Sistemas Informáticos
Curso 2003-2004

Cluster de PlayStation 2

Alberto Fernández Sánchez
Diego Arnáiz García
Javier Martín Esquifino

Dirigido por:
Prof. Manuel Prieto Matías y Luis Piñuel Moreno
Dpto. DACYA

Facultad de Informática
Universidad Complutense de Madrid

Índice

0.- Resumen

La necesidad de obtener rendimientos cada vez mejores ha llevado a la invención de ordenadores más potentes que los convencionales denominados superordenadores. Estas máquinas son capaces de multiplicar el rendimiento de un ordenador de sobremesa para la ejecución de tareas específicas como son la simulación de sistemas, el modelado y creación de mundos tridimensionales y el diseño de sistemas de búsqueda de información masiva y detección de errores en tiempo real, entre otros. Para la construcción de un superordenador existen diversos criterios respecto a la relación calidad y coste. Una posible alternativa es la de difundir diferentes procesos entre varias computadoras, para luego poder recoger los resultados que dichos procesos deben producir, a este concepto se le conoce con el nombre de 'cluster'. Para la construcción de un cluster necesitamos un conjunto de máquinas que puedan intercambiar información mediante una red y que a su vez estén gobernadas por otro sistema denominado front-end. Nuestro proyecto está formado por cuatro máquinas del mismo tipo conectadas en red y gobernadas por un front-end que consta de un solo ordenador. Las máquina que utilizamos para la realización del cluster es una consola de videojuegos llamada PlayStation 2. La PlayStation 2 es una máquina con un alto rendimiento en operaciones en punto flotante respecto al precio de la misma. Para la puesta en marcha del cluster necesitamos adaptar la consola a un entorno de trabajo y configurarla mediante la interfaz de un sistema operativo. Necesitamos adquirir el kit de linux para PlayStation 2, que incluye una tarjeta de red, un disco duro, un teclado y un disco de arranque entre otros accesorios. Una vez montado debemos de configurar la máquina con un sistema operativo que en nuestro caso es una distribución GNU de linux llamada BlackRhino (basada en Debian). A partir de este punto debemos de instalar todos los sistemas necesarios para la comunicación y ejecución de procesos entre las diferentes consolas, que se pueden englobar como NIS, NFS, PBS y MIP.

The need for high-end perfomance has derived into the creation of more powerful computers than the mainstream computers available in the current market. This brand of super computers are able to multiply the perfomance of an usual computer by a great factor in certain duties such as systems simulations, modeling and creation of tridimensional worlds, design of massive information finders and real time error detection. To make a super computer, there are several points to consider regarding quality and price. One possibility is to distribute different process between different computers, to get later the results produced by those processes. This concept of computing is known as clustering. In order to build a cluster it´s needed a set of computers with comunication capabilities enabled between them and also is needed a master computer ruling all of them. This master computer is known as front end. Our project is built around 4 farms or computing machines, and one standard PC computer representing the front end. The computing machine used in our project is the PlayStation2. PS2 is a high perfomance/low cost machine in floating point operations. The PS2´s need to be prepared for network action in order to create the cluster. To achieve this we have used the Sony´s Linux Kit for Playstation2, composed by a network adapter, a hard disk drive, a keyboard, a VGA adapter and the DVD with the linux distribution. We have used the BlackRhino (Debian based) distribution instead of the red hat included in the original DVD from Sony. From this point on, the process of creation of the cluster involves all the needed services to achieve comunication and partnering between the cluster elements. This services are NIS, NFS, PBS and MIP.

1.- Arquitectura del Cluster de la PlayStation 2

1.1.- Descripción y objetivos del Cluster.

El objetivo docente de este proyecto apunta a la utilidad de una videoconsola de uso específico como máquina básica en la construcción de un supercomputador. Se pretende que sea capaz de elevar rendimientos y disminuir tiempos de ejecución de trabajos que puedan correr en esta máquina.

Tradicionalmente los mainframes y las supercomputadoras han estado construidas solamente por unos fabricantes muy concretos y para un colectivo elitista que necesitaba gran potencia de cálculo, como pueden ser grandes empresas o universidades. Muchos colectivos no pueden afrontar el gasto económico que supone adquirir una máquina de estas características, y aquí es donde toma la máxima importancia la idea de poder disponer de esa potencia de cálculo, pero a un precio muy inferior.

El concepto de cluster nació cuando los pioneros de la supercomputación intentaban difundir diferentes procesos entre varias computadoras, para luego poder recoger los resultados que dichos procesos debían producir. Con un hardware más barato y fácil de conseguir se pudo perfilar que podrían conseguirse resultados muy parecidos a los obtenidos con aquellas máquinas mucho más costosas, como se ha venido probando desde entonces.

Un clúster es un conjunto de ordenadores o máquinas electrónicas denominadas nodos unidos mediante una red de interconexión a los que un determinado software convierte en un sistema de mayores prestaciones.

Su topología es sencilla, pudiendo alternar entre una red de máquinas u ordenadores homogénea como es nuestro caso, o incluso hacer un cluster con diferentes tipos de equipos. En ocasiones, los cluster sirven para la reutilización de ordenadores desfasados, en los que solo interesa que su ‘cpu’ pueda servir de apoyo al resto de la red

del cluster. Nuestra visión del Cluster es diferente, nuestro objetivo primordial es obtener rendimientos pico elevados, y una vez que esté montado que se puedan correr trabajos vía remota para medir tiempos y realizar comparativas. Es importante resaltar el que los rendimientos que queremos adquirir solo se ciñen a ciertas aplicaciones y operaciones propias de la máquina, como el tratamiento de gráficos vectoriales, tratamiento de matrices y operaciones en punto flotante. Entre otras peculiaridades de la máquina, como ya especificaremos más adelante, su falta de memoria hace que su rendimiento medio disminuya.

Básicamente existen tres tipos de clusters: **Fail-over, Load-balancing y HIGH Performance Computing**. Los clusters Fail-over consisten en dos o más computadoras conectadas en red con una conexión heartbeat separada entre ellas. La conexión heartbeat entre las computadoras es usualmente utilizada para monitorear cuál de todos los servicios está en uso, así como la toma de servicio de una máquina por otra cuando una de las maquinas se haya caído.

El concepto en los load-balancing se basa en distribuir la carga entre diferentes máquinas, por ejemplo, que cuando haya una petición entrante a un servidor web, el cluster verifica cuál de las máquinas disponibles posee mayores recursos libres, para luego asignarle el trabajo pertinente.

Actualmente un cluster load-balancing es también fail-over con el extra del balanceo de la carga y a menudo con mayor número de nodos.

La última variación en el clustering son los High Performance Computing. Estas máquinas han estado configuradas especialmente para centros de datos que requieren una potencia de computación extrema. Los clusters Beowulf han sido diseñados específicamente para estas tareas de tipo masivo que refleja parte del objetivo del proyecto.

