that execute several scientific workloads and/or expose their resulting data sets, given the heterogeneity in the software configuration (operating system, libraries and applications that they rely on), which sometimes

[3] http://skyserver.sdss.org/

conflicts with the libraries or versions that are needed for another project. Moreover, most of the Grid infrastructures do not allow administrators to isolate and partition the performance of the resources they devote to the Grid. In this way, the applications of a Grid user can affect the execution of other Grid or local users (Tapiador et al., 2007).

Therefore, virtual machines could be used as the base technology to dynamically modify the computing elements of a Grid, thus providing an adaptive environment. Figure 2.9 shows the Grid architecture that allows to dynamically adapt the underlying hardware infrastructure to changing demands. The backend of the system is able to provide on-demand virtual worker nodes to existing clusters and integrate them in any Globus-based Grid. This allows to deploy self-adaptive Grids, which can support different projects and needs in shared physical infrastructures and dynamically adapt its software configuration. The experimental results on a testbed show less than a 10% overall performance loss including the hypervisor overhead.

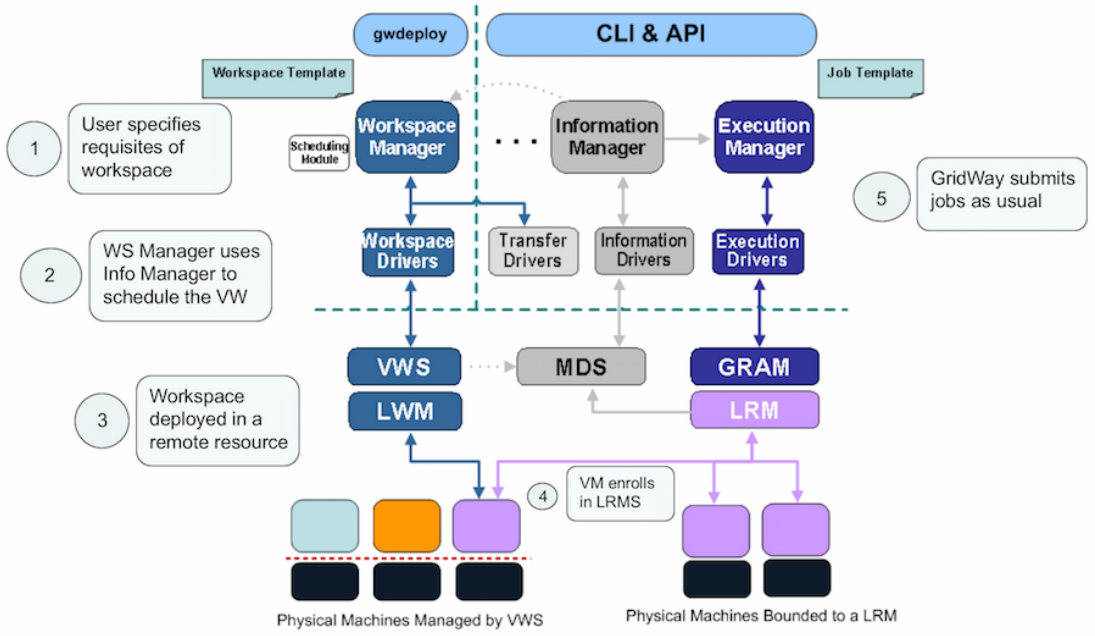


Figure 2.9: Process flow and interactions among GridWay components, Grid middleware and Grid resources for dynamically adapting the infrastructure to the workload.

The process flow is the following (see Figure 2.9):

1. The user specifies the requisites of a workspace to be deployed because tasks being run on that type of environment are queueing up in the scheduling system (i.e. Gridway[4]). This should ideally be done by some intelligent agent that moni-

[4]http://gridway.org/

tored the workload and, given some heuristics, deployed (or disposed) the relevant workspace type.

2. The workspace is deployed remotely in the appropriate data centre by using the Virtual Workspace Service (VWS) (Keahey et al., 2005a,b).

3. The virtual workspace (virtual machine), which is preconfigured to enrol to a particular LRMS (in the case of this prototype SGE).

4. The Grid information system, i.e. the Monitoring and Discovery System (MDS), detects that there are more available resources for a particular job type. Gridway meta-scheduler submits more jobs to those newly deployed workspaces as usual.

5. Once the workload decreases, the workspace is disposed and the physical resources are released.

With this approach, the flexibility of the Grid infrastructure can be increased without requiring dedicated hardware, or modifying neither existing applications nor software configurations. The final goals are thus:

- Dynamically adapt a shared infrastructure to support different virtual organizations (devoted to different projects or missions), by balancing the physical resources allocated to each of them.

- Reduce the operational cost of the Grid infrastructure, by providing a simple way to provide on-demand software configurations to the users.

- Minimize the *Gridification* time, by executing specific virtual organization applications in a well known pre-defined environment.

This prototype showed a way in which Cloud computing could be effectively used in current Grid setups. However, its potential goes way beyond that. It can support the same ideas in conjuction with other (more advanced and recent) Big Data technologies like those presented in Chapter 3, e.g. scale up or down in premise or externally through hybrid clouds, share resources among different missions to cover for peaks in demand when reprocessing archives with new calibration techniques or for correcting biases not previously accounted for, etc. If this type of (re-)analyses is not enabled in astronomical archives and databases, much of the value of these expensive missions or experiments will be lost.

2.4 Collaborative Archives

The use case depicted in Section 2.3.3 is just an example of what Cloud computing can offer to scientific archives. There are other areas in which it can contribute to a new way of implementing *collaborative* archives, where the scientific data and tools exposed to the community represent a platform where scientists can seamlessly work on more ambitious

challenges that can increase the return on investment and produce synergies. The key elements for such archives include (O'Mullane, 2011):

- The possibility to allow ligther clients (only the visualization layer) through the usage of richer environments that run on the data centre (e.g. virtualization or by other means). This is already ongoing in the Canadian Advanced Network for Astronomical Research (CANFAR) (Gaudet et al., 2010) and in the SKA[5].
- Outreach oriented where the basic interface can be used by the general public but which can be configured with more advanced features for professionals.
- Living archive allowing derived data products, often by using other external missions' data sets and being normally exposed by the scientific community, to be externally published. This might comprise an archive as a model, where the catalogue can be described as a model of the data, also allowing to compare among different models once richer data sets are available.
- Scientific PaaS, Astro applications store and social networking. These ideas encourage archives to become a platform where scientists from different institutes can apply (and pay) on-demand for some allocated resources to perform certain tasks of the data and/or develop applications that can be released to the community. These applications might then be shared in a similar way as the *App Store* or *Google Play*.

Furthermore, there are other areas that will certainly contribute to this new concept for science and research in which collaboration plays an important role. One of such areas is reproducible research. Science is reportedly in the middle of a reproducibility crisis (Peng, 2011; Iqbal et al., 2016). Reproducibility seems laudable and is frequently called for. In general, the argument is that research that can be independently reproduced is more reliable than research that cannot be independently reproduced. The availability of a collaborative environment or platform where scientists of different institutions can get access to, largely improves reproducibility by letting peers internally go through the process and use the same environment, tools and data sets for addressing a scientific hypothesis and build up evidence for or against it.

Interoperability, interactivity and scalability are other key ingredients for truly collaborative archives. Interoperability enables the possibility to merge data sets coming from different missions and surveys (being the VO the main contributor), but it also refers to the tools being used for processing the data, they would ideally need to be exchangeable (i.e. integrated ecosystem), letting scientists use the right tool for the right purpose. Interactivity enables exploration and discovery of data. Moreover, it facilitates the execution and fine tuning of recipes created by other users (both for reproducibility and for evolution of a specific use case). Last but not least, scalability is required both for coping with the bigger and bigger data sets being collected, and mainly for moving away

[5]http://www.cyberska.org/

from the *query the metadata and then work with the selected data* approach, to *query all data* mindset, despite the useful descriptions and aggregations that may summarize the available data sets.

2.4.1 Consolidating Operations, Archiving and Data Exploitation

The concept of consolidating in the same infrastructure the different data-related parts of an astronomical mission is not something new (Arviset et al., 2011). These areas involve the SOC, which is where the processing of the data coming from the Mission Operations Centre (MOC) occurs, the resulting scientific products archiving, and its scientific exploitation. Figure 2.10 represents the workflow of the ESA Herschel mission data processing pipeline. It clearly shows the interrelationship of the three elements aforementioned.

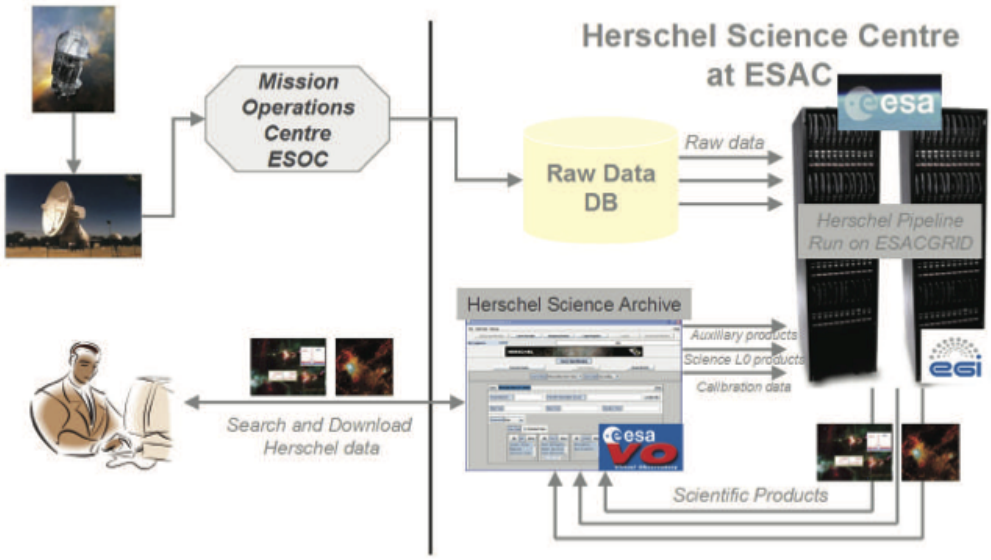
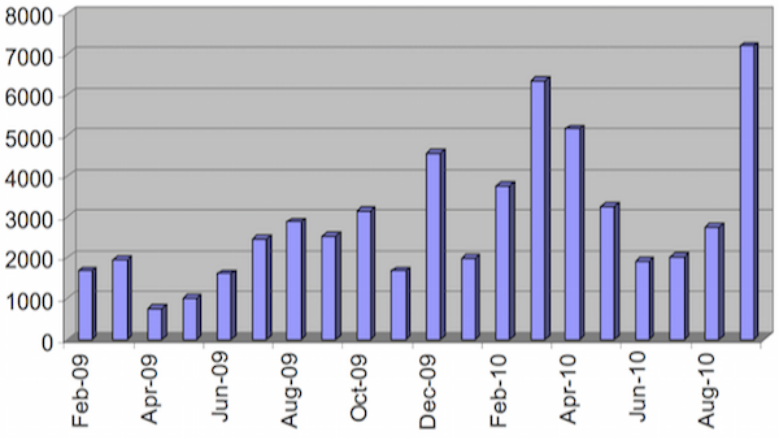
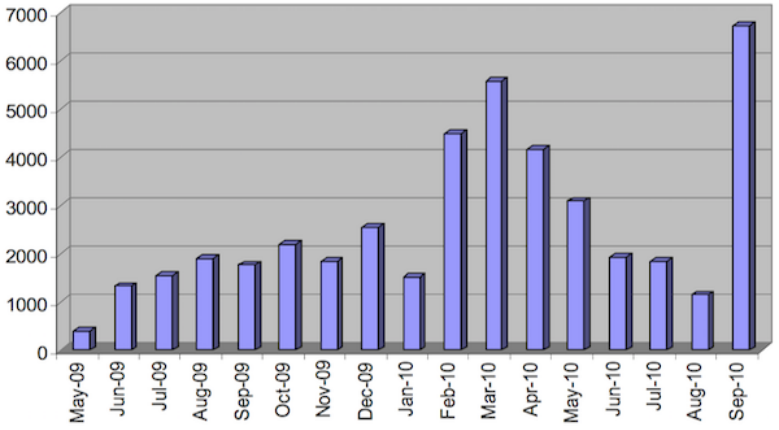


Figure 2.10: Herschel data processing architecture and workflow.

The idea behind this consolidation is twofold. On one hand we reduce operational costs as the same infrastructure can be utilized along the mission pipeline, and on the other hand all software, algorithms and techniques that are developed for the data processing (in the SOC) can later on be made available to the community as open source under the GNU General Public License (GPL), along with the infrastructure and architecture on which the software run. This would allow for a straightforward reuse, bug fixes, and so on, and would facilitate the integration of two areas that typically sit under different domains, encouraging a leaner approach. Figure 2.11 correlates the activity of the SOC with the ingestion activity in the archive. The development and operational costs of both infrastructures might be reduced by just integrating further.



(a) Herscheldata processing jobs.



(b) Observations ingested into the Herschel archive.

Figure 2.11: Correlation of the Herschel SOC data processing and the archive ETL pipeline.

Chapter 3

Enabling Large Scale Data Science and Data Products

If we have data, let's look at data.
If all we have are opinions, let's go with mine.
Jim Barksdale, former Netscape CEO

Distributed computing has always been closely related to research and innovation, typically playing a key role in supporting scientific breakthroughs like the one produced by the LHC (Geddes, 2012; Britton and Lloyd, 2014). More specifically, it has ceaselessly struggled over time to come up with new architectures that can process larger data sets, and so be leveraged in more demanding scenarios. With the arrival of the Internet, large scale analytical data has become widespread in web companies and across industries. Turning these data into value has posed many issues in areas like data exploration, modeling, agile prototyping, scalability, cost efficiency, visualization, etc.

To deal with these challenges, monolithic architectures (based on SMP) are not suitable any more because of their inherent limitations to scale efficiently to the volumes of data present in scientific research. Therefore, even though traditional Relational Database Management System (RDBMS) are very stable, with around 30 years of existence, and with a lot of commercial and open source implementations (like Oracle, MySQL or PostgreSQL), they pose some concerns when dealing with the data sets coming from the most ambitious experiments and missions:

- Scientific data sets are multidimensional. Examples in the astronomical realm might include a star being observed in several scans (transits) and the data collected in each scan comprises spectral information, i.e. values for many different wavelengths. We would then have dozens of transits per star, and hundreds of spectrum values per transit. Storing this in a traditional RDBMS is quite challenging given the structures that are available. It is then mandatory to normalize.
- Normalization kills performance. Storage was a scarce resource when RDBMS were first devised. They tackled this issue with a mechanism named normalization, which

simply splits the data into different entities to prevent duplicated information. As the database grows in volume, the computational cost of *joins* among tables becomes higher and higher. It is important to remark that the *join* primitive is probably the most computationally expensive operation in a database, given the nature of the actions that need to take place (shuffling data, ordering, matching, etc).

- Data must conform to a well known schema. This schema needs to be defined before any ingestion can take place and must be optimized for the workflow that will have to support. This is sometimes too rigid as new ways of analyzing the data will certainly need different ways to access it.
- They follow the ACID rules (Atomicity, Consistency, Isolation and Durability)[1]. Some of these rules are not required in databases that will be populated once and read many times. Then, the mechanisms that ensure these unneeded rules reduce the performance and scalability of the system.
- They are well suited to only host structured data. They lack ways to store unstructured information like text or images, which is more and more common these days.

New scalable approaches are thus needed to efficiently manage the increasing volumes, variety and velocity of current data sets. The content of this chapter sets out the key technological foundations over which the research presented in Chapters 4 and 5 has been performed. The original contribution is the implementation of an ad-hoc column-oriented data store on top of HDFS that was leveraged in the framework laid out in Section 4.3 to prove the value of this technique for scientific data sets. Furthermore, there is an extensive study of the different alternatives that can be used for scientific data archiving, processing and mining, showing why the selected engine is the most adequate in these scenarios. The publications backing these contributions are Tapiador et al. (2014, 2017).

3.1 Massively Parallel Processing Databases

To address some of these limitations, new *shared-nothing* architectures (Stonebraker, 1986; Valduriez, 2009) started to catch on. In this approach, all the separate nodes of

[1] ACID rules:

- *Atomicity* requires that if one part of the transaction fails, then the entire transaction fails, and the database state is left unchanged.
- *Consistency* ensures that any transaction will bring the database from one valid state to another. All nodes and queries will see the same new data once the transaction has finished.
- The *Isolation* property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.
- *Durability* ensures that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors.

the infrastructure participate in the coordinated computations. Each query is then split into a set of coordinated processes executed by the nodes of the MPP in parallel, running faster than in traditional SMP RDBMS. The main advantage is scalability. We can easily add more nodes to the infrastructure and both the available space and the performance of the queries will increase. This architecture works well with commodity hardware which typically mounts Direct-Attached Storage (DAS) storage. The data sets are usually split (sharded) between the nodes in a way that each node only processes its local data. Data locality (Guo et al., 2012) occurs when each node takes the roles of both computation and storage. It reduces network traffic, which is normally a bottleneck in data-intensive workloads.

Parallel databases (MPP) are often confused with distributed databases. There is a clear distinction between the both. A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. Its architecture allows to manage the different distributed instances, which may be placed at geographically different locations. It also makes the distribution of the data transparent to the end users and is commonly used for On Line Transaction Processing (OLTP) scenarios (high transaction rates). A truly parallel database or MPP is a system where multiple nodes are used to execute and run queries in parallel (as stated above). The data is partitioned across multiple disks for parallel I/O (horizontal partitioning). Furthermore, the individual relational operators (sort, join, etc) are executed in parallel. Last, each node usually works on its own data partition.

There are two levels of parallelism attained in MPP, i.e. interquery and intraquery, which increase throughput and performance respectively. MPP Database Management System (DBMS) are normally referred to as On Line Analytical Processing (OLAP). OLAP is characterized by relatively low volume of transactions. Queries are often very complex and involve aggregations. The response time is a measure of the effectiveness. OLAP applications are widely used by data mining techniques. On the contrary, OLTP systems are identified as they process a large number of short on line transactions (e.g. *INSERT*, *UPDATE*, *DELETE*). The main emphasis is put on very fast query processing, maintaining data integrity in multi-access environments and an effectiveness measured by number of transactions per second.

Some examples of MPP databases include Teradata, Vertica, Netezza, Greenplum and Apache Impala. All of them have complex and mature SQL optimizers developed specifically for improving the operations to be performed over the data. These databases are normally closed-sourced enterprise solutions (pay per TB or number of CPU cores), but this is changing in the last years as there are more and more open source solutions, not only in the DBMS realm, that are being made available to the community (see Section 3.2). Some concrete examples of this trend range from Apache Impala (Kornacker et al., 2015), which is the first open source MPP SQL engine for Hadoop, or Greenplum[2], which is the first enterprise MPP solution that has transformed its license into an open source one. The tests carried out in Section 4.2 are performed on top of the later.

[2]http://greenplum.org/

3.2 The Big Data Landscape

3.2.1 MapReduce and the Lambda Architecture

There are still some limitations and drawbacks present in MPP databases. Some of them have already been remarked above, like the need to define a schema before any data can be stored, or the lack of data structures that can naturally host multidimensional fields without normalization and joins (which kill performance). The main concern though, may be related to the scalability of the architecture, given the trade-off in the design in which data consistency is preferred over system availability (in the presence of failures).

The CAP theorem (Gilbert and Lynch, 2002) states that it is impossible for a distributed system to simultaneously provide all three guarantees of:

1. *Consistency.* All nodes see the same data at the same time.

2. *Availability.* Every request receives a response about whether it succeeded or failed.

3. *Partition tolerance.* The system continues to operate despite arbitrary partitioning due to network failures.

MPP and other RDBMS normally focus on getting consistency when there is a network partitioning, whereas new Not Only SQL (NoSQL) approaches sacrifice consistency for achieving high availability, implementing the so called *eventually consistent* model. This model guarantees that, if no new updates are made to a given data item, eventually all accesses to that item will return the last updated value. The approach is nowadays fair for many modern scenarios where either the data is not updated frequently, or the situation of having two processes read different values of the same data item would happen in extremely rare conditions. Even in those rare situations, the consequences would obviously have to be acceptable for the particular use case. This approach makes the distributed system scale to even bigger data sizes, which might be out of the scope of the data sets being produced by some of the scientific missions currently in operations (like Gaia), but which are certainly the most adequate solution for others to come (like for instance the SKA), not to account for other business oriented areas also producing massive data sets like in the Internet of Things (IoT) or web applications.

This and other challenges have lately been extensively explored, and new computing paradigms, such as MapReduce (Dean and Ghemawat, 2008), have appeared. MapReduce proposes a new programming model (originating from functional programming) that emphasizes the scalability and availability of the distributed system over the organization and structure of the information (preferred way in traditional relational databases). The key idea behind this paradigm is to isolate developers from the burden of coping with the inherent complexity of a distributed system, letting them focus on the business domain. The resulting implementations done on top will naturally scale up (horizontally) to bigger input data sets by just adding more hardware, which is unquestionably less expensive than any new software development or adjustment.

An interesting feature of this new type of data management system (MapReduce) is that it does not impose a declarative language, like SQL, but it allows users to plug in their algorithms no matter the programming language they are written in and let them run and visit every single record of the data set (always brute force in MapReduce, although this may be worked around if needed by grouping the input data in different paths using certain constraints). This may also be accomplished to some extent in traditional SQL databases through UDF, although code porting is always an issue as it depends a lot on the peculiarities of the database and debugging is not straightforward (Pavlo et al., 2009). However, scientists and many application developers are more experienced at, or may feel more comfortable with, embedding their algorithms in a piece of software (i.e. a framework) that sits on top of the distributed system, while not caring much about what is going on behind the scenes or about the details of the underlying system. This is actually the case of the framework proposed in Section 4.3.

If we compare MapReduce model with parallel DBMS, there are some differences and similarities worth remarking:

- The data loading into the system (ingestion) is much faster and simpler in the MapReduce model. This is because data can be directly dumped into the distributed file system, typically HDFS (Shvachko et al., 2010) (see Section 3.2.2). This comes at a cost, it precludes various I/O optimizations which can increase CPU cost later on for runtime parsing of the data.

- MapReduce and other models based on the data lake concept (see Section 3.2.2) are schema free, i.e. they do not require any data schema to be defined prior to the ingestion. Data can then be written in any arbitrary format, which makes much easier on the developer to rapidly prototype and deploy a solution. This is important in today's fast changing world, where agile and lean product development is crucial for the success of any initiative.

- Scalability with regard to the amount of data is higher in the MapReduce model because of the trade-off presented above and due to the less syncrhonization required among the distributed nodes.

- Native and integrated support for UDF in the MapReduce model. This is not so easy in parallel glsDBMS as support is limited and they have to be implemented in complex (and sometimes specific) languages when available. MapReduce poses an easy API, where developers can implement anything they want (in several different languages).

- The system tuning is more difficult to achieve in parallel DBMS, typically requiring the expertise of a Database Administrator (DBA).

- Parallel DBMS give a better performance up to a determined amount of data (with a non negligible configuration and data setup cost). This is due to the extensive usage of indexes (e.g. B-tree) which speed up the execution of *SELECT* operations,

or thanks to sophisticated parallel algorithms for querying, joining and aggregating large amounts of relational data.

- Tasks start-up takes a lot of time in MapReduce. It is clearly not conceived as a low latency system but as a batch processing one. It follows a brute force approach as previously remarked where the whole data set must be read. It is then suitable for answering those questions that need to go through the whole data set like in the use case presented in Section 4.3. The remaining question refers to the volume of data from which the MapReduce model starts to outperform a parallel DBMS.

- Complex operations like joins or algorithms in graph processing are implemented in similar ways (behind the scenes) in both approaches, but it is more complex to do so in the MapReduce model, as it requires both deep knowledge into the MapReduce internals and significant boilerplate code as we can see in Blanas et al. (2010); Lin and Schatz (2010).

MapReduce seems to be suited then for those workloads that require a significant amount of time to finish, or those which involve a large amount of data, meaning way higher than what other parallel DBMS can cope with given their inherent limitations in scalability due to synchronization and other mechanisms. Because of this specialization of the MapReduce model on batch processing, it is not suitable for other approaches that need to work in a (near-) real time fashion like many Internet services or even scientific pipelines operating on telemetry with time constraints. Therefore, a variety of engines have been developed to cover for these shortcomings or complement and enhance the solution. Examples may range from distributed processing counterparts for streaming (e.g. Apache Storm[3]), to ETL and publish/subscribe tools like Apache Nifi[4] or Apache Kafka[5] respectively. The picture is completed by adding other tools that focus on providing higher level primitives to MapReduce (Sakr et al., 2013). Apache Hive[6] is the SQL interface to MapReduce, and Cascading[7] represents the higher level counterpart for data applications. Both leverage MapReduce behind the scenes and provide more user friendly interfaces and API.

The Lambda architecture[8] (see Figure 3.1) suggests an approach to overcome the trade-off posed by the CAP theorem. On one hand, a design guaranteeing consistency (commonly a DBMS) has a lower availability (and scalability). On the other hand, a system increasing its availability (and thus its scalability), does so by trading off with consistency. In real life scenarios, there is the need to:

1. Process and query on historic data sets that need to scale to big volumes (Petabyte (PB) of data). This might be the historical records of an Internet application, but also the scientific raw data that is made available to the community.

[3]http://storm.apache.org/
[4]https://nifi.apache.org/
[5]http://kafka.apache.org/
[6]https://hive.apache.org/
[7]http://www.cascading.org/
[8]http://nathanmarz.com/blog/how-to-beat-the-cap-theorem.html

2. There is also the need to process and query (near-) real time streams of data. These again may encompass the updates being produced by users interacting with any Internet application and the telemetry being download and process in (near-) real time.

Therefore, a different distributed system is leveraged for each use case, one of them focusing on historic data sets (older than a few hours or days) and using Hadoop tools like MapReduce and Cascading, and the other one processing and exposing the (near-) real time live streams using tools like Apache Storm. This way, we use the best of breed engine for each use case, i.e. use the right tool for the right purpose, as the idea of "one size fits all" seems to be gone (Stonebraker and Cetintemel, 2005). Anyway, one data flow or another, the data is normally consolidated at the end in the Data Lake.

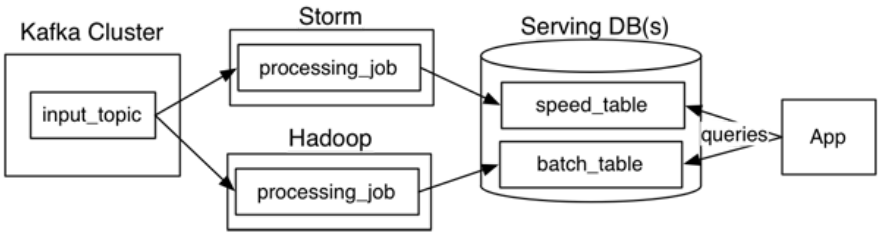


Figure 3.1: The Lambda architecture.

3.2.2 The Data Lake

A data lake is a storage repository that holds a vast amount of raw data in its native format, including structured, semi-structured, and unstructured data. The data requirements are not defined until the data is needed. It then facilitates the colocation of data in variant schemas and structural forms. Its main capabilities are:

- Capture and store raw data at scale for a low cost. One of the main drivers for its adoption is to be able to store more historic data (including raw) on cheaper commodity hardware. The cost is also reduced by offloading data from more expensive systems (like MPP databases).

- Store different types of data in the same repository, including normalized, multidimensional and free-form like images or text.

- Allow exploration and different transformations of the data by using different tools and egines. This allows new types of processing to be applied, i.e. not only acces through declarative queries but also any potential transformation and use case like batch, near-real time and so on.

- The structure of the data is defined at the time it is used. This is referred to as schema on read, meaning that the data is stored in flexible and extensible formats and serialization techniques, allowing for different applications and optimizations based on those.

Data lakes are then different from MPP databases and traditional RDBMS (see Table 3.1 for a full comparison). In general, they naturally scale to larger data sets in a more cost-effective way, and enable the acquisition and storage of larger data sets, no matter their current format and the evolution they might take.

Data lakes are now mature and mainstream in most of the capabilities they offer. HDFS is the de facto standard for a data lake in the Hadoop ecosystem. Figure 3.2 shows the architecture of this distributed file system. Data is collocated with the nodes performing the processing (although there may be some specialised ones for I/O), and there is a *namenode* that contains the physical locations of the blocks pertaining to each file. Clients reading the data do it directly from the data nodes.

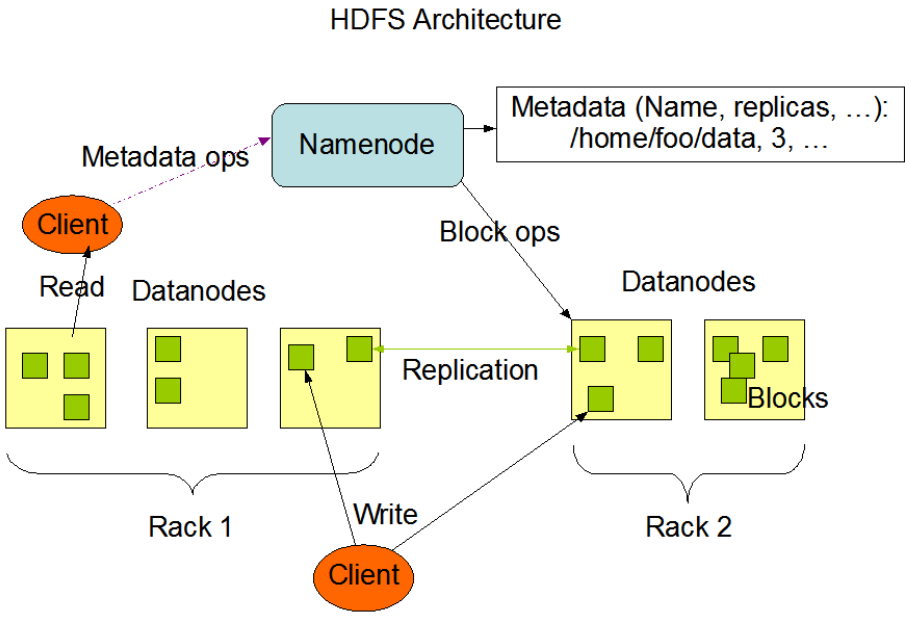


Figure 3.2: HDFS architecture.

Furthermore, support for replication is native, and files can be stored using different serialization techniques available in the Hadoop ecosystem like Apache Avro[9] or Protocol Buffers[10]. These serialization formats are a good way of integrating data in the data lake as they provide ways to evolve the schemas and be backwards compatible with data sets that were generated with old data models (obviously with some constraints).

[9]https://avro.apache.org/
[10]https://developers.google.com/protocol-buffers/

Table 3.1: Comparison between the Data Lake and other MPP databases or traditional RDBMS

Dimension	MPP Database	Data Lake
Data	Structured and processed once it lands. Schema on write: required data is identified and modeled in advance. It offers performance, security and integration.	Structured, semi-structured, unstructured and raw. Schema on read means data must be captured in the software for each program accessing it. It offers agility and ease of data capture, but requires work at the end of the process (when defining what to read).
Workload	Interactive analytics of a lot of concurrent users. Some sort of batch processing as well	Batch processing of data at scale, currently improving capabilities to support more interactive exploration and processing (see Section 3.4)
Scale	Can scale to large data volumes at a moderate cost (given new open source MPP solutions)	It scales to extreme data volumes at low cost.
Access	The data is accessed through standard declarative ways like SQL. Good integration with visualization and reporting tools operating on top of SQL.	The data is normally accessed through programs created by developers. There is increasing support for SQL and other mixed methods (e.g. SQL with some embedded code).
SQL	Standard SQL, ACID compliant (with the possibility to perform updates and deletes over specific data items).	Flexible programming model with non-negligible improvements in interactivity (and the possibility to update and delete specific data records under certain circumstances)
Agility	Highly-structured repository by definition. It is not technically hard to change the structure, but it can be very time-consuming given all the processes that are tied to it.	It lacks the predefined structure, which gives developers and data scientists the ability to easily configure and reconfigure their models, queries, and applications on-the-fly.
Users	Business proffessionals, scientists or data analysts.	Data scientists, i.e. with field domain, Big Data and modeling acumen.
Security	More mature as technology has been available for a while. The fact that a structured data model has to be created in advance, makes it easier to define the security levels for the different stakeholders.	Technology is rapidly maturing. It is not a question of whether this will happen but when will this catch up.

This way, compatibility of the different pipelines (for a variety of purposes) is obtained through the data sets, as they can be accessed by several engines and tools. This makes it easier to work in a multidisciplinar environment (like in any scientific field) where software engineers can work in a specific typed programming language like Java or Scala (very suitable for operational workflows) but yet allow scientists to explore the data sets with other more interactive and/or scripting languages with no compile time type checking (Python for instance). This approach requires some kind of semantic layer and its associated governance for e.g. data lineage. In addition, metadata helping describe this semantic layer can be best collected at ingestion time, facilitating a proper governance. Last but not least, building a data lake can effectively break the silos of information that naturally emerge in organizations.

HDFS is not the only implementation of a data lake. There are other equivalent solutions like Amazon Simple Storage Service (S3), that does not provide collocation of data but which is designed to store long-lasting data sets in a public cloud environment (as computing power gets instantiated and destroyed on-demand). GlusterFS[11] is another popular distributed file system. Another promising solution, complementing other distributed file systems, is Alluxio[12] (formerly known as Tachyon), which provides a better integration with in-memory workloads.

3.3 Column Orientation

An important novel (and complementary) technique for storing data is column orientation. It has made its way to the de facto file format implementation of the data lake, as well as to other interactive (and web-scale) ad-hoc query systems like Dremel (Melnik et al., 2010), and its open source counterpart Apache Drill[13]. The idea, which originates from the relational DBMS space, has now been incorporated in Big Data ecosystems like Hadoop. The two common implementations fully integrated with HDFS are Apache Parquet[14] and Apache Optimized Row Columnar (ORC)[15]. Figure 3.3 shows an example of the ORC file format to illustrate how each column is stored contiguously. The main features of column orientation are enumerated below:

- Data is stored contiguously on disk by columns rather than rows. Then, each block in HDFS contains a range of rows of the dataset and there is some metadata that can be used to seek to the start or the end of any column data, so if we are reading just two columns, we do not have to scan the whole block, but just the two columns data. This way, it is not necessary to create many different files which might incur in extra overhead for the HDFS name node.

[11] https://www.gluster.org/
[12] http://www.alluxio.org/
[13] https://drill.apache.org/
[14] https://parquet.apache.org/
[15] https://orc.apache.org/

- Compression ratio for each column data will be higher than the row oriented counterpart due to the fact that values for the same column are usually more similar, overall for scientific data sets involving time series, because new values representing certain phenomena are likely to be similar than those just measured. These implementations go beyond a simple compression for the column data by allowing different compression algorithms for different types of data, or even do so on the fly as we create the dataset by trying several alternatives. For instance, for a column representing a measure (floating point number) of a determined sensor or instrument, it would be reasonable to use a delta compression algorithm where we store the differences between values which will probably require less bits for their representation. It is important to remark that I/O takes more time than the associated CPU time (de)compressing the same data. For other columns with enumerated values, the values themselves will be stored in the metadata section along with a shorter (minimum) set of bits which will be the ones being used in the column data values. This of course requires (de)serializing the whole block (range of rows) for building back the original values of each column, but this technique usually performs well and takes less time. Examples of compression techniques are null supression, delta/dictionary/run-length encoding, etc (Abadi et al., 2006).

- Predicate push-down. Not only we can specify the set of columns that will be read for a determined workflow, but for those that need to be queried or filtered with some constraints, we can also push down the predicates so that the data not needed are not even deserialized in the worker nodes, or even not read from disk (only the metadata available is accessed). This is achieved through the usage of bloom filters (Bloom, 1970), which are a compact data structure that is used to test whether an element is a member of a set. False positive matches are possible (trade-off allowing a much more reduced size), but false negatives are not, which make them perfect for columnar stores.

- Complex nested data structures can be represented in the format (not just simple flattening of nested namespaces like in the naive implementation used in Section 4.3). The data types range from simple integers, floating point and strings, to structs, lists, maps, unions and arbitrary binary data. One example of a technique for implementing this feature is presented in more detail in Melnik et al. (2010).

- This data format representation is agnostic to the data processing, data model or even the programming language, being Parquet the best suited for this interoperability in systems and languages.

- Further improvements of the column-based approach include the ability to split files without scanning for markers, some kind of indexing per chunk or stripe and the ability to perform operations for updating or deleting rows, which makes it ACID compliant. However, this does not obviously intend to provide OLTP requirements. It can support millions of rows updated per a transaction, but it can not support millions of transactions an hour.