El modelo del Clusters tipo Beowulf se basa en componentes y periféricos para la plataforma x86 común para obtener un rendimiento sin precedentes a un costo muy bajo siendo la tecnología de Clusters de Alto Rendimiento para Linux más conocida. Los servidores de un Cluster de Alta Disponibilidad normalmente no comparten la carga de procesamiento que tiene un Cluster de Alto Rendimiento. Tampoco comparten la carga de tráfico como lo hacen los Clusters de Balance de Carga. Su función es la de esperar listos para entrar inmediatamente en funcionamiento en el caso de que falle algún otro servidor. La característica de flexibilidad que proporciona este tipo de tecnología de Cluster, lo hacen necesario en ambientes de intercambio intensivo de información.

Los clusters de alta disponibilidad permiten un fácil mantenimiento de servidores. Una máquina de un cluster de servidores se puede sacar de línea, apagarse y actualizarse o repararse sin comprometer los servicios que brinda el Cluster. Cuando el servidor vuelve a estar listo, se incorpora al Cluster y se puede dar servicio al siguiente servidor del grupo.

1.2 Historia y descripción de la PlayStation 2.

1.2.1.- ¿Cómo surgió la máquina?.

En 1988, Sony estableció un acuerdo con Nintendo para desarrollar un lector de CD-ROM, el SuperDisc, como un accesorio para la Super Nintendo. Pero debido a problemas en el contrato y las licencias el SuperDisc nunca salió al mercado. En cambio, en 1991 Sony sacó al mercado la PlayStation con una versión modificada del SuperDisc. La PlayStation pretendía ser el dispositivo multimedia para el hogar aunque únicamente salieron a la venta 200 unidades pues se decidió retirar el producto del mercado. El nuevo diseño dio lugar a la PlayStation X, o PSX, un dispositivo que ejecutaba videojuegos en CD-ROM. La PSX rápidamente se convirtió en la videoconsola más popular. Después del gran éxito de ventas alcanzado por la PSX, Sony fabricó la PS2 (PlayStation 2) que fue uno de los productos más esperados del 2001.

Esta consola ha sido diseñada desde el principio con un claro objetivo: Juegos 3D. Es por eso que todo el hardware está orientado a que se puedan realizar juegos que lleven al usuario a una inmersión en un mundo audiovisual en tres dimensiones. La PS2 es la evolución de la primera consola desarrollada por Sony, la PlayStation (PSX), que fue puesta en el mercado en el año 1995. La PSX fue de las primeras consolas cuyos juegos se distribuían en soporte CDROM. Los juegos de la PS2 se distribuyen en DVD aunque también es capaz de reproducir los juegos de la PSX en CDROM. Gracias a la gran capacidad de los DVD's (4.7 GB), los juegos de la PS2 están repletos de videos, música y sonido en 3D. Además de juegos, la PS2 puede reproducir CD's de audio y películas en DVD por tanto es una completa plataforma de entretenimiento.

El panel frontal contiene, además de la bandeja para los DVD's o CD's, dos ranuras para las tarjetas de memoria que son de 8 MB normalmente, aunque el formato es el mismo que las de la PSX por lo que se pueden intercambiar, otros dos ranuras para los nuevos controles, también funcionan los controles antiguos de la PSX. También consta de dos puertos USB que pueden ser usados con cualquier dispositivo USB compatible como teclados, ratones, impresoras, ... y por último 1 puerto Firewire de alta velocidad.

La parte trasera contiene conectores para la salida de televisión, para televisión de alta definición y salidas de sonido Surround, DTS y Dolby Digital 5.1.

1.2.2.- Comparando consolas.

La siguiente tabla comparativa muestra algunas consolas disponibles en el mercado en el año de presentación de la PS2 y las consolas más recientes. Los datos proporcionados por Sony y Microsoft no son realistas, son los datos máximos, mientras que los de Nintendo y Sega están medidos en un juego real:

Consola	Polígonos/seg	Frecuencia	Memoria Principal
PlayStation 2	75millones	300 MHz	32 MB RAMBUS + 4 MB DRAM de video
Xbox	150 millones	733 MHz	64 MB RAM
Gamecube	6-12 millones	485 MHz	24MB SRAM+16MB DRAM+3Mb SRAM de video
PlayStation	360.000	33.86 MHz	2 MB + 1 MB de video
Dreamcast	3 millones	200 MHz	16 MB + 8 MB de video
Nintendo 64	150.000	93.75 MHz	4 MB Direct RAMBUS

El alto grado de paralelismo de la arquitectura de la PS2 permite obtener un rendimiento superior a las últimas consolas funcionando a una velocidad de reloj menor.

1.3 Arquitectura de la PlayStation 2.

1.3.1.- Introducción.

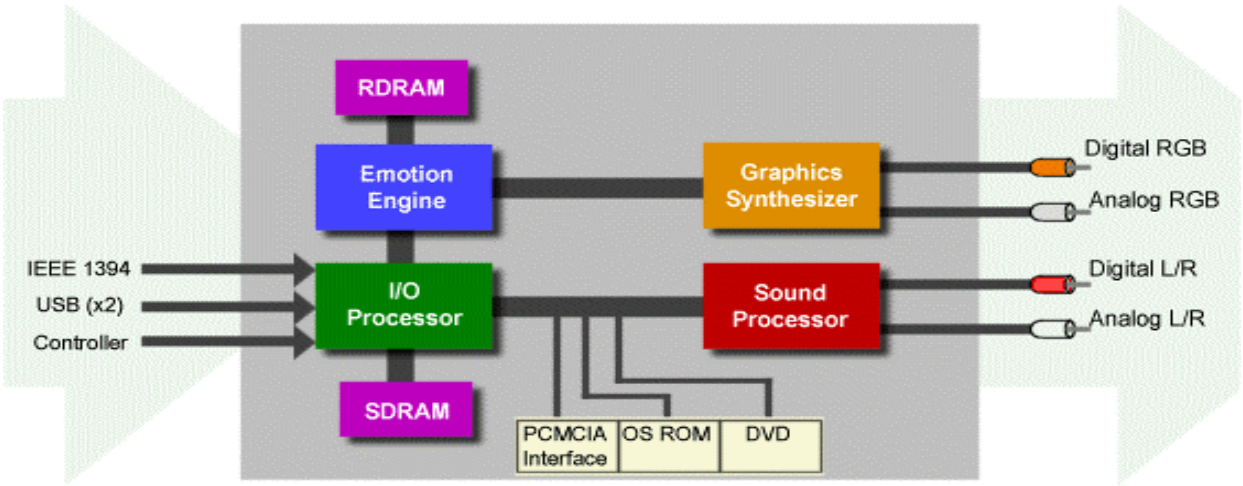
La arquitectura de la PlayStation 2 está formada principalmente por los siguientes componentes:

Procesador principal Emotion Engine (EE): El *Emotion Engine* es el componente principal de la PS2, y parte que la hace única. Está preparado para realizar todo tipo de operaciones en punto flotante con dos modos de programación (micro y macro). Tiene un rendimiento pico elevado que deriva de una buena programación de la aplicación a ejecutar, en su mayor medida juegos tridimensionales. El trabajo de esta CPU acaba con resultado de secuencias de comandos de *rendering* que son enviados al sintetizador gráfico denominadas *display lists*.