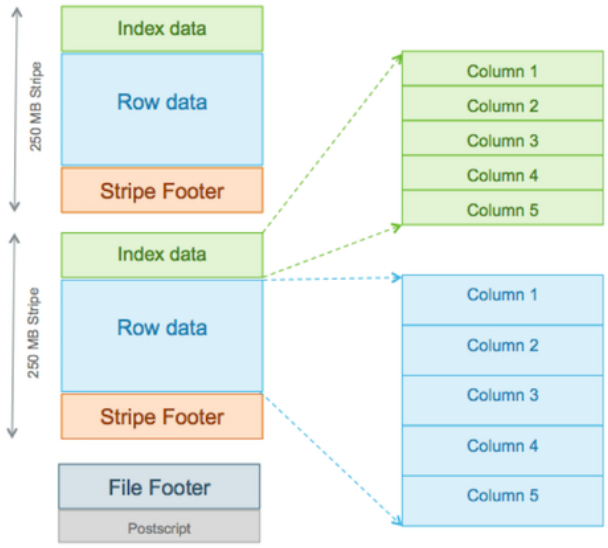


Figure 3.3: ORC file layout. Each stripe is independent of each other and contains only entire rows so that rows never straddle stripe boundaries. The data for each column is stored separately and there are statistics about each column both at file and stripe level. The stripe footer contains the encoding of each column among other things.

These storage formats are crucial in any architecture that is meant to support scientific workflows because typical scientific data sets are multidimensional, especially those found in astronomical surveys where a set of instruments are observing different phenomena of the same area of the sky (astrometry, photometry, etc). Workflows accessing them do not normally need to access all features, but just those relevant to the particular use case being studied. By having the data organized in columns, scanning time will be very much reduced as only those required dimensions will be fetched from disk. But not only that, the amount of memory consumed will also be less, resulting in a more efficient use of a scarce resource, which will eventually let machine learning algorithms or models (normally iterative) run faster (on data that stays in memory). Figure 3.4 shows an example of a real astronomical data set (GUMS version 10) being compressed in a popular MPP database (Greenplum). We observe that the columnar format compresses even further the data set. Using different compression techniques for each column would certainly compress it even more.

This innovation is the best match for information that is going to be stored in the data lake. It facilitates exploration of large data sets as the end user decides what to read from disk. The compatibility of the two major implementations with the main programming languages and data processing engines makes it suitable for scientific data processing and archiving.

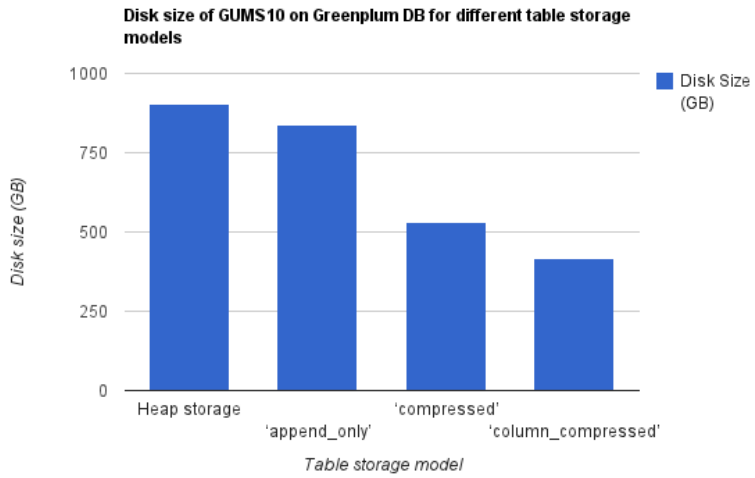


Figure 3.4: Disk size of the GUMS version 10 data set for different storage models selected in Greenplum DBMS.

3.4 General Purpose Large Scale Data Processing Engines

Even though MapReduce programming model became pervasive for large scale data processing, we can easily identify some flaws when it is applied to iterative algorithms (like the ones found in machine learning and graph processing) (Zaharia et al., 2012) or when it is utilized in typical workflows that make use of reference data sets (which are normally not so big in volume) (Blanas et al., 2010). These flaws do not seem to be the result of a misconception in the MapReduce programming model, which still has a clear niche in batch processing, but more of a wrong perception (probably due to its popularity (Rowstron et al., 2012)) that it had to be leveraged for any use case involving complex transformations and/or large data sets. Further constraints in which MapReduce has struggled to provide a good fit are:

- The offering of high-level primitives that truly hide the inherent complexity of parallel processing operations like aggregations or joins (Blanas et al., 2010), or typical operators for filtering, counting, getting the distinct elements of a collection, etc.
- The possibility to perform interactive exploration and data mining, enabling ad-hoc queries, sampling, model training and so on.
- The lack of abstractions to leverage distributed memory in scenarios that require a real time view of the data or the models' outcome (through streaming capabilities). This in the end gave rise to architectures that used two different solutions (with similar functionality) to separately focus on the batch and real-time flows

of data. This concept refers to the *lambda architecture* (see Section 3.2.1), which although a good interim workaround, leads to duplication of the code base and higher operational costs (maintenance and debugging of more systems).

Furthermore, empirical research (Rowstron et al., 2012; Elmeleegy, 2013) shows that at least two analytics production clusters at Microsoft and Yahoo have median job input sizes under 14 Gigabyte (GB), and 90% of jobs on a particular Facebook cluster have input sizes under 100 GB (Ananthanarayanan et al., 2012). Also, memory is getting cheaper and cheaper and even the heaviest workloads dealing with TB of data, normally need to access reference data sets that are much smaller (in the range of MB or several GB).

In addition, new file formats (see Section 3.3) have proven to significantly compress typical data sets collected from business operational processes or scientific workflows (Abadi et al., 2006). The fact that this data can stay in memory in a compressed form (as decompression is not so computationally expensive when using certain light-weight techniques), makes the transition towards in-memory processing paradigms even more attractive.

Last but not least, other solutions traditionally focusing on batch processing (i.e. Hive) are now shifting towards a more memory centric approach, reducing latencies in queries and applying state-of-the-art technological advances (Huai et al., 2014). Those advancements include the previously introduced ORC file format and some others:

- MapReduce stages were designed to dump intermediate data sets to disk so that a long running task (of days) could recover from the point of failure. This disk I/O increase the latency for more interactive workloads. Apache Tez[16] solves this issue by defining a memory based complex Directed Acyclic Graph (DAG) of tasks that are executed in memory (data is not stored on disk in between stages).

- Effectively leverage the field indexes and statistics available in the ORC file format, and build more optimized query plans.

- Vectorization to effectively utilize the capabilities of modern CPU. In this case, an operation being done over a column gets vectorized by taking several rows and applying the operation all at once over the values of the same columns (each belonging to a different row). Vectorization is extensively use in the implementation of the Grand Challenge presented in Chapter 5.

With all these improvements, tools and engines, we observe that the ecosystem is more and more complex, with highly specialised solutions that focus on a particular problem, i.e. MapReduce for batch processing, Apache Storm for real time workflows or streaming, Apache Giraph[17] for graph manipulation and so on. This specialisation, together with the old architectural approach stated in the Lambda architecture, makes data processing application development more difficult.

[16] https://tez.apache.org/
[17] http://giraph.apache.org/

First, we have more tools that are available to us. These tools and engines fit well to a particular use case (be it batch, streaming, etc). Given the compatibility through the data formats and the fact that all data is stored in the same data lake, it would be easy to cherry-pick the best of breed depending on the use case and integrate through the data model each tool or engine consumes/produces. However, this might lead to an excessive complexity in the workflows, which, to some extent, is what typically happens in many scientific workflows where data being collected through certain instrumentation has to go all the way through a multitude of platforms and systems transforming and analysing it. This complexity may not be needed if we consider the non-negligible overlap of many of these systems, e.g. MapReduce over Apache Tez will never beat any specific solution for streaming, but may be enough for many use cases that do not require real time enforcements. Furthermore, the level of specialization of each engine requires a different mindset when developing on top. The resulting code will certainly have overlaps as well, leading to duplication, which raises costs both for maintaining the software and the engines where it runs on.

It would be ideal then to build an intermediate engine that can deal with most of the use cases (near-real time or streaming, batch, in-memory, SQL, machine learning, etc) in a reasonable way, and for those scenarios in which a specialised solution is required (for whatever reason), a selection of the best tool from the ecosystem can be made, and compatibility is achieved through data stored in the data lake.

3.4.1 Apache Spark

Apache Spark (Zaharia et al., 2012) has gained a lot of traction and attention because it addresses most of these shortcomings. It proposes a generic distributed processing engine that can be leveraged for many different purposes, i.e. streaming use cases (Zaharia et al., 2013), machine learning workloads[18], graph processing (Xin et al., 2013), SQL access to data (Armbrust et al., 2015) for exploration and visualization purposes, and other innovations on top for automatic model selection and parametrization (Kraska et al., 2013; Sparks et al., 2013), or for approximate query processing on extremely large volumes of data (Agarwal et al., 2012, 2013).

It was originally developed for machine learning algorithms, which are typically iterative (scanning the data over and over again for optimising a certain function), or for interactive data mining through a console that enables exploration. These two objectives are enabled by providing the relevant primitives for working with the data sets in memory. The fact that Apache Spark takes care of the memory management and the variety of operations made available to the user, allow to both expose a console to the final user and also perform these iterative workflows faster (as the data can stay in memory among iterations).

The idea of creating a general purpose distributed processing engine that can be reused for different use cases is proven by looking at the lines of code of Apache Spark, and comparing to other niche solutions for batch, streaming, MPP database and graph

[18] http://spark.apache.org/mllib/

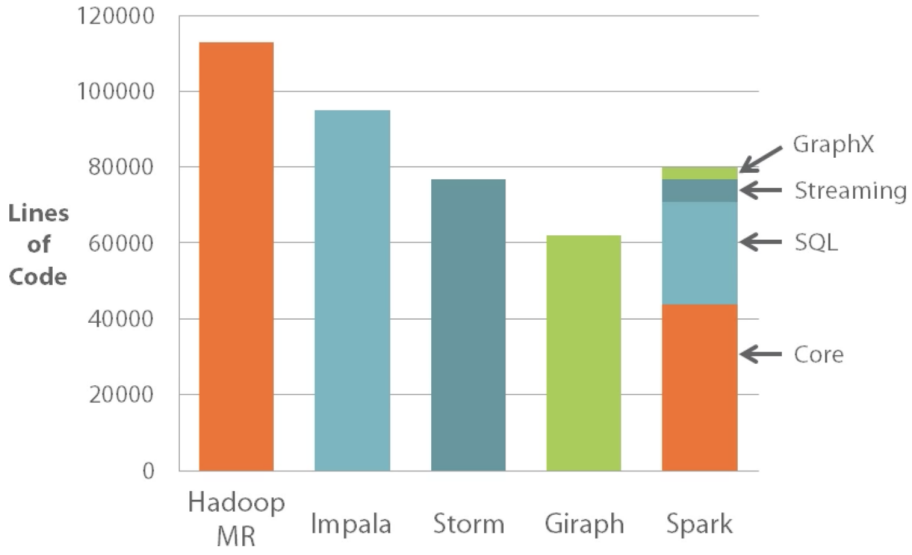


Figure 3.5: Lines of code comparison for major Big Data frameworks.

processing (see Figure 3.5). It is important to remark that Apache Spark is implemented in Scala[19], whose software typically produces more compact code with less lines of code. Even so, the difference among the other counterparts is noticeable. Lesser code means easier management and optimization. Furthermore, any optimization in the core is automatically reflected on the libraries built on top.

Even if Apache Spark is internally coded in Scala, it provides several other language bindings, i.e. Python, Java and recently R. This makes it more suitable for the scientific community as they may feel more comfortable with these languages given the conducted research put forward in Section 4.3.6. Having alternatives proves to work fine in these complex scenarios where the architecture needs to be leveraged by many different stakeholders that might tend to use different programming or scripting languages.

Figure 3.6 shows the stack of Apache Spark core and its main libraries. These libraries are being bundled by the same team of developers. This way, they can be implemented in a more effective way by leveraging the knowledge of the main core engine. These libraries are:

- **Apache Spark core engine**. Spark core engine is the central part of the stack that binds all components. It is responsible for all basic spark functionalities including memory management, fault-recovery, storage system interaction and more. One of the major programming abstraction of spark is called Resilient Distributed Dataset (RDD), which is a collection of items distributed across multiple nodes, typically stored in the memory of these nodes. They can be manipulated in parallel through

[19]http://www.scala-lang.org/

a set of high-level actions like transformations, aggregations, joins and so on.

- **Spark SQL** is the package for working with structured data. It allows querying data via SQL as well the Hive Query Language (HQL), supporting many sources of data, including Hive tables, Parquet or ORC formats, and JSON. Moreover, it provides a SQL interface that allows developers to write programs that manipulate the RDD in other languages (Scala, Python, Java or R) along with the SQL queries. This is a key ingredient that allows for the combination of SQL with complex analytics coded in those programming languages, providing the means for the straightforward implementation of astronomical services like those in the VO, e.g. TAP (see Section 2.2). Apache Hive might be another option for a SQL interface that scales well. However, its main drawback may be the lack of ways to easily embed code in the SQL instructions, or the interoperability of both in different stages (i.e. start the workflow with a set of queries and then continue with a complex pipeline written in any of the programming languages that are available in Apache Spark).

- **Spark Streaming** is the component that allows processing of live streams of data (topics coming from Apache Kafka, telemetry, initial data treatments, Twitter streams, or other messages queues containing regular updates). The streaming API extends the RDD API and hence it facilitates code reuse among the batch and streaming processing workflows. This is one of the main concepts of Apache Spark, the idea of unifying several types of processing under the same engine. It helps reduce the high complexity of systems implementing the Lambda architecture (see Section 3.2.1) as the code for dealing with the batch view can be reused in the processing applied to the near-real time one. Furthermore, all transformations of incoming data as well as its final exposure (through Spark SQL) will leave in the same framework. This makes it very suitable for unifying architectures for several SOC and scientific archives as stated in Section 4.4. A potential competitor for Spark Streaming might be Apache Storm, which goes beyond the concept of the Spark Streaming *micro-batch* (with minimum latency of around half a second) and can implement scenarios demanding smaller latencies. However, for most of the real life use cases, accepted latencies are above that minimum threshold.

- **MLlib** is the library that provides basic machine learning functionality. Its current offering contains machine learning algorithms like regressions, clustering, classifications, collaborative filtering and gradient boosting mechanisms among others. All these algorithms are designed to scale to large data sets and also serve as examples of how to approach new developments for mission specific modeling like those stated in Section 4.5. One of the competitors for MLlib might be Apache Mahout[20] which implements a large variety of machine learning algorithms that used to run on a series of MapReduce jobs. Nowadays, it is being transformed to a programming

[20] http://mahout.apache.org/

environment for building scalable algorithms on top of Apache Spark (reinforcing the idea of the power of its core distributed processing engine).

- **GraphX** is a library for manipulating graphs and performing graph-parallel computations. Like the other components, GraphX extends the Spark RDD API, allowing the definition of directed graphs with arbitrary properties attached to each vertices and edges. GraphX also provides various operators for manipulating graphs (e.g. subgraph and the mapping of vertices) and a library of common graph algorithms (e.g. PageRank (Page et al., 1999) and triangle counting).

All the features stated above make Apache Spark as the best suitable framework for the Gaia Archive Data Mining sub-workpackage as it enables the possibility for some of the concepts laid out in Section 2.4:

1. It empowers the idea of a *living archive* by providing a view of any data set being derived through well-known interfaces like SQL.
2. It facilitates an *archive as a model*, which could be developed, executed and tweaked on the platform (and from the raw data), allowing for a straightforward release of the results to the community through the mentioned interfaces (and the acompanying code).
3. The generic distributed processing engine is suited to be run on a Cloud environment, where both resources and middleware can be made available to the community in isolated and on-demand environments (some kind of PaaS).
4. Interoperability is obtained by both the possibilities enabled by the data lake, but also given the variety of programming languages for which there are bindings available as remarked above.
5. Interactivity is supplied not only by the availability of a console, but also with the integration of multi-purpose notebooks like Apache Zeppelin[21], that help build data-driven, interactive and collaborative documents with declarative (i.e. SQL) and programming (i.e. Scala) languages. These notebooks are a step forward towards a collaborative reproducible research.
6. Consolidation of the SOC, archiving and data exploitation is easier when a generic distributed processing engine that can cope with streaming, batch and declarative workflows is leveraged.
7. Existing models are easy to port, and new ones are easy to develop. Chapter 5 shows an example of two models that have been developed from scratch. These models are successfully using existing tools for MCMC without any modifications being done to them. This would prove challenging when using other frameworks (e.g. MapReduce).

[21] https://zeppelin.apache.org/

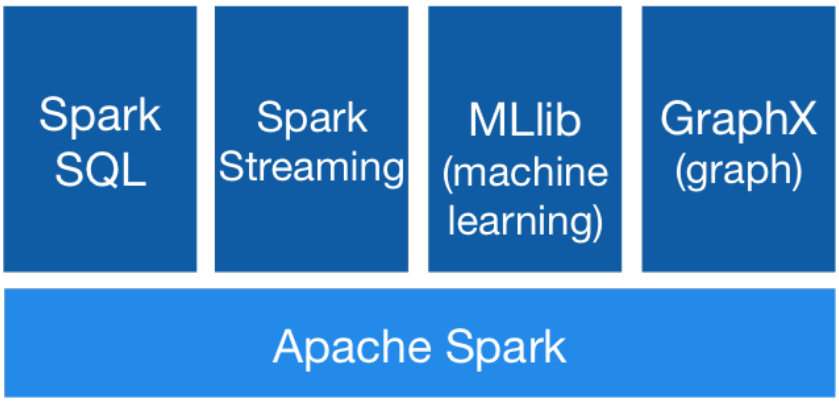


Figure 3.6: Generality of Apache Spark combining SQL, streaming, machine learning and graph processing over the same distributed processing engine.

3.5 Data as a Service

A typical scientific archive follows the Data as a Service (DaaS) model as the scientific information is distributed over a network (the Internet). The VO may be seen as the standardization of a set of DaaS services for astronomical data. Its benefits include the possibility to move data easily from one platform to another, outsource the presentation (or visualization) layer, preservation of data integrity by implementing access control measures, compatibility among different platforms, collection of automated data metrics for quality purposes and global accessibility among others. It consists of an entity that comprises both the platform and the data.

However, it should not be limited to the well-defined interfaces to the data, but also the ability to interact and process it, *data wrangling*, or a higher level set of operations, i.e. some kind of Domain Specific Language (DSL), that helps close the gap between the scientific and technological fields. This will enhance the adoption among those users who are less experienced in technology, by providing functionality that can be used and customized on-the-fly. Many of the traditional machine learning and data mining techniques (the most generic ones) will be made available in the distributed processing engines like Apache Spark (through MLlib), but others more specific to science and astronomy will not. Examples might be operations as simple as a geometrical search for selecting stars in a particular area of the sky, a more complex implementation of a cross-match in different wavelengths (with a configurable lambda that defines what a match is), a combination/composition of both, etc. The design of these higher level operations should take into account the potential reusability they might have and so provide proper configuration options (and default values) in the same way it is done in some popular programming languages used for analytics like R or Python. A powerful DSL allows to query any single data set and ease the exploration of data or any development on top.

Moreover, the ability to seamlessly provide with a semantic layer, not just for the

official data deliverables but for any other derived or incorporated data sets, will both encourage collaboration and empower the reusability of intermediate results, something required in order to unlock the potential of a real living archive (Brown, 2012). This does not only refer to the metadata, which can be queried to identify a subset of the data archive with certain features, but also to the data lineage information of when and how a particular data set was created, along with the inputs that it took and the software or model that was run.

In addition, an interactive access to the platform (both data processing engines and data sets) will facilitate community engagement and will be the key to enable exploration of data sets and any preliminary testing of hypotheses or assumptions (like remarked above). It will also soften the barriers for bringing the software and models to the data centre, encouraging collaborative and reproducible research by being able to share intermediate results otherwise impossible for the biggest in size. The proper interoperability with visualization engines and tools will effectively close the loop, increasing the experience offered throughout the platform.

Last but not least, Functional Programming is getting traction as the best programming paradigm for Big Data, given the scalability that it offers. This makes it very suitable for DaaS. Functional Programming is based on the idea of immutable state, i.e. the state of data does not change over time and thus it is not needed to synchronize the accesses to data from the different threads (which prevents scalability). This programming paradigm has been boosted by the current trend in CPU multicore architectures, which lower the clock speed and provide with a significant number of cores that can execute independent flows of instructions. The unlimited scalability of Cloud computing has also contributed to the popularity of this programming paradigm, which exists since many decades. The attributed benefits that Functional Programming provides range from a better error handling and modularity of the code, to shorter (more expressive) code and increased developer productivity.

Chapter 4

Architecture and Techniques for the Gaia Mission Archive

If you want something new,
you have to stop doing
something old.

Peter F. Drucker.

The ESA Gaia mission represents a breakthrough in astrophysics, a cornerstone mission aimed at producing the most accurate three dimensional map of the Milky Way to date. The resulting stereoscopic census of our Galaxy will represent a giant leap in astrometric accuracy complemented by the only full sky homogeneous photometric survey with an angular resolution comparable to that of the Hubble Space Telescope (HST), as well as the largest spectroscopic survey ever undertaken. The scientific bounty will be immense, not only unravelling the formation history and evolution of our Galaxy but also revealing and classifying thousands of extra-solar planetary systems, minor bodies within our solar system and millions of extragalactic objects, including some 500,000 quasars. Moreover, such a massive survey is bound to uncover many surprises that the universe still holds in store for us.

The Gaia mission poses several challenges for current data archiving technologies, mainly due to the unprecedented amount of data that will be produced. The final data delivery will not only include the catalogue of one billion sources but also the single epoch Charge Coupled Device (CCD) transit data that was used in its computation (including spectra), making an estimated total data set of around 1 PB.

New challenges arise in two main areas: state of the art computing technologies that need to be applied for the ingestion, data management and storage processes of the OAIS model (see Figure 2.4), and new methodologies and protocols will have to arise, based in current VO technology (see Section 2.2), to fulfil new requirements in the access process. The combination of both processes will give the opportunity to deliver unprecedented levels of accessibility to perform high performance computation over large amounts of

scientific data. Additionally, Gaia presents a great opportunity for the utilization and development of existing and new modeling techniques respectively, as well as innovative visualisation tools.

The storage and management of PB data volumes cannot be easily dealt with the traditional monolithic approaches. Then, distributed systems are required to properly address its scientific exploitation. Data analysis requires making use of emerging architectures and techniques like those presented in Chapter 3, but applied to Gaia data (only simulations available at the time this research took place)[1]. This chapter provides an extensive view of these innovative solutions comprising the study of ways in which a skewed astronomical catalogue can be organised and partitioned in a MPP system, as well as a specific high-level and scientist-ready data aggregation framework that runs over MapReduce for bridging the gap between the scientific community and the latest technological advances. Furthermore, hybrid capabilities are put forward for coping with the variety of use cases found in the context of a demanding scientific mission, paying special attention to the unification of approaches for the SOC, archiving and data access through data lakes and innovative distributed processing engines as described in Chapter 3. Last but not least, features for enabling science in the archive are laid out as an introduction to the Grand Challenge (that will be covered in Chapter 5).

The publication supporting the innovative contributions of this chapter is Tapiador et al. (2014). There are also two Gaia CU 9 (Archive and Catalogue Access) technical reports (Tapiador, 2012, 2011) backing up the results put forward in this same chapter. The work carried out in this research contributed to the definition and preparation of the winning proposal (Luri et al., 2013) for the ESA Gaia CU 9 Announcement of Opportunity (AO). It took place within the Gaia Archive Preparation (GAP) working group, involving many of the Data Processing and Analysis Consortium (DPAC) members across different countries and national space agencies. This thesis' author has also co-authored three recent publications as member of the Gaia Collaboration, i.e. the Gaia mission (Gaia Collaboration et al., 2016b) and the ones referred to the Gaia Data Release 1 (Gaia Collaboration et al., 2016a, 2017).

4.1 The Gaia Mission Archive

The Gaia DPAC (Mignard and Drimmel, 2007) is a European collaboration including the ESA Gaia SOC and a broad international science community. The consortium is structured around a set of several CU each in charge of a specific aspect of the data processing for the initial data treatment, the main data processing referred as Astrometric Global Iterative Solution (AGIS) (O'Mullane et al., 2011), simulations, system architecture and data archiving and access among other things. Figure 4.1 shows a schematic overview of how DPAC is structured around scientific units (CU), in charge of developing the algorithms for the data processing, and the Data Processing Centre (DPC), which provide the infrastructure where to run the scientific pipelines.

[1]The first Gaia data release (Gaia Collaboration et al., 2016a) has taken place on September 14th,

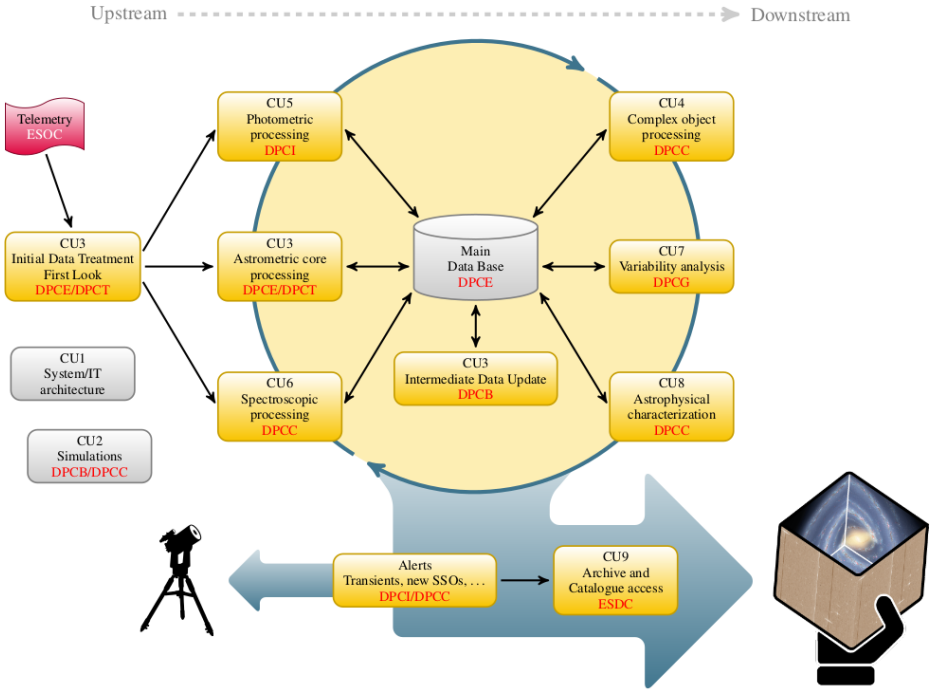


Figure 4.1: Schematic overview of the Gaia DPAC.

Gaia will provide an unprecedented census of our Galaxy in size, scope, and accuracy, encompassing astrometry, radial velocities and multi-colour photometry for over one billion objects in the sky. The primary scientific aim of the Gaia mission is to map the structure of the Galaxy and unravel its formation history. The main mission goal for the Gaia archive is to provide a comprehensive repository of the rich data products to be generated by Gaia, and a range of access mechanisms and associated helper applications to enable effective access to the Gaia data by the end user science community (Luri et al., 2013).

One of the major science cases for Gaia concerns building up an improved understanding of the current structure of our Galaxy, and using this to gain an insight into the formation and evolutionary processes at work. During the lifetime of Gaia, it will generate, year on year, improved information on a large sample of up to one billion stars in the Milky Way. The high precision parallaxes determined for these stars will, in principle allow the distances of these to be determined, and thus, with the large numbers of objects, astronomers will be able to build up a three dimensional view of the Milky Way. However, from the perspective of the Gaia archive, the end user astronomers will not simply issue a bulk query requesting the distance for the complete set of Gaia stars. The analysis is a significantly more complicated process, and will necessarily involve the interplay between observational data and detailed models.

2016.

All of the above imply complex queries to the Gaia archive, exposing the rich data attributes available for each Gaia object together with access to related information, either singularly or as an ensemble, from external ancillary data sources. These might include ground based survey data, or other space based surveys. All in all, although an archive or catalogue may be considered products as such, their full potential is only realised when the information content is efficiently accessible and widely used. This enables and fosters new scientific discoveries, improves our present understanding of Nature in significant ways, and leads to serendipitous results unthinkable at the time the mission/experiment was conceived.

It is probable that 80% of research may be produced by 20% of the archived data (i.e. the catalogue). However, the most impactful research will likely be generated out of the remaining 80% of the data available in the archive (raw data, transits, observations, etc). Therefore, the organisation of the archive needs to provide ways to tackle both issues by letting scientists access the catalogue in an easy and efficient way, as well as mechanisms to perform the most complex use cases, sometimes leveraging brute force approaches supplied by new computing architectures. This is referred to as a *hybrid architecture*, where the community will have different tools to tackle different problems. On the one hand, SQL like interfaces that are easy to understand and which allow some kind of pre-filtering of results, easy joining with other catalogues or scientists' data tables. On the other hand, more scalable solutions for running complex algorithms and models, often both by writing code in a programming language and leveraging higher level engines, tools and frameworks.

Figure 4.2 shows the work packages for CU9 (CU for the archive and catalogue access). They include extensive efforts to document and validate the catalogue (semantic layer), provide education and outreach (increasing scientific return and public awareness), science enabling applications (see Section 4.5) and some other architectural and operational matters. Furthermore, the products coming from any archive release will be twofold. First, a set of core products, i.e. the archive and the catalogue itself, and its documentation. Second, there will be a set of services and tools, which comprise the science enabling and visualisation tools, auxiliary data sets, web content, visualization tools and again its documentation.

4.2 Partitioning Astronomical Catalogues

Despite the limitations of MPP databases laid out in Section 3.2.2, especially in flexibility and scalability, RDBMS are still prevalent when low latencies are seeked. Parallel databases are then a good option to explore in case larger volumes of data need to be ingested. In the case of Gaia, its catalogue will not be so large in terms of volume (see Figure 3.4 for an example of the size of a simulated catalogue resembling the one Gaia will produce). However, other larger catalogues being currently planned or produced by ground or space based telescopes and missions will also need to be cross-matched to Gaia. Furthermore, any scientific catalogue being released will also have to provide some disk space allocation for scientists to use, some kind of area where they can create and

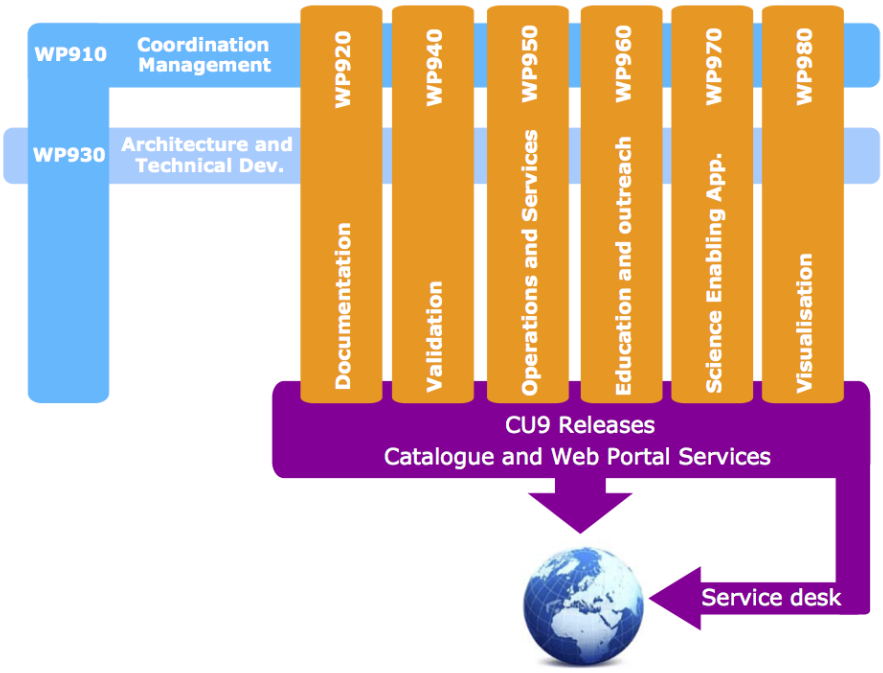


Figure 4.2: CU9 Work Packages.

share their own data tables. In order to deal with these concerns in a cost effective way, some kind of parallelism will be needed, enabling the possibility of iteratively growing the infrastructure without changing the architecture of the solution.

Parallelism aims at breaking things into smaller pieces and then deal with each of them in a separate resource. When it comes to data, it means that some kind of partitioning needs to be done, so that each chunk of data is stored in a different place. The idea is to split the data in a meaningful way so that later on queries can be optimized based on their constraints and only relevant partitions are read from disk, thus performing the whole operation much faster. One of the main issues when partitioning data sets is how to minimize *skew*, which occurs when the data is not well balanced among the processing elements. *Skew* may also happen in intermediate steps of database operations.

There are several ways in which we can partition a data set, some of the main ones are the following:

- Round-Robin. Data records are assigned to the different places in equal portions and in circular order among them. With this approach, the data set is uniformly distributed among the elements thus preventing any kind of skew, although the main pitfall is that there are no semantics into where it is stored what. Subsequent queries run on top will need to scan the whole data set to be able to return the matching records.

- Hash partitioning. A hash key is used to distribute rows evenly across the different partitions. This is ideal when we are looking for exact match queries by an identifier. For example, queries looking for specific astronomical sources through their identifier will only need to scan partitions whose hash identifiers are those being searched.

- Range partitioning. This type is used to assign each partition a range of values generated by the partitioning expression. Ranges must be ordered, contigous and non-overlapping so that each possible record can only be placed in one partition. This type is ideal for range queries over attributes used in the partitioning. For example, when looking at temporal data, we may specify a range of dates and only the relevant partitions need to be accessed. In the case of astronomical catalogues, each partition can be a different spatial region, and each source is stored in the partition based on its location. HEALPix[2] (Górski et al., 2005) is a sphere tessellation (pixelization) framework that subdivides a spherical surface into equal area pixels. More details on this technique are provided in Section 4.3.1. Q3C (Koposov and Bartunov, 2006) is another sphere tessellation approach that let us do similar things. When these techniques are used in range partitioning, the idea is to assign the pixel (small spatial area) identifier to each partition, so that the two dimensional space can be split with integer identifiers (values), and all sources falling into each spatial region are then stored in the relevant partition. We will be using both techniques in the experiments shown in this section, as common spatial queries are much faster when only the partitions overlapping with the area of interest are accessed at execution time.

Astronomical catalogues are skewed by nature, i.e. there are regions in the sky having more sources than others (see Figures 4.3 and 4.8). Furthermore, astronomers typically query these catalogues with spatial constraints (they are interested in a specific area of the sky). This factual reality makes range partitioning a suitable candidate when distributing the catalogues among the processing elements. However, as previously remarked, skewed partitions are counterproductive as they do not balance the work evenly among the nodes.

One practical way of tackling this issue is to create small splits of data, i.e. smaller spatial areas (more subdivisions on the sphere), and then use a technique based on histograms. This technique consists in:

1. Choose the partitioning attribute (i.e. HEALPix index) and create a histogram with a large number of partitions, i.e. more subdivisions on the sphere (see Figure 4.3 for an example).

2. Check data skew. If the data is not uniformly distributed, identify the biggest split and group the others so that each partition has the same number of elements. This grouping has to be done meaningfully, e.g. join splits that are spatially consecutive. In this example, consecutive identifiers belong to nearby areas in the HEALPix (*nested*) and Q3C sphere tesselations.

[2] http://healpix.sourceforge.net/

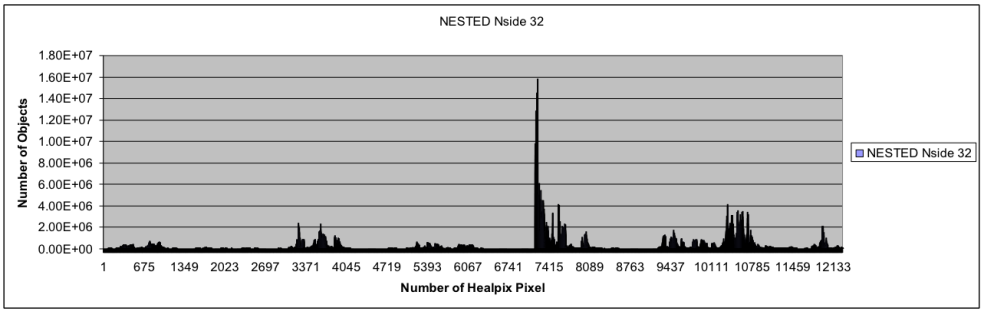


Figure 4.3: Histogram of the number of sources per HEALPix pixel subdivisions (area on the sky). See Figure 4.8 for a spatially projected view of this histogram.