Procesador de Entrada/Salida (IOP): El IOP maneja el USB, el Firewire, y todo el tráfico de los mandos de control, así como el teclado y cualquier elemento externo. El IOP se comunica directamente con el EE para actuar en tiempo real.

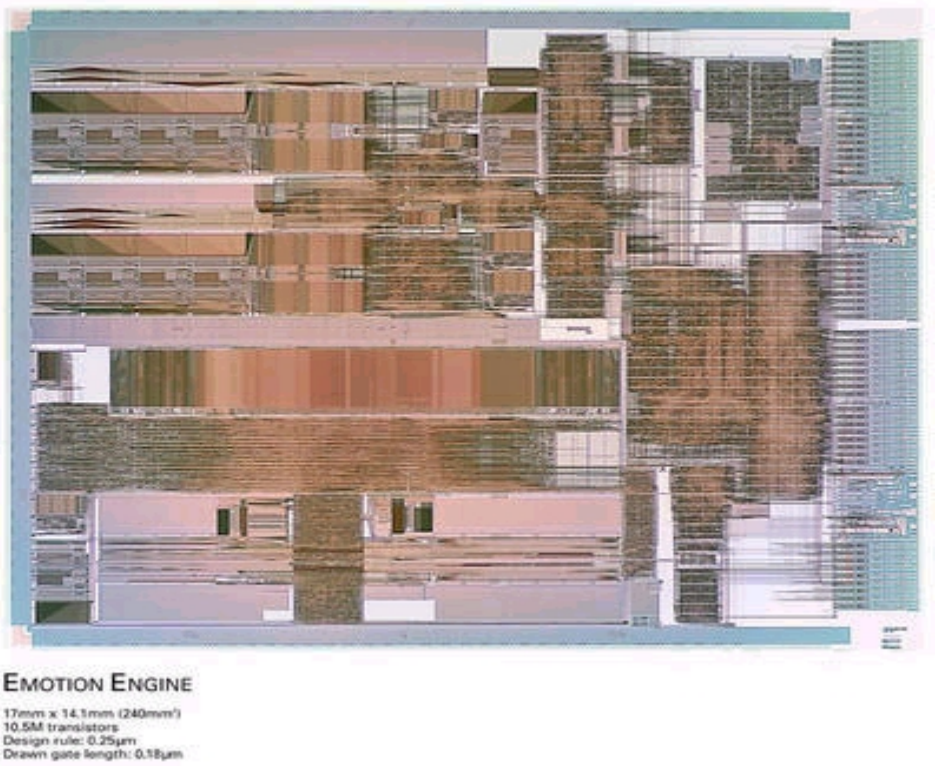
Sintetizador Gráfico (GS): El sintetizador gráfico recoge los *display lists* que le envía el EE y los presenta en pantalla. Esta tarjeta gráfica no libera al EE del trabajo que le supone el cálculo de los gráficos tridimensionales, esto se debe a que las aplicaciones que se utilizan están programadas en el EE de manera óptima, ya que no es una máquina de propósito general.

Procesador de sonido (SP): Representa la tarjeta de sonido con 48 canales de la PS2. Es capaz de reproducir sonido digital 3D.



1.3.2.- Emotion Engine.

El *Emotion Engine* es un procesador de 128-bits desarrollado por Sony y por Toshiba. Al igual que la PlayStation X, el *Emotion Engine*, es un procesador RISC (Reduced Instruction Set Computing), en concreto es superescalar, permitiendo ejecutar múltiples instrucciones a la vez. La combinación de las capacidades del EE para ejecutar múltiples instrucciones a gran velocidad, le proporcionan una mayor potencia que otros chips con más velocidad de reloj. Este implementa un subconjunto de instrucciones del procesador MIPS-IV funcionando con una velocidad de reloj de 300 MHz.

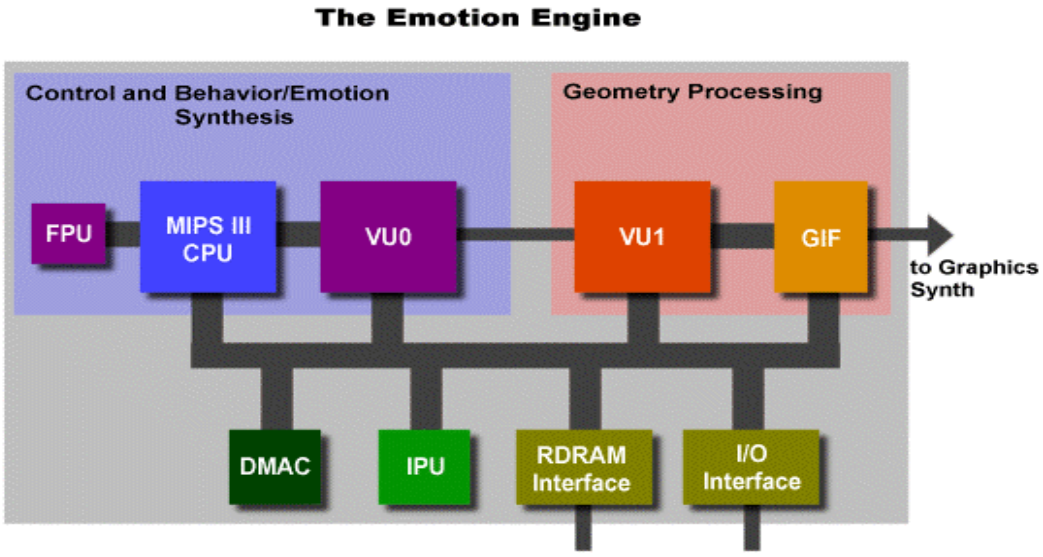


El EE no es de propósito general, como anteriormente apuntábamos, a diferencia de cualquier CPU de un PC actual, por este motivo, cuesta sacar un buen rendimiento en aplicaciones no programadas pensando en la arquitectura del micro. El EE ha sido diseñado específicamente para ejecutar videojuegos en 3D y está compuesto por dos unidades vectoriales (VU) y una CPU MIPS III. El EE realiza básicamente 2 tipos de cálculos, los cálculos geométricos y los cálculos de comportamiento y simulación del entorno del juego. Cuando todo se ha calculado, la tarea del EE es crear las listas de visualización (display lists) que son secuencias de comandos para la generación de la imagen, estas se envían directamente al sintetizador de gráficos (GS).

La capacidad de cálculo en punto flotante es muy superior a la de los ordenadores personales corrientes. La CPU incorpora dos unidades de enteros (IU) de 64-bits, una unidad SIMD de 128 bits con 107 instrucciones para el procesamiento multimedia, dos unidades independientes de cálculo de vectores en punto flotante (VU0, VU1), un circuito decodificador de MPEG-2 y controladores DMA.