3. Keep on consecutively grouping splits with low number of records until the amount of elements is similar to that of the biggest split. Then, each partition will contain a roughly similar number of records, although ranges of splits will obviously vary among partitions.

4. If the number of partitions is too low (given the resources available), there is always the solution of subdividing more the sphere in the first place and perform the algorithm again.

This technique is somehow not trivial, but it only needs to be done once, as subsequent astronomical catalogues are likely to have the same skew. This is obviously because of the fact that different surveys are observing the same reality (the Universe), although it considers that the distribution of sources per region of the sky will not change when we observe at fainter magnitudes (in current or future experiments). Other more complex techniques are Hybrid Range Partitioning Strategy (HRPS) (Ghandeharizadeh and DeWitt, 1990) and Multiattribute Grid Declustering (MAGIC) (Ghandeharizadeh and DeWitt, 1994).

For the tests carried out in subsequent subsections, Greenplum MPP is used, although we can assume relatively similar results had we used other similar MPP databases. The details of the infrastructure on AWS are as follows:

- The Amazon Elastic Compute Cloud (EC2) instance is *m1.xlarge* as, among other things, around 16 GB of memory per node is required for Greenplum.

- This instance contains 15 GB of memory, four disks with a total of 1,690 GB of DAS and four cores.

- With regard to the most optimum number of instances, the key point is the storage that is going to be needed. With eight instances (*m1.xlarge*), plus the instance for the master, we get around 13.52 TB of data, which makes 6.72 TB of user data with a replication factor of two.

- Regarding the storage architecture of the physical nodes, the best alternative is always the ephemeral disks, i.e. DAS, grouped with a zero-level Redundant Array of Independent Disks (RAID) (striped) for performance, as although Amazon Elastic Block Store (EBS) disks offer a much better redundancy, it is not recommended for this deployment as Greenplum is thought to run on shared-nothing commodity hardware that is likely to fail, i.e. segments holding data can be mirrored and when they go down the DBMS will still perform well. Once the node is back the segments will be resynchronized automatically.

4.2.1 Ingestion of the Catalogue

Greenplum DBMS allows to distribute data among the different computing nodes by using either round-robin and hash-based approaches (range distribution is not provided because it is prone to produce skew). Data physically distributed to different nodes can be then partitioned with both hashing and range methods. For the ingestion of the catalogue, the chosen data distribution technique was hash-based. This allows exact match queries and joins because when looking for a specific source we just need to compute its hash and go to that node and collect it. In addition, join operations, which are one the most computationally complex and time consuming operations in a DBMS, are also speeded up because they will be computed locally on each node (without shuffling any data). In the case of Gaia, this exact match queries and local joins are feasible because most of the generated products delivered by the different CU (e.g. observations, transits, auxiliary data and so on) have a source identifier as primary key. The identifier comprises a prefix from the corresponding HEALPix pixel of the source location, and a suffix from a running counter. This works fine with the hashing method used for distribution, storing sources of the same region across the different nodes of the MPP cluster running the database.

The only drawback of this approach, for speeding up exact match queries and perform joins locally without any data shuffling, might occur in case these tables containing observations or transits are not well distributed by that field. It would then produce skew among the nodes. This might happen when there is a *1:n* multiplicity (one source is made up by several observations) between the table holding the information of the source and the other one (e.g. containing the observations), and the distribution of data is not roughly equivalent for every source, i.e. there may not be data associated for certain sources whereas there may be a lot for other ones. This is the case for Gaia: the number of transits per star varies roughly between approximately 40 and 220 with bright stars havind also (heavy) Radial Velocity Spectrometer (RVS) spectra. The first issue is clearly dependent on sky position and thus HEALPix pixel[3]. The second is independent of sky position.

In this case, the distribution of the data will have to be carefully studied and the convenience of this distribution policy must be assessed to prove that the time saved by locally joining the two tables is greater than the one wasted by retrieving data from less

[3]More details at http://www.cosmos.esa.int/web/gaia/transits

disks concurrently (the second table data is skewed, so only a few nodes will retrieve data). However, other techniques like denormalization (Sanders and Shin, 2001) together with column orientation (see Section 3.3) may figure this out, provided there is some support for multidimensional data (e.g. for storing spectra) and the use cases on top demand for such grouping of data in the same table.

It is important to remark that different MPP implementations like Cloudera Impala only support one level of partitioning (or data distribution) and thus each eventual partition would go to one particular node. In our case, Greenplum will allocate catalogue sources in computing nodes by using hashing on the source identifier, and all partitioning will be done afterwards on each node. Therefore, if the source identifier is not skewed, all computing nodes will contain some sources for each partition, and then one spatial query looking at a particular partition will make all nodes work on it. This is ideal for queries with some analytical workload, as all processors may be leveraged for those computations in a parallel way, boosting the operation and making the solution for such type of queries more scalable (more nodes, less data to process on each node, faster response time). Range distribution would not then be recommended for an astronomical catalogue, because the parallelism of a query would be much reduced. There would only be a few disks being accessed concurrently which would slow down the query execution. This would become worse and worse as we had to perform some other analyses over the rows being retrieved like aggregations, ordering, or most importantly joins with other tables, because the disks would not be the only bottleneck.

The most efficient way of ingesting data was to perform a parallel ingestion from several nodes from data stored in Comma Separated Values (CSV) files (compatible in most MPP databases out-of-the-box). Indexes or any transformation to the ingested records are better done in subsequent operations (once the data has been ingested). There is also the possibility to ingest (or export) from (to) the de facto implementation of the data lake (HDFS), and the estimated time would be similar when a fully parallel approach is taken (without intermediate landing servers).

4.2.2 Partitioning and Clustering

The partitioning of the catalogue is done by a field that represents the spatial location of the source in the sky. This field will be either the HEALPix or Q3C index, and the technique based on histograms introduced above will be used to group consecutive ranges so that each partition has the same number of elements. This is based on the fact that consecutive pixel identifiers mean consecutive areas on the sphere.

For MPP databases that only provide a partitioning technique (e.g. Cloudera Impala), skew has to be corrected for them to perform well. This means that if we are using the spatial pixel identifier for partitioning the catalogue, it would be advised to ensure that each partition contains the same number of elements (for even query processing). However, this is not strictly needed for Greenplum, as skew is handled by distributing the data with hashing over the source identifier column. Therefore, partitions do not need to be merged together in bigger partitions, provided the number of partitions is kept under reasonable margins (e.g. a few thousands). This approach would be straightforward to

implement as it would not be required to implement the technique based on histograms, but it might result in different response times depending on the partitions being accessed in the concrete queries to be run afterwards, as each of them might have a different number of records in it.

The idea behind partitioning an astronomical catalogue is to be able to use an indexing scheme that helps retrieve data faster (as only a few of the partitions are searched and this scan operations are done in parallel among all nodes in the MPP). The operation for clustering each partition physically reorders data based on an index. This operation can take a lot of time to execute, but once that it is performed, the data is accessed in an ordered way, speeding up even more the search operations that can leverage the index that was used to cluster the data. The best way to create a clustered table from an indexed table is by using an intermediate table that will become the final one once the process completes.

Sometimes, neither partitioning nor clustering are needed at all. In situations where the data can be kept in memory because the cluster memory can be scaled up to the size of the table or because there is no field over which most of the queries are made, a brute force approach will always be used, and latencies for query response can be tuned by adding or removing resources (the scan to the table is done in a parallel way among all nodes of the cluster). This is more than enough in most of the scenarios (this one being one of them), as the execution time for each particular query can be lowered below any threshold by simply increasing the number of resources. Indeed, there are some engines providing a SQL interface that do not implement partitioning or clustering capabilities (e.g. Apache Spark), also because they are more oriented towards analytical workloads (not OLTP ones). However, the tests carried out in Section 4.2.3 include the alternatives of partitioning and clustering the table as it is an available feature in Greenplum MPP.

4.2.3 Experiments

The spatial queries executed in the experiments are cone searches[4] using Q3C and PgSphere[5]. They are shown in Listing 4.1. The queries shown in Listing 4.1 are contextualised for each particular test case with the values shown in Table 4.1. Those comprise different configurations with regard to the size of the region being searched, and the amount of sources falling into that region. All tests have been run with GUMS version 10 simulated catalogue. Whenever a result is not shown for a determined test case, it means that the query did not finish due to out of memory errors or similar.

[4] A cone search is a relatively simple and common request for astronomical information, retrieving whatever is available for a given position in the sky and a given radius about that position.

[5] http://pgsphere.projects.pgfoundry.org/

Listing 4.1: Queries for the benchmark.

```
-- Q3C cone search
SELECT * INTO geometry_tests FROM gums_um_stellar_source WHERE
  q3c_radial_query(alpha, delta, <alpha>, <delta>, <radius>);

-- PgSphere cone search
-- 'pos' contains the position for each source
SELECT * INTO results_temp FROM gums10_um_stellar_source
  WHERE scircle'<(<alpha>d, <delta>d),<radius>d>'~pos;
```

Test Id	Region size - Amount of sources	<alpha >	<delta >	<radius >
Geom1	Small - Small	212.948209	51.824572	0.5
Geom2	Small - Medium-sized	315.3453	42.1262	0.5
Geom3	Small - Medium-sized	269.23942	-29.09853392	0.5
Geom4	Medium-sized - Medium-sized	315.3453	42.1262	2.5
Geom5	Medium-sized - Medium-sized	269.23942	-29.09853392	2.5
Geom6	Large - Relatively small	212.948209	51.824572	5
Geom7	Large - Huge	269.23942	-29.09853392	5
Geom8	Large - Very huge	269.23942	-29.09853392	7.5
Geom9	Very Large - Very huge	269.23942	-29.09853392	10
Geom10	Extremely Large - Extremely huge	269.23942	-29.09853392	20

Table 4.1: Test battery for geometrical queries benchmark.

Before deep diving into the test set executed, we can highlight some of the main conclusions (which do not vary much when compared to those that would have been obtained in a traditional monolithic RDBMS):

- The usage of indexes speeds up geometrical query execution.
- Table partitioning by using geometrical identifiers extremely speeds up geometrical query execution as only relevant partitions for the query will be scanned. This is only up to a limit, where sequential disk scans start outperforming indexed accesses to data.
- When the number of results grow, the clustering of the table improves the data retrieval stage as there is a sequential read from the disk (the most efficient way of reading data) due to the fact that data is physically ordered on the disk.

Figure 4.4 shows the results obtained for the geometrical benchmark when Q3C (both with and without table partitioning and clustering) and PgSphere are used. It does

not matter whether the table is partitioned or clustered for PgSphere as it is always processed sequentially. At a first glance we observe that for queries returning less than a certain number of rows (100-200 million rows) the approach with Q3C with the table partitioned and clustered by the Q3C index works best. However, for queries returning a higher number of rows it might be needed that the query is executed with a sequential (brute force) approach using PgSphere (as Q3C is clearly optimized to only be used with indexes). Therefore, it is proven that the same type of query (cone search) might need different approaches depending on the estimated number of rows to be retrieved. This conclusion poses some constraints to the design of any layers on top of the database, as it might be needed to take different approaches depending on certain parameters (estimated size of rows returned and so on).

We could of course tune the memory parameters of the database, and that would probably speed up queries, making more of them work fine (like the ones returning a lot of objects). However, there will always be a limit as we increase the amount of data. Therefore, the best configuration in terms of scalability seems to be the usage of indexes up to a determined point (e.g. a determined amount of objects being returned, a determined cone search area, etc), which has to be decided, and let the query run sequentially from that point on as it will be the most optimum way as a significant part of the table data has to be scanned. It is important to highlight that sequential query plans are the default ones for query execution in analytical databases (like Greenplum), so in this case, for queries returning a relatively small amount of objects (still to be determined), it seems more appropriate to change that and run the query by using Q3C (with indexes and the table partitioned/clustered), but from that point on, a sequential scan (e.g. PgSphere) seems to be the way to go.

Figure 4.5 shows a comparison of the sequential query execution (no index usage) of the geometrical benchmark for Q3C and PgSphere. This figure shows that when sequential scanning is used, the time to process the query is kept relatively constant (as expected). Furthermore, any increment in the number of rows to be retrieved implies the corresponding proportional query processing time rise. Then, we could certainly assert that performance would scale and speed up well (almost linearly) in case the number of nodes devoted to Greenplum cluster increased. In addition, the fact that sequential scan query plans always work fine whichever the number of rows to retrieve, justifies in a way that this is the default query plan for these types of queries in an analytical DBMS such as *Greenplum* (also given that these query plans scale and speed up much better as they involve a sequential scan of the whole dataset). One way or another, for queries returning a relatively small amount of objects (still to be determined), the index usage seems to be the most sensible approach as stated above.

So far, all tests have assumed that the user is interested in all source columns, but this is not normally the case. Normally, scientists will only need a subset of them (e.g. astrometry, photometry and so on), especially for elements that contain so much data like in the case of the Gaia catalogue. Therefore, two further tests have been carried out to find out more about the behaviour of these approaches. Note that both Q3C and PgSphere geometry modules have been used for the use cases where they perform well

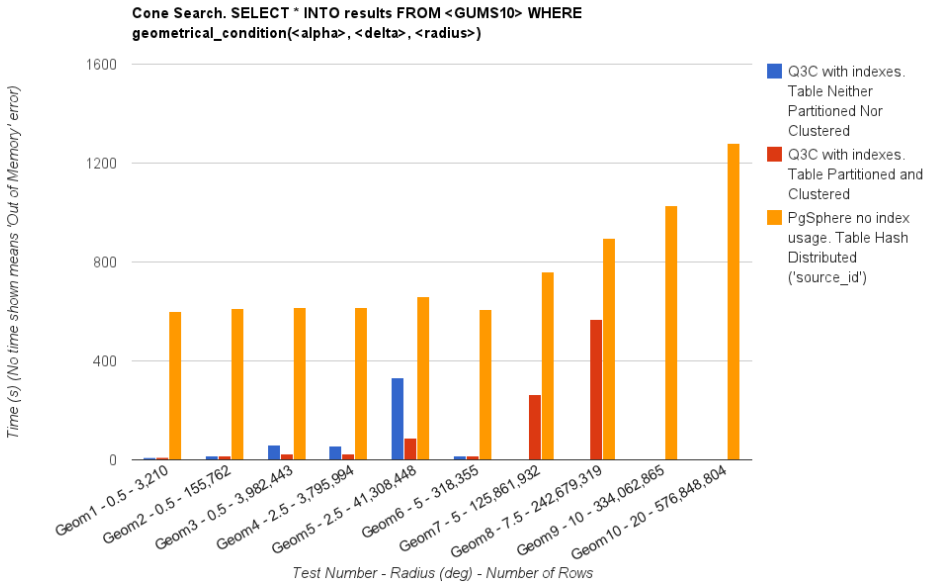


Figure 4.4: Q3C (indexed) with and without table partitioning and clustering, and PgSphere (sequential) geometry benchmark (GUMS version 10).

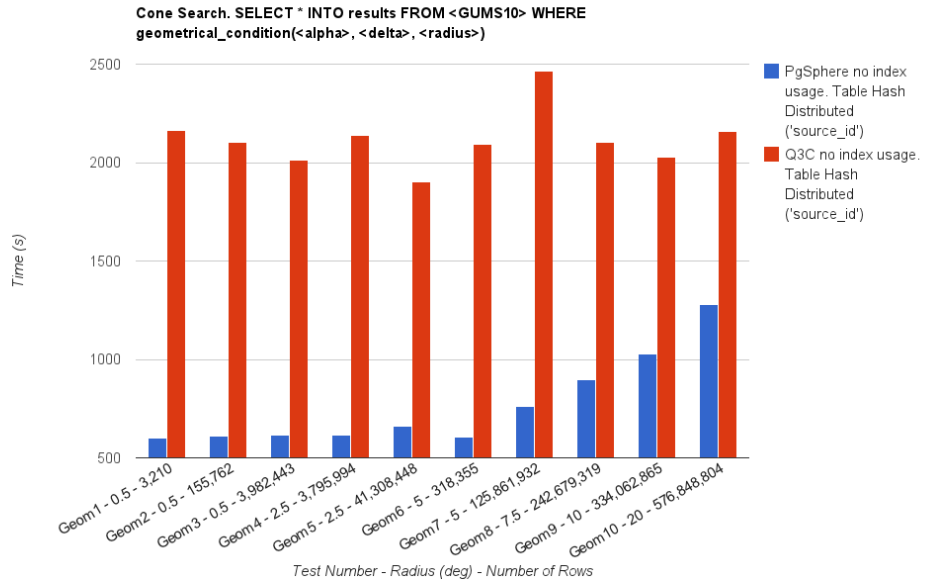


Figure 4.5: Geometry benchmark for Q3C and PgSphere (no index usage) for GUMS version 10 data set.

(Q3C using its index for small regions returning a low or moderate amount of objects and PgSphere sequential analysis for large regions returning a large amount of objects). The results are depicted in Figures 4.6 and 4.7.

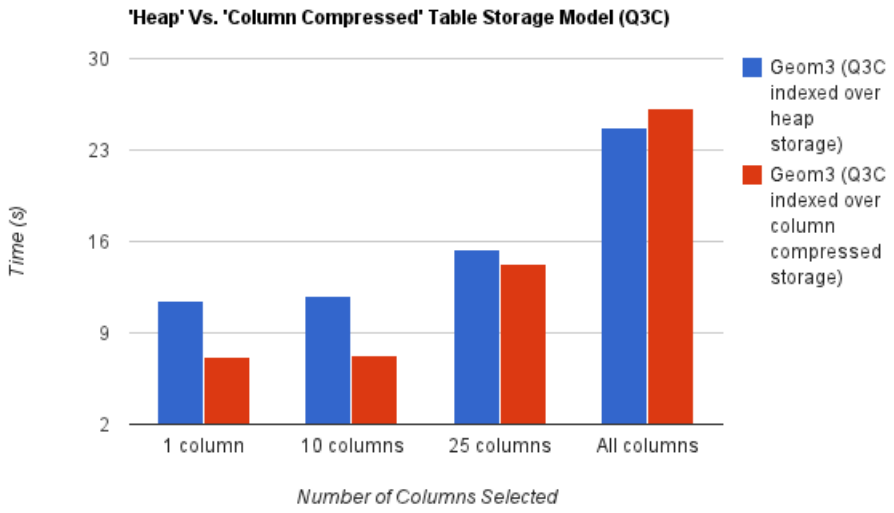


Figure 4.6: Heap and column compressed storage model comparison for different number of columns with Q3C GUMS version 10.

Looking at Figure 4.6, we can certainly conclude that even though the processing time for queries is low (less than 25 seconds), we obtain significant benefits when using the columnar store. This is of course due to the fact that less data is read in the column-oriented layout. The differences are greater when there is more data to be retrieved (Figure 4.7) with a performance improvement of ten times in the best scenario for the column approach, i.e. read one single column. It is remarkable that query processing time is kept roughly the same (or a bit better when using PgSphere) for both approaches, when all the columns are accessed (worst case scenario for the columnar store).

4.3 Higher Level Frameworks for Scientists

Some of the more widely used tools in data mining and statistics are multidimensional hypercubes and histograms, as they can provide summaries of different and complex phenomena (at a coarser or finer granularity) through a graphical representation of the data being analyzed, no matter how large the data set is. These tools are useful for a wide range of disciplines, in particular in science and astronomy, as they allow the study of certain features and their variations depending on other factors, as well as for data classification aggregations, pivot tables confronting two dimensions, etc. They also help

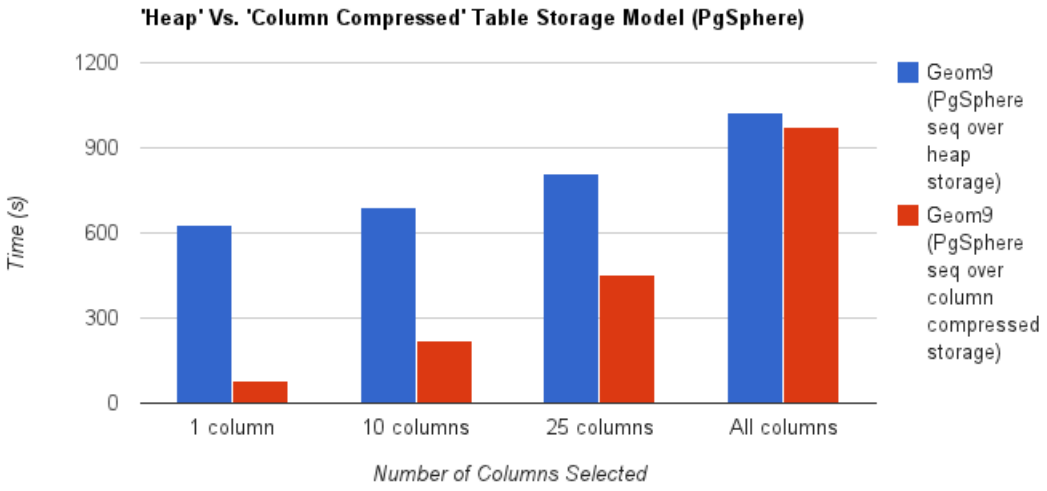


Figure 4.7: Heap and column compressed storage model comparison for different number of columns with PgSphere (GUMS version 10). Heap storage is row-oriented (one row after the other on disk)

scientists validate the generated data sets and check whether they fit within the expected values of the model or the other way around (also applicable to simulations).

As multidimensional histograms can be considered a very simple hypercube which normally contains one, two or three dimensions (often for visualization purposes) and whose measure is the count of objects given certain concrete values (or ranges) of its dimensions, we generally refer to hypercubes and will only mention histograms when the above conditions apply (hypercubes with one to three dimensions whose only measure is the object count).

4.3.1 Data Analysis in the Gaia Mission

In the case of the Gaia mission, many histograms will be produced for each data release in order to summarize and document the catalogues produced. Furthermore, a lot of density maps will have to be computed, e.g. for visualization purposes, as otherwise it would be impossible to plot such a large amount of objects. All these histograms and plots (see Robin, A. C. et al. (2012) for examples), the so-called *precomputed statistics*, will have to be (re)generated in the shortest period of time and this will imply a load peak in the data centre. Therefore, the solution adopted should be able to scale to the Cloud just in case it is needed due to e.g. the absence of a local infrastructure that can execute these work flows (as it would mean a high fixed cost for hardware which is underutilized most of the time).

Figures 4.8 and 4.9 show two simple examples of histograms that have been created

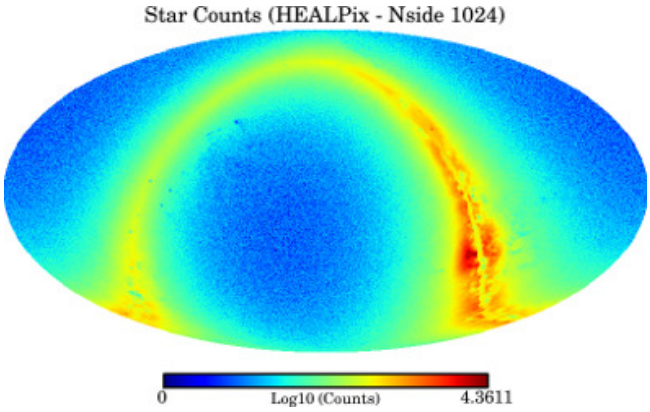


Figure 4.8: Star density map using HEALPix.

with the framework and which we will use throughout this section for presenting the different results obtained. The GUMS version 10 data set (Robin, A. C. et al., 2012), from which histograms have been created, is a simulated catalogue of stars that resembles the one that will be produced by the Gaia mission. It contains a bit more than two billion objects with a size of 343 GB in its original delivery form (binary and compressed with Deflate). Since there was no Gaia observed data at the time this work took place, all Gaia data processing software is verified against simulated observations of this universe model (Robin, A. C. et al., 2012).

The histogram shown in Figure 4.8 is a star density map of the sky. It has been built by using a sphere tessellation (pixelization) framework named HEALPix[6] (Górski et al., 2005), which among other things provides a set of routines for subdividing a spherical surface into equal area pixels, and for obtaining the pixel number corresponding to a given pair of angular coordinates. HEALPix is widely known not only in astronomy but also in the field of earth observation. HEALPix also allows indexing of geometrical data on the sphere for speeding up queries and retrievals in relational databases. The resolution of the pixels is driven by a parameter called N_{side}, which must be a power of two. The higher this parameter is, the more pixel subdivisions the sphere will have. For $N_{side} = 1024$ (used in Figure 4.8 and in the rest of tests in this section) there are 12 582 912 pixels.

The example in Figure 4.9 is a theoretical Hertzsprung-Russell diagram (two dimensional) which shows the effective temperature of stars vs. their luminosity. This is a widely used diagram in astronomy, which contains information about the age (or mixture of ages) of the plotted set of stars as well as about the physical characteristics and evolutionary status of the individual stars.

For the scenario just sketched the MapReduce approach is the most reasonable one. This is not only due to the fact that it scales up very well (also in the Cloud) or that

[6] http://healpix.sourceforge.net/

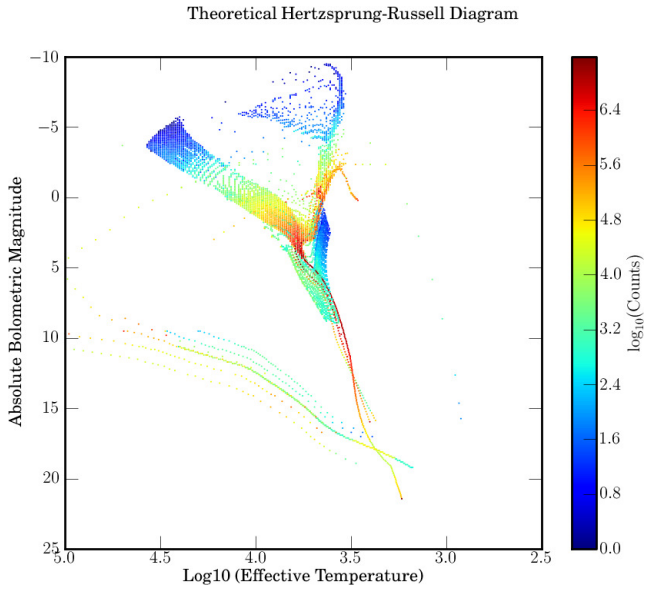


Figure 4.9: Theoretical Hertzsprung-Russell diagram. The horizontal axis shows the temperature of the stars on a logarithmic scale and the vertical axis shows a measure of the luminosity (intrinsic brightness) of the stars (also logarithmic, brighter stars are at more negative values).

there are open-source solutions already available like Hadoop[7] (which we use for the implementation of this framework), but also because the generation of a hypercube fits perfectly into the MapReduce paradigm. This is not necessarily true for other parallel computing paradigms such as Grid computing, where the processing is efficiently distributed but the results are cumbersome to aggregate afterwards, or MPI[8], where the developer has to take care of the intrinsic problems of a distributed system. In the case of a parallel DBMS, these two simple examples could be easily created either by using UDF with external HEALPix libraries (something already done for *Microsoft SQL Server* at the SDSS) or directly with a SQL query for the theoretical Hertzsprung-Russell diagram. However, more complex histograms or hypercubes would be much more difficult to generate. Furthermore, the scalability will be better in Hadoop as the data set grows due to the inherent model of MapReduce. Last but not least, the generation of several histograms and/or hypercubes each one with different constraints in terms of filtering or aggregation (e.g. several star density maps at different N_{side} granularities) will be more efficient using this framework on top of Hadoop (provided the amount of data is very large) as they will be computed in one single scan of the data set.

[7] http://hadoop.apache.org
[8] www.mpi-forum.org

4.3.2 Framework Description

The framework (implemented in *Java*) has been conceived considering the following features:

- Thin layer on top of Hadoop that allows users or external tools to focus only on the definition of the hypercubes to compute.
- Hide all the complexity of this novel computing paradigm and the distributed system on which it runs. Therefore, it provides a way to deal with a cutting-edge distributed system (Hadoop) without any knowledge of Big Data internals.
- Possibility to process as many hypercubes as possible in one single scan of data, taking advantage of the brute-force approach used in Hadoop jobs, thus reducing the time for generating the precomputed statistics required for each data release.
- Leverage the capabilities offered by this new computing model so that the solution is scalable.
- *Java generics* have been used throughout the framework in order to ease its integration in any domain and permit a straightforward embedding of any already existing source code.

Hypercubes defined in the framework (see Figure 4.10 for an example) may have as many dimensions (categories) as required. They may define intervals (for a continuous function) or be of a discrete type (for discrete functions), depending on the use case. Users may supply their own algorithms in order to specify how the value of each dimension will be computed (by implementing the corresponding *getField* method). This value might be of a custom user-defined type in case of need. The input object being processed at each time is obviously available for performing the relevant calculations.

It is important to highlight that the possibility of defining as many dimensions as needed is what allows us to easily build data mining hypercubes or concrete pivot tables, and do so on-the-fly (at runtime) without losing generality. This is a key feature for scientific data analysis because the data is always exploited in many different ways due to the diversity of research studies that can be done with them. Furthermore, the cubes generated may be further analyzed (i.e. slice, dice, drill-down and pivoting operations) within the framework by defining a new Hadoop input format that reads the output of the cube generation job and delivers it to the next analytical job. The results can also be exported to a database in order to perform the subsequent analysis in there.

We must also set the value that will be returned for each entry being analyzed. This will usually be a value of '1' when performing e.g. counts of objects falling into each combination of categories, but it might also be any other derived (user-implemented) quantity for which we want to know the maximum or minimum value, the average, the standard deviation, or any other linear statistical value. There is only one MapReduce phase for the jobs so more complex statistics cannot currently be calculated, at least not efficiently and in a scalable way (e.g. the median, quartiles and the like). We may however

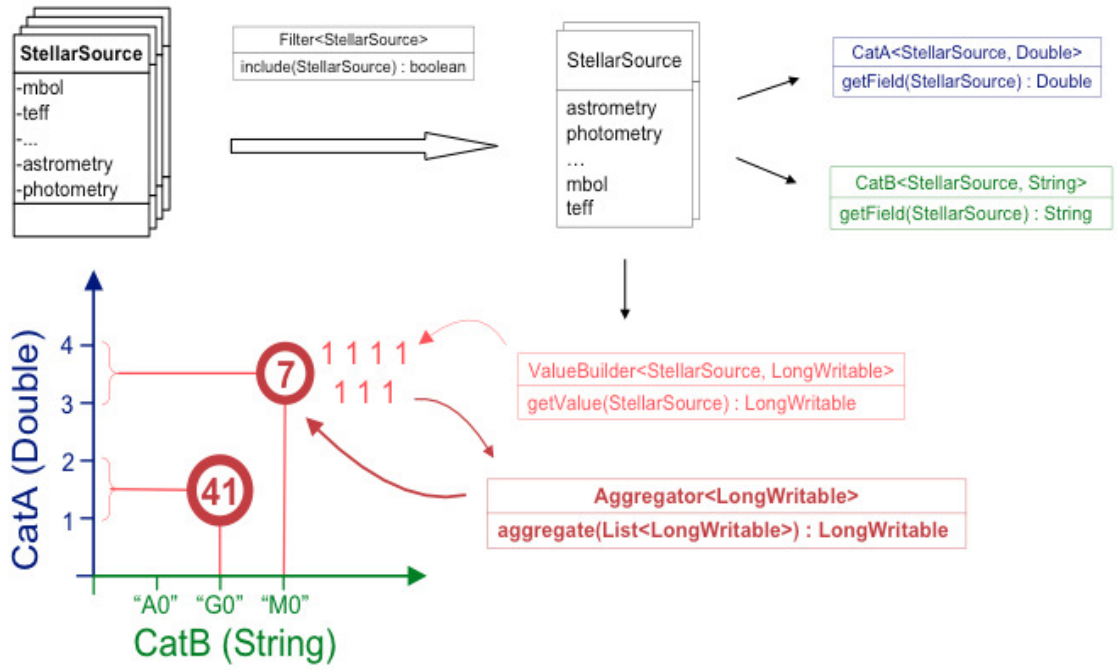


Figure 4.10: Data workflow through the framework and main interfaces to implement for each hypercube. The sample in the figure shows a hypercube with two dimensions (discrete for the x axis and continuous with intervals for the y axis) that counts the number of elements falling into each combination of the categories.

define a custom type such that the cells of the hypercube contain more information than just a determined measure. We can also define a filter which is used to decide which input objects will be analyzed and which ones will be discarded for each hypercube included in the job.

The current implementation offers a lot of helpers that can be plugged in many different places for many different purposes. For instance, if we just want to get a field out of the input object being processed for a certain dimension (or for the value returned), we can just use a helper reader that obtains that field at runtime through *Java reflection*, and avoid the generation of a new class whose only method would just return the field. The user just needs to specify the field name and make sure that the object provides the relevant accessor (*getter* method).

Last but not least, the manner in which the data is aggregated as well as whether the aggregator can be used as the Hadoop combiner for the job (recommended whenever possible (Kwon et al., 2011)) can also be defined by the user (and could also be defined per hypercube with minor changes), although most of the time they will just set one of the currently available helpers (for computing counts, the minimum/maximum value, the average, the standard deviation, etc). For more complex hypercubes (i.e. several measures aggregated differently on each cell of the hypercube), we could create an aggregator along with a custom type for cell values so that the different measures are aggregated differently (the count for some of them, the average for others, etc).

Before running the job it is required to set some configuration properties for the definition of the input files and the corresponding input format to use, the path where to leave the results, the class that will define the hypercubes to create (Listing 4.2 shows the skeleton of this class for the example shown in Figure 4.10), and some other parameters like the type of the output value returned for them (mandatory for any Hadoop job). This last constraint forces all entities computed in the same job to have the same return type (the reducer output value, e.g. the count, the maximum value, etc), although this can be easily worked around if needed by setting more generic types (a *Double* for holding both integers and floating point numbers, a *String* for numbers and text, etc), or as stated above, developing a custom type (implementing the Hadoop *Writable* interface) that holds them in different fields, along with the corresponding aggregator.

Listing 4.2: Custom class defining the hypercube(s) to compute.

```
public class MyHypercubeBuilder
extends BuilderHelper<StellarSource,LongWritable> {

  @Override
  public List<Hypercube<StellarSource,LongWritable>>
  getHypercubes() {
    // Create list holding the hypercubes
    // Create CatA instance (with ranges)
    // Create CatB instance
    // Create Filter instance
    // Create ValueBuilder (use Helper)
    // Create Aggregator (use Helper)
    // Create Hypercube instance
```

```
    // Add to hypercubes list
    // Return hypercubes list
  }
}
```

The output files of the job have two columns, the first one for identifying the hypercube name as well as the combination of the concrete values for its dimensions (split by a separator defined by the user), and the second one holding the actual value of that combination of categories (see Listing 4.3 for a sample of the output for the Theoretical Hertzsprung-Russell diagram shown in Figure 4.9). The types used for discrete categories must provide a method to return a string which unequivocally identifies each of the possible values of the dimension. For categories with intervals, the string in the output file will contain information on the interval itself with square brackets and parentheses as appropriate (closed and open ends respectively), but again they must ensure that the types of the interval ends (bin ends) supply a unequivocal string representation. This unequivocal representation might be the primary key of the dimension's concrete value (for more advanced hypercubes) so that it can later on be joined with the rest of the information of that dimension as it usually happens in data mining *star* schemas.

Listing 4.3: Sample of the output for the Theoretical Hertzsprung-Russell diagram shown in Figure 4.9.

```
[...]
TheoreticalHR/[3.58, 3.5825)/[16.7,16.725)/     998
TheoreticalHR/[3.58, 3.5825)/[8.0,8.025)/       883
TheoreticalHR/[3.5875, 3.59)/[-4.875,-4.85)/    328
TheoreticalHR/[3.5875, 3.59)/[-5.4,-5.375)/     391
TheoreticalHR/[3.6075, 3.61)/[-0.9,-0.875)/   87031
TheoreticalHR/[3.6075, 3.61)/[-3.6,-3.575)/    2780
TheoreticalHR/[3.6075, 3.61)/[-3.925,-3.9)/   12384
[...]
```

One straightforward but important optimization that has been implemented is the usage of sorted lists for dimensions that define continuous, non-overlapping intervals. This way the number of comparisons to do per input object is considerably lowered, reducing by a factor of 20 the time taken for the execution. Therefore, although non-continuous (and non-ordered) interval categories are allowed in the framework, it is strongly recommended to define continuous (and non-overlapping) ranges even though some of them may be later on discarded.