Para analizar la potencia del procesador podemos fijarnos en cuántas operaciones en punto flotante de multiplicación y suma pueden realizarse, la respuesta es que son 10 las unidades de FMAC (Floating-Point Multiply Adder Calculator) que pueden ejecutarse a la vez. Como ya veremos en un dibujo posterior, son tres los componentes que pueden realizar operaciones en punto flotante y en paralelo: el co-procesador 1 FPU con 1 FMAC y 1 FDIV, el coprocesador 2 VU0 con 4 FMAC y 1 FDIV y la unidad de proceso vectorial (VU1) con 5 FMAC y 2 FDIV. Los FDIV son unidades que realizan operaciones de división y son mucho más lentas, llegando a tener una latencia de 7 ciclos con respecto a la suma.

Además de procesar los datos a 128-bits, es posible procesar y transferir volúmenes masivos de datos multimedia. Aunque es una máquina que carece de una memoria de gran capacidad, los 32 MB de RAM de memoria principal que soportan la velocidad de la CPU son Direct Rambus DRAM de dos canales para conseguir un ancho de banda de 3.2 GB/seg. Con la incorporación del decodificador MPEG-2 en un chip, es posible procesar en paralelo datos gráficos 3D de alta resolución e imágenes DVD de alta calidad. Con una capacidad de cálculo en punto flotante de 6.2 GFLOPS/seg, el rendimiento de esta CPU alcanza el de algunos supercomputadores. Cuando es aplicado al procesamiento de transformaciones de perspectiva y geométricas, que son las que se usan normalmente para el cálculo de gráficos en 3D, el rendimiento llega a 66 millones de polígonos por segundo. Este rendimiento es comparable con las estaciones gráficas usadas en la producción de películas de animación.



Como se puede observar en el diagrama superior, el EE está formada por las siguientes unidades principalmente:

Una CPU MIPS III como núcleo.
Un coprocesador en coma flotante (FPU) para la CPU.
Dos unidades vectoriales (VPU0 y VPU1).
La unidad de procesado de imagen (IPU): un decodificador MPEG-2.
Un controlador DMA de 10 canales (DMAC).

CPU: MIPS III y subconjunto IV + SIMD 128 bits

Característica	Valor
Frecuencia	300 MHz
Registros	32 x 128 bits
Microarquitectura	2 vias, dos unidades enteras 64 bits, 1 FPU
Segmentación	6 estados
Caché instrucciones	16K, 2 vias asociativa por conjunto
Caché datos	16K, 2 vias asociativa por conjunto
Scratch-Pad Ram	16K
TLB	

El núcleo de la PS2 sigue el estándar RISC ISA con algunas modificaciones que ha realizado la misma Sony. Esta CPU tiene dos unidades de enteros de 64 bits y está basada principalmente en una arquitectura MIPS III aunque el núcleo también incluye algunas instrucciones de la arquitectura MIPS IV. Pero en lugar de tener únicamente las instrucciones SIMD (Single Instruction Multiple Data) con enteros de 64 bits estándares del MIPS, Sony ha definido por completo un nuevo conjunto de instrucciones SIMD con enteros de 128 bits usando las dos unidades de enteros en paralelo para realizarlas. Trabajando juntas ambas unidades de enteros pueden realizar 4 operaciones con enteros de 32 bits, 8 de 16 bits o 16 de 8 bits por ciclo.

Las unidades vectoriales son dos procesadores VLIW/SIMD de 128 bits que complementan la unidad de coma flotante básica (FPU) para realizar los cálculos 3D T& L (Transform and Lighting).

Vector Unit 0	
Unidades	4 FMAC, 1 FDIV
Memoria	8K instrucciones, 8K datos

Vector Unit 1	
Unidades	5 FMAC, 2 FDIV
Memoria	16K instrucciones, 16K datos

La VPU0 situada justo al lado de la CPU realizará los cálculos de simulación y comportamiento. En cambio VPU1 ha sido dotada con el doble de memoria caché de instrucciones y de datos porque trabaja como un procesador de cálculos geométricos y necesita mantener muchos más datos en memoria para generar las listas de visualización. Las unidades vectoriales alcanzan una increíble velocidad de cálculo de 4
operaciones MAC de coma flotante por ciclo al igual que una división en coma flotante cada 7 ciclos. Las operaciones MAC (Multiply-ACcumulate) son usadas en gran variedad de cálculos vectoriales, el más común es el producto vectorial.

Si se programa basándose únicamente en la potencia de la CPU sin usar paralelamente las unidades vectoriales el rendimiento global baja enormemente. La

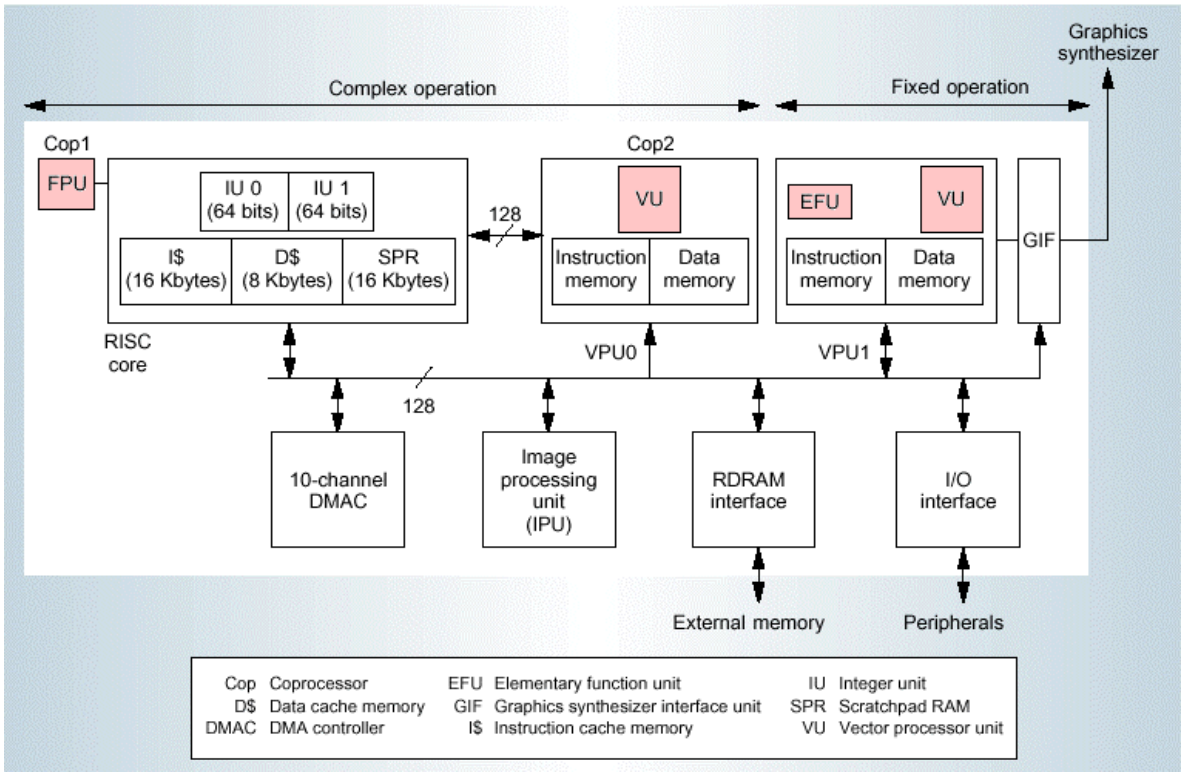
verdadera potencia de la EE proviene de usar estas unidades de forma conjunta durante la generación del entorno del juego.