4.3.3 Cloud Deployment

Recently there has been a blossoming of commercial Cloud computing service providers, for example AWS, Google Compute Engine, Rackspace Cloud, Microsoft Azure and several other companies or products sometimes focused on different needs (Dropbox, Google Drive, etc). AWS has become one of the main actors in this Cloud market, offering a wide range of services such as the ones that have been used for this work: Amazon

Elastic MapReduce (EMR)[9]), S3[10]) and EC2[11]). The way these three services are used is as follows: EC2 provides the computers that will run the work flows, S3 is the data store where to take the data and leave the results, and EMR is the Hadoop ad-hoc deployment (and configuration) provided for MapReduce jobs. Amazon charges for each service, and not only for the computing resources but also for the storage on S3 and the data transfers in and out of their infrastructure. They also provide different instances (on-demand, reserved and spot instances) which obviously have different prices at different levels of availability and service. The EMR Hadoop configuration is based on the current operational version of Hadoop with some bug fixes included. Furthermore, the overall experience with EMR is very good and it has been quite easy to start submitting jobs to it through command line tools openly available. Debugging is also quite easy to do as *ssh* access is provided for the whole cluster of nodes.

The deployment used for testing and benchmarking consists of eight worker nodes each one having the Hadoop data and task tracker nodes running on them. There is also one master node which runs the name node and the job tracker. The AWS instance chosen is *m1.xlarge*, which has the following features:

- 4 virtual cores (64-bit platform).
- 15 GB of memory.
- High I/O performance profile (1 Gbps).
- 1690 GB of local Direct Attached Storage (DAS), which sums up to a bit more than 13 TB of raw storage which may be cut down by half or more depending on the HDFS (Shvachko et al., 2010) replication factor chosen.

With this layout, EMR Hadoop deployment launches a maximum number of 8 mappers and 3 reducers per worker node (64 and 24 for the entire cluster respectively). It is important to remark that the time taken for starting the tasks of a job in Hadoop is not negligible (more than one minute for the tests carried out) and certainly affects the performance of short jobs (Pavlo et al., 2009), as it imposes a minimum amount of time that a job will always last (sequential workload) no matter the amount of data to process. This is one of the reasons why Hadoop is mostly advised for very big workloads (Big Data), where this effect can just be disregarded.

4.3.4 Data Storage Model Considerations

Scientific raw data sets are not normally delivered in a uniformly sized set of files as the parameters chosen for placing the data produce a lot of skew due to features inherent to the data collection process (some areas of the sky are more densely populated, a determined event does not occur at regular intervals, etc). This is also true for the data

[9] http://aws.amazon.com/elasticmapreduce/
[10] http://aws.amazon.com/s3/
[11] http://aws.amazon.com/ec2/

set being analyzed with this framework (GUMS version 10) as it comprises a set of files each one holding the sources of the corresponding equal-area sky region (see Figure 4.8 for its histogram drawn in a sky projection). This may be a problem for binary (and often compressed) files when stored in HDFS as the records cannot be split into blocks (there is no delimiter as in the text format). The data formats studied in this paper are of this type (binary and compressed with no delimiters), defined by the Gaia mission SOC. Thus we have to read each of them sequentially in one Hadoop mapper and their size must be roughly the same and equal to the defined HDFS block size to maximize performance through data locality. Figure 4.11 shows the performance obtained when computing the different histograms shown in Figures 4.8 and 4.9: a HEALPix density map and a theoretical Hertzsprung-Russell diagram. As we can see, once we group the data into equally sized files and set the HDFS block to that size, the time consumed for generating them is approximately 2/3 the time taken when the original highly-skewed delivery is used. The standard format chosen for data deliveries within the Gaia mission is called 'GBIN', which contains *Java*-serialized binary objects compressed with Deflate (ZLIB).

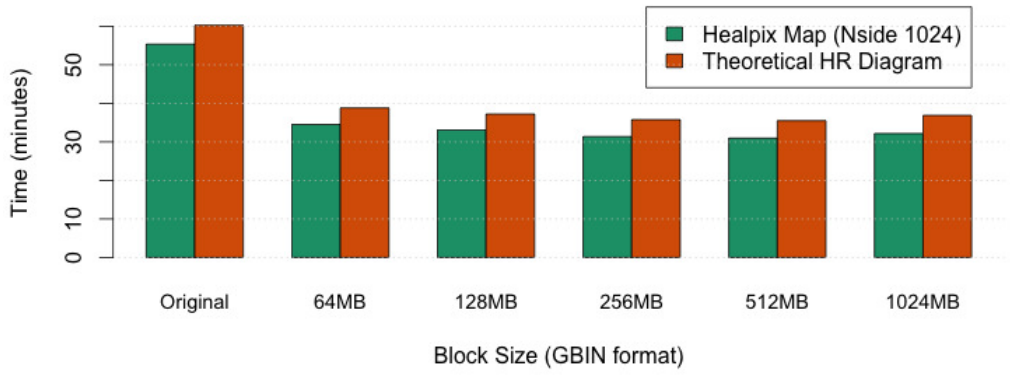


Figure 4.11: Performance for different HDFS block and file sizes (files in GBIN format are binary and compressed with Deflate).

In Figure 4.11 we can also see that there is a block size which performs slightly better than the others (512 MB) which is a consequence of the concrete configuration used for the testbed, as more files mean more tasks (Hadoop mappers) being started which is known to be slow in Hadoop as already remarked above. Furthermore, less but bigger files may produce a slowdown in the data shuffling period (each Hadoop mapper outputs more data which then has to be combined and shuffled). Therefore, we will use the best configuration (data files and block size of 512 MB) for the comparison with other data storage techniques and for benchmarking.

To analyze the effects of the different compression techniques available and study

how they perform for an astronomical data set, a generic data input format has been developed. This way, different compression algorithms and techniques may be plugged into Hadoop, again without dealing with any Hadoop internals (input formats and record readers). This is more or less the same idea as the generic input format interface provided by Hadoop but more focused on binary (non-splittable) and compressed data. The data reader to use for the job must be configured through a property and its implementation must provide operations for setting up and closing the input stream to use for reading, and for iterating through the data objects. The readers developed so far store *Java* serialized binary objects with different compression techniques which are indicated below as: GBIN for Deflate (ZLIB), Snappy for Google Snappy[12] compression and Plain for no compression.

Figure 4.12 shows the results obtained when these compression techniques are used with the GUMS version 10 data set for creating the same histograms as before. It is important to remark that no attention has been paid to other popular serialization formats currently available like Thrift[13], Avro[14] etc, as the time to (de)serialize is always negligible compared to the (de)compression one. Furthermore, as stated above, the data is always stored in binary format as the textual counterpart would lead to much worse results (a proof of this is the battery of tests presented in Pavlo et al. (2009)).

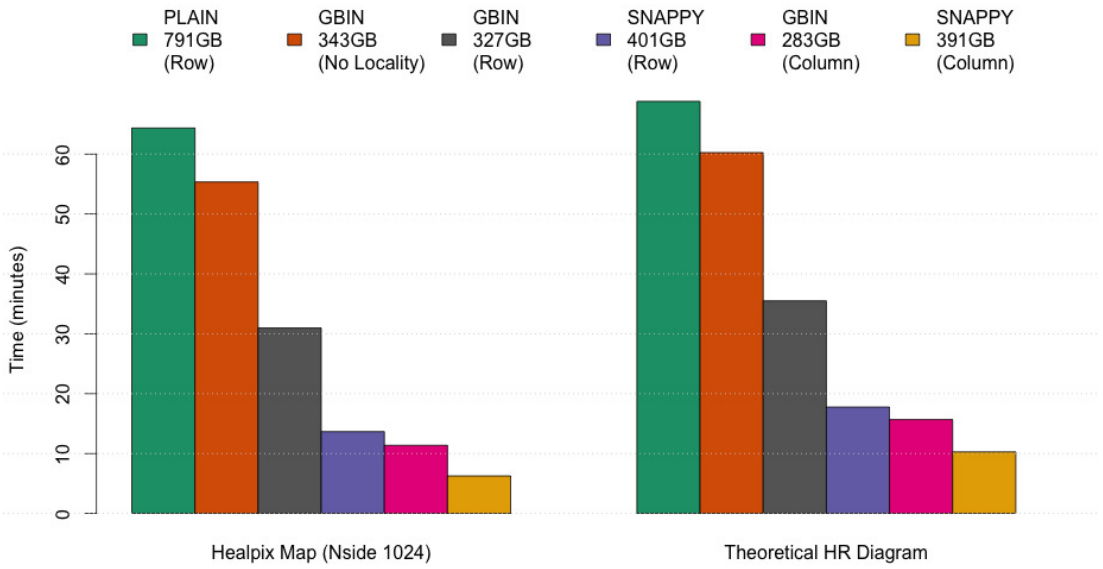


Figure 4.12: Data storage model approaches performance comparison.

[12] http://code.google.com/p/snappy/
[13] http://thrift.apache.org/
[14] http://avro.apache.org/

Google Snappy codec gives a much better result as the decompression is faster than Deflate (GBIN). It takes half of the time to process the histograms (50%) and the extra size occupied on disk is only around 23% (see Figure 4.13). This confirms the suitability of this codec for data to be stored in HDFS and later on analyzed by Hadoop MapReduce work flows.

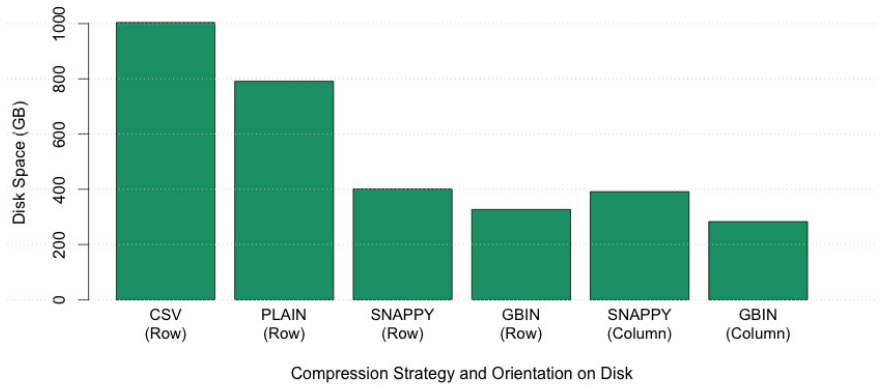


Figure 4.13: Data set size for different compression and format approaches.

Figure 4.12 also shows the performance obtained with another data storage model developed ad-hoc using a new Hadoop input format, column-oriented (see the rightmost two columns in both histograms), which resembles the one presented in Floratou et al. (2011), although it integrates better in the client code as the objects returned are of the relevant type (the information that we want to populate from disk still has to be statically specified though as in Floratou et al. (2011)). Furthermore, we obviously expect that the improvements made by the column input format are more significant as the data set grows larger (both in number of rows and columns), although we can also state that performance will decrease when most of the data set columns are required in a job, due to overheads incurred in the column-oriented store mechanism (several readers used at the same time, etc). These issues have to be carefully considered for each particular use case before any of the formats is chosen.

The Hadoop operational version at the time of performing these tests did not yet provide a way to modify the block data placement policy when importing data into HDFS (newer alpha/beta versions did support this to some extent although these could not be used in EMR). The current algorithm for deciding what data node is used (whenever an input stream is opened for a certain file), chooses the local node if there is a replica in there, then another random node in the same rack (containing a replica) if it exists, and if there is no one serving that block in the same rack it randomly chooses another data node in an external rack (containing a replica of course). Considering this algorithm, if we set the replication policy to the number of cluster worker nodes, we ensure that there

will always be a local replica of everything on every node and thus we can simulate that the column files corresponding to the same data objects have been placed in the same data node (and replicas) for data locality of input data readers. This is never to be used in an operational deployment of course, but it has served its purpose in this framework's study. Meanwhile, new techniques that overcome these issues were being put in place (i.e. embed data for all columns in the same file, and split by row ranges).

These new techniques, whose main implementations are Parquet and ORC as described in Section 3.3, should always be chosen instead of ad-hoc developments like the one presented here. This ad-hoc development is put forward just for reference, as again, at the time of running these experiments none of the current main implementations were available.

Figure 4.13 shows that the level of compression achieved by the naive implementation of the column-oriented approach (compared to the row-oriented counterpart) is not as good as it might be expected. This may be caused by the fact that an entire row may much resemble the next row (similar physical properties), so the whole row may be considered a column, but at a higher granularity, leading to a relatively good compression in the row-oriented storage model as well. Another more plausible explanation for this may be that we do not use deltas for adjacent data in columns as is usual (Krueger et al., 2010), but the values themselves. Last but not least, there is a variety of other features that are currently embedded in the existing implementations, i.e. Parquet and ORC, (see Section 3.3).

Contrasting the results in Figures 4.12 and 4.13, we see that the column-oriented storage model (using Snappy codec) takes more disk space than the row-oriented one (with Deflate). This extra cost overhead (64 GB) amounts to $8 per month in Amazon S3 storage (where the data are taken from at cluster initialization time), which is much less than the price incurred in a typical workload where many histograms and hypercubes have to be computed, as e.g. the cluster must be up 24 minutes more in the case of a HEALPix density map computation (which comes to a bit more than $3 extra per job) or 25 minutes more for a single theoretical Hertzsprung-Russell diagram (again a bit more than $3 extra per job). Therefore, for typical larger workloads of several (and more complex) histograms and/or hypercubes, we can expect larger and larger cost savings with the column-oriented approach as computation is much more expensive than the extra overhead in S3 storage, mainly due to the amount of jobs to be run as well as the non-negligible cost of the cluster nodes.

4.3.5 Benchmarking

Two powerful and well-known products have been chosen for the benchmark, Pig[15] 0.11.0 and Hive[16] 0.10.0. These open source frameworks, which also run on top of Hadoop, offer an abstraction of the MapReduce model, providing users with a general purpose, high-level language that could be used not only for the hypercubes described above, but also

[15] http://pig.apache.org/
[16] http://hive.apache.org/

for other data processing workflows such as ETL. However, they might require some further work to do in case we wanted to build more complex hypercubes (involving several dimensions and values, some of them computed with already existing custom code), or generating several hypercubes in one scan of the input data (something not neatly expressed in a SQL query).

The tests carried out use a row-oriented scheme with compressed (Snappy) binary data and have been run on the infrastructure described in Section 4.3.3. For the purpose of benchmarking, we will carefully analyze different scenarios:

- Simple one-dimensional hypercube with the key encoded as text (the default for the framework), and as binary (more efficient but less flexible).
- Two-dimensional hypercube with low key cardinality where the aggregation factor is high.
- Several hypercubes at the same time with different key cardinalities.
- Different aggregation algorithms (default Merge Sort and Hash-based).
- Scalability tests by increasing the dataset size, but keeping the same cardinalities for the keys.

Figure 4.14 shows the results obtained when generating the hypercube plotted in Figure 4.8. Two approaches (with the key encoded as text and as binary) have been considered in order to prove that the proportions in execution time for the different solutions are kept, although the framework is supposed to always work with text in its current version. We can see that the framework performs considerably better (33% in the case of Pig and up to 40% for Hive) and we argue that this may be due to its simplicity in the design yet the efficient core logic built inside, which cannot be achieved by general purpose frameworks that are supposed to span a very wide domain of applications. Therefore, this generality has a high impact in cost when we focus on a particular use case like the one described here.

The results shown in Figure 4.15 refer to the scalability of the alternatives studied for different computations and configurations. The same dataset (GUMS version 10) is used to enlarge the input size, although it is important to notice that the output size will remain the same as the number of bins will not change as we increase the input data.

We can see that the framework performs significantly better in the use cases studied, which proves that for well-known, operational workloads, it is usually better to use a custom implementation (or an ad-hoc framework) rather than using general purpose tools which are more suited for exploration or situations where performance is not so important. However, we can be certain that these general purpose and higher level implementations are catching up fast enough if we look at the optimizations they are currently releasing, such as hash-based aggregation. This technique (known as *In-Mapper* combiner) tries to avoid data serialization and disk I/O by aggregating data in a hash table in memory. The mappers that run in the same Java Virtual Machine (JVM) do

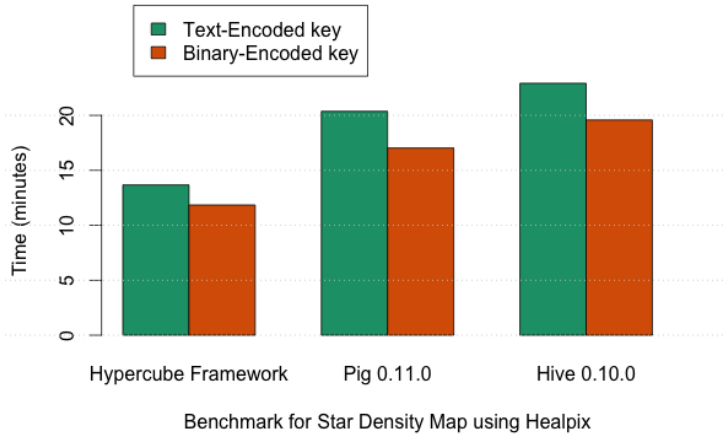
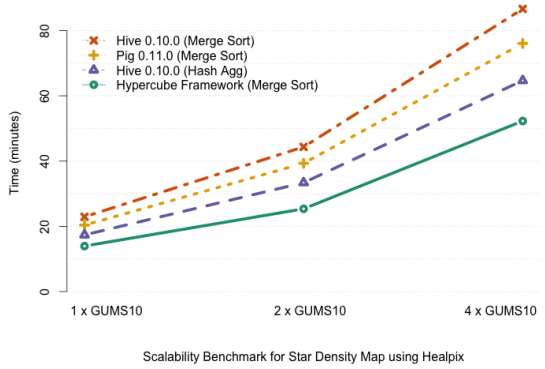


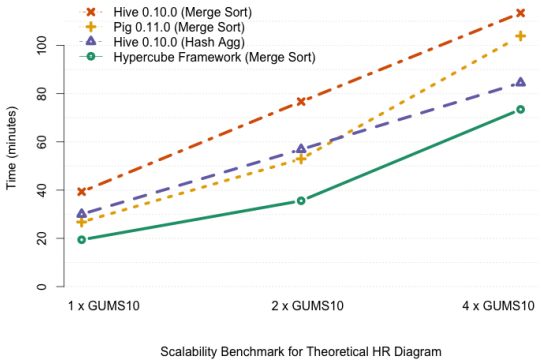
Figure 4.14: Comparison among the framework presented and other popular data analysis tools currently available. All tests have been run using the Hadoop standard Merge-Sort algorithm for data aggregation.

not emit data until they are all finished (as long as the aggregation ratio is high enough). The steps for serializing data and the associated disk I/O before the combiner is executed are not needed anymore, thus improving performance dramatically. The logic built-in for accomplishing this new functionality is rather complex not only because Hadoop was not designed for this kind of processing in the first place, but also due to the dynamic nature of the implementations, which can switch on-the-fly between hash-based and merge-sort aggregations by spilling to disk what is inside the hash table once a certain configured aggregation threshold is not met by the workflow at run time. The implementation of this automatic switching is something that will make these higher level tools much more efficient, but it will also require much more expertise from users for tuning the best configuration for the workflows.

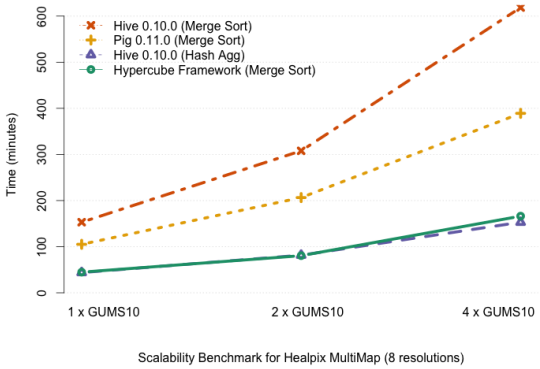
Furthermore, the current implementation for Hive shows a very good performance gain as shown in Figure 4.15 (c) but is not significantly better than the merge-sort counterpart when using Hadoop directly. This may be caused by the fact that there is not much I/O due to the small size of each pair of keys and values, compared to the savings produced for a better in-memory aggregation. Results for hash-based aggregation in Pig have been omitted due to its very poor performance in the release used, which proves that a more robust implementation must properly handle the memory consumed by the hash table, allowing to switch to Merge Sort dynamically whenever the cardinality goes beyond a predefined threshold. This dynamism in query execution may become an asset for Hadoop-based processing comparing to parallel DBMS, where the query planner picks one alternative at query parsing time and usually sticks to it till the end. In Hadoop, this is more dynamic and gives more flexibility and adaptability at run time.



(a) Star density map using Healpix (one dimension).



(b) Theoretical HR diagram (two dimensions).



(c) Star density map using Healpix at eight different resolutions (one dimension, eight different hypercubes in the same run).

Figure 4.15: Scalability benchmark for (a) star density map, (b) theoretical Hertzsprung-Russell diagram and (c) star density map at eight different resolutions. The approaches shown encompass different alternatives from the Hadoop ecosystem and the two main algorithms used for aggregating data, i.e. Merge Sort (the default for Hadoop) and Hash Aggregation (whose implementation is known in the Hadoop ecosystem as *In-Mapper* combiner). The results for Hash aggregation are only shown for Hive, which is the only one that showed some improvements in the tests run. The dataset is enlarged one, two and four times with the same data (1xGUMS10, 2xGUMS10 and 4xGUMS10 respectively) and therefore the cardinality of the key space for each hypercube being computed remains unchanged.

One of the most common features of data pipelines is that they are often DAG and not linear pipelines. However, SQL focuses on queries that produce a single resultset. Thus, SQL handles trees such as joins naturally, but has no built in mechanism for splitting a data processing stream and applying different operators to each sub-stream. It is not uncommon to find use cases that need to read one data set in a pipeline and group it by multiple different grouping keys and store each as separate output. Since disk reads and writes (both scan time and intermediate results) usually dominate processing of large data sets, reducing the number of times data must be written to and read from disk is crucial to good performance.

The inclusion of the *GROUPING SETS* clause in Hive has also contributed to the improvements shown in Figure 4.15 (c), comparing to those in Figure 4.15 (a) and (b), as it allows that the aggregation is made with different keys (the ones specified in the clause) yet only one scan of data is needed. This fits perfectly in the scenario posed in the test shown in Figure 4.15 (c), where we compute several hypercubes at the same time in the same dataset. However, *GROUPING SETS* clause is complex since the keys specified have to be in separate columns. Therefore, when trying to compute results in the format of a single key plus its corresponding value, we will always get the key which the row refers to, plus the rest of keys with empty values (*null*).

We have found other usability issues in Hive, which we believe will be addressed soon, but which may currently lead to a worse user experience, such as the lack of aliases on columns. This is a minor problem, but most of the users and client applications are used to relying on them everywhere for reducing complexity or increasing flexibility, overall when applying custom UDF or other built-in operators. Furthermore, there is no way to pass parameters to the UDF when initializing, which has made the execution of the multi-hypercube workflow more difficult to run, as several different UDF had to be coded, even though they all share the same functionality and the only difference is the parameter that sets the resolution of the map.

Comparing Pig and Hive, results show that Pig performs significantly better than Hive when the (Hadoop) standard Merge Sort algorithm is chosen. However, the implementation of the hash-based aggregation in Hive seems more mature and gives a better performance than the Merge Sort alternative in Hive, for those cases where the aggregation factor is high enough (see the tendency of Figure 4.15 (b) where the aggregation factor is increased as we enlarge the dataset due to the same data being duplicated).

One of the most important conclusions to take away when looking at these results is that there is no solution that fits all problems. Therefore, special care has to be taken when choosing the product to use as well as when setting the algorithm and tuning its parameters. Results show that a bad decision may even double execution time for certain workloads. In this case, an ad-hoc solution fits better than more generic ones, even though already implemented hash-based algorithms in Hive and Pig may seem more appropriate upfront. In addition, there are other optimizations that could be easily made, such as sort avoidance, because it is normally not needed when processing aggregation workflows.

Another remark worth mentioning is that when we double the input size, the exe-

cution time is a bit less than the expected (double) one. This is due to the fact that Hadoop inherent overhead starting jobs is compensated by the larger workload, which proves that Hadoop is not well suited for small datasets as there is a non-negligible (and well-known) latency starting tasks in the worker nodes.

4.3.6 User Experience

The hypercube generation framework is packaged as a Java Archive (JAR) file and has a few dependencies on other packages (mainly on those of Hadoop distribution). To make use of the framework, the user has to write some code that sets what hypercubes to compute (see Listing 4.2), as well as any extra code that will be executed by the framework, e.g. when computing the concrete categories or values for each hypercube and input record in the dataset being processed. Furthermore, the user is expected to package all classes into a JAR, create a file containing (at least) the properties specified in Section 4.3.2 (input and output paths, etc), and run that JAR on a Hadoop cluster following Hadoop documentation.

The framework has been tested by offering it to a variety of user groups. One of the early users, without any background in computer science (but with experience in *Java* programming), tried out the framework as it was being developed. He had no significant problems in understanding how to write pieces of *Java* code needed to generate hypercubes for specific categories or intervals and was able to quickly write a small set of classes for supporting the production of hypercubes for quantities (e.g. energy and angular momentum of stars) that involve significant manipulation of the basic catalogue quantities. How to write additional filters based on these quantities was also straightforward to comprehend.

Subsequently the framework (together with the small set of additional classes) was offered to students attending a school on the science and techniques of Gaia. During this school the students were asked to produce a variety of hypercubes (such as the ones in Figures 4.8 and 4.9) based on a subset of the GUMS version 10 data set. The aim was to give the attendants a feel for working with data sets corresponding to the *Big Data* case. The programming experience of this audience (mostly starting PhD students) ranged from almost non-existent, to experience with procedural and scripting languages, to very proficient in *Java*. Hence, although the conceptual parts of the framework were not difficult to understand for the students (what is a hypercube, what is filtering, etc), the lack of knowledge in both the *Java* programming language, and in its philosophy and methods proved to be a significant barrier in using the framework.

Widespread opinions were for instance: "Given the *Java* programming language learning curve the framework is a huge amount of work for short term studies but is very useful for long term and more complex studies" and "*Java* needs a complete change of mind with respect to the way we are used to programming". The framework was also offered at a workshop on simulating the Gaia catalogue data and there the attendants consisted of a mix of junior and senior astronomers. The reactions to the use of the framework were largely the same.

On balance we believe that once the language barrier is overcome the framework

provides a very flexible tool to work with. The way of obtaining the data and the fact that the user may choose the treatment of these data, enables a wide range of possibilities with regard to scientific studies based on the data.

The fact that it operates under a Hadoop system makes it an efficient way of serving data analysis of huge amounts of data with respect to conventional database systems, due to the nature of the requests presented by the users in the seminars: "They must be completely customizable for any statistics or studies a scientist wanted to develop with the source data".

Another interesting point is the way the data is presented on output. It can be parsed with any data mining software that can represent graphical statistical data due to its simple representation, and it can also be understood by the users themselves without major issues.

4.4 Towards Scalable and Unified Architectures

Sections 4.2 and 4.3 discuss two specialized approaches for relatively similar challenges. If we allow some tradeoffs, there would be no need to roll out and maintain two infrastructures and so a bigger one could be put in place, coping with both types of pipelines. Something like a data lake with Apache Spark on top, and if low-latency queries are strictly needed, also with the latest versions of Apache Hive or Cloudera Impala. However, the data stays in the same data lake, and the computing resources are shared among the different engines and tools released to the community (including developers and modelers). For low-latency queries, it would be needed to perform sequential scans, but both by bringing the full catalogue to memory and having a larger infrastructure (which can scale up nicely and cheaply), this tradeoff is mitigated.

Furthermore, another advantage is the possibility of embedding some code in the SQL declarative language in an easy way, something needed both to implement higher level access tools and services like TAP. In addition, there is no need to ingest data back and forward for exposing it in the different engines. Whenever something is ingested in the platform, it automatically becomes visible to any tool and service built on top.

This consolidated approach turns out to be more cost efficient and thus it is expected to see a convergence towards these scalable and unified architectures that gravitate around the data lake as more and more pressure is being put on publicly funded scientific ventures and especially those that require significant investments like any spatial mission. In addition, the return on investment is difficult to measure, and it takes some time to realize about its value and importance, which makes it even harder.

In this scenario, operationally efficient solutions become crucial. There are significant efforts worldwide to come up with cheaper ways of launching payloads to space. There are also efforts to optimize further the way ground segments operate, especially those related to scientific data exploitation, starting in the SOC all the way down through the data archiving and exploitation. In today's typical pipeline, a similar concept of the Lambda architecture is leveraged. The pipelines that process the scientific data sets and telemetries coming from the satellite in (near-)real time are run on different

infrastructures than those where the results get dumped afterwards. For instance, Gaia SOC software has been open sourced, but this is not enough, as the systems where it runs cannot be easily replicated. This may prevent scientists from processing the raw data with that software, tuning specific areas of their interest, and making it all run smoothly in the same architecture.

In this matter, there is a clear opportunity to consolidate, make the software and pipelines more reusable inter and intra missions, and share the infrastructure, the engines and techniques on top across different use cases. Current technology allows this in a way in which data sets keep on evolving but stay backwards compatible (e.g. as many of the file formats available today for the data lake like Apache Avro, Apache Parquet and so on). This will also help preserve the archived data, will lead to faster development cycles, will make it easier to deploy and expose new data sets and will allow scientists to reuse and/or exchange not only data but also their models and any other processing performed over it.

4.5 The Science Enabling Applications Work Package

The data access and analysis tools are being developed in the frame of the Science Enabling Applications Work Package (see Figure 4.2). Some existing astronomical data querying and exploration tools will be adapted and integrated into the archive infrastructure, but also new data tools, frameworks and models will be developed as a way of enabling the community to work with the archive. The models put forward for the Grand Challenge (see Chapter 5) are one example that serves the purpose of both enabling science through a well known problem in astronomy and showing appropriate ways in which other similar ones can be developed for other problems. All these tools will obviously put requirements on the archive framework itself and vice versa, will evolve over time depending on the trends in Big Data, and will require a close tailoring to the archive framework to ensure its performance and usability.

The tasks performed in this work package are divided into four sub-work packages presented below. They will serve as a reference that helps position the research performed in the Grand Challenge (see Chapter 5) in the wider initiative of the full Science Enabling Applications Work Package.

Advanced data access tools The goal of this sub-work package is to deliver client-side applications to facilitate querying in the Gaia archive. This category of applications addresses the need for advanced tools to access the massive amounts of data contained in the repository. These applications fall into categories like:

- Development of VO tools and adoption of its standards for data querying such as the Simple Spectral Access Protocol (SSAP)[17] and TAP (Dowler et al., 2010). In addition, the development of user defined functions in ADQL (Osuna et al., 2008) may be necessary to ease the execution of Gaia specific use cases.

[17] http://www.ivoa.net/documents/SSA/

- Existing VO-Tools such as TOPCAT (Taylor, 2005) will be adapted to serve Gaia data.
- Integration of the Gaia archive into services like Aladin (Boch and Fernique, 2014) and VizieR (Ochsenbein et al., 2000) will allow to select and download excerpts of the Gaia catalogue into the Aladin sky atlas, and display it as individual sources over a large palette of reference surveys like Digitized Sky Survey (DSS)[18], SDSS[19], the Two Micron All Sky Survey (2MASS)[20] and Galaxy Evolution Explorer (GALEX)[21].
- Extending query functionalities in RDBMS for astrometric and astronomical purposes. Providing fast access to the 6-dimensional phase space of Gaia enhancing visualisation capabilities of large data volumes.
- Non-interactive tools for accessing and downloading Gaia data for users with their own tools.
- Advanced Gaia data navigation and exploration by means of searching through clustering will be developed. Precomputed clusters, obtained by means of a measure of similarity, can help researchers to find what they are looking for, since they will be able to explore the data quickly.

Data Mining The application of data mining algorithms is essential for a full scientific exploitation of the data archive. The benefits of astronomical archives with data mining enabling capabilities in general (and specifically for the Gaia mission archive) have been extensively discussed in the scientific community (Sarro et al., 2014). The application of such techniques will reveal patterns and relationships within the astronomical data that can lead to the detection of new types or exotic objects that represent rapid stages of stellar evolution and/or new astrophysical scenarios. The data mining framework to apply these techniques and methodologies will be covered in this sub-work package.

The Gaia archive will be available using both traditional architectures for storing data such as RDBMS as mentioned in Section 4.1, but also with parallel ones like those based on distributed file systems (leveraging column-oriented file formats like Parquet or ORC as discussed above). Tools to ensure data synchronization between different archiving systems are being developed as well.

Feature wise, the framework will provide a set of data mining techniques classified as follows:

- Dimensionality Reduction: Feature selection and feature extraction methodologies.
- Supervised classification and regression techniques.
- Unsupervised classification and clustering techniques.

[18] http://archive.eso.org/dss/dss
[19] http://www.sdss.org/
[20] http://www.ipac.caltech.edu/2mass/
[21] http://www.galex.caltech.edu/

- Model evaluation.
- Advanced and tailored techniques. These are already existing algorithms, such as HMAC (Li et al., 2007), or the MCMC samplers (Gelman et al., 2013) used in Chapter 5, that prove to be useful for certain astronomical use cases, and that will be integrated as well in the data mining framework.

From the technological perspective, the Gaia data mining platform is based on the latest state-of-the-art Big Data technologies remarked in Chapter 3. Hadoop framework as the Big Data infrastructure and Apache Spark as the general purpose distributed engine:

- Apache Hadoop framework comprising HDFS, Yet Another Resource Negotiator (YARN) for job scheduling and cluster resource management, and common libraries, utilities and engines for supporting different pipelines.
- Apache Spark for large-scale data processing (see Section 3.4.1). Apache Spark includes (as already depicted) additional modules with capabilities in Big Data analytics and machine learning. The most relevant ones for the Data Mining subwork package are obviously MLlib for scalable machine learning and Spark SQL for structured data and SQL-like queries and operations on distributed data.
- Other services such as web notebooks for interactive data science and scientific computing will be offered to the community.

The Gaia data lake is based on an HDFS data repository with read-only Gaia data products plus several additional catalogues, which will be fully synchronized with the rest of archive repositories such as those using RDBMS. In addition, it will also be possible to publish intermediate results from data processing workflows and models, so that they are available to the community as well (implementing the concept of the *living data archive*).

A collaborative documentation platform based on recipes will be offered as well, enlarging the scientific use cases and the re-usability of the techniques and methodologies (contributing to extend the semantic layer). This obviously applies to the Grand Challenge (see Chapter 5), as a representative example of this approach.

Cross-matching The Gaia astrometric, spectrophotometric and radial velocity data will empower high impact science in a wide range of research fields. However, many scientific questions will be best addressed through the combination of Gaia data with existing and forthcoming surveys such as 2MASS[20], VISTA[22], Hipparcos-2 (van Leeuwen, 2007), Tycho-2 (Høg et al., 2000) and the LSST[23] among many others.

Although such external data are not taken into account for the production of the Gaia catalogue, the Gaia mission does use external data for the calibration of the data and for

[22]https://www.eso.org/public/unitedkingdom/teles-instr/surveytelescopes/vista/surveys/
[23]http://www.lsst.org/

training of some of its algorithms. Mechanisms for source matching between catalogues are also being addressed in this work package.

The cross-matching of astronomical catalogues is a complex and challenging problem both scientifically and technologically. The higher angular resolution in Gaia (with respect to other existing catalogues), requires many-to-one algorithms as more than one object in the Gaia catalogue will be matched to the same object in other catalogues.

The challenges reside in the heterogeneity of the data as well as in the variety of the scientific communities preparing the catalogues. These range from solar system objects to distant galaxies, some already organized within their own databases, web portals and dissemination tools, some self-organized on local machines, some with their own well established standards, etc.

On one hand, the matching of large surveys (usually in the optical range) involves processing a huge amount of data. On the other hand, the cross-matching of data sets from largely different wavelength ranges involves dealing with very heterogeneous data.

In order to further facilitate the usage of the Gaia data, the cross-matching results will be integrated into existing services, such as the well-known Centre de Données astronomiques de Strasbourg (CDS) Xmatch[24] service. This will widen the opportunities for cross-matching the Gaia catalogue with any of the other 10,000 catalogues available in the service.

Science alerts During the lifespan of the mission, some photometric-related phenomena will be detected, requiring immediate attention and study due to its scientific value and (often) short duration. These phenomena could be unexpected and could be detected as rapid changes in the flux, spectrum or position of sources or appearance of new objects.

Scientific alerts contain basic characterisation information for each event, including parameters such as estimated object type like supernovae (SNe) events.