La organización del EE se divide en dos partes que trabajan conjuntamente para realizar todos los cálculos y generar las listas de visualización.

La primera parte, consta de la CPU, FPU y VPU0. Estas unidades están unidas por un bus especial de 128 bits y disponen de una memoria especial, la Scratch Pad RAM. La SPRAM es una memoria muy rápida situada en la CPU que puede ser utilizada por la CPU y la VPU0. Es como un espacio de trabajo compartido para ambas unidades.

La VPU1 y GS (representado por el GIF) representarían esa segunda parte funcional. La VPU1 además del bus principal de 128 bits dispone de una conexión adicional de 128 bits con el GIF.

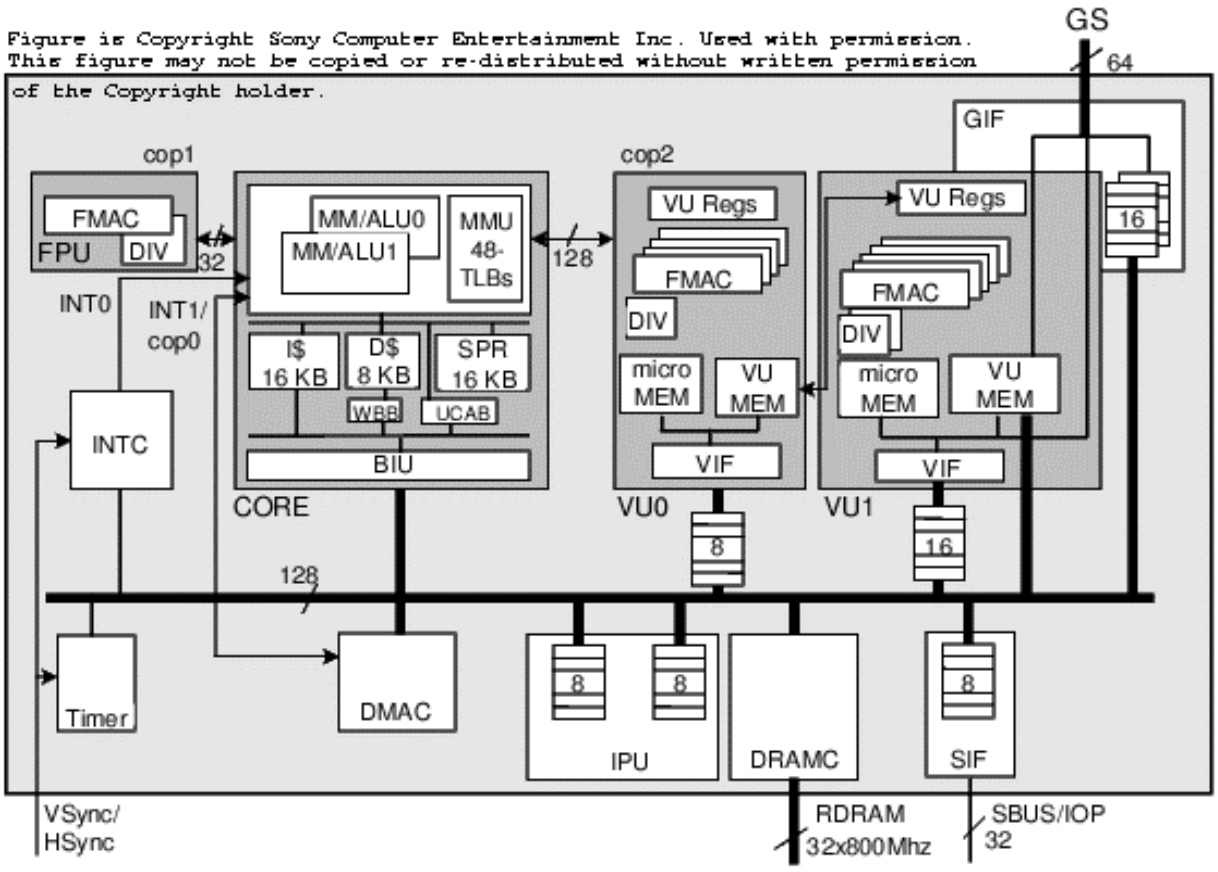
Aunque las tareas de los componentes son fijas, pueden mantener cierta flexibilidad. Los programadores pueden escoger como dividir el trabajo entre las dos partes. Trabajando en paralelo, el bloque CPU/FPU/VPU0 puede generar listas de visualización y procesar la geometría paralelamente con la VPU1. Y ambos enviar listas de visualización al GS a través del GIF. O trabajando en serie, con el bloque CPU/FPU/VPU0 actuando como un preprocesador para la VPU1. Entones la CPU y su equipo generarán la información del entorno del juego que la VPU1 recibirá para transformarla en las listas de visualización que serán enviadas al GS.



1.3.3.- Programando el EE.

Dentro de la programación del procesador existen dos modos de ejecución:

Es posible programar en “micromodo” sobre ambas unidades vectoriales VPU0 y VPU1. Esto funciona de la siguiente manera: Se carga una subrutina en la micromemoria de la VPU correspondiente y esto se ejecuta independientemente del pipeline del EE, de esta forma si se establece este modo sobre el VPU1, el VPU0 se queda libre como coprocesador matemático para el EE.



Como ya hemos apuntado anteriormente, existen dos unidades de procesamiento vectorial, VU0 y VU1, apuntaremos alguna que otra diferencia entre ambos. En primer lugar, el VU0 puede funcionar en macromodo interactuando como coprocesador del núcleo del Emotion Engine, para instrucciones del EE directamente. A su vez tenemos la posibilidad de hacerlo funcionar en micromodo a diferencia del VU1 que funciona solamente en micromodo no interactivo, en el cual un programa entero se envía a la micromemoria, y entonces el núcleo del EE transmite una señal al VU para comenzar a funcionar el programa enviado.

El VU1 tiene una trayectoria prioritaria respecto del sintetizador de gráficos (GS). A su vez el VU1 tiene ventajas de cálculos respecto al VU0 por tener una unidad elemental de cálculo de funciones trigonométricas, exponenciales, .. además de incluir como el primero (VU0) adición en punto flotante para un vector de cuatro elementos (x,y,z,w), incluyendo multiplicaciones rápidas y operaciones con enteros.

Comparando la memoria de ambos, dividimos la misma en memoria para código y memoria para datos, como ya habíamos comentado anteriormente, la memoria VU1 duplica a la VU0.

1.3.4.- IO, GS y SP.