This work-package takes up two key tasks to ensure that science alerts can be accessed and visualised utilising standard tools such as Aladin or the WorldWideTelescope. These are:

- Develop the interfaces required to connect the real time science alerts classification processing to the main Gaia data products. As the mission evolves, and more knowledge is accumulated about objects measured by Gaia in its successive scans of the sky, there will be opportunities to cross reference new alerts against previous knowledge of that sky point, as well as previous alerts against new information.
- The Alerts Access will provide linkages to external data resources provided through the Gaia archive interfaces.

[24] http://cdsxmatch.u-strasbg.fr/xmatch

Chapter 5

The Grand Challenge

An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.

John Tukey, Mathematician

All the innovations presented in previous chapters represent an heterogeneous but well-integrated ecosystem for Big Data and Data Science. As described, there are several open source and commercial distributions available for institutions to roll out. However, its successful adoption in scientific contexts is not yet a straightforward task as it will require a transition period in which code, services and many other tools will have to be ported or adapted to the new environment. This endeavour will be exhausting in certain cases unless the proper guidance and a set of examples is provided to the community.

In this chapter, we concentrate on one particular astrophysical problem: the inference of a full spatio-temporal Milky Way galaxy model from Gaia observations of billions of stars. This is the ultimate goal of the Gaia mission and, in its most complex version, is certainly beyond present-day computing facilities. Its solution involves computational models of the stellar interiors and their evolution in time, dynamical evolution models for the kinematics of N-body systems (with N the number of stars in the Milky Way) and the modelling of the Gaia instruments themselves as well as the translation of their measurements into physically meaningful quantities. The work presented here, on a simplified yet still very ambitious version of the problem, represents a major step forward towards making these challenges feasible by solving some aspects of the problem. Future (and already ongoing) work will continue with these efforts for building the full model and use it with the observations being produced by Gaia.

Furthermore, we will use this simplified solution as a testbench for the infrastructure that will be made available to the community of the Gaia archive users as part of the Science Enabling Applications work package described in Section 4.5. In addition, we show some of the pitfalls that will be faced when getting through the previously mentioned transition, giving some hints on how they can be worked around. These contributions will certainly help make the right decisions of the engines or approaches to be taken, i.e.

use the right tool for the right purpose, as the idea of "one size fits all" seems to be gone (Stonebraker and Cetintemel, 2005). The original contributions presented in this chapter are covered in publication Tapiador et al. (2017).

5.1 Motivation

The Gaia archive will provide astrophysicists with observations and inferred quantities, and associated uncertainties for both of them. One of the Gaia DPAC Work Packages, i.e. Final Luminosity, Age and Mass Estimation (FLAME) in CU 8 (Bailer-Jones et al., 2013), will populate the Gaia archive with, amongst other quantities, masses and ages of the stars. These are quantities not directly observed, but inferred from the observations; in particular, they are inferred directly from the apparent brightness of the star, its parallax, and its effective temperature, which is in turn derived from the low-resolution spectra. In the test case presented in this chapter, we develop a HBM (Gelman et al., 2013) to infer the properties of the population of Milky Way stars from the sample that will be observed by Gaia. This kind of models have been successfully used in Dries et al. (2016); Mandel et al. (2016); Angus and Kipping (2016); Sale (2012). The properties we will infer are the PDMF and PDAD. The IMF is a Probability Density Function (PDF) $p(m)$ that describes the probability that a star formed in the Milky Way has mass m. Since the most massive stars evolve rapidly beyond the stellar regime, what we observe now is the remainder of the initial population (this is often known in the statistical literature as a truncated sample). In order to go from the PDMF to the IMF (or from the PDAD to the Star Formation History (SFH)), we would need to include stellar evolution models into the HBM, something which is out of the scope of this experiment. The PDAD is again a PDF that describes the probability that a star observed now has a given age (or alternatively, was formed at time t). A detailed description of the theory of star formation is beyond the scope of this article, but suffice it to say that it is the process by which dense clouds of gas and dust contract gravitationally until nuclear reactions ignite in the core. It is the final mass of the newly born star that is described by the IMF; and the amount of mass created per unit time in these processes is described by the SFR.

In order to infer the properties of the PDMF and PDAD from the properties of the observed stars, we will establish a probabilistic framework. In this framework, the quantities derived by FLAME will be treated as random variables that can be described by parametric PDF or by random samples from a MCMC realization. This scheme is repeated in several levels or layers of the model, and it represents a latent statistical model often referred to by the name of Hierarchical or multi-stage Bayesian Model (HBM). This will become clearer as the explanation goes on in the following paragraphs. Our main objective will be to infer the parameters that describe the PDMF and PDAD from the data set available in the Gaia mission archive.

The main advantages of HBM compared with other non-Bayesian techniques are:

1. The consistent propagation of uncertainties from observations to parameters through probability distributions in different levels of the model.

2. The possibility to derive probability distributions for individual parameters of interest by integrating the full probability distribution for the entire set of parameters over the uninteresting ones (the so-called nuisance parameters). This integration is known in the statistical literature as marginalisation.

3. Another important advantage of Bayesian analysis (not necessarily hierarchical) stems from the possibility to compare different models using Bayes factors. Model comparison is a crucial task in science that cannot be carried out in the framework of likelihood analysis without incurring in the danger of overfitting. The advantage of going hierarchical is the objectivity of the analysis. However, the degree of dependence of the bayesian posteriors on the assumed priors is a common controversy in the scientific community. Priors are often seen as subjective contamination of the Platonic scientific method idealised as a fully objective method. On the following, we alleviate this criticism by including the priors as part of the model we aim to infer.

On the contrary, one of the main drawbacks of using HBM is the higher complexity of the models implied by the hierarchies involved. These lead to the nested distributions needed to explain the relationships amongst parameters. Also, the increase in complexity requires a much larger number of model parameters and a higher dimensionality of the space of solutions. As a consequence, the probability distribution of the model parameters given the observations (i.e. the posterior distribution in Bayesian jargon) cannot be expressed analytically in general and one has to resort to sampling techniques like MCMC samplers. This adds a high computational overhead in the context addressed here of large data set sizes.

In the work presented here, we will consider two different kinds of models. For the PDMF we will infer a *parametric model*. The parametric model is an analytical expression for the probability of the observations expressed in terms of a series of model parameters. This analytical expression (the likelihood in Bayesian terms) condenses our knowledge about how probable an observation is in our model. In the case of the PDMF we will have a physical model for this likelihood term: an analytical expression of the parameters of which we ignore and wish to infer from the observed masses. The second model is a *non-parametric model* of the PDAD. It is non-parametric in the sense that the analytical expression depends on many parameters none of which has a specific meaning in statistical terms. This large number of parameters allows for a lot more flexibility (and hence expressivity) of the model, which implies that by inferring the parameters we are indeed inferring the model itself.

In the following, we assume that we have some sort of estimate of the masses and ages of the stars, with their associated uncertainties. These will be available as posterior probability distributions of the mass and age given the Gaia observations, in the form of MCMC samples produced by the FLAME work package. Let these posterior probabilities be

$$p_i = p(m_i, t_i|\mathcal{D}) \tag{5.1}$$

where $i : 1, 2, ..., N$ indexes stars and N is the total number of stars. $\mathcal{D}$ represents the data (Gaia observations) from which masses and ages are inferred: parallaxes, apparent brightness and low resolution spectro-photometry. These posterior probabilities p_i are assumed independent by the FLAME inference module. Departures from this assumption may be expected due to correlations in the measurements of the parallax and of the low resolution spectra. However, the current definition of the FLAME module operates source by source and does not take these correlations into account. Therefore, we will adopt the assumption of independence too, for the sake of consistency with the inputs expected from the Gaia catalogue.

Let us define $\psi(m, t)$ as the fraction of stars of mass m created at time t, per unit time and unit mass (and, for the sake of this simplified version of the Grand Challenge, we will neglect the mass evolved past the stellar stage). Then, $\psi(m, t)$ is a proper PDF in the sense that

$$\int_0^{13.8} \int_{m_{min}}^{m_{max}} \psi(m, t) \cdot \mathrm{d}m \cdot \mathrm{d}t = 1, \tag{5.2}$$

where m_{min} and m_{max} are the minimum and maximum masses defined for the PDMF, and 13.8 is taken to be the age of the Universe (Planck Collaboration et al., 2015). If we assume that the PDMF is constant in time, we can factorize $\psi(m, t)$ into the product of two PDF: the PDMF and the PDAD. This is a common assumption in Astrophysics even though it is not expected on physical grounds. Early in the history of the Galaxy, the chemical composition of the molecular clouds that collapse to form stars was different from the present-day composition. As stars are born, exhaust their hydrogen (and other species) and die, they enrich the insterstellar medium with the products of the nuclear reactions that take place in their interior. This chemical enrichment is theorised to change the range of masses that can be formed and the shape of the PDMF. The reason why the PDMF is often assumed constant in time is the relative lack of knowledge about its exact shape even in the present time, let alone its evolution through time. In this research we assume constancy of the PDMF only for the sake of simplicity, but real applications of the HBM will relax this assumption and model the PDMF and PDAD concurrently.

Previous work modelling the IMF and the SFR can be found in Weisz et al. (2013), where a probabilistic approach for inferring the parameters of the present-day power-law stellar mass function of a resolved young star cluster is shown. The authors use MCMC sampling to come up with estimates of the model parameters. However, they do not approach the problem from the computational complexity that is incurred with data sets as big as the ones Gaia (and other similar surveys) will produce. The amount of stars being observed in the Gaia mission makes this problem intractable, unless proper innovative ways are put in place. This is precisely one of the goals of the Grand Challenge.

5.2 Markov Chain Monte Carlo Techniques

The MCMC algorithm has played a significant role in statistics, physics, econometrics and computing science over the last decades. The reasoning behind it is to select a

statistical sample to approximate a hard combinatorial problem. MCMC techniques are often applied to solve integration and optimisation problems in large dimensional spaces, where analytical approaches are intractable. These are often found in bayesian inference and learning, e.g. for normalisation, marginalisation (the case for the models built below) and expectation (Andrieu et al., 2003).

The concept of Monte Carlo simulation is to draw an independent, identically distributed set of samples $\{x^{(i)}\}_{i=1}^{N}$ from a target density $p(x)$ defined on a high-dimensional space X (e.g. the space on which the posterior is defined, the set of possible configurations of a system, or the combinatorial set of feasible solutions). These N samples can be used to approximate the target density. The advantage of Monte Carlo integration over deterministic integration emerges from the fact that Monte Carlo positions the integration grid (i.e. the samples) in regions where the probability is high.

Therefore, MCMC is a strategy for generating samples $x^{(i)}$ while exploring the state space X using a Markov chain mechanism. This mechanism is built so that the chain spends more time in the most important regions, i.e. the samples $x^{(i)}$ imitate the samples that would be drawn from the target distribution $p(x)$. The usage of MCMC is for those cases where we cannot draw samples from $p(x)$ directly, but can evaluate $p(x)$ up to a normalising constant. Last, the evolution of the chain in a space X depends only on the current state of the chain.

One of the most well-known algorithms for MCMC is Metropolis-Hastings. It can draw samples from any probability distribution $p(x)$, provided the computation of the value of a function $f(x)$, that is proportional to the density of $p(x)$, is feasible. This last relaxed requirement that $f(x)$ is just proportional to the density makes this algorithm very useful, because calculating the necessary normalization factor is often extremely difficult in practice. The vague idea behind this algorithm is to draw samples from the probability distribution so that the number of samples drawn at a specific location is proportional to the height of the distribution at that location.

In the Metropolis-Hastings algorithm, there is a two stage process for generating the next sample x_{n+1}. First, a candidate x^* is generated. The value of x^* is created from the proposal distribution $Q(x^*|x_n)$, which depends on the current state x_n of the Markov chain. This may just be a normal distribution centred on the current state x_n with some standard deviation that needs to be specified. The second step is to accept or reject the new sample. For that, the acceptance probability $A(x_n \to x^*)$ is calculated (see Equation 5.3).

$$A(x_n \to x^*) = \min\left(1, \frac{P(x^*)}{P(x_n)} \times \frac{Q(x_n|x^*)}{Q(x^*|x_n)}\right) \tag{5.3}$$

The idea behind Equation 5.3 (acceptance probability) is to calculate the ratio of the probability of the new sample with regard to the old one (no need to account for the normalising constant) and multiply it by the ratio of the probability of generating the current sample x_n given the new sample x^*, over the probablity of generating a x^* as the candidate given that the current state is x_n. This second ratio corrects any bias that the proposal distribution may induce. Then, we accept the new candidate x^* with

a probability equal to the acceptance probability, or reject it otherwise (staying in the current state for another iteration).

The Gibbs sampler (Casella and George, 1992) is a special case of the Metropolis-Hastings algorithm. It is a technique for generating random variables from a (marginal) distribution indirectly, without having to calculate the density. The key intuition is that given a multivariate distribution, it is simpler to sample from a conditional distribution than to marginalize by integrating over a joint distribution. Then, it differs from Metropolis-Hastings in two ways:

- The candidate point is always accepted.
- The full conditional distributions of the parameters need to be known.

In the algorithm we have an n-dimensional x and the expression of the full conditionals $p(x_j|x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_n)$. Then, we initialise the vector $x^0_{1:n}$ and then we run N iterations sampling each element of the vector in order, based on the conditional distribution with the rest of elements:

- $x_1^{(i+1)} \sim p(x_1|x_2^{(i)},x_3^{(i)},\ldots,x_n^{(i)})$.
- $x_2^{(i+1)} \sim p(x_2|x_1^{(i+1)},x_3^{(i)},\ldots,x_n^{(i)})$.
- ...
- $x_j^{(i+1)} \sim p(x_j|x_1^{(i+1)},\ldots,x_{j-1}^{(i+1)},x_{j+1}^{(i)},\ldots,x_n^{(i)})$.
- $x_n^{(i+1)} \sim p(x_n|x_1^{(i+1)},x_2^{(i+1)},\ldots,x_{n-1}^{(i+1)})$.

For the implementation of the PDMF and PDAD within the efforts of the Grand Challenge, we have selected *emcee* (Foreman-Mackey et al., 2013), which is an affine-invariant ensemble sampler (Goodman and Weare, 2010) for MCMC. There are several advantages of such a sampler with regard to the traditional ones (e.g. Metropolis-Hastings and Gibbs). One of these advantages is that it requires hand-tuning of way less parameters (1 or 2 compared to $\sim N^2$, for a traditional algorithm in an N-dimensional parameter space). Furthermore, it has a very good performance as measured by the autocorrelation time (i.e. function calls per independent sample). In addition, it is insensitive to covariances among parameters.

The algorithm for *emcee* involves simultaneously evolving an ensemble of K walkers $S = X_k$ where the proposal distribution for one walker k depends on the current positions of the $K-1$ walkers in the complementary ensemble $S_{[k]} = X_j, \forall j \neq k$. In this context, *position* means a vector in the N-dimensional, real-valued parameter space (Foreman-Mackey et al., 2013).

For the models computed below, we have used the autocorrelation time as the burn-in and convergence criteria. It is especially applicable for *emcee* because it is the affine invariant measure of the performance. Another important measurement to be taken is

the acceptance fraction. In the tests carried out, the acceptance fraction was inside reasonable boundaries (approximately 0.62 for the PDMF and 0.4 for the PDAD). With regard to the autocorrelation time, we observed values between 40 to 60 for the PDMF and around 60 for the PDAD. We then let the chain run for one and two thousand iterations for the PDMF and PDAD respectively, which would be around 20 and 30 autocorrelation times (more than enough for these models considering the number of parameters used for each one (Foreman-Mackey et al., 2013)).

Figures 5.1, 5.2 and 5.3 show the chains for the parameters in the PDMF model as an illustration. It is important to remark that burn-in is a bit higher as we did not use any of the potential techniques that might have been leveraged to speed it up as suggested in Foreman-Mackey et al. (2013).

5.3 The Present-Day Mass Function

The IMF describes the distribution of initial *true* masses for a population of stars. There is no consensus about the exact shape of this function (the main alternatives due to Chabrier (2003); Kroupa (2002); Miller and Scalo (1979); Salpeter (1955)). In our case (where we neglect a key element of the stellar evolution past the stellar regime), we will not be inferring the IMF but the PDMF. Anyhow, and for the purpose of illustrating the method, we will use the broken power law proposed by Kroupa (2002) which is often assumed by the astronomical community (but it should be borne in mind that this is an analytical prescription for the IMF and not for the PDMF). We establish the hypothesis that the PDMF can be expressed as

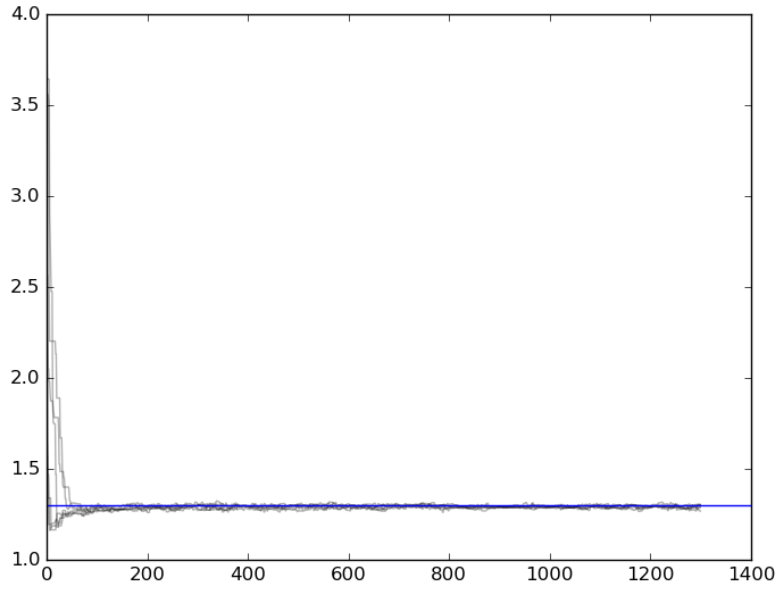
$$\xi(m;\boldsymbol{\theta}) = c_j m^{-\theta_j},\ M_j < m \leq M_{j+1}, \quad j = 1,2,3, \tag{5.4}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ is the parameter to infer, and the M_j are the mass limits that define the support of the broken power law. These will be held fixed in our model for the sake of simplicity, but could be inferred from the data as well. Finally, the c_j are computed for each $\boldsymbol{\theta}$ such that the function $\xi(\boldsymbol{\theta})$ is continuous at the boundaries M_j, and represents a proper PDF. This involves solving the simple system of equations:

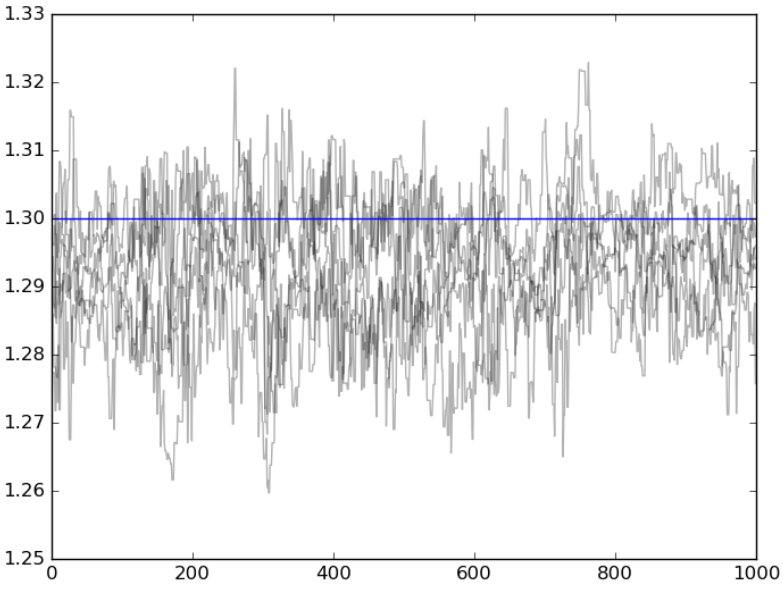
$$\begin{aligned} c_j M_i^{-\theta_j} &= c_{j+1} M_{j+1}^{-\theta_{j+1}}, \\ \sum_{j=1}^{3} \int_{M_j}^{M_{j+1}} c_j m^{-\theta_j}\, \mathrm{d}m &= 1. \end{aligned}$$

In order to explain what hierarchical or multi-level models are, we will start with a very simple model depicted in Figure 5.4. It shows how observed masses, $\hat{m}_i$ are related to the true masses, m_i, for each star.

In Figure 5.4, circles represent random variables and squares denote quantities held fixed in the analysis. Arrows denote the existence of a probabilistic dependence while grey nodes represent random variables that are inferred from Gaia data. The number

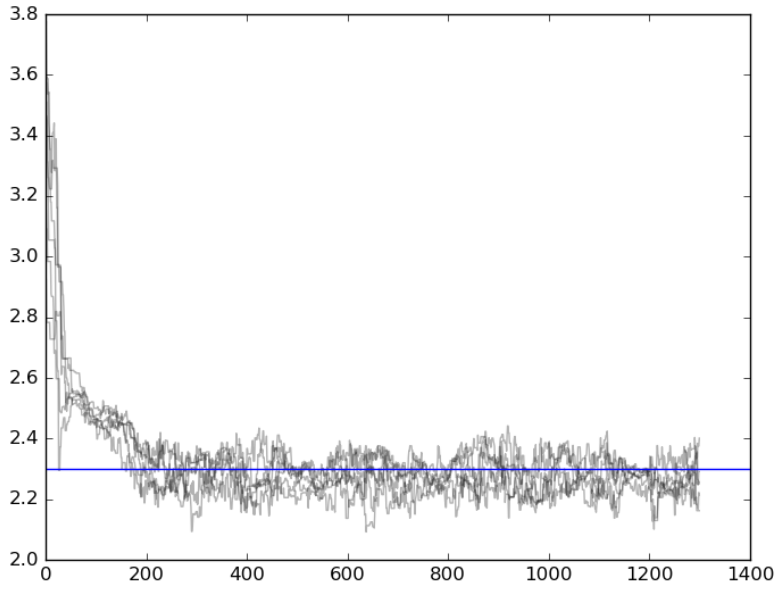


(a) Chain for PDMF parameter θ_1 in 1300 iterations without any burn-in.

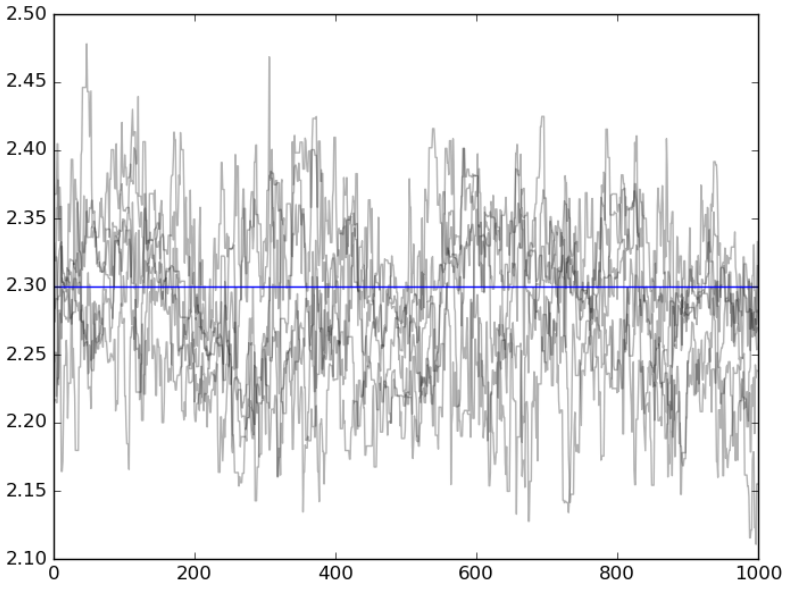


(b) Chain for PDMF parameter θ_1 in 1000 iterations after a burn-in of 300 iterations.

Figure 5.1: Walker chains for parameter θ_1.

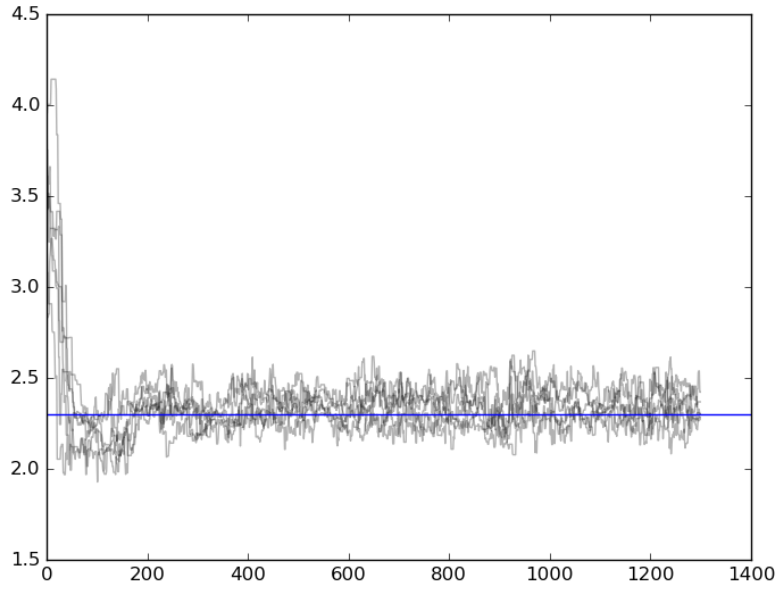


(a) Chain for PDMF parameter θ_2 in 1300 iterations without any burn-in.

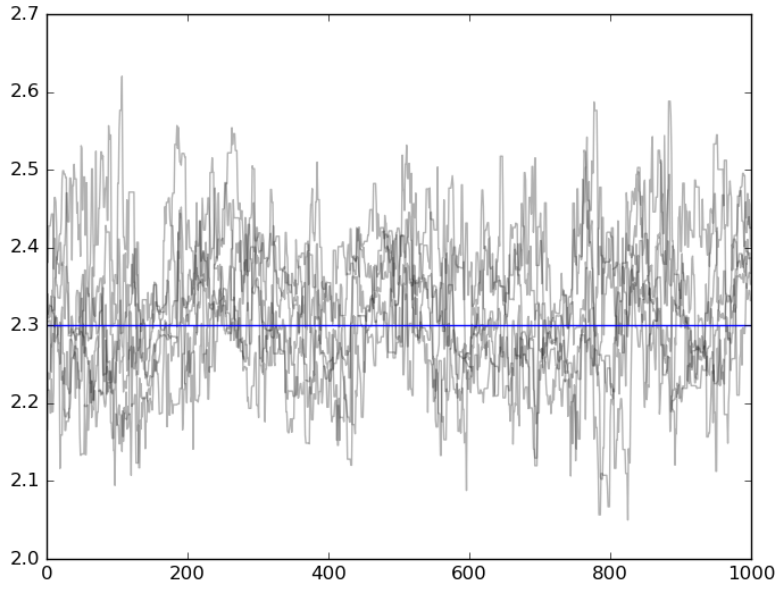


(b) Chain for PDMF parameter θ_2 in 1000 iterations after a burn-in of 300 iterations.

Figure 5.2: Walker chains for parameter θ_2.



(a) Chain for PDMF parameter θ_3 in 1300 iterations without any burn-in.



(b) Chain for PDMF parameter θ_3 in 1000 iterations after a burn-in of 300 iterations.

Figure 5.3: Walker chains for parameter θ_3.

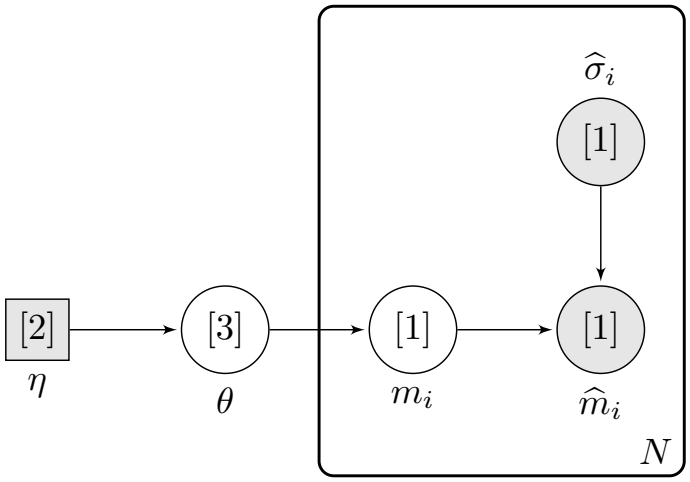


Figure 5.4: Hierarchical model using plate notation. The circles represent random variables and the squares refer to fixed quantities. Arrows denote the existence of a statistical dependence. Grey nodes represent measured random variables. For instance, $\hat{m}_i$ is the observed mass of the i-th star. The big rectangle (plate) represents the repetition of the variables inside. The value N in the corner is the number of these repetitions and match the number of sources/stars with true (m_i) and observed ($\hat{m}_i$) masses. η is the hyperparameter that governs the distribution of the prior probability distribution of the slopes $p(\boldsymbol{\theta}|\eta)$. Finally the number inside the brackets represents the dimension of the variables.

in square brackets in the node denotes the dimensionality of the parameter and arrows represent conditional dependence probabilistic relations between variables. The N in the corner indicates the number of equivalent variables in the plate (in our case, one for each mass in the data set).

Let us begin the description of the plate notation with the two right-most arrows. Under the hypothesis that the measurement uncertainties are Gaussian, this probabilistic relationship would be

$$p_i(\hat{m}_i|\mathcal{D}) = \log \mathcal{N}(\hat{m}_i; m_i, \hat{\sigma}_i), \tag{5.5}$$

where $\hat{m}_i$ and $\hat{\sigma}_i$ are the mean and the standard deviation of the distribution given in Equation 5.1. Both $\hat{m}_i$ and $\hat{\sigma}_i$ are estimations from the posterior distribution of masses delivered by FLAME team. In our case, and for the sake of simplicity in the explanation, we are considering the sample mean and the sample standard deviation as estimators. Of course, other functional relationships can be thought of to represent the uncertainty in the measurements. This is especially true for masses because they are obtained through convolved procedures whereby uncertainties can be far from Gaussian. This is the case of evolved stars that can have very different masses and ages, and yet be characterised by the same observed properties (or, in the astronomical terminology, loci in the Hertzsprung-Russell diagram where several evolutionary tracks cross). This results in multi-modal uncertainties if correctly inferred (via, for example, MCMC techniques).

In the plate example, we assume that the PDMF is modelled as a function of three parameters $\xi(m_i; \theta_1, \theta_2, \theta_3)$ that yields the probability density of generating a star of mass m_i. The number 3 between square brackets inside the node denotes exactly this dimension of three in the parameters. Hence, the middle arrow and its two connected nodes represent the PDMF.

In our HBM, the vector of model parameters $\boldsymbol{\theta}$ is itself treated as a random variable, the distribution of which we aim to infer. By inferring the probability distribution of $\boldsymbol{\theta}$ given the data $\mathcal{D}$, we infer the PDMF. As a random variable in our model, we need to specify a PDF for it. Since $\boldsymbol{\theta}$ does not depend on any other model parameter or random variable (η is held fixed) the PDF for $\boldsymbol{\theta}$ is its *a priori* or *prior* distribution. The arrow that joins the fixed value of η with the parameter $\boldsymbol{\theta}$ represents then the model element or conditional probabilistic relationship that encodes the *a priori* distribution $p(\boldsymbol{\theta}|\eta)$ of possible slopes $(\theta_1, \theta_2, \theta_3)$ of the PDMF. η is the hyperparameter that governs the distribution of the prior probability distribution of the slopes $p(\boldsymbol{\theta}|\eta)$. It is specified based on a priori knowledge of the values of $\boldsymbol{\theta}$ that are reasonably consistent with the estimated stellar masses in general. For the purpose of the example below, we consider $\eta = (0, 5)$ which means that θ_i will be generated by a uniform distribution between 0 and 5.

Summarizing, we have a probabilistic latent model to explain the distribution of the observed data, and a generative model for the observed masses $\hat{m}_i$:

1. draw three slopes according to a uniform distribution with PDF $p(\boldsymbol{\theta}|\eta)$.
2. for each star, draw a true mass according to the PDF $\xi(m_i; \boldsymbol{\theta})$.

3. for each star, draw an observed mass according to the PDF $\log\mathcal{N}(\hat{m}_i; m_i, \hat{\sigma}_i)$.

The main goal of astronomers is to move in the inverse way: from observations to model parameters. In our example, from the masses and their uncertainties, $\{\hat{m}_i, \hat{\sigma}_i\}_{i=1}^N$, to the parameter $\boldsymbol{\theta}$. This is a classical problem in science: infer model parameters from observed data, and that is exactly what Bayesian inference provides: the posterior probability is nothing but the probability distribution of the model parameters given the observed data. We obtain the posterior by multiplying the so called likelihood (i.e. $p(\{\hat{m}_i\}_{i=1}^N|\{\hat{\sigma}_i\}_{i=1}^N, \theta, \eta)$, the generative model we have just exemplified) times the prior $p(\boldsymbol{\theta}|\eta)$.

Assuming a sample of independent observations $\mathcal{M}$, the posterior distribution can be written according to

$$\begin{aligned} p(\boldsymbol{\theta}|D,\eta) &\propto p(\{\hat{m}_i\}_{i=1}^N|\{\hat{\sigma}_i\}_{i=1}^N, \boldsymbol{\theta}, \eta)p(\boldsymbol{\theta}|\{\hat{\sigma}_i\}_{i=1}^N, \eta) \\ &= p(\{\hat{m}_i\}_{i=1}^N|\{\hat{\sigma}_i\}_{i=1}^N, \boldsymbol{\theta})p(\boldsymbol{\theta}|\eta) \\ &= \prod_{i=1}^N p(\hat{m}_i|\hat{\sigma}_i, \boldsymbol{\theta}) \quad p(\boldsymbol{\theta}|\eta), \end{aligned}$$

and by using the log function

$$\log p(\boldsymbol{\theta}|\mathcal{D},\eta) = \sum_{i=1}^N \log p(\hat{m}_i|\hat{\sigma}_i, \boldsymbol{\theta}) \quad + \log p(\boldsymbol{\theta}|\eta) - C, \tag{5.6}$$

where $C = \log p(\mathcal{D})$ is a constant (the log-evidence) that can, for our purposes, be ignored (we will only need to compute it explicitly in order to compare and select amongst alternative PDMF models like Chabrier (2003); Kroupa (2002), etc).

Let us now focus on the likelihood $p(\hat{m}_i|\hat{\sigma}_i, \boldsymbol{\theta})$. We only have observed masses available, not the true ones, so every true (unknown) mass m_i will also be a model parameter. This is a severe drawback in the Gaia context of billions of sources. We avoid this problem by marginalising the likelihood:

$$\begin{aligned} p(\hat{m}_i|\hat{\sigma}_i, \boldsymbol{\theta}) &= \int p(\hat{m}_i, m|\hat{\sigma}_i, \boldsymbol{\theta})\mathrm{d}m \\ &= \int p(\hat{m}_i|m, \hat{\sigma}_i, \boldsymbol{\theta})p(m|\hat{\sigma}_i, \boldsymbol{\theta})\mathrm{d}m \\ &= \int p(\hat{m}_i|m, \hat{\sigma}_i)p(m|\boldsymbol{\theta})\mathrm{d}m. \end{aligned}$$

In order to obtain the posterior probability in equation 5.6, we must first evaluate the integral above for every mass observed. We approximate the integral by using the trapezoidal rule.