El GS es la aceleradora de video que realiza todas las funciones de aceleración de video y genera las imágenes a partir de las listas de visualización creadas por el EE. El bus que une el EE con el GS es el GIF (Graphic Interface Unit) y es un bus de 64 bits dedicado a obtener múltiples listas de visualización de las diferentes unidades de la EE y combinarlas para permitirle al GS componer las imágenes. El GIF gestiona todo lo relacionado con las listas de visualización para que el EE no se vea obligada a perder tiempo gestionándolas.

En el GS se han implementado diversos efectos gráficos por hardware como el canal alpha, las superficies de bezier, la corrección de perspectiva o el mip mapping.

La PS2 usa el canal alpha para añadir efectos de transparencia a los objetos. Este es un modo especial de gráficos usado para video digital, animación y videojuegos para conseguir ciertos aspectos gráficos. Se usan 24 bits para definir las cantidades de rojo, verde y azul, con 8 bits para cada color, definiendo así un color específico. Además otros 8 bits son usados para crear la máscara de balanza de grises que actúa como una capa separada para representar los niveles de transparencia de un objeto. El grado de transparencia se define por el nivel de oscuridad que tiene el gris en el canal alpha. Si por ejemplo defines una rea de máscara con gris muy oscuro, puedes crear un objeto que parezca muy transparente. En cambio si defines el área con un gris claro puedes crear efectos especiales de niebla y agua.

Las superficies de bezier son un proceso de modelado 3D que calcula cuántos polígonos se necesitan para crear un objeto. Se basa en el nivel de detalle necesario para que el objeto tenga una apariencia suave para el espectador. La PS2 sólo ejecuta estos cálculos para objetos modelados con superficies de bezier que están en el videojuego.

La corrección de perspectiva crea el mapa de textura escalado al mismo ratio que el objeto donde está mapeado.

El mip mapping es una forma de mapeado de texturas que se realiza a través de diferentes tamaños del mismo mapa de textura. En esencia, el procesador actualiza la apariencia de un objeto con mayor detalle de imagen cuando nos situamos a menor distancia de dicho objeto en el videojuego. A continuación se describen los pasos que realiza la PS2 para usar mapas de textura con trilinear mip mapping:

El IOP processor es el procesador de E/S de la PS2 y manipula todos los puertos USB, el FireWire y los mandos del juego. Este procesador es una versión idéntica a la CPU de la PSX lo que le proporciona la capacidad de ejecutar todos los juegos que se desarrollaron originalmente para la PSX.

El Sound Processor es la tarjeta de sonido de la PS2, dispone de 48 canales de sonido permitiendo sonido digital 3D usando Dolby Digital Sound, AC-3 (Audio Compression) y DTS (Digital Theather Systems) con una calidad de sonido superior a la de CD, calidad DAT (Digital Audio Tape). Si se dispone de un receptor de audio de calidad podremos sacar el máximo rendimiento a la PS2. Además en el aspecto de la reproducción de DVD, la consola tiene una calidad de sonido al nivel de los mejores reproductores de DVD.

1.4 Configuración de la PlayStation 2.

1.4.1.- ¿Qué opción es la adecuada?.

Antes de montar el Cluster hay que sopesar las diferentes herramientas que existen para configurar y desarrollar aplicaciones en nuestra máquina. Como nuestro objetivo es el de minimizar el coste unido a la necesidad de que todas las máquinas se repartan las tareas, una vez acabado este punto seremos capaces de decidir cual de estas tres opciones es la más adecuada.

(a) Kit de desarrollo de Sony.

Sony vende el Sony Computer Entertainment Development Kit DTL-T10000 para desarrollar programas y juegos para la PS2. El coste por unidad es de unos 20.000 $, lo que lo hace inaccesible para usuarios normales, incluso para empresas pequeñas ya que normalemente sobre un kit pueden trabajar 1 o 2 personas por lo que para un equipo de desarrollo habrá que comprar varios kits con el consiguiente desembolso.

El KIT contiene dos PS2 de desarrollo:

1.- La PS2 TEST, es igual que una PS2 normal pero puede leer cdr sin necesidad de modchips.

2.- La PS2 TOOL, es bastante más grande que una PS2 normal debido al aumento de los componentes con respecto a esta, entre las diferencias, destacan:

-128 MB de memoria principal
-Disco duro
-Tarjeta de red

Sony proporciona a los desarrolladores esta máquina de tal forma que compilan su código y se ejecuta sobre el hardware de esta PS2. Sony sólo proporciona el hardware y las librerias,así como asesoría y ejemplos.

Existen una serie de entornos de desarrollo comerciales. Entre ellos los dos mas conocidos:

-ProDG de Snsys. Tienen diferentes mÛdulos, incluido uno muy interesante llamado proview que permite mediante el uso del interfaz firewire conectar una PS2 modelo DTL-H 18 Test Unit a un pc. Este modelo de PS2 tiene la peculiaridad de que es capaz de leer y ejecutar cdr (sin chip ni nada parecido dado que es una unidad de testeo para desarrolladores). Snsys te proporciona con el proview los archivos de una iso que corre en esta PS2 a modo de programa monitor permitiendo la comunicación con el software que corre en el PC para subir el código que tu compilas directamente y ejecutarlo sin necesidad de grabar un cd por cada prueba. También tiene herramientas de debug que redirigen la salida al lado del pc para facilitar el desarrollo.

-CodeWarrior de Metrowerks. Al igual que Snsys nos proporciona un entorno de desarrollo completo con multitud de herramientas.

Para tener acceso a estas herramientas es necesario ser desarrollador o_cial de Sony y estar registrado como tal.

(b) Kit de linux.

El kit de Linux para PS2 permite ejecutar una distribución Linux en la PS2. Esto convierte a la consola en un ordenador de sobremesa plenamente funcional. El kit de Linux contiene:

-un disco duro interno de 40 GB.
-un teclado usb.
-un ratón usb.
-una tarjeta de red 10/100 Base-T.
-dos discos DVD.

El primero contiene el entorno de ejecución (RTE) y los manuales de la PS2 que Sony suele incluir en el kit de desarrollo. El segundo disco contiene todo el software de la distribución que se puede instalar en el disco duro.

El kernel de Linux contiene drivers que ocultan el hardware e impiden el acceso directo a la IOP. Sony proporciona solo los binarios de estos drivers por lo que existen limitaciones a la hora de programar la PS2 con este kit. Por ejemplo estos drivers no proporcionan interfaz con el puerto Firewire, por lo que es imposible de programar este puerto. El Kit viene con el compilador de GNU gcc, las xfree y muchas otras utilidades. Por lo que tenemos un completo entorno de desarrollo. Aunque los programas o juegos que se desarrollen sobre el kit de Linux solo podrán ejecutarse en una PS2 que tenga el kit.

(c) Programación RAW.

La programación RAW consiste en escribir programas para la PS2 sin utilizar ni el kit de desarrollo de Sony, ni el kit de Linux. El desarrollo se realiza en un PC mediante el uso de compiladores cruzados. Necesitamos trabajar con los siguientes procesadores:

El procesador principal es un MIPS R5900, este procesador implementa todas las instrucciones del MIPS III ISA, algunas del MIPS IV y un conjunto propietario de instrucciones multimedia.