In order to test the hierarchical model with a small simulated data set, 10,000 masses were drawn from an PDMF PDF with $\boldsymbol{\theta} = (1.3, 2.3, 2.3)$. For each *true* mass obtained,

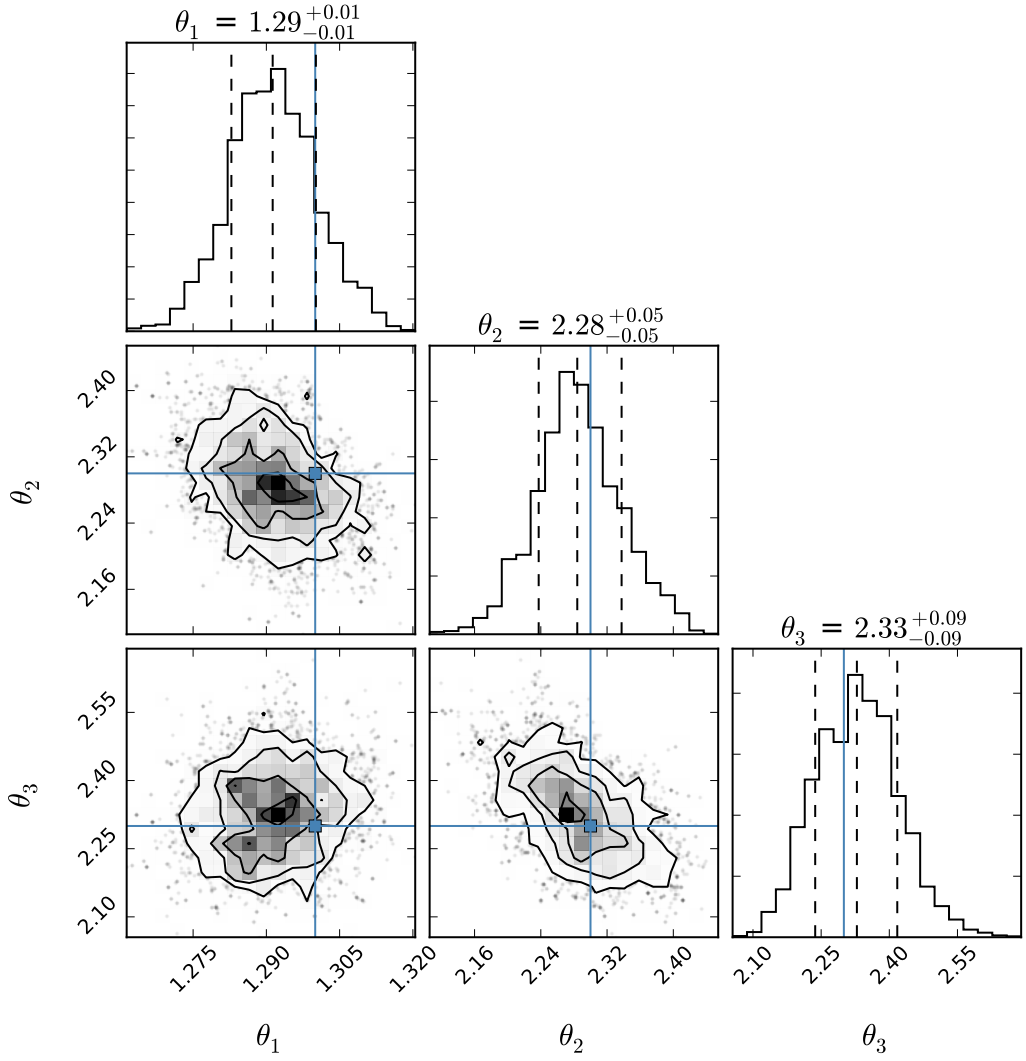


Figure 5.5: Samples drawn from the posterior distribution in Equation 5.6 by *emcee* algorithm (Foreman-Mackey et al., 2013). Blue lines represent true values of $\boldsymbol{\theta} = (1.3, 2.3, 2.3)$. Dashed lines represent quantiles 0.16, 0.5, 0.84, for each θ_i. The title above each 1-D histogram shows the 0.5 quantile with the upper and lower errors supplied by the quantiles. The 2-D plots show the contour lines for levels 0.11, 0.39, 0.67 and 0.86. See Foreman-Mackey (2016) for more information.

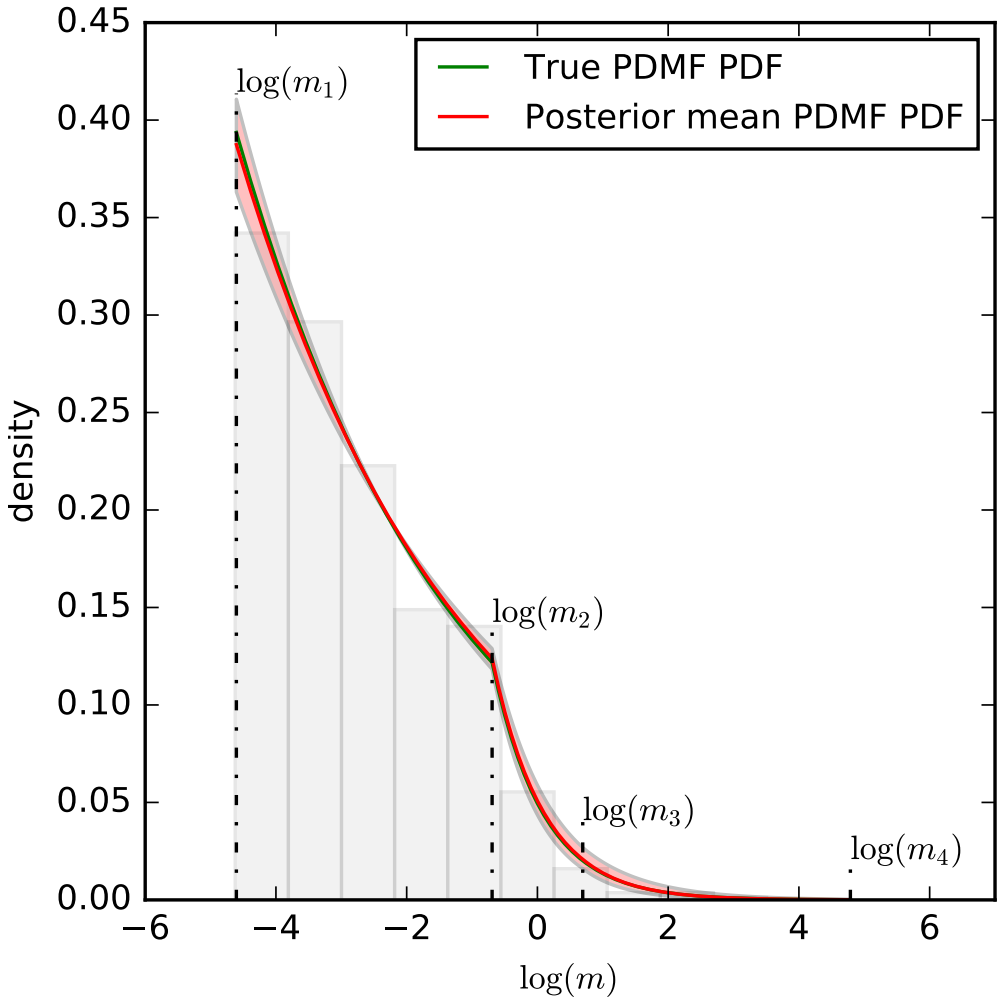


Figure 5.6: True and estimated PDMF PDF function are shown in green and red respectively. The red ribbon represents the 3σ confindence band for estimated PDMF PDF. The graph was done by using the log transformation in x-axis to improve the visualization. Ticks on masses doing the intervals for the PDMF PDF support are shown. Note that the figure shows PDMF PDF with a change of variable to $\log(m)$ instead of m, in order to improve the visualization.

we drew 50 samples from a log-normal distribution that reproduces the expected output from the FLAME Work Package[1], at least during the first cycles of data processing. All the log-normal distributions considered (one per star) were centred at the true mass, and had a standard deviation equal to 0.1.

Figures 5.5 and 5.6 show the posterior samples obtained by applying the *emcee algorithm* (Foreman-Mackey et al., 2013) to the hierarchical model described above. The dashed line in Figure 5.5 represents the quantiles 0.16, 0.50 and 0.84 of the posterior samples. These quantiles result in the following estimates: $\hat{\theta}_1 = 1.29^{+0.01}_{-0.01}$, $\hat{\theta}_2 = 2.28^{+0.05}_{-0.05}$ and $\hat{\theta}_3 = 2.33^{+0.09}_{-0.09}$. The bulk of the marginal posterior distribution for θ_1 seems to have a bias with respect to the true value, but this displacement is only five thousandths from the maximun a posteriori value, which fits well for the purposes of this test example. Figure 5.6 shows a comparative between the true PDMF PDF (green) and the estimated posterior mean PDMF PDF (red) with a confidence band of 3σ. Note that the figure shows the PDMF PDF as a function of $\log(m)$ instead of m, for the sake of clarity.

5.4 The Present-Day Age Distribution

5.4.1 Gaussian Processes

Gaussian processes define a distribution on functions with a continuous domain. This continuous domain is typically time (like in the model below for the PDAD) or space. The intuition behind it is that we observe some data points and aim at assigning probabilities to every way in which a line could be drawn through those observed points. The goal is to find the functions (lines) with highest probability (given the observations), hoping that they are very similar to the true function that cannot be observed.

Every draw of a Gaussian process represents a function for which we need to measure its fit to the observations. The prior will constrain the set of concrete functions (lines) that can be drawn (obviously with big enough margins), and we can take measurements less rigidly. Those measurements refine the prior into a posterior with an increased confidence.

To show the relationship of Gaussian processes with Gaussian distributions, we can state that:

- Univariate Gaussians are distributions over real valued variables.
- Multivariate Gaussians are distributions over pairs, triplets, etc, of real valued variables.

[1]Let us notice that, in general, the distributions delivered by FLAME will neither be normal (or log-normal), nor homoscedastic, as it has been assumed in this use case. We expect a variety of distributions from the observations performed by Gaia, i.e. one for each star. In particular, given the degeneracy of the inference problem solved by FLAME, we expect a significant fraction of their posterior distributions to be multimodal. However, this will not be a drawback for the model, but rather it will produce posterior distributions that are closer to the reality, showing real characteristics of the PDMF. This will empower a direct usage of the model with real observed data once it is available.

- Gaussian processes are functions of (infinite numbers of) real valued variables. It allows us to have a Gaussian distribution over infinite numbers of variables.

This statistical model drives us to the notion of regression, which has some interesting features and problems which Gaussian processes benefit from or can address nicely:

- Denoising and smoothing, i.e. do not follow every wiggle of the data points (observations) but rather get a good description of what is signal and what is noise.
- Prediction and forecasting with some certainty.
- Sort out the dangers of parametric models which may miss some of the features and even result in weird predictions (e.g. in quadratic models when taken far away of the intervals for which there are observations).
- The problem of overfitting and underfitting the data points.

5.4.2 Problem Description

In this Section we address the problem of inferring the Star Formation History of our Galaxy (actually, the PDAD because as mentioned before, we will not be including the relics of stellar evolution). The SFH is defined as $\zeta(t)$, the total mass (expressed in units of the solar mass) transformed into stars at time t. With this definition, the SFR is not a PDF. We will find that a slightly different definition proves useful in reformulating the problem in probabilistic terms.

In principle, we would like to be able to infer any arbitrary shape of the PDAD function, except maybe discontinuous ones. If we assume continuity and smoothness of the PDAD, we can define it as:

$$\zeta(t) = \sum_{k=1}^{K} w_k \phi_k(t) = \boldsymbol{\phi}^T(t) \cdot \boldsymbol{w}, \tag{5.7}$$

where $\boldsymbol{w} \in \mathbb{R}^K$ would be the parameters of our model, and $\boldsymbol{\phi} : \mathbb{R} \to \mathbb{R}^K$ is the *feature space mapping* (also called *characteristic function*) that fulfills the continuity and smoothness conditions stated above. For the sake of clarity, and in order to fix ideas and exemplify the procedure, we can choose

$$\boldsymbol{\phi}(t) = (e^{-0.5(t-t_{01})^2}, \quad e^{-0.5(t-t_{02})^2}, \ldots, e^{-0.5(t-t_{0K})^2})^T \tag{5.8}$$

where t_{0k} are some reference times, and the components $\phi_k(t) = e^{-0.5(t-t_{0k})^2}$ correspond to the same function, centred at different times t_{0k}. Thus, we have expressed the PDAD in equation 5.7 as a linear combination of some basis functions.

The probability density function of the formation time of a star is simply a normalized version of the PDAD $\zeta(t)$ defined above, such that its integral between the origin of time and the present time is one. In order to define a PDF that reflects the information encoded in the PDAD, it will be convenient to work in terms of a modified version of

it. Instead of $\zeta(t)$ which has units of solar masses per unit time, we will work with $\zeta'(t)$ defined as the fraction of stars (of whatever mass) created per unit time:

$$\zeta'(t) = \frac{\sum_k w_k \phi_k(t)}{\sum_k w_k \int_a^b \phi_k(t)\,\mathrm{d}t}. \tag{5.9}$$

Thus, the PDAD defines a non-stationary Poisson process with intensity function $\zeta'(t)$, $t \geq 0$.

Given the data set available from Gaia (the posterior samples of mass and age $p_i(m, (13.8 - t)|\mathcal{D})$ produced by the FLAME module) we could estimate[2] the function ζ' at a set of times $\boldsymbol{t}_1 = \{t_{11}, t_{12}, \ldots, t_{1Q}\}$ as

$$\hat{\zeta}'(t_{1j}) = \frac{1}{N} \int_{m_{min}}^{m_{max}} \sum_{i=1}^{N} p_i(m, (13.8 - t_{1j})|\mathcal{D}) \cdot \mathrm{d}m. \tag{5.10}$$

where again, N is the total number of stars observed by Gaia. We have assumed that the origin of time is defined by the age of the Universe (13.8 Gyrs ago (Planck Collaboration et al., 2015)). The age of a star is then defined as 13.8 Gyr minus its birth time.

Since as stated above, $\zeta'(t)$ is a non-stationary Poisson process, $\zeta'(t)$ and $\hat{\zeta}'(t)$ will be related by

$$\begin{aligned} \hat{\zeta}'(t) &\sim \mathcal{P}(\zeta'(t)) \\ &\sim \mathcal{N}(\zeta'(t), \sigma(t)), \end{aligned} \tag{5.11}$$

where we approximate the Poisson distribution with a Normal distribution centred at ζ' and with standard deviation $\sigma(t)$, the standard deviation of the associated Poisson counting experiment in the asymptotic limit of large intensities. In Chapter 6, we describe extensions of this work that allow for the inclusion of the Poissonian distribution.

Equation 5.10 represents an estimate of the PDAD PDF in some set of arbitrary but ordered points $\boldsymbol{t}_1$. Unfortunately, this estimate is not enough except in cases of PDAD that are smooth even on infinitesimal scales. In order to accomodate PDAD that change smoothly, but very rapidly on very small time scales, we would need a prohibitively large number of points, as we show next.

[2] In reality, this oversimplification does not apply: the total number of stars observed is only a fraction of the total number of stars because any astronomical survey (and Gaia is no exception in this respect) is biased. Gaia only detects stars down to a certain apparent brightness limit, so intrinsically bright stars are detected further from us while faint stars are only detected locally around the Sun. Since the density of stars in the Galaxy is far from uniform, incorporating all the biases into our model would require a full model of the spatial density distribution of stars in the Milky Way. Furthermore, the relationship between mass, age and brightness is rather complex and involves also spatial gradients in the chemical composition of the stars. All these complexities (and more that are not mentioned here) are far beyond the scope of this research and we will simply assume that what we see (with the eyes of Gaia) is what there is.

Let $\zeta'_{\boldsymbol{t}_1}$ be a finite restriction of ζ' to values at reference times $\boldsymbol{t}_1 = [t_{11}, \ldots, t_{1Q}]$. Then,

$$p(\hat{\zeta}'_{\boldsymbol{t}_1}|\zeta'_{\boldsymbol{t}_1}) = p(\hat{\zeta}'_{\boldsymbol{t}_1}|\boldsymbol{w}, \boldsymbol{\phi}_{\boldsymbol{t}_1}) = \mathcal{N}(\hat{\zeta}'_{\boldsymbol{t}_1}; \boldsymbol{\phi}^T_{\boldsymbol{t}_1}\boldsymbol{w}, \Sigma_{\boldsymbol{\varepsilon}}), \tag{5.12}$$

where $\Sigma_{\boldsymbol{\varepsilon}}$ is a $Q \times Q$ diagonal matrix containing $\sigma^2(t_{1i})$ in the diagonal, and $\boldsymbol{\phi}_{\boldsymbol{t}_1}$ is the $K \times Q$ matrix

$$\boldsymbol{\phi}_{\boldsymbol{t}_1} = \begin{pmatrix} 1 & \phi_1(t_{12}) & \ldots & \phi_1(t_{1Q}) \\ \phi_2(t_{11}) & 1 & \ldots & \phi_2(t_{1Q}) \\ \ldots & \ldots & \ldots & \ldots \\ \phi_K(t_{11}) & \phi_K(t_{12}) & \ldots & 1 \end{pmatrix}.$$

Let us assume a K-dimensional multivariate normal distribution for the prior probability distribution of $\boldsymbol{w}$,

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{\mu}_w, \Sigma_w), \tag{5.13}$$

and consider the vector $\zeta'_{\boldsymbol{t}_2}$ for some finite sequence of times $\boldsymbol{t}_2 = \{t_{21}, t_{22}, \ldots, t_{2M}\}$ that includes and extends the times $\boldsymbol{t}_1$ (that is, $\boldsymbol{t}_1 \subset \boldsymbol{t}_2$). It is possible to reorder $\zeta'_{\boldsymbol{t}_2}$ as $(\zeta'_{\boldsymbol{t}}, \zeta'_{\boldsymbol{t}_1})^T$, where $\boldsymbol{t}$ contains the points in $\boldsymbol{t}_2$ excluding $\boldsymbol{t}_1$. Furthermore, $(\zeta'_{\boldsymbol{t}}, \zeta'_{\boldsymbol{t}_1})^T = \zeta'_{\boldsymbol{t}_2} = \phi^T_{\boldsymbol{t}_2}\boldsymbol{w}$ is also distributed as a multivariate normal because it is a linear combination of $\boldsymbol{w}$, which is assumed normal according to Equation 5.13.

In this situation, we can apply the property of closure under conditioning (5.4.2) thus obtaining

$$p(\zeta'_{\boldsymbol{t}}|\hat{\zeta}'_{\boldsymbol{t}_1}, \boldsymbol{\phi}_{\boldsymbol{t}_1}) = \mathcal{N}(\zeta'_{\boldsymbol{t}}; \boldsymbol{\mu}^{\text{post}}, \Sigma^{\text{post}}), \tag{5.14}$$

where

$$\begin{aligned} \boldsymbol{\mu}^{\text{post}} &= \boldsymbol{\phi}^T_{\boldsymbol{t}}\boldsymbol{\mu}_w + \boldsymbol{\phi}^T_{\boldsymbol{t}}\Sigma_w\boldsymbol{\phi}_{\boldsymbol{t}_1}(\boldsymbol{\phi}^T_{\boldsymbol{t}_1}\Sigma_w\boldsymbol{\phi}_{\boldsymbol{t}_1} + \Sigma_{\boldsymbol{\varepsilon}})^{-1}(\hat{\zeta}'_{\boldsymbol{t}_1} - \boldsymbol{\phi}^T_{\boldsymbol{t}_1}\mu_w), \\ \Sigma^{\text{post}} &= \boldsymbol{\phi}^T_{\boldsymbol{t}}\Sigma_w\boldsymbol{\phi}_{\boldsymbol{t}} - \boldsymbol{\phi}^T_{\boldsymbol{t}}\Sigma_w\boldsymbol{\phi}_{\boldsymbol{t}_1}(\boldsymbol{\phi}^T_{\boldsymbol{t}_1}\Sigma_w\boldsymbol{\phi}_{\boldsymbol{t}_1} + \Sigma_{\boldsymbol{\varepsilon}})^{-1}\boldsymbol{\phi}^T_{\boldsymbol{t}_1}\Sigma_w\boldsymbol{\phi}_{\boldsymbol{t}}. \end{aligned}$$

Proposition 5.4.1 (Closure under multiplication)**.** Let two Gaussian distributions $\mathcal{N}(x; \mu_1, \Sigma_1)$ and $\mathcal{N}(x; \mu_2, \Sigma_2)$ then

$$\mathcal{N}(x; \mu_1, \Sigma_1)\mathcal{N}(x; \mu_2, \Sigma_2) = \mathcal{N}(x; \mu_3, \Sigma_3)\mathcal{N}(\mu_1; \mu_2, \Sigma_1 + \Sigma_2),$$

where $\Sigma_3 = (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}$ and $\mu_3 = \Sigma_3(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2)$.

Proposition 5.4.2 (Closure under conditioning)**.** Let two Gaussian distributions $\mathcal{N}(x; \mu_x, \Sigma_{xx})$ and $\mathcal{N}(y; \mu_y, \Sigma_{yy})$ with $\text{cov}(x, y) = \Sigma_{xy}$ then

$$p(x|y) = \mathcal{N}(x; \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}).$$

At this point we see that the size of the matrices increases with the number of features K, which enters the calculations via $\boldsymbol{w}$. This is a big drawback, because increasing the value of K in order to be able to accomodate the steep rise times of known PDAD, results in computational costs that quickly become unacceptable (see Section 5.5). We solve this problem by defining a Gaussian Process that samples in the space of functions rather than in the space of vectors ($\boldsymbol{w}$).

For the sake of clarity, we will consider a simple matrix

$$\Sigma_w = \frac{\sigma^2(t_{\max} - t_{\min})}{K}\boldsymbol{I},$$

where $t_{\max} \geq 13.8$ and $t_{\min} \leq 0$ are constant values and σ^2 is an unknown parameter. The elements $\boldsymbol{\phi_t}^T\Sigma_w\boldsymbol{\phi_t}$ will be

$$\boldsymbol{\phi_t}^T\Sigma_w\boldsymbol{\phi_t}(i,j) = \frac{\sigma^2(t_{\max} - t_{\min})}{K}\Sigma_{k=1}^{K}\phi_k(t_i)\phi_k(t_j),$$

where $\phi_k(t) = e^{-0.5(t-t_{0k})^2}$, i.e.,

$$\begin{aligned}\boldsymbol{\phi_t}^T\Sigma_w\boldsymbol{\phi_t}(i,j) &= \frac{\sigma^2(t_{\max} - t_{\min})}{K}\Sigma_{k=1}^{K}e^{-0.5(t_i-t_{0k})^2}e^{-0.5(t_j-t_{0k})^2} \\ &= \frac{\sigma^2(t_{\max} - t_{\min})}{K}e^{-0.5(t_i-t_j)^2}\Sigma_{k=1}^{K}e^{-0.5(t_{0k}-0.5(t_i+t_j))^2}.\end{aligned}$$

Increasing K such that the number of features in δt becomes $\frac{K\delta t}{t_{\max}-t_{\min}}$, we have

$$\boldsymbol{\phi_t}^T\Sigma_w\boldsymbol{\phi_t}^T(i,j) = \sigma^2 e^{-0.5(t_i-t_j)^2}\int_{t_{\min}}^{t_{\max}} e^{-0.5(t-0.5(t_i+t_j))^2}\mathrm{d}t,$$

and letting $t_{\min} \to -\infty$ and $t_{\max} \to +\infty$ we obtain

$$\boldsymbol{\phi_t}^T\Sigma_w\boldsymbol{\phi_t}^T(i,j) \to \sqrt{2\pi}\sigma^2 e^{-0.5(t_i-t_j)^2}, \tag{5.15}$$

which does not depend on the number of features K. This infinite covariance matrix is also called the *kernel function*, $\kappa(t_i,t_j) = \sqrt{2\pi}\sigma^2\exp\{-0.5(t_i-t_j)^2\}$ and the following paragraphs link the previous statistical development with the Gaussian Process (GP) methodology.

Definition 1. A function $\kappa : \mathbb{X}\times\mathbb{X} \to \mathbb{R}$ is a Mercer kernel if, for any finite collection $X = [x_1,\ldots,x_K]$, the matrix $\kappa_{XX} \in \mathbb{R}^{K\times K}$ with elements $\kappa_{XX}(i,j) = \kappa(x_i,x_j)$ is positive semidefinite.

There are many kernels defined in the literature (see for instance Bishop (2006)) and one of the multiple approximations to construct them is through a feature space mapping $\boldsymbol{\phi}(t)$ which ensures a valid kernel κ when the values $\kappa(x_i,x_j)$ conform a positive semidefinite matrix according to

$$\kappa_{XX}(i,j) = \kappa(x_i,x_j) = \boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}(x_j).$$

Definition 2. Let $\nu : \mathbb{X} \to \mathbb{R}$ be any function, called the mean function, and let $\kappa : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ be a Mercer kernel. A Gaussian Process $p(f) = \mathcal{GP}(f; \nu, \kappa)$ is a probability distribution over the function $f : \mathbb{X} \to \mathbb{R}$, such that every finite restriction to function values $f_X := [f_{x_1}, \ldots, f_{x_K}] = [f(x_1), f(x_2), \ldots, f(x_K)]$ is Gaussian distributed $p(f_X) = \mathcal{N}(f_X; \nu_X, \kappa_{XX})$

We tackle the problem of inferring the PDAD PDF as a GP, by substituting every term which includes the feature function in $\boldsymbol{\mu}^{\text{post}}$ and Σ^{post} in the posterior distribution (Equation 5.14), with their mean and kernel functions counterparts, namely

$$\begin{aligned} \boldsymbol{\mu}^{\text{post}} &= \nu_{\boldsymbol{t}} - \kappa_{\boldsymbol{t}\boldsymbol{t}_1}(\kappa_{\boldsymbol{t}_1\boldsymbol{t}_1} + \Sigma_{\boldsymbol{\varepsilon}})^{-1}(\hat{\zeta}'_{\boldsymbol{t}_1} - \nu_{\boldsymbol{t}_1}) \\ \Sigma^{\text{post}} &= \kappa_{\boldsymbol{t}\boldsymbol{t}} - \kappa_{\boldsymbol{t}\boldsymbol{t}_1}(\kappa_{\boldsymbol{t}_1\boldsymbol{t}_1} + \Sigma_{\boldsymbol{\varepsilon}})^{-1}\kappa_{\boldsymbol{t}_1\boldsymbol{t}} \end{aligned} \tag{5.16}$$

where $\kappa_{\boldsymbol{t}\boldsymbol{t}}$ is a Mercer kernel and $\nu_{\boldsymbol{t}}$ is the mean function.

5.4.3 The Hierarchical Model

The only parameter involved in the GP defined above is σ^2 which does neither depend on the number of stars observed (N), nor on the dimension of the feature space mapping (K). In order to give our HBM more expressivity, we have chosen alternative (but more complex) kernel and mean functions:

$$\nu_{\boldsymbol{t}} = \mathbf{0}, \quad \kappa_{\boldsymbol{t}\boldsymbol{t}}(i,j) = \eta^2 \exp(-\rho^2(t_i - t_j)^2), \tag{5.17}$$

where η^2, ρ^2 are parameters that define the actual shape of the PDAD and are inferred as part of the hierarchical model.

Figure 5.7 shows the hierarchical Bayesian model in plate notation. The arrow between nodes m_j and $p_i(\cdot)$ represents the kernel density estimation via diffusion (Botev et al., 2010) derived from the FLAME output. Note the bidirectional relationship between the nodes $\zeta'(\cdot)$ and $p_i(\cdot)$: (i) the dashed arrow represents the set up of the GP, where we estimate the $p_i(\cdot)$ on the set of $\boldsymbol{t_1}$ times. This procedure is executed just once; and (ii), the solid line represents the evaluation of the likelihood of the estimated $p_i(\cdot)$ under the $\zeta'(\cdot)$ proposed by the GP. The node $\zeta'(\cdot)$ is the result of the GP defined by $\nu(\cdot) = 0$ and $\kappa(\cdot,\cdot)$ given by Equation 5.17. The prior probabilities for η and ρ defined as a half Cauchy distribution with the same scale hyperparameters, $\eta_{\text{sc}} = \rho_{\text{sc}} = 5$.

The posterior distribution of the HBM parameters in the simple case of a single star observed would be

$$\begin{aligned} p(\eta, \rho | p_1(t), \zeta'_{\boldsymbol{t}}, \hat{\zeta}'_{\boldsymbol{t}_1}, \Sigma_\varepsilon) \propto\ & p(p_1(t) | \zeta'_{\boldsymbol{t}}, \hat{\zeta}'_{\boldsymbol{t}_1}, \Sigma_\varepsilon) \cdot \\ & p(\zeta'_{\boldsymbol{t}} | \hat{\zeta}'_{\boldsymbol{t}_1}, \Sigma_\varepsilon, \eta, \rho) \cdot \\ & p(\eta) \cdot p(\rho), \end{aligned} \tag{5.18}$$

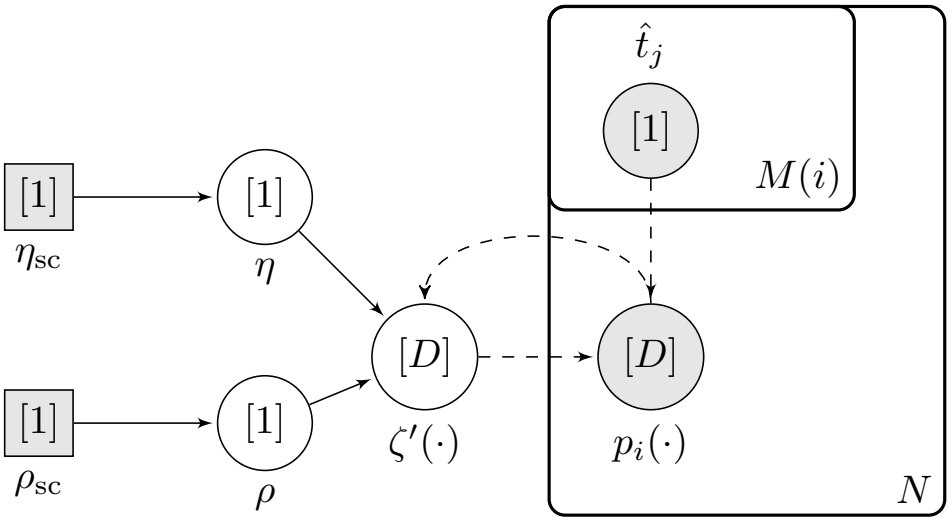


Figure 5.7: The PDAD PDF Hierarchical Bayesian Model in plate notation. The circles represent random variables and the squares refer to fixed quantities. Arrows denote the existence of a statistical dependence. Grey nodes represent measured random variables. For instance, $\hat{t}$ is the observed age for each star. The outer rectangle (plate) represents the repetition of the variables inside, and N at the bottom right corner is the number of these repetitions (i.e. number of sources). The inner rectangle represents a sample of ages with size $M(i)$ for each star i. Finally the number inside the brackets represents the dimension of the variables.

where $p(\eta)$ and $p(\rho)$ are the hyperprior distributions, and $p(\zeta'_{\boldsymbol{t}}|\hat{\zeta}'_{\boldsymbol{t_1}}, \Sigma_\varepsilon, \eta, \rho)$ is the GP defined by the model specified in Equation 5.14 with the parametrization given by Equation 5.16.

Note we do not have a unique measurement of the age of a given star. Rather, we have a distribution of ages $p_i(t)$ be it as a collection of MCMC samples, or in the form of a set of summary statistics for a given parametric distribution. Hence, the term $p(p_1(t)|\zeta'_{\boldsymbol{t}}, \hat{\zeta}'_{\boldsymbol{t_1}}, \Sigma_\varepsilon)$ instead of just $p(t_1|\zeta'_{\boldsymbol{t}}, \hat{\zeta}'_{\boldsymbol{t_1}}, \Sigma_\varepsilon)$. In practice, we evaluate this term as the expected likelihood under the posterior PDF of the ages produced by FLAME, according to

$$p(p_1|\zeta'_{\boldsymbol{t}}, \hat{\zeta}'_{\boldsymbol{t_1}}, \Sigma_\varepsilon) = \int_0^{13.8} p_1(t)\zeta'(t)\mathrm{d}t. \tag{5.19}$$

In order to test the HBM we have simulated 10,000 ages according to an ideal PDAD PDF defined as

$$\zeta'(t) = \frac{\gamma}{1 - \exp(-13.8 \cdot \gamma)} \exp(-\gamma t),$$

which is proposed in Czekaj, M. A. et al. (2014) with $\gamma = 0.12$. For every age in the simulation, 50 samples were drawn from a normal distribution with mean set as the true age and with a standard deviation of 0.1 (100 Myr). We do this because this is the expected output from FLAME: posterior samples. The mass and age posterior PDF will definitely be multimodal and/or asymmetric for a significant fraction of the stars observed by Gaia and thus, they cannot be summarized with the usual mean and variance summary statistics[3].

Figures 5.8 and 5.9 show the results obtained by executing the *emcee* algorithm to sample the posterior probabilities of the PDAD HBM parameters. The histogram in Figure 5.9 represents the distribution of the 10,000 simulated ages in 20 bins. Error bars are centred at the estimates of the $\zeta'_{\boldsymbol{t_1}}$ obtained using Kernel Density Estimation (KDE) via diffusion (Botev et al., 2010). The true ($\zeta'(t)$) and the estimated posterior mean ($\boldsymbol{\mu}^{\text{post}}$) PDAD PDF are drawn in green and red, respectively. Finally a confidence band (red ribbon) derived by using the diagonal of the matrix Σ^{post} in Equation 5.16 is also plotted.

[3] As stated above, the distributions delivered by FLAME will neither be normal (or log-normal), nor homoscedastic, as it has been assumed in this use case. We expect a variety of distributions from the observations performed by Gaia, i.e. one for each star. In particular, given the degeneracy of the inference problem solved by FLAME, we expect a significant fraction of their posterior distributions to be multimodal. However, this will not be a drawback for the model, but rather it will produce posterior distributions that are closer to the reality, showing real characteristics of the PDAD. This will empower a direct usage of the model with real observed data once it is available.

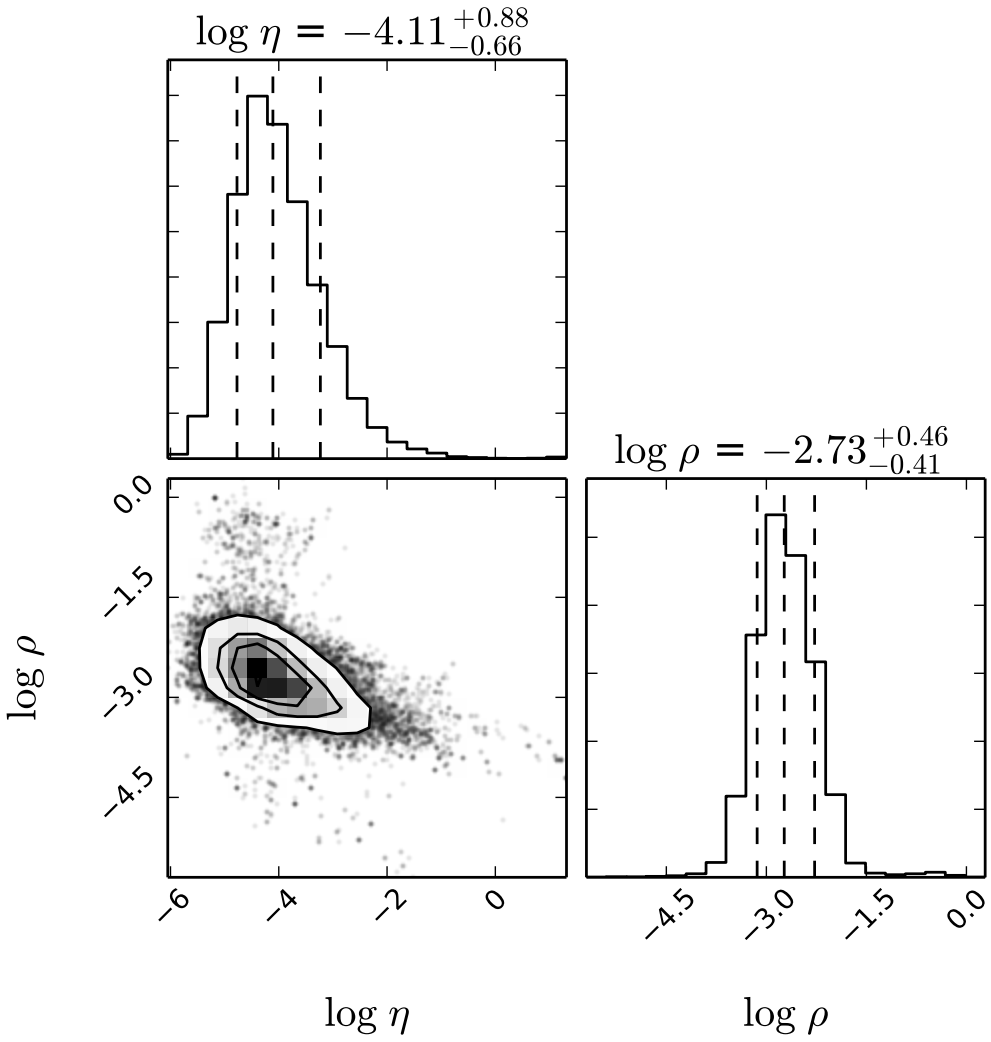


Figure 5.8: Posterior samples obtained by *emcee* for the hyperparameters η and ρ. Dashed lines represent the 0.16, 0.5 and 0.84 quantiles. The title above each 1-D histogram shows the 0.5 quantile together with the spread given by the 0.16 and 0.84 quantiles.