Unidades vectoriales (VU0, VU1)

IOP, es un MIPS R3000

Existen herramientas libres de desarrollo capaces de generar código para todos los procesadores de la PS2. Estas herramientas son:

binutils Es una colección de herramientas multiplataforma para trabajar con ficheros ejecutables, entre las que destacan: ld, el linkador de GNU; as, el ensamblador de GNU.

gcc Es la colección de compiladores de GNU. Estos compiladores son capaces de generar código para una gran cantidad de plataformas. Nos interesa especialmente el compilador de C.

ps2lib Es una librería open-source para desarrollar directamente con la PS2. Esta librería ha sido desarrollada a base de ingeniería inversa. La librería proporciona un gran número de funciones básicas para acceder al hardware de la PS2. Como la PS2 tiene dos CPUs (EE e IOP), la librería está formada por dos partes, cada una con funciones para una CPU.

Newlib Es la librería que implementa las funciones estándar de C (libc) y las funciones matemáticas (libm). Es una librería de Cygnus usada en muchos sistemas empotrados debido a su pequeño tamaño que ha sido adaptada para la PS2.

Cuando comenzó la programación RAW de la PS2, para probar los programas que se estaban desarrollando había que grabarlos en un CD y ejecutarlos con alguno de los métodos vistos anteriormente. Esto hacía arduo, engorroso y caro el desarrollo. Con el tiempo han ido surgiendo programas que mediante la conexión de la PS2 al PC permiten enviar los programas compilados del PC a la PS2 para que esta la ejecute. De esta forma el proceso de desarrollo se ha dinamizado bastante.

Existen tres programas de este tipo, uno para cada forma de conectar la PS2 con el PC:

Pukklink y InLink: que usa la tarjeta de red que se encuentra en el kit de Linux para transmitir los ejecutables.

Naplink: usa un cable usb.

El método más común, por ser el más barato, es hacerlo con el Naplink. El único problema con este método es que hay que usar un cable USB que tenga el chip PL-2301 o PL-2302.

Una vez que tengamos el cable, conectamos el PC a la PS2. En la PS2 ejecutamos el Naplink servidor, que tenemos que tener previamente grabado en un CD, y en el PC ejecutamos el Naplink cliente. Desde el cliente podremos indicarle a la PS2 el ejecutable a cargar. Existen versiones del cliente para Windows y para Linux.

Como acabamos de observar, lo más factible, eficiente y apropiado para nuestro proyecto es utilizar el kit de linux, por sus altas prestaciones, su bajo coste respecto al kit de desarrollador y la posibilidad de comunicar las máquinas mediante una red ethernet. A estos puntos habría que incluir que el software a utilizar es de libre distribución y dentro del mismo tenemos la posibilidad de elegir entre varias distribuciones.

1.4.2.- Distribuciones.

Llegado a este punto deberíamos tener en cuenta dos aspectos fundamentales. Primero tener claro cómo vamos a programar en la máquina y en segundo lugar qué sistema operativo nos acercará a la misma. Existen algunas distribuciones de Linux para el manejo de la consola, que desglosamos a continuación.

La distribución de **Sony** dotada con parches gráficos, es la primera que se pondrá en nuestro camino. Está creada para usuarios con conocimientos mínimos, y la potencia de adaptabilidad de esta distribución no llega lejos. Viene con el kit de linux junto con software adicional para hacer más fácil el entorno de trabajo.

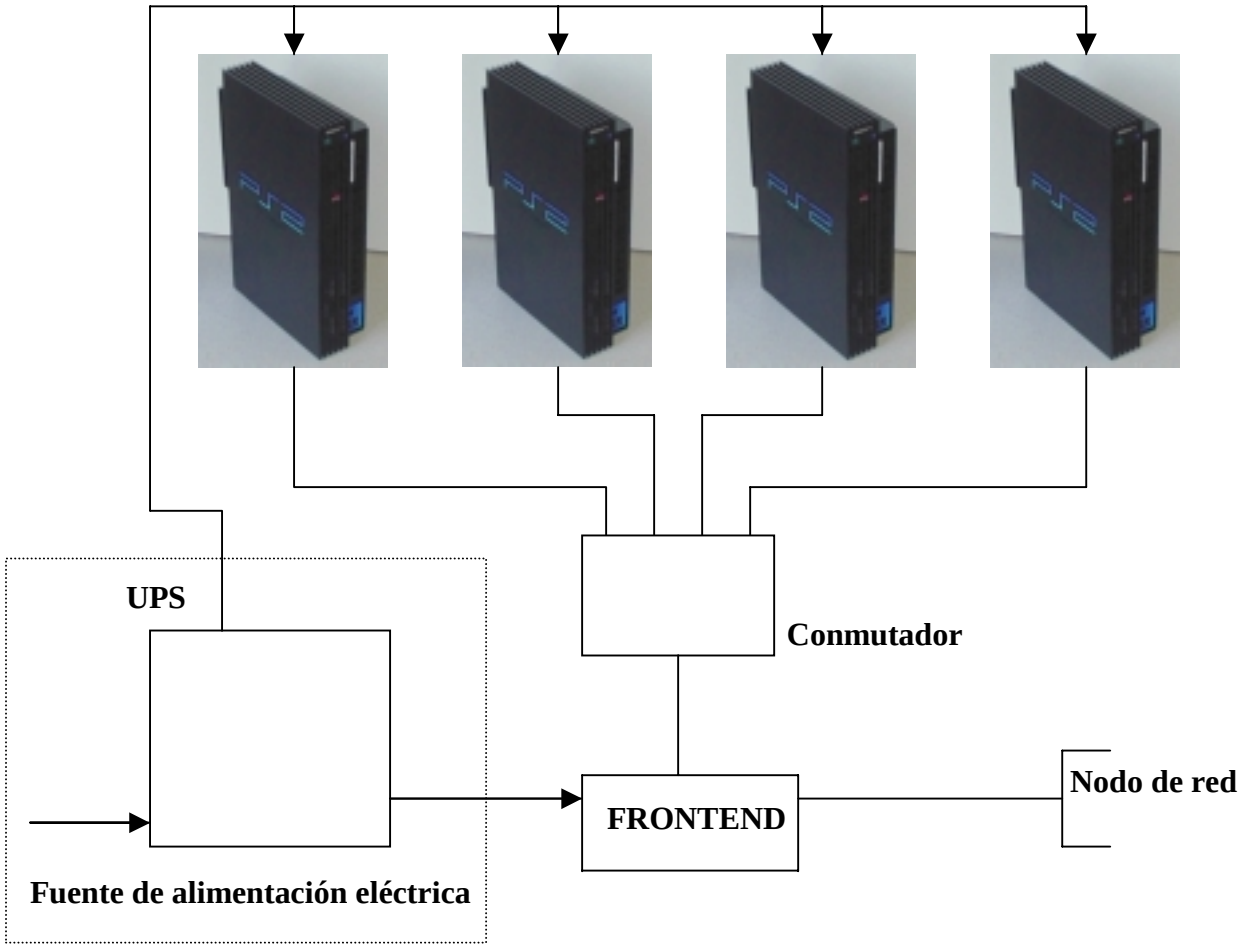
En segundo lugar tenemos la **Black Rhino**, que es una apuesta por enmascarar a la conocida Debian, y que se presenta sin entorno gráfico. Es un paso intermedio entre usuarios que está acostumbrados a utilizar el modo consola en sus puestos de trabajo y los que no. En cierto modo esta distribución permite la modificación de parámetros internos de la máquina de una forma más flexible. También debemos de comentar que la resolución de problemas de incompatibilidades en esta distribución respecto a paquetes genéricos es más problemática que la anterior.