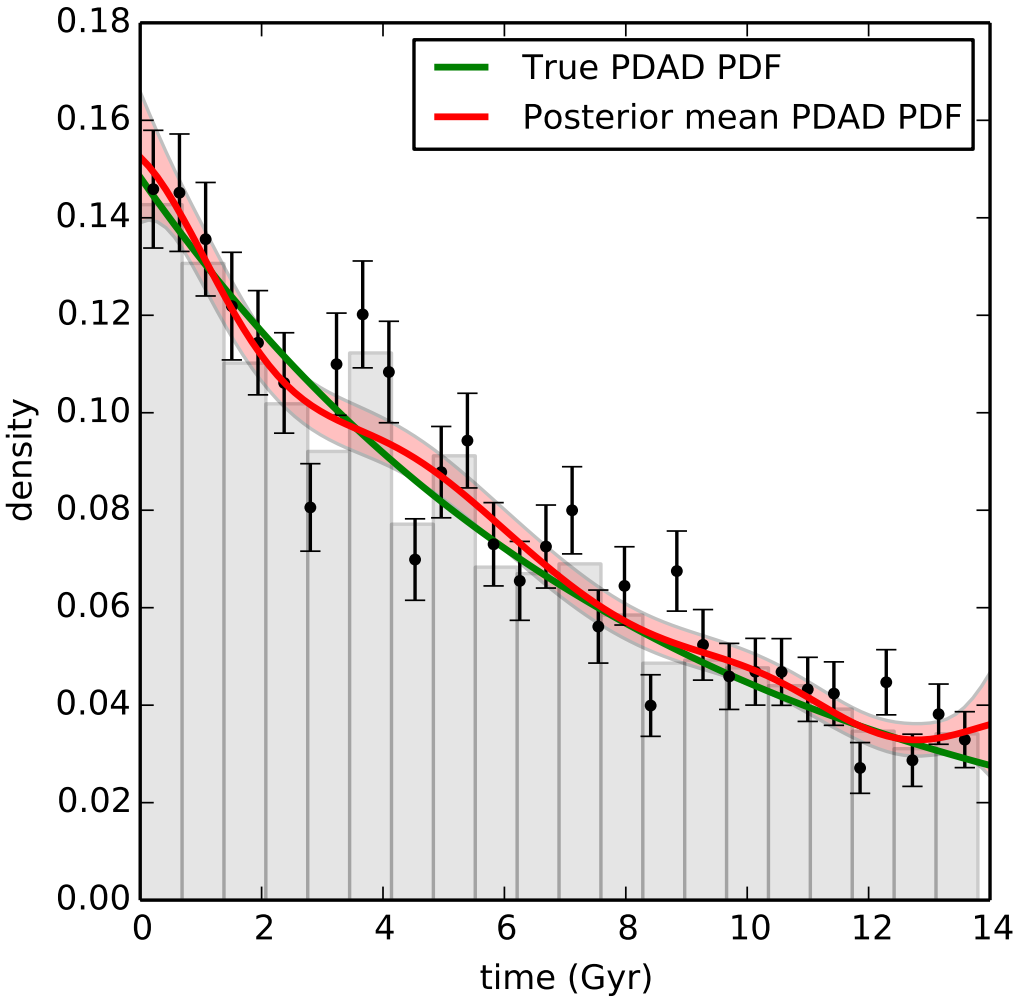


Figure 5.9: Histogram of the simulated ages. True and estimated PDAD PDF functions are shown in green and red respectively. The red ribbon represents the confindence band derived by using the diagonal of the matrix Σ^{post} in Equation 5.16.

5.5 Experiments

As previously discussed in Chapter 3, there are quite a few reasons why Apache Spark is a good candidate for implementing any kind of modeling that needs to scale to large data sets. Some of them obviously refer to its suitability for running iterative algorithms like those found in machine learning, or to the generality of the distributed processing engine (yet providing high level operations). However, the main enablers for a smooth adoption are probably the out of the box interoperability with already existing tools and frameworks (Hadoop ecosystem aside), and the possibility to have an interactive console where to try them out quickly (even locally). The fact that it provides several language bindings (Scala, Python, Java and R) makes it even more attractive. This has indeed been the case for this exercise with the PDMF and PDAD models. On one hand, we wanted to leverage a well known implementation of Goodman & Weare's Affine Invariant MCMC Ensemble sampler (Goodman and Weare, 2010), i.e. *emcee* (Foreman-Mackey et al., 2013), developed in Python. On the other hand, the scientific models built on top should be able to scale to the vast data sets produced by the Gaia mision without any modification or adaptation, further than configuration changes in the amount of resources allocated.

The programming paradigm proposed in Apache Spark allows us to accomplish both goals in an easy way. First of all, it is based on functional programming and monadic transformations (Wadler, 1995), which makes it very suitable for parallel and concurrent workloads due to, among other things, the immutability of the data (Hammond, 2011). Furthermore, programs are self contained, facilitating the integration with existing tools. We can then identify two parts in any program on top of Apache Spark: the code that is run on the *driver* (master node) which can only be scaled to several cores or CPU in the same host, and the operations performed over the data sets (Resilient Distributed Datasets, RDD (Zaharia et al., 2012)) that are run in parallel across the worker nodes. These transformations over the RDD can naturally scale up to more and more resources, thus being capable of dealing with bigger and bigger data sets. It is important to remark that other implementations (e.g. using MPI[4]) might be more efficient in terms of performance (lighter framework, more control and predictability of thread execution times, etc). However, the time to readiness would typically be much longer, and model maintenance or evolution costs would be higher, mainly because Apache Spark API hides the inherent complexity of a distributed system and is easier to experiment with, debug and evolve. This has precisely been the reason why MPI has not been benchmarked. Its development would have required a reimplementation of *emcee*, which would certainly be one step backwards, given that the goal is exactly the opposite, i.e. show that we can leverage already existing tools thus free scientists time and let them focus on the modeling, not on the intricacies of a distributed environment. Other approaches like MapReduce are not conceptually suitable for this problem as they read and write the data from/to disk in every iteration, which is again exactly the opposite of what is needed, i.e. let the data stay in memory so that the iterations can be performed faster.

[4] www.mpi-forum.org

In our scenario, for implementing these models with *emcee* on top of Apache Spark, we have leveraged the fact that *emcee* lets the user define the function to calculate the posterior probability for every MCMC iteration. Then, the workflow looks like the following:

- The data are loaded in the beginning of the execution as RDD and persisted (cached) in memory.
- The posterior probability function is developed by using the primitives provided by Apache Spark over the RDD. Those include operations for computing the equations laid out in Sections 5.3 and 5.4, typically calculating the likelihood of each star mass or age and then summing up the results.
- The *emcee* sampler object is built with the posterior probability function just developed, along with any arguments it may need (like constants) further than the parameters that will be created on each MCMC iteration. As previously remarked in Section 5.2, it is important to notice that *emcee* uses several internal samplers (known as *walkers*) that follow orthogonal paths to better explore the parameter space. The number of *walkers* has to be at least twice the number of parameters.
- The sampler is then started and runs a set of iterations as a *burn-in*, gets reset and starts sampling the parameter space based on the posterior probability of the parameters drawn.

Due to the fact that only the posterior probability function is parallelized through Apache Spark, the parameters are drawn in the master node. This may become a bottleneck for those use cases that require a huge amount of parameters, as the sampling of those parameters might effectively take longer than the computation of their associated posterior probability. This potential limitation might be easily alleviated by assigning more computing power to the *driver* (more cores). However, for a complete parallel approach leveraging Apache Spark when drawing parameter samples, some tweaking to *emcee* might be required. In our case, this originally posed some constraints for the PDMF model as one possible implementation might have one parameter per mass. This would not scale for the Gaia archive data set as we would need millions (or even billions) of parameters to be generated per iteration. The fact that the number of *walkers* must be at least twice the number of parameters would make it even harder. We tackled this limitation by setting a slightly different model that needed one parameter for each of the mass ranges (slopes) in the PDMF characteristic function (Chabrier, 2005). For the PDAD model, the number of parameters is again completely decoupled from the number of stars being considered, as we approached the problem in a similar way as for the PDMF (effectively sorting out any concern in this respect).

The workload of these problems is mostly computational, requiring very little memory or other complex I/O operations and data shufflings that may occur when joining or aggregating large amounts of data. The input data set size for the PDAD model (when executed with 4 million ages) is around 4.7 GB. This is further reduced in the early stages of the processing. The PDMF model input data sets are similar in size. In this scenario, enough memory to load the masses or ages of the stars will suffice, apart from the resources consumed when performing the sum of the individual likelihoods of each mass/age at the end of every iteration. However, the best configuration for such setup is not trivial as there are several parameters to be configured in order to get the best performance of the implementation. The methodology used for the experiments consisted in running each of the reported tests four times (with a representative number of walkers and iterations, typically 60) and then average out the results. The deployment used for testing and benchmarking comprises six nodes, each one having 16 cores, 64 GB of memory and 12 TB of disk. With this configuration, the maximum number of cores allocated for worker tasks was 80, considering one node was left out to act as the master (i.e. the *driver*).

We first tried to confirm that the tasks launched by Apache Spark (each one processing the likelihood of a set of records) were homogeneous in execution time. For that, we simply set a given number of executors (each executor runs in a separate Java Virtual Machine) and configured one core to each of them. In order to distribute the load among all executors/cores available, we explicitly repartitioned the RDD containing the masses/ages to create the same number of partitions, only to realize that it was not properly balanced because Apache Spark uses a hash partitioner by default, which in this case turned out to generate partitions with differences of up to 25% in the number of elements they contained, thus making execution times vary significantly. One way to alleviate this issue is to increase the number of partitions, so that the skew in the number of records per partition was adjusted. This did not smooth out the differences and in the end we set up the number of partitions at read time. A custom partitioner (following a round-robin distribution of records to partitions) would have been a better solution, but even though it is currently available in the Scala API, it has not yet been exposed in the Python one. It is worth remarking that Apache Spark automatically assigns one partition per HDFS block (at read time), typically 64 or 128 MB of data, which in our case would concentrate all masses/ages in very few partitions.

Another constraint faced throughout development and testing relates to the minimum amount of memory needed to start each executor (a bit less than 512 MB). If we set up one core per executor we would be wasting a lot of memory as the workload is purely computational. It then becomes necessary to set up more cores per executor so that this can be softened. Figure 5.10 shows the dependence of the average execution time per walker and iteration on the number of partitions used in the simulation. It proves that setting up more cores per executor is also beneficial in terms of performance, as having several cores per executor will help process more partitions at the same time, which will eventually lead to shorter execution times. Furthermore, setting less partitions than available cores is counterproductive as the allocated resources cannot be fully utilized in

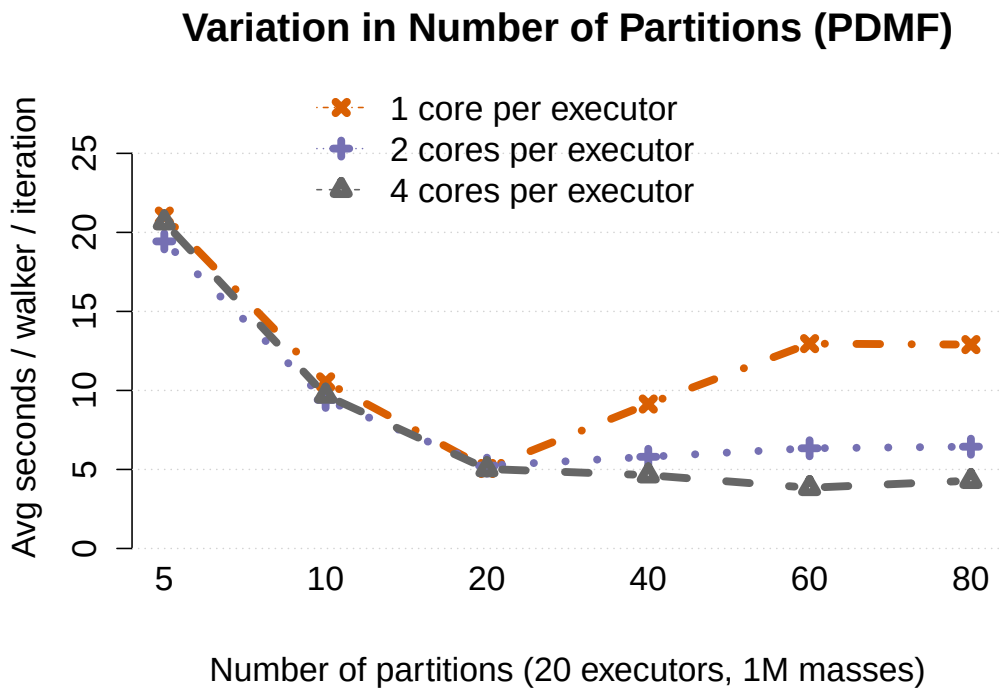


Figure 5.10: Average time per walker and iteration when increasing the number of cores per executor. The most cost effective configuration is when the number of partitions correspond to the number of executors times the number of cores per executor. Furthermore, the more cores per executor the smaller the total amount of memory allocated for this mainly computationally intensive workload.

the workflow. In addition, configuring more partitions than the number of executors times the number of cores per executor does not yield any significant improvement, probably due to the increasing latencies when summing the likelihoods at the end of each iteration (of every *walker*).

It is clear that a coarse-grained parallelism can be easily achieved with the described tools and engines, but the implementation of the PDMF model required a calculation of a complex integral (once per mass, walker and iteration). This meant that for a one million masses sample with six walkers and one thousand iterations, 6 billion integrals would need to be computed, and each of these computations would have to take place in a single core (i.e. not possible to parallelize in a multi-core way given the approach taken). We leveraged Python vectorization capabilities for this, and managed to reduce the time taken to approximate the integral (through the trapezoidal rule) by around 40 times with respect to the first naive approach (from 34 to 0.86 seconds per walker/iteration). Moreover, an increase of 10 times in the number of bins used for the integral computation (for more accurate results) only augmented 2.69 times the total execution time, confirming that vectorization should obviously be leveraged as much as possible in this sort of workflows (as briefly mentioned in Section 3.4). This way, there are two levels of parallelism that were taken advantage of, one coarser-grained with Apache Spark primitives, and the other (more fine-grained) through Python vectorization.

Figure 5.11 shows the speedup of the PDMF model implementation. There is a reduction in execution time as we increase the number of resources. As expected in any distributed system, this improvement decays as we have more and more workers collaborating in the job (which need to coordinate with each other). In this particular case, there are more and more likelihoods to be summed up, which are coming from different hosts, and even though this is an extremely fast operation as the sums are performed locally and only one number per machine gets shuffled, it gets reflected in the average time as each stage computation time per iteration/walker (just a few seconds) is not so high to make this negligible.

Even though reducing the execution time when increasing resources is important for any implementation over a distributed system, the scientific domain where the PDMF model will operate requires a good scaleup, i.e. keep the execution time similar when both the workload and the resources are increased proportionally. The results shown in Figure 5.12 confirm this is the case. The minor and gradual increase in the average time taken per walker and iteration is again due to latencies and other communication overheads produced when a greater number of resources have to cooperate in a determined task.

Figures 5.13 and 5.14 show the same pattern for the speedup and scaleup of the PDAD implementation, i.e. adding more resources results in shorter execution times and if we increase the workload and resources simultaneously, the execution time is kept relatively constant. It is also important to remark that previous implementations of the PDAD model set as many parameters as features K (see Section 5.4). Due to the way the *emcee* sampler works, it was required to have a large amount of walkers (at least twice the number of parameters but sometimes ten times are recommended for a proper exploration of the feature space), which turned out to be quite computationally

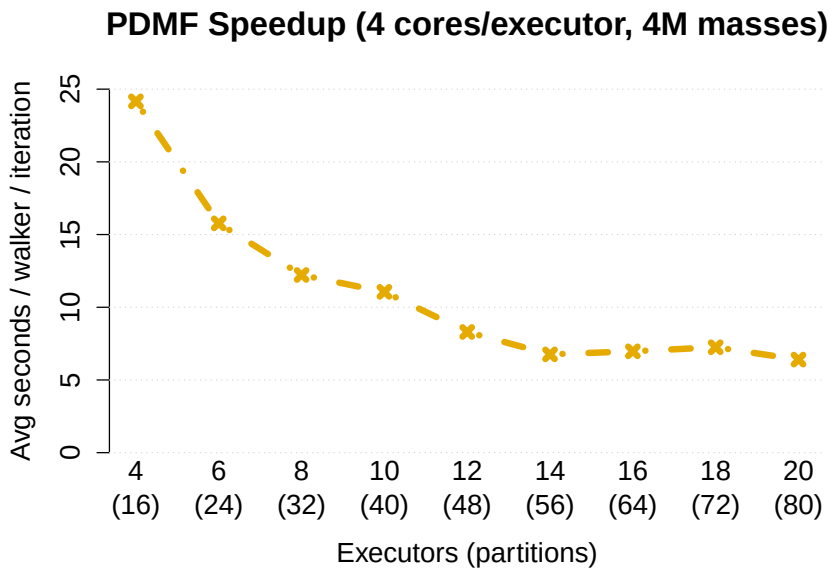


Figure 5.11: PDMF speedup with a four million masses data set. Adding resources decreases the average time taken per walker/iteration as the workload per partition/core is reduced. At some point (around 12 or 14 executors), the improvements are less abrupt, mainly due to the time taken to aggregate the likelihoods for the masses and any other latencies produced by the increase in the number of workers involved.

expensive. Some changes to *emcee* and/or Apache Spark would be needed to be able to scale up in these situations, because configuring several threads in the former (to allow the computation of all walkers to be sent in parallel to Apache Spark executors) raises an exception. This functionality would also be useful in situations where the work to be done per walker/iteration is really small (similar to the time taken to distribute the lambdas to the executors running on the worker nodes). Then, we would make it scale up by sending the computations of several walkers to Apache Spark executors at the same time (in parallel).

Last but not least, broadcast variables should be leveraged as much as possible, even for small reference variables (like vectors of a few elements). This will not only reduce the amount of data being sent over the network (that may be significant if the number of iterations is very high), but will also prevent memory issues or exceptions being raised because the serialized tasks are larger than the default maximum configured in Apache Spark.

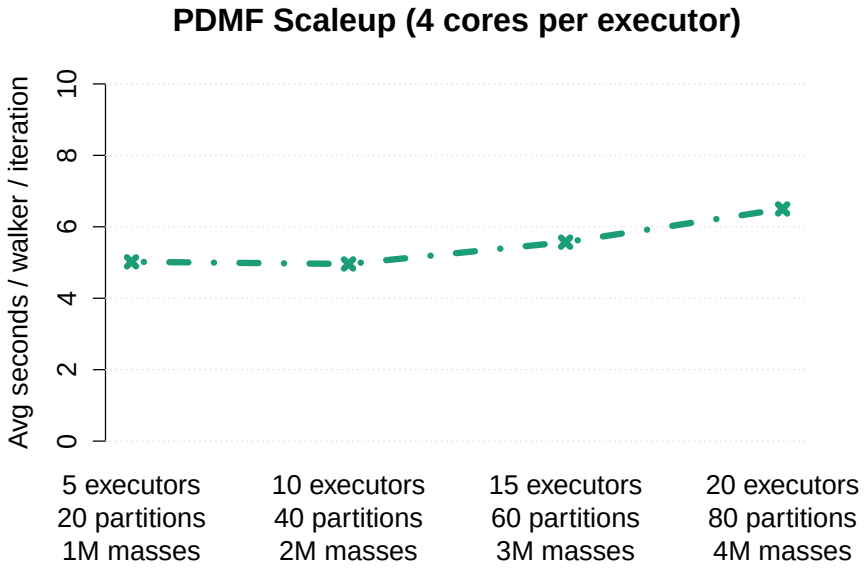


Figure 5.12: PDMF scaleup (reported with execution time per iteration instead of number of iterations per unit of time). The plot shows a scaleup very close to the linear (theoretical) one as we increase the data set and the resources proportionally.

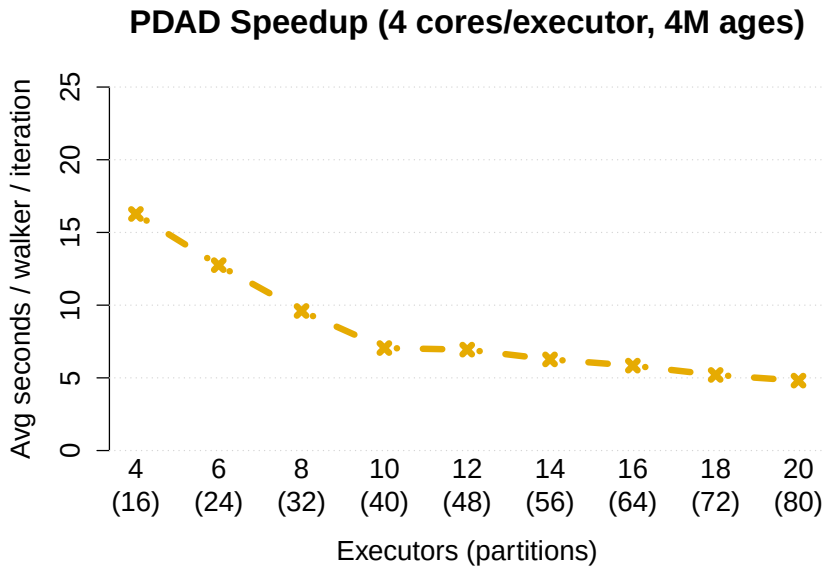


Figure 5.13: PDAD speedup with a four million ages data set.

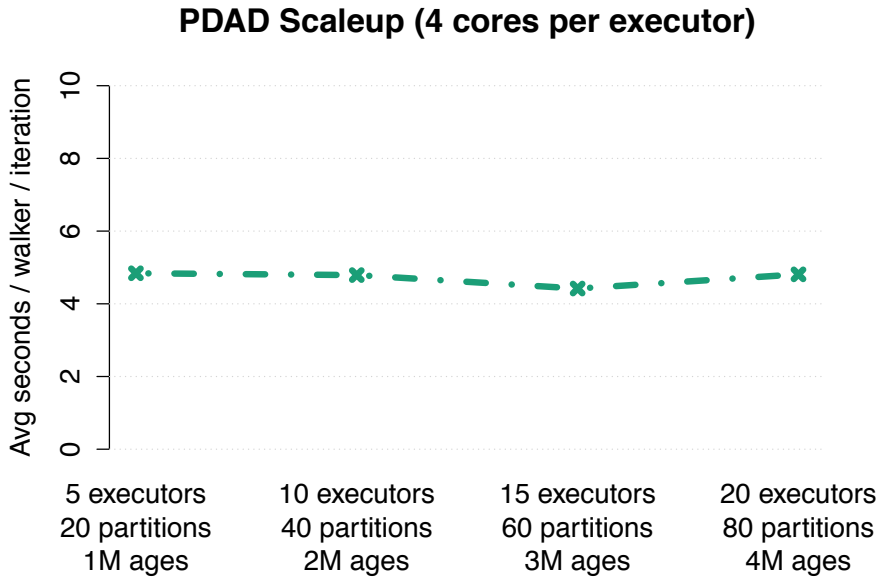


Figure 5.14: PDAD scaleup (reported with execution time per iteration instead of number of iterations per unit of time).

Chapter 6

Conclusions and Future Work

The original content of this thesis is diverse and twofold. On one hand, it addresses the challenges for facilitating the exploitation of massive scientific data archives. To accomplish this task, we present some innovative techniques and architectures that can be leveraged in complex analyses involving both the final data products, and the raw data before some refinements and reductions are performed by the SOC. These techniques include column-orientation for scientific data sets, the concept of bringing the software to the data, higher-level (and ad-hoc) frameworks for data aggregation, and how to approach a complex modeling that needs to scale to bigger data sets and reuse existing software packages. They have been validated in a variety of ESA missions like XMM-Newton, Herschel, and most importantly Gaia.

On the other hand, it presents the foundations of the so-called Grand Challenge, which focuses on an ambitious statistical problem that is inherent to the Gaia mission. This problem requires significant computing capabilities that can scale to the data sets that will be made available, and is a key step in the path to construct a full framework wherein scientists can deploy the most demanding analysis techniques for Knowledge Discovery in scientific archives. It consists in inferring a set of parameters from a probabilistic point of view for the PDMF and PDAD.

The main conclusions drawn as a result of this research can be enumerated as follows:

- A shift of the data analysis from the scientist environment towards the data centre. This widens the possibilities because there is more computing power available to the end user, lower latencies when accessing the data, on-the-fly and bulk reprocessing when new (or better) pipelines become available, and a better scalability and dimensioning of the architecture. The transformation has to be complete to be effective, i.e. the infrastructure hosting the data should be the one performing the analyses over it. These changes empower scientists to perform more complex analyses that will increase the scientific output.

- Higher level frameworks and ad-hoc implementations on top of Big Data tools and engines are crucial for new technological advances by the scientific community. Special attention has to be paid to the programming languages being exposed as

there is a non-negligible learning curve. Furthermore, these ad-hoc frameworks can produce better performance than other more generic solutions like Hive or Pig. This has to do with the well-known trade-off of performance versus generality. For analyses that will run operationally it makes sense to consider this approach, especially when the counterpart functionality in generic tools is not yet available. In addition, they can eliminate the gaps between the scientific and technological disciplines, boosting the adoption of cutting-edge technological advances.

- Data aggregation is a common operation in massive scientific archives. The techniques to be leveraged depend on certain domain specific factors like the key cardinality, the aggregation factor for each key and whether it is possible to perform several of them at the same time. The selection of generic approaches may be appropriate when the workload is not known a priori. However, in certain scenarios it is more suitable to develop custom tools and packages that better fit the domain. In the particular case studied in Section 4.3, a custom framework outperforms other generic solutions, even when the ad-hoc implementation leverages a worse algorithm (i.e. MergeSort) than the generic one (i.e. Hash aggregation) for the specific domain being considered. Furthermore, the development of a use case specific framework can work around the potential limitations of the lower level engine being leveraged, i.e. by implementing an aggregation technique that runs in memory (hash aggregation) on a purely batch, disk-based solution like MapReduce. One way or another, generic distributions eventually catch up with these improvements, but it is worth pursuing these custom developments when the workflow will become an operational and repeated task over time, leveraging a more control of the solution versus more uncertainty of the available features in generic tools.

- Column-orientation is crucial in any architecture that supports scientific workflows because scientific data sets are multidimensional and the pipelines accessing them do not normally need to access all available features for specific analyses. The scan time is thus reduced, but not only that, the compression ratio achieved is much higher given the similarity of adjacent data (e.g. it represents the same phenomena being observed in consecutive time stamps). A custom implementation is laid out as part of this research (when there was not any other one available on top of Hadoop), and the results prove significant performance gains when leverating this technique for data aggregation.

- Porting scientific data sets to Hadoop requires a thorough study of the different storage models. The compression and serialization techniques have to be carefully assessed. The conclusions drawn in the research show that lighter compression techniques like Google Snappy give a much better result for data analysis as the decompression is obviously faster than other approaches focusing on a better compression ratio. In the benchmarks performed, we observe a 50% reduction in execution time (workflows that will be run over and over again) with an increase of only 23% on the storage taken by the light compression technique. When we translate these percentages to on-demand Cloud environments like AWS (pay as

you go), we observe a significant reduction in the costs incurred, as storage is way much cheaper than the computing resources and also because the analyses to be performed will be repeated over and over again.

- As astronomical catalogues become larger and larger, the effective way of cross-matching them with others (e.g. at different wavelengths) is becoming a challenge. Furthermore, these scientific catalogues are normally skewed (much higher concentration of stars in certain areas) thus making its partitioning more difficult to accomplish. Techniques subdividing the sky in smaller spatial regions and then aggregating them into equally sized (and spatially close) partitions are assessed in a popular parallel DBMS. The data is then distributed by an identifier (hash-based) and the partitions are applied to the data placed in each of the nodes of the parallel architecture. This approach allows to perform a parallel cross-match of the astronomical sources that uses all computing resources available in the DBMS. Furthermore, this distribution and partitioning is suitable for analytical workloads that need all resources to participate even though the tasks are performed over a spatial subset of the catalogue. The test battery carried out shows that even for OLTP workloads (spatial queries), the performance is optimal, making this approach a good way of exposing catalogues for both query and analytical operations.

- Probabilistic models are one of the most demanding techniques for data analysis. We present an implementation that is linearly scalable and whose speedup is also linear, leveraging an existing software package for MCMC. This is the first step towards a full spatio-temporal Milky Way galaxy model from Gaia observations of billions of stars. The model is ready to start producing science with the first release of data coming from the Gaia mission. This research also enables a *learn by example* for other similar problems that require a significant amount of computing resources and validates the architecture that has been set up for knowledge discovery in the Gaia mission archive.

- An extensive performance and scalability analysis is performed for validating the suitability of the approach taken. In particular, different granularities of paralelism in modern general purpose distributed processing engines are leveraged, i.e. Apache Spark RDD for coarse-grained and CPU vectorization capabilities for fine-grained parallelism. Ensuring the evenness of the distribution of elements among the different partitions is crucial for achieving the best performance, as otherwise there may be some *stragglers* that increase the overall execution time significantly.

With regard to the future work enabled by this research, there are quite a few lines, both in the architectural and scientific streams. Those referred to the architectural techniques are enumerated below:

- Scientific archives are transitioning towards collaborative approaches, i.e. outreach oriented, interactive, interoperable, etc. This is something like a multi-mission scientific PaaS where scientists can perform their analyses and share the results

publicly. The definition of a generic and scalable architecture supports this to a certain extent, but further work and experimentation needs to be done to orchestrate all these features.

- Extend the research of the best approaches for data aggregation to current in-memory processing engines and contextualize to specific workflows with scientific data sets depending on the aggregation ratio and key cardinality, and most importantly studying how to switch from one approach to another on-the-fly.

- Apply columnar techniques to multidimensional data (e.g. spectra) and identify how and when to apply different compression techniques (even while in-memory) depending on its usage for data mining and the query patterns.

For the Grand Challenge, the future work includes:

- Usage of these models (alongside the proposed architecture) with real observations coming from the Gaia mission. This will lead to early scientific results. Furthermore, the approach taken in the implementation of these models will certainly spark other ideas, shortening the time required to develop them and ensuring the scalability to the enormous scientific data sets today available (and those to come).

- One of the most obvious improvements to the models presented in this thesis is the inclusion of the astrophysical ingredients left aside in this work for the sake of clarity. Including the observational biases is key for the scientific profitability of the results of these models.

- With regard to the framework of Bayesian analysis, we propose two lines of future research. The first one would consist in increasing the accuracy (and complexity) by assuming dependency between the variables representing the masses and ages. This can be achived by considering a generalization of a Poisson Process which includes, in the same stochastic process, the PDMF and PDAD PDF functions. In any case, it will be interesting to include the joint PDF for masses and ages given by FLAME in a general hierarchical model that could reveal probabilistic relationships between the PDMF and PDAD PDF. The second line of future improvements would require the inclusion of other sampling techniques in the framework, complementing the *emcee* sampler. Algorithms like Nested Sampling or the No-U-Turn Sampler (NUTS) (Hoffman and Gelman, 2011) can be adapted to deal with multimodal PDF, a strong requirement in this context of high level probabilistic inferences in scientific data sets.

- Another extension might be the proper integration of *emcee* with Apache Spark, so that it can be instantiated with several threads and thus scale up in the number of walkers, i.e. run several walkers at the same time on the same executors (not just the task belonging to one and then continue with the next in the same iteration).

- A benchmark against other architectures like Graphics Processing Unit (GPU) or MPI libraries would help assess whether there is any trade-off when considering the different approaches in terms of performance.

Bibliography

Abadi, D., Madden, S., Ferreira, M., 2006. Integrating compression and execution in column-oriented database systems. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. SIGMOD '06. ACM, New York, NY, USA, pp. 671–682.

Accomazzi, A., Budavári, T., Fluke, C., Gray, N., Mann, R. G., O'Mullane, W., Wicenec, A., Wise, M., 2013. Astronomy and computing: A new journal for the astronomical computing community. Astronomy and Computing 1, 1 – 4.
URL http://www.sciencedirect.com/science/article/pii/S2213133712000029

Agarwal, S., Iyer, A. P., Panda, A., Madden, S., Mozafari, B., Stoica, I., Aug. 2012. Blink and it's done: Interactive queries on very large data. Proc. VLDB Endow. 5 (12), 1902–1905.

Agarwal, S., Mozafari, B., Panda, A., Milner, H., Madden, S., Stoica, I., 2013. Blinkdb: Queries with bounded errors and bounded response times on very large data. In: Proceedings of the 8th ACM European Conference on Computer Systems. EuroSys '13. ACM, New York, NY, USA, pp. 29–42.

Ananthanarayanan, G., Ghodsi, A., Wang, A., Borthakur, D., Kandula, S., Shenker, S., Stoica, I., 2012. Pacman: Coordinated memory caching for parallel jobs. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12. USENIX Association, Berkeley, CA, USA, pp. 20–20.
URL http://dl.acm.org/citation.cfm?id=2228298.2228326

Anderson, C., 2008. The End of Theory: The Data Deluge Makes the Scientific Method Obsolete. Wired 16 (07).

Andrieu, C., de Freitas, N., Doucet, A., Jordan, M. I., 2003. An introduction to MCMC for machine learning. Machine Learning 50 (1), 5–43.

Angus, R., Kipping, D. M., May 2016. Probabilistic Inference of Basic Stellar Parameters: Application to Flickering Stars. The Astrophysical Journal Letters 823, L9.

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., Zaharia, M., 2015. Spark SQL: Relational data pro-

cessing in Spark. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. SIGMOD '15. ACM, New York, NY, USA, pp. 1383–1394.

Arviset, C., Alvarez, R., Gabriel, C., Osuna, P., Ott, S., Jul. 2011. Synergy Between Archives, VO, and the Grid at ESAC. In: Evans, I. N., Accomazzi, A., Mink, D. J., Rots, A. H. (Eds.), Astronomical Data Analysis Software and Systems XX. Vol. 442 of Astronomical Society of the Pacific Conference Series. p. 215.

Arviset, C., Barbarisi, I., de La Calle, I., Fajersztejn, N., Freschi, M., Gabriel, C., Gomez, P., Guainazzi, M., Ibarra, A., Laruelo, A., Leon, I., Micol, A., Parrilla, E., Ortiz, I., Osuna, P., Salgado, J., Stebe, A., Tapiador, D., Aug. 2008. ESA Science Archives, VO tools and remote Scientific Data reduction in Grid Architectures. In: Argyle, R. W., Bunclark, P. S., Lewis, J. R. (Eds.), Astronomical Data Analysis Software and Systems XVII. Vol. 394 of Astronomical Society of the Pacific Conference Series. p. 227.

Bailer-Jones, C. A. L., Andrae, R., Arcay, B., Astraatmadja, T., Bellas-Velidis, I., Berihuete, A., Bijaoui, A., Carrión, C., Dafonte, C., Damerdji, Y., Dapergolas, A., de Laverny, P., Delchambre, L., Drazinos, P., Drimmel, R., Frémat, Y., Fustes, D., García-Torres, M., Guédé, C., Heiter, U., Janotto, A.-M., Karampelas, A., Kim, D.-W., Knude, J., Kolka, I., Kontizas, E., Kontizas, M., Korn, A. J., Lanzafame, A. C., Lebreton, Y., Lindstrøm, H., Liu, C., Livanou, E., Lobel, A., Manteiga, M., Martayan, C., Ordenovic, C., Pichon, B., Recio-Blanco, A., Rocca-Volmerange, B., Sarro, L. M., Smith, K., Sordo, R., Soubiran, C., Surdej, J., Thévenin, F., Tsalmantza, P., Vallenari, A., Zorec, J., Nov. 2013. The Gaia astrophysical parameters inference system (Apsis). Pre-launch description. Astronomy and Astrophysics 559, A74.

Bar-Yam, Y., Aug. 2013. The Limits of Phenomenology: From Behaviorism to Drug Testing and Engineering Design. ArXiv e-prints.

Bell, C., Chen, R., Rege, S., 1972. Effect of technology on near term computer structures. Computer 5 (2), 29–38.

Bell, G., Jan. 2008. Bell's law for the birth and death of computer classes. Commun. ACM 51 (1), 86–94.
URL http://doi.acm.org/10.1145/1327452.1327453

Bishop, C. M., 2006. Pattern Recognition and Machine Learning. Springer.

Blanas, S., Patel, J. M., Ercegovac, V., Rao, J., Shekita, E. J., Tian, Y., 2010. A comparison of join algorithms for log processing in mapreduce. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. SIGMOD '10. ACM, New York, NY, USA, pp. 975–986.
URL http://doi.acm.org/10.1145/1807167.1807273

Bloom, B. H., Jul. 1970. Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13 (7), 422–426.
URL http://doi.acm.org/10.1145/362686.362692

Boch, T., Fernique, P., May 2014. Aladin Lite: Embed your Sky in the Browser. In: Manset, N., Forshay, P. (Eds.), Astronomical Data Analysis Software and Systems XXIII. Vol. 485 of Astronomical Society of the Pacific Conference Series. p. 277.

Borne, K. D., Jacoby, S., Carney, K., Connolly, A., Eastman, T., Raddick, M. J., Wallin, J., Becla, J., Castelez, M., Connors, A., Hamilton, T., Lintott, C., McCollum, B., Fox, P., Mahabal, A., Olsen, J., Pesenson, M., Ptak, A., Ross, N., Schweitzer, A., Teays, T., Way, M., Wood-Vasey, M., 2009. The Revolution in Astronomy Education: Data Science for the Masses. In: astro2010: The Astronomy and Astrophysics Decadal Survey. Vol. 2010 of ArXiv Astrophysics e-prints. p. 7P.

Botev, Z. I., Grotowski, J. F., Kroese, D. P., 10 2010. Kernel density estimation via diffusion. Ann. Statist. 38 (5), 2916–2957.
URL http://dx.doi.org/10.1214/10-AOS799

Britton, D., Lloyd, S. L., 2014. How to deal with petabytes of data: the LHC Grid project. Reports on Progress in Physics 77 (6), 065902.
URL http://stacks.iop.org/0034-4885/77/i=6/a=065902

Brock, D., Moore, G., 2007. Understanding Moore's law - four decades of innovation. Journal of Chemical Education 84 (8), 1278.
URL http://dx.doi.org/10.1021/ed084p1278

Brown, A. G., 2012. Science from Gaia: How to deal with a complex billion-source catalogue and data archive. In: Sarro, L. M., Eyer, L., O'Mullane, W., De Ridder, J. (Eds.), Astrostatistics and Data Mining. Vol. 2 of Springer Series in Astrostatistics. Springer New York, pp. 17–29.

Casella, G., George, E. I., 1992. Explaining the Gibbs sampler. The American Statistician 46, 167–174.

Ceballos, M. T., Campos, I., Orviz, P., Tapiador, D., Álvarez, R., Ibarra, A., Gabriel, C., Rodón, J. R., 2010. XMM–Newton Data Analysis with SAS Software Over Grid Technology. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 479–479.
URL http://dx.doi.org/10.1007/978-3-642-11250-8_136

Chabrier, G., Jul. 2003. Galactic Stellar and Substellar Initial Mass Function. Publications of the Astronomical Society of the Pacific 115, 763–795.

Chabrier, G., Jan. 2005. The Initial Mass Function: from Salpeter 1955 to 2005. In: E (Ed.), The Initial Mass Function 50 Years Later. Vol. 327 of Astrophysics and Space Science Library.

Cleveland, W. S., Apr. 2001. Data Science: an Action Plan for Expanding the Technical Areas of the Field of Statistics. International Statistical Review 69 (1), 21–26.
URL http://dx.doi.org/10.1111/j.1751-5823.2001.tb00477.x

Council, N. R., 1995. Preserving Scientific Data on Our Physical Universe: A New Strategy for Archiving the Nation's Scientific Information Resources.

Czekaj, M. A., Robin, A. C., Figueras, F., Luri, X., Haywood, M., 2014. The Besançon Galaxy model renewed. A&A 564, A102.
URL http://dx.doi.org/10.1051/0004-6361/201322139

Dagum, L., Menon, R., Jan. 1998. OpenMP: An industry-standard API for shared-memory programming. IEEE Comput. Sci. Eng. 5 (1), 46–55.
URL http://dx.doi.org/10.1109/99.660313

de Bruijne, J. H. J., Sep. 2012. Science performance of Gaia, ESA's space-astrometry mission. Astrophysics and Space Science 341, 31–41.

Dean, J., Ghemawat, S., Jan. 2008. Mapreduce: simplified data processing on large clusters. Commun. ACM 51 (1), 107–113.

Dewdney, P., Hall, P., Schilizzi, R., Lazio, T., aug. 2009. The Square Kilometre Array. Proceedings of the IEEE 97 (8), 1482 –1496.

Dowler, P., Rixon, G., Tody, D., Mar. 2010. Table Access Protocol Version 1.0. IVOA Recommendation 27 March 2010.

Dries, M., Trager, S. C., Koopmans, L. V. E., Nov. 2016. A hierarchical Bayesian approach for reconstructing the initial mass function of single stellar populations. Monthly Notices of the Royal Astronomical Society 463, 886–912.

Elmegreen, B. G., Scalo, J., 2006. The effect of star formation history on the inferred stellar initial mass function. The Astrophysical Journal 636 (1), 149.

Elmeleegy, K., Aug. 2013. Piranha: Optimizing short jobs in hadoop. Proc. VLDB Endow. 6 (11), 985–996.
URL http://dx.doi.org/10.14778/2536222.2536225

Fajersztejn, N., Arviset, C., Baines, D., Barbarisi, I., Castellanos, J., Cheek, N., Costa, H., Fernandez, M., Gonzalez, J., Laruelo, A., Leon, I., Martinez, B., Ortiz, I., Osuna, P., Rios, C., Salgado, J., Sarmiento, M. H., Tapiador, D., Jul. 2011. Storing Data in Science Archives: Striving for a Common Architecture. In: Evans, I. N., Accomazzi, A., Mink, D. J., Rots, A. H. (Eds.), Astronomical Data Analysis Software and Systems XX. Vol. 442 of Astronomical Society of the Pacific Conference Series.

Fernandez, M., Arviset, C., Barbarisi, I., Castellanos, J., Cheek, N., Costa, H., Fajersztejn, N., Gonzalez, J., Laruelo, A., Leon, I., Ortiz, I., Osuna, P., Salgado, J., Stebe, A., Tapiador, D., Dec. 2010. ESA New Generation Science Archives: New Technologies Applied to Graphical User Interface Creation. In: Mizumoto, Y., Morita, K.-I., Ohishi, M. (Eds.), Astronomical Data Analysis Software and Systems XIX. Vol. 434 of Astronomical Society of the Pacific Conference Series. p. 313.

Floratou, A., Patel, J. M., Shekita, E. J., Tata, S., Apr. 2011. Column-oriented storage techniques for mapreduce. Proc. VLDB Endow. 4 (7), 419–429.

Foreman-Mackey, D., 2016. corner.py: Scatterplot matrices in Python. The Journal of Open Source Software 24.
URL http://dx.doi.org/10.5281/zenodo.45906

Foreman-Mackey, D., Hogg, D. W., Lang, D., Goodman, J., 2013. emcee: The MCMC hammer. Publications of the Astronomical Society of the Pacific 125 (925), 306–312.
URL http://www.jstor.org/stable/10.1086/670067

Foster, I., 1995. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Foster, I., 2005. Globus toolkit version 4: Software for service-oriented systems. In: Proceedings of the 2005 IFIP International Conference on Network and Parallel Computing. NPC'05. Springer-Verlag, Berlin, Heidelberg, pp. 2–13.
URL http://dx.doi.org/10.1007/11577188_2

Foster, I., Kesselman, C. (Eds.), 1999. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Foster, I., Kesselman, C., Nick, J. M., Tuecke, S., 2002. The physiology of the Grid: An open grid services architecture for distributed systems integration.

Foster, I., Kesselman, C., Tuecke, S., Aug. 2001. The anatomy of the Grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. 15 (3), 200–222.
URL http://dx.doi.org/10.1177/109434200101500302

Foster, J. G., Rzhetsky, A., Evans, J. A., 2013. Tradition and innovation in scientists' research strategies. Computing Research Repository abs/1302.6906.
URL http://dblp.uni-trier.de/db/journals/corr/corr1302.html#abs-1302-6906

Gabriel, C., Ibarra, A., de La Calle, I., Salgado, J., Osuna, P., Tapiador, D., Aug. 2008. RISA: Remote Interface for Science Analysis. In: Argyle, R. W., Bunclark, P. S., Lewis, J. R. (Eds.), Astronomical Data Analysis Software and Systems XVII. Vol. 394 of Astronomical Society of the Pacific Conference Series. p. 183.

Gaia Collaboration, Brown, A. G. A., Vallenari, A., Prusti, T., de Bruijne, J. H. J., Mignard, F., Drimmel, R., Babusiaux, C., Bailer-Jones, C. A. L., Bastian, U., et al., Nov. 2016a. Gaia Data Release 1. Summary of the astrometric, photometric, and survey properties. Astronomy and Astrophysics 595, A2.

Gaia Collaboration, Prusti, T., de Bruijne, J. H. J., Brown, A. G. A., Vallenari, A., Babusiaux, C., Bailer-Jones, C. A. L., Bastian, U., Biermann, M., Evans, D. W., et al., Nov. 2016b. The Gaia mission. Astronomy and Astrophysics 595, A1.

Gaia Collaboration, van Leeuwen, F., Vallenari, A., Jordi, C., Lindegren, L., Bastian, U., Prusti, T., de Bruijne, J. H. J., Brown, A. G. A., Babusiaux, C., et al., Mar. 2017. Gaia Data Release 1. Open cluster astrometry: performance, limitations, and future prospects. ArXiv e-prints.

Gaudet, S., Hill, N., Armstrong, P., Ball, N., Burke, J., Chapel, B., Chapin, E., Damian, A., Dowler, P., Gable, I., Goliath, S., Ghiurea, I., Fabbro, S., Gwyn, S., Jenkins, D., Kavelaars, J., Major, B., Ouellette, J., Paterson, M., Peddle, M., Penfold-Brown, D., Pritchet, C., Schade, D., Sobie, R., Woods, D., Yeung, A., Zhang, Y., Jul. 2010. CANFAR: the Canadian Advanced Network for Astronomical Research. In: Software and Cyberinfrastructure for Astronomy. Vol. 7740 of Proceedings of the International Society for Optical Engineering. p. 77401I.

Geddes, N., 2012. The large hadron collider and grid computing. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 370 (1961), 965–977.

Gelman, A., Carlin, J. B., Stern, H. S., Rubin, D. B., 2013. Bayesian Data Analysis, Third Edition. Chapman and Hall/CRC.

Gentzsch, W., 2001. Sun grid engine: Towards creating a compute power grid. In: Proceedings of the 1st International Symposium on Cluster Computing and the Grid. CCGRID '01. IEEE Computer Society, Washington, DC, USA, pp. 35–.
URL http://dl.acm.org/citation.cfm?id=560889.792378

Ghandeharizadeh, S., DeWitt, D. J., 1990. Hybrid-range partitioning strategy: A new declustering strategy for multiprocessor database machines. In: Proceedings of the 16th International Conference on Very Large Data Bases. VLDB '90. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 481–492.
URL http://dl.acm.org/citation.cfm?id=645916.671988

Ghandeharizadeh, S., DeWitt, D. J., May 1994. Magic: A multiattribute declustering mechanism for multiprocessor database machines. IEEE Trans. Parallel Distrib. Syst. 5 (5), 509–524.
URL http://dx.doi.org/10.1109/71.282561

Ghemawat, S., Gobioff, H., Leung, S.-T., 2003. The google file system. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. SOSP '03. ACM, New York, NY, USA, pp. 29–43.
URL http://doi.acm.org/10.1145/945445.945450

Ghoshal, D., Canon, R. S., Ramakrishnan, L., 2011. I/o performance of virtualized cloud environments. In: Proceedings of the Second International Workshop on Data Intensive Computing in the Clouds. DataCloud-SC '11. ACM, New York, NY, USA, pp. 71–80.
URL http://doi.acm.org/10.1145/2087522.2087535

Gilbert, S., Lynch, N., Jun. 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33 (2), 51–59.
URL http://doi.acm.org/10.1145/564585.564601

Goodman, J., Weare, J., Jan. 2010. Ensemble samplers with affine invariance. Communications in Applied Mathematics and Computational Science 5 (1), 65–80.
URL http://dx.doi.org/10.2140/camcos.2010.5.65

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., Bartelmann, M., 2005. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. The Astrophysical Journal 622 (2), 759.

Graham, M., 2012. The art of data science. In: Sarro, L. M., Eyer, L., O'Mullane, W., De Ridder, J. (Eds.), Astrostatistics and Data Mining. Vol. 2 of Springer Series in Astrostatistics. Springer New York, pp. 47–59.
URL http://dx.doi.org/10.1007/978-1-4614-3323-1_4

Guo, Z., Fox, G., Zhou, M., 2012. Investigation of data locality in mapreduce. In: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012). CCGRID '12. IEEE Computer Society, Washington, DC, USA, pp. 419–426.
URL http://dx.doi.org/10.1109/CCGrid.2012.42

Hammond, K., 2011. Reliable software technologies - ada-europe 2011: 16th ada-europe international conference on reliable software technologies, edinburgh, uk, june 20-24, 2011. proceedings. Springer Berlin Heidelberg, Ch. Why Parallel Functional Programming Matters: Panel Statement, pp. 201–205.

Hanisch, R., 2015. Astronomy and Computing Special Issue: The Virtual Observatory: II. Elsevier Science, volume 11, Part B, Pages 73-210.
URL http://www.sciencedirect.com/science/journal/22131337/11/part/PB

Herbst, N. R., Kounev, S., Reussner, R., 2013. Elasticity in cloud computing: What it is, and what it is not. In: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13). USENIX, San Jose, CA, pp. 23–27.
URL https://www.usenix.org/conference/icac13/technical-sessions/presentation/herbst

Hoffman, M. D., Gelman, A., Nov. 2011. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. ArXiv e-prints.

Høg, E., Fabricius, C., Makarov, V. V., Urban, S., Corbin, T., Wycoff, G., Bastian, U., Schwekendiek, P., Wicenec, A., Mar. 2000. The Tycho-2 catalogue of the 2.5 million brightest stars. Astronomy and Astrophysics 355, L27–L30.

Huai, Y., Chauhan, A., Gates, A., Hagleitner, G., Hanson, E. N., O'Malley, O., Pandey, J., Yuan, Y., Lee, R., Zhang, X., 2014. Major technical advancements in apache hive. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. SIGMOD '14. ACM, New York, NY, USA, pp. 1235–1246.
URL http://doi.acm.org/10.1145/2588555.2595630

Huedo, E., Montero, R. S., Llorente, I. M., 2004. A framework for adaptive execution in grids. Softw., Pract. Exper. 34 (7), 631–651.
URL http://dblp.uni-trier.de/db/journals/spe/spe34.html#HuedoML04

Ibarra, A., de La Calle, I., Tapiador, D., Loiseau, N., Gabriel, J., C. S., Osuna, P., Oct. 2007. Remote Interface to Science Analysis Tools for Grid Architecture: The XMM-Newton SAS Case. In: Shaw, R. A., Hill, F., Bell, D. J. (Eds.), Astronomical Data Analysis Software and Systems XVI. Vol. 376 of Astronomical Society of the Pacific Conference Series. p. 85.

Ibarra, A., Tapiador, D., Félix, F., Gabriel, C., Arviset, C., Hoar, J., Ansari, S., Jul. 2006. Optimizing SAS tasks for Grid Architectures. In: Gabriel, C., Arviset, C., Ponz, D., Enrique, S. (Eds.), Astronomical Data Analysis Software and Systems XV. Vol. 351 of Astronomical Society of the Pacific Conference Series. p. 520.

Ibarra, A., Tapiador, D., Huedo, E., Montero, R. S., Gabriel, C., Arviset, C., Llorente, I. M., Apr. 2006. On-the-fly XMM-Newton spacecraft data reduction on the Grid. Sci. Program. 14 (2), 141–150.
URL http://dx.doi.org/10.1155/2006/739583

Iqbal, S. A., Wallach, J. D., Khoury, M. J., Schully, S. D., Ioannidis, J. P. A., 01 2016. Reproducible research practices and transparency across the biomedical literature. PLoS Biol 14 (1), 1–13.
URL http://dx.doi.org/10.1371%2Fjournal.pbio.1002333

Ivezic, Z., Tyson, J. A., May 2008. LSST: from Science Drivers to Reference Design and Anticipated Data Products. ArXiv e-prints.

Joszczuk-Januszewska, J., 2010. Development trends in intelligent transport systems in respect to environmental protection. In: Mikulski, J. (Ed.), Transport Systems Telematics. Vol. 104 of Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 183–193.
URL http://dx.doi.org/10.1007/978-3-642-16472-9_20

Keahey, K., Foster, I., Freeman, T., Zhang, X., Oct. 2005a. Virtual workspaces: Achieving quality of service and quality of life in the grid. Sci. Program. 13 (4), 265–275.
URL http://dx.doi.org/10.1155/2005/351408

Keahey, K., Foster, I., Freeman, T., Zhang, X., Galron, D., 2005b. Euro-Par 2005 Parallel Processing: 11th International Euro-Par Conference, Lisbon, Portugal, August 30 - September 2, 2005. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, Ch.

Virtual Workspaces in the Grid, pp. 421–431.
URL http://dx.doi.org/10.1007/11549468_49

King, T., Merka, J., Walker, R., Joy, S., Narock, T., 2008. The architecture of a multi-tiered virtual observatory. Earth Science Informatics 1 (1), 21–28.
URL http://dx.doi.org/10.1007/s12145-008-0006-3

Koposov, S., Bartunov, O., Jul. 2006. Q3C, Quad Tree Cube – The new Sky-indexing Concept for Huge Astronomical Catalogues and its Realization for Main Astronomical Queries (Cone Search and Xmatch) in Open Source Database PostgreSQL. In: Gabriel, C., Arviset, C., Ponz, D., Enrique, S. (Eds.), Astronomical Data Analysis Software and Systems XV. Vol. 351 of Astronomical Society of the Pacific Conference Series. p. 735.

Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., Erickson, J., Grund, M., Hecht, D., Jacobs, M., Joshi, I., Kuff, L., Kumar, D., Leblang, A., Li, N., Pandis, I., Robinson, H., Rorke, D., Rus, S., Russell, J., Tsirogiannis, D., Wanderman-Milne, S., Yoder, M., 2015. Impala: A modern, open-source sql engine for hadoop. In: Proceedings CIDR'15.

Kraska, T., Talwalkar, A., Duchi, J. C., Griffith, R., Franklin, M. J., Jordan, M. I., 2013. Mlbase: A distributed machine-learning system. In: CIDR. www.cidrdb.org.

Kroupa, P., 2002. The initial mass function of stars: Evidence for uniformity in variable systems. Science 295 (5552), 82–91.
URL http://science.sciencemag.org/content/295/5552/82

Krueger, J., Grund, M., Tinnefeld, C., Plattner, H., Zeier, A., Faerber, F., 2010. Optimizing write performance for read optimized databases. In: Proceedings of the 15th international conference on Database Systems for Advanced Applications - Volume Part II. DASFAA'10. Springer-Verlag, Berlin, Heidelberg, pp. 291–305.

Kwon, Y., Balazinska, M., Howe, B., Rolia, J., 2011. A study of skew in mapreduce applications. In: 5th Open Cirrus Summit.

Laureijs, R., Amiaux, J., Arduini, S., Augueres, J. ., Brinchmann, J., Cole, R., Cropper, M., Dabin, C., Duvet, L., Ealet, A., et al., Oct. 2011. Euclid Definition Study Report. ArXiv e-prints.

Laurent, A. M. S., 2004. Understanding Open Source and Free Software Licensing. O'Reilly Media, Inc.

Lavoie, B. F., January 2004. The Open Archival Information System Reference Model: Introductory Guide. Tech. rep., Digital Preservation Coalition/OCLC.
URL http://www.dpconline.org/docs/lavoie_OAIS.pdf

Leon, I., Arviset, C., Baines, D., Barbarisi, I., Castellanos, J., Cheek, N., Costa, H., Fajersztejn, N., Fernandez, M., Gonzalez, J., Laruelo, A., Ortiz, I., Osuna, P., Salgado,

J., Stebe, A., Tapiador, D., Dec. 2010. ESA New Generation Science Archives: State of the Art Data Management Techniques for SOHO and EXOSAT Science Archives. In: Mizumoto, Y., Morita, K.-I., Ohishi, M. (Eds.), Astronomical Data Analysis Software and Systems XIX. Vol. 434 of Astronomical Society of the Pacific Conference Series. p. 201.

Li, J., Ray, S., Lindsay, B. G., Dec. 2007. A Nonparametric Statistical Approach to Clustering via Mode Identification. J. Mach. Learn. Res. 8, 1687–1723.
URL http://dl.acm.org/citation.cfm?id=1314498.1314555

Li, N., Thakar, A. R., 2008. Casjobs and mydb: A batch query workbench. Computing in Science and Engineering 10 (1), 18–29.

Lin, J., Schatz, M., 2010. Design patterns for efficient graph algorithms in mapreduce. In: Proceedings of the Eighth Workshop on Mining and Learning with Graphs. MLG '10. ACM, New York, NY, USA, pp. 78–85.
URL http://doi.acm.org/10.1145/1830252.1830263

Luri, X., O'Mullane, W., Alves, J., Arenou, F., Brown, A., Els, S., Hambly, N., Helmi, A., Jordan, S., Krone-Martins, A., van Leeuwen, F., Masana, E., Di Matteo, P., Mercier, E., Moitinho, A., Osuna, P., Salgado, J., Tapiador, D., Walton, N., February 2013. Delivering the promise of Gaia, response to ESA's Announcement of Opportunity, GAIA-C9-PL-UB-XL-033.
URL http://www.rssd.esa.int/doc_fetch.php?id=3165853

Mandel, K. S., Scolnic, D., Shariff, H., Foley, R. J., Kirshner, R. P., Sep. 2016. The Type Ia Supernova Color-Magnitude Relation and Host Galaxy Dust: A Simple Hierarchical Bayesian Model. ArXiv e-prints.

Marr, B., 2015. Big Data: Using SMART Big Data, Analytics and Metrics To Make Better Decisions and Improve Performance. Wiley.
URL https://books.google.ae/books?id=OfglBgAAQBAJ

Mell, P. M., Grance, T., 2011. Sp 800-145. the nist definition of cloud computing. Tech. rep., Gaithersburg, MD, United States.

Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., Vassilakis, T., 2010. Dremel: Interactive analysis of web-scale datasets. In: Proc. of the 36th Int'l Conf on Very Large Data Bases. pp. 330–339.
URL http://www.vldb2010.org/accept.htm

Merín, B., Salgado, J., Giordano, F., Baines, D., Sarmiento, M.-H., López Martí, B., Racero, E., Gutiérrez, R., Pollock, A., Rosa, M., Castellanos, J., González, J., León, I., Ortiz de Landaluce, I., de Teodoro, P., Nieto, S., Lennon, D. J., Arviset, C., de Marchi, G., O'Mullane, W., Dec. 2015. ESA Sky: a new Astronomy Multi-Mission Interface. ArXiv e-prints.

Mignard, F., Jan. 2005. Overall Science Goals of the Gaia Mission. In: Turon, C., O'Flaherty, K. S., Perryman, M. A. C. (Eds.), ESA SP-576: The Three-Dimensional Universe with Gaia. pp. 5–+.

Mignard, F., Drimmel, R., April 2007. DPAC: Proposal for the Gaia Data Processing, GAIA-CD-SP-DPAC-FM-030-02.
URL http://www.rssd.esa.int/doc_fetch.php?id=2720336

Miller, G. E., Scalo, J. M., Nov. 1979. The initial mass function and stellar birthrate in the solar neighborhood. The Astrophysical Journal, Supplement 41, 513–547.

Moore, G. E., 2000. Readings in computer architecture. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Ch. Cramming More Components Onto Integrated Circuits, pp. 56–59.
URL http://dl.acm.org/citation.cfm?id=333067.333074

Ochsenbein, F., Bauer, P., Marcout, J., Apr. 2000. The VizieR database of astronomical catalogues. aaps 143, 23–32.

Ochsenbein, F., Williams, R., Nov. 2009. VOTable Format Definition Version 1.2. IVOA Recommendation 30 November 2009.

O'Leary, D. E., 2014. Embedding AI and crowdsourcing in the Big Data lake. IEEE Intelligent Systems 29 (5), 70–73.

O'Mullane, W., 2011. Blue skies and clouds, archives of the future. Tech. Rep. GAIA-TN-PL-ESAC-WOM-057-01, Gaia Science Operations Centre, European Space Agency.

O'Mullane, W., Lammers, U., Lindegren, L., Hernandez, J., Hobbs, D., 2011. Implementing the gaia astrometric global iterative solution (AGIS) in Java. Experimental Astronomy 31 (2), 215–241.
URL http://dx.doi.org/10.1007/s10686-011-9248-z

O'Mullane, W., Li, N., Nieto-Santisteban, M., Szalay, A., Thakar, A., Gray, J., July 2005. Batch is back: Casjobs, serving multi-TB data on the Web. In: IEEE International Conference on Web Services (ICWS'05). pp. 33–40 vol.1.

Osuna, P., Arviset, C., Baines, D., Barbarisi, I., Castellanos, J., Cheek, N., Costa, H., Fajersztejn, N., Fernandez, M., Gonzalez, J., Laruelo, A., Leon, I., Ortiz, I., Salgado, J., Stebe, A., Tapiador, D., Dec. 2010. ESA New Generation Science Archives: SOHO and EXOSAT. In: Mizumoto, Y., Morita, K.-I., Ohishi, M. (Eds.), Astronomical Data Analysis Software and Systems XIX. Vol. 434 of Astronomical Society of the Pacific Conference Series. p. 3.

Osuna, P., Ortiz, I., Lusted, J., Dowler, P., Szalay, A., Shirasaki, Y., Nieto-Santisteban, M. A., Ohishi, M., O'Mullane, W., VOQL-TEG Group, VOQL Working Group., Oct. 2008. IVOA Astronomical Data Query Language Version 2.00. IVOA Recommendation 30 October 2008.

Page, L., Brin, S., Motwani, R., Winograd, T., November 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, previous number = SIDL-WP-1999-0120.
URL http://ilpubs.stanford.edu:8090/422/

Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D. J., Madden, S., Stonebraker, M., 2009. A comparison of approaches to large-scale data analysis. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. SIGMOD '09. ACM, New York, NY, USA, pp. 165–178.

Peng, R. D., 2011. Reproducible research in computational science. Science 334 (6060), 1226–1227.
URL http://science.sciencemag.org/content/334/6060/1226

Perryman, M. A. C., Lindegren, L., Kovalevsky, J., Hoeg, E., Bastian, U., Bernacca, P. L., Crézé, M., Donati, F., Grenon, M., Grewing, M., van Leeuwen, F., van der Marel, H., Mignard, F., Murray, C. A., Le Poole, R. S., Schrijver, H., Turon, C., Arenou, F., Froeschlé, M., Petersen, C. S., Jul. 1997. The HIPPARCOS Catalogue. Astronomy and Astrophysics 323.

Perryman, M. A. C., et al., 1997. The Hipparcos and Tycho Catalogues. Astrometric and Photometric Star Catalogues derived from the ESA Hipparcos Space Astrometry Mission. European Space Agency Publications Division, c/o ESTEC Noordwijk, The Netherlands.

Pilbratt, G. L., 2008. Herschel mission overview and key programmes.
URL http://dx.doi.org/10.1117/12.789431

Planck Collaboration, Ade, P. A. R., Aghanim, N., Arnaud, M., Ashdown, M., Aumont, J., Baccigalupi, C., Baker, M., Balbi, A., Banday, A. J., et al., Dec. 2011. Planck early results. I. The Planck mission. Astronomy and Astrophysics 536, A1.

Planck Collaboration, Ade, P. A. R., Aghanim, N., Arnaud, M., Ashdown, M., Aumont, J., Baccigalupi, C., Banday, A. J., Barreiro, R. B., Bartlett, J. G., et al., Feb. 2015. Planck 2015 results. XIII. Cosmological parameters. ArXiv e-prints.

Pryor, G., 2012. Why manage research data? In: Managing research data. Facet Publishing London, pp. 1–16.

Resch, M., Küster, U., 2008. Hpc in industrial environments. In: Krause, E., Shokin, Y., Resch, M., Shokina, N. (Eds.), Computational Science and High Performance Computing III. Vol. 101 of Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Springer Berlin Heidelberg, pp. 8–13.
URL http://dx.doi.org/10.1007/978-3-540-69010-8_2

Robin, A. C., Luri, X., Reylé, C., Isasi, Y., Grux, E., Blanco-Cuaresma, S., Arenou, F., Babusiaux, C., Belcheva, M., Drimmel, R., Jordi, C., Krone-Martins, A., Masana,

E., Mauduit, J. C., Mignard, F., Mowlavi, N., Rocca-Volmerange, B., Sartoretti, P., Slezak, E., Sozzetti, A., 2012. Gaia universe model snapshot. Astronomy and Astrophysics 543, A100.

Rodríguez, M., Tapiador, D., Fontán, J., Huedo, E., Montero, R. S., Llorente, I. M., 2008. Dynamic provisioning of virtual clusters for Grid computing. In: Euro-Par 2008 Workshops - Parallel Processing, VHPC 2008, UNICORE 2008, HPPC 2008, SGS 2008, PROPER 2008, ROIA 2008, and DPA 2008, Las Palmas de Gran Canaria, Spain, August 25-26, 2008, Revised Selected Papers. pp. 23–32.
URL http://dx.doi.org/10.1007/978-3-642-00955-6_4

Rowstron, A., Narayanan, D., Donnelly, A., O'Shea, G., Douglas, A., 2012. Nobody ever got fired for using Hadoop on a cluster. In: Proceedings of the 1st International Workshop on Hot Topics in Cloud Data Processing. HotCDP '12. ACM, New York, NY, USA, pp. 2:1–2:5.
URL http://doi.acm.org/10.1145/2169090.2169092

Rzhetsky, A., Foster, J. G., Foster, I. T., Evans, J. A., Nov. 2015. Choosing experiments to accelerate collective discovery. Proceedings of the National Academy of Sciences 112 (47), 14569–14574.
URL http://dx.doi.org/10.1073/pnas.1509757112

Sakr, S., Liu, A., Fayoumi, A. G., Jul. 2013. The family of mapreduce and large-scale data processing systems. ACM Comput. Surv. 46 (1), 11:1–11:44.
URL http://doi.acm.org/10.1145/2522968.2522979

Sale, S. E., Dec. 2012. 3D extinction mapping using hierarchical Bayesian models. Monthly Notices of the Royal Astronomical Society 427, 2119–2131.

Salpeter, E. E., Jan. 1955. The Luminosity Function and Stellar Evolution. The Astrophysical Journal 121, 161.

Sanders, G., Shin, S., 2001. Denormalization effects on performance of RDBMS. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3 - Volume 3. HICSS '01. IEEE Computer Society, Washington, DC, USA, pp. 3013–.
URL http://dl.acm.org/citation.cfm?id=820558.820646

Sarro, L. M., Eyer, L., O'Mullane, W., De Ridder, J., 2014. Astrostatistics and Data Mining. Springer Publishing Company, Incorporated.

Shafer, J., 2010. I/O virtualization bottlenecks in cloud computing today. In: Proceedings of the 2nd Conference on I/O Virtualization. WIOV'10. USENIX Association, Berkeley, CA, USA, pp. 5–5.
URL http://dl.acm.org/citation.cfm?id=1863181.1863186

Shvachko, K., Kuang, H., Radia, S., Chansler, R., may 2010. The Hadoop distributed file system. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. pp. 1 –10.

Sicular, S., 2013. Gartner's Big Data definition consists of three parts, not to be confused with three "V"s. Forbes.

Sparks, E. R., Talwalkar, A., Smith, V., Kottalam, J., Pan, X., Gonzalez, J. E., Franklin, M. J., Jordan, M. I., Kraska, T., 2013. MLI: An API for distributed machine learning. CoRR abs/1310.5426.

Stonebraker, M., 1986. The case for shared nothing. Database Engineering 9, 4–9.

Stonebraker, M., Cetintemel, U., 2005. "one size fits all": An idea whose time has come and gone. In: Proceedings of the 21st International Conference on Data Engineering. ICDE '05. IEEE Computer Society, Washington, DC, USA, pp. 2–11.

Sutter, H., March 2005. The free lunch is over: A fundamental turn toward concurrency in software. Dr. Dobb's Journal 30 (3).

Taniar, D., Leung, C. H. C., Rahayu, W., Goel, S., 2008. High Performance Parallel Database Processing and Grid Databases. Wiley Publishing.

Tapiador, D., 2011. Deployment of Intersystems Caché with GUMS on Amazon EC2. Tech. Rep. GAIA-C9-TN-ESAC-DTP-002-01, ESAC Science Archives and VO Team, European Space Agency.

Tapiador, D., 2012. Deployment of Greenplum parallel DBMS on Amazon EC2 with GUMS. Tech. Rep. GAIA-C9-TN-ESAC-DTP-004-1, ESAC Science Archives and VO Team, European Space Agency.

Tapiador, D., Berihuete, A., Sarro, L., Julbe, F., Huedo, E., 2017. Enabling data science in the Gaia mission archive: The present-day mass function and age distribution. Astronomy and Computing 19, 1 – 15.
URL http://www.sciencedirect.com/science/article/pii/S221313371630083X

Tapiador, D., O'Mullane, W., Brown, A., Luri, X., Huedo, E., Osuna, P., 2014. A framework for building hypercubes using mapreduce. Computer Physics Communications 185 (5), 1429 – 1438.

Tapiador, D., Rubio-Montero, A., Huedo, E., Montero, R., Llorente, I., 2007. Two approaches for the management of virtual machines on Grid infrastructures. In: Proceedings of the Spanish Conference on e-Science Grid Computing, March 1-2, 2007. Madrid (Spain).

Taylor, M., Boch, T., Taylor, J., 2015. SAMP, the Simple Application Messaging Protocol: Letting applications talk to each other. Astronomy and Computing 11, Part B, 81 – 90, the Virtual Observatory: {II}.
URL http://www.sciencedirect.com/science/article/pii/S2213133714000821

Taylor, M. B., Dec. 2005. TOPCAT STIL: Starlink Table/VOTable Processing Software. In: Shopbell, P., Britton, M., Ebert, R. (Eds.), Astronomical Data Analysis Software and Systems XIV. Vol. 347 of Astronomical Society of the Pacific Conference Series. p. 29.

Valduriez, P., 2009. Shared-Nothing Architecture. Springer US, Boston, MA, pp. 2638–2639.
URL http://dx.doi.org/10.1007/978-0-387-39940-9_1512

van Leeuwen, F., Nov. 2007. Validation of the new Hipparcos reduction. Astronomy and Astrophysics 474, 653–664.

Vazquez, A., de la Calle, I., Contreras, J. L., Ibarra, A., Tapiador, D., 2010. Migration of Monte Carlo simulation of high energy atmospheric showers to Grid infrastructure. Journal of Physics: Conference Series 219 (7), 072057.
URL http://stacks.iop.org/1742-6596/219/i=7/a=072057

Vines, T. H., Albert, A. Y., Andrew, R. L., Débarre, F., Bock, D. G., Franklin, M. T., Gilbert, K. J., Moore, J.-S., Renaut, S., Rennison, D. J., 2014. The availability of research data declines rapidly with article age. Current Biology 24 (1), 94 – 97.

Wadler, P., 1995. Monads for functional programming. In: Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques-Tutorial Text. Springer-Verlag, London, UK, UK, pp. 24–52.
URL http://dl.acm.org/citation.cfm?id=647698.734146

Weisz, D. R., Fouesneau, M., Hogg, D. W., Rix, H.-W., Dolphin, A. E., Dalcanton, J. J., Foreman-Mackey, D. T., Lang, D., Johnson, L. C., Beerman, L. C., Bell, E. F., Gordon, K. D., Gouliermis, D., Kalirai, J. S., Skillman, E. D., Williams, B. F., 2013. The panchromatic Hubble Andromeda treasury. IV. A probabilistic approach to inferring the high-mass stellar initial mass function and other power-law functions. The Astrophysical Journal 762 (2), 123.

Xin, R. S., Gonzalez, J. E., Franklin, M. J., Stoica, I., 2013. Graphx: A resilient distributed graph system on spark. In: First International Workshop on Graph Data Management Experiences and Systems. GRADES '13. ACM, New York, NY, USA, pp. 2:1–2:6.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., Stoica, I., 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12. USENIX Association, Berkeley, CA, USA, pp. 2–2.

Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I., 2013. Discretized streams: Fault-tolerant streaming computation at scale. In: Proceedings of the Twenty-Fourth

ACM Symposium on Operating Systems Principles. SOSP '13. ACM, New York, NY, USA, pp. 423–438.