Por último, contamos con la distribución de **Gentoo**. Muchos administradores ven en esta versión el sistema operativo más moldeable que pueda existir. Desde la instalación, el usuario debe de actuar por su cuenta, esta distribución carece de instalador y es un proyecto que cada día se renueva con nuevas ideas.

En nuestro caso hemos seleccionado la Black Rhino para la realización del cluster, por ser un punto medio entre la dificultad y la flexibilidad del proyecto.

2.- Instalación y configuración

2.1.- Topología del Cluster.

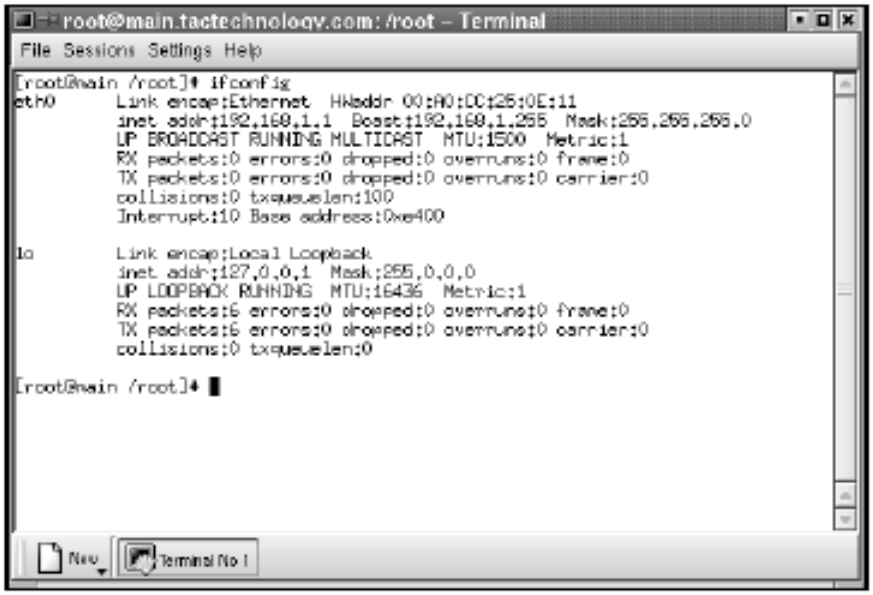


En la imagen anterior se puede observar que el Cluster consta de cuatro máquinas conectadas en red mediante un conmutador de ocho puertos de los cuales utilizamos cinco y al quinto se conecta el Frontend que a su vez se puede conectar a una red externa e incluso obtener acceso a Internet. Para no perder datos en caídas de energía eléctrica incorporamos una ups que apague todo el sistema correctamente si detecta que errores en el mantenimiento eléctrico.

2.2.- Configurando red en Frontend (Red Hat).

Bajo **Red Hat** todas la comunicaciones de redes consisten en enlazar un dispositivo físico con unas determinadas interfaces software.

Lo primero que tenemos que ver es si nuestro sistema reconoce la tarjeta de red instalada. Para ello ejecutamos el comando *ifconfig* el cual nos mostrará la siguiente ventana:



Como se puede ver en la figura el primer bloque describe la configuración de nuestra tarjeta de red ‘eth0’ y el segundo como esta configurado el loopback.

El siguiente paso sería configurar los parámetros de nuestra tarjeta definida por el interfaz eth0. Para ellos necesitamos definir distintos parámetros como son la direccion IP, la máscara, la puerta de enlace…Esto se puede hacer de dos modos:

- Asignando una serie de opciones al comando ifconfig.
- Utilizando la herramienta netconfig que proporciona Red Hat. Se trata de una herramienta gráfica muy sencilla e intuitiva, en la cual seleccionamos un interfaz y nos pide una serie de datos. Para nuestro caso utilizamos los siguientes datos:

IP address	*198.168.0.2*
Netmask	*255.255.255.241*
Default gateway	*198.168.0.1*
Primary server	*…*

A continuación y una vez configurada la tarjeta de red, hay que editar el fichero /etc/hosts donde se incluirán las direcciones del resto de computadoras que forman la red.

2.3.- Configurando red en PlayStation 2.

En cuanto a la configuración de la red bajo Debian es muy similar a Red Hat, a diferencia de alguna ruta, o de que la configuración de la tarjeta hay q hacerla a través de un fichero de configuración.

Lo primero es configurar la tarjeta, mediante la edicción del fichero */etc/network/interfaces* donde se define la IP de la máquina, la mascara… quedando de la siguiente forma:

address	*192.168.0.3*
netmask	*255.255.255.240*
network	*192.168.0.0*
broadcast	*192.168.0.15*

Para asignar un nombre a la máquina se hace a través del fichero /etc/hostname, al que se le edita una línea con el nombre elegido:

estaca.cluster

Y finalmente al igual que para el frontend en el fichero */etc/hosts* añadimos una linea por cada máquina perteneciente a la red.

2.4.- Network Information System (NIS).

2.4.1.- ¿Qué es Network Information System (NIS)?

Network Information Service distribuye información que necesita ser compartida a partir de una red Linux a todas las maquinas de la red. La información mas comúnmente distribuida por la red usando NIS consiste en una base de datos de usuarios e información de autentificación, como /etc/passwd y /etc/group. Por ejemplo si la entrada de un password de usuario es compartida por todos los login de los hosts via NIS, entonces ese usuario puede acceder a la red desde cualquier puesto. Pero también se puede usar para más cosas, como para distribuir un directorio de una lista telefónica o una lista de códigos.

NIS usa una arquitectura cliente-servidor. Las configuraciones NIS giran alrededor del concepto de dominio. Un dominio NIS no es lo mismo que un dominio DNS o un dominio en internet. Un nombre de dominio NIS referencia a un grupo de sistemas, normalmente en un LAN que usa los mismos mapas NIS. Los dominios NIS son usados como herramientas de administración de sistemas, es un método de organizar grupos de máquinas que necesitan acceder a una información compartida a través de una red usando un conjunto común de mapas NIS.

Las bases de datos NIS son almacenadas en formato DBM que es un formato de fichero binario basado en los sencillos ficheros de texto ASCII.

Cada dominio NIS debe tener al menos un sistema que funcione como servidor NIS para ese dominio. Un servidor NIS administra la información q va ser compartida a través de la red, mientras que un cliente NIS no es mas que un programa que hace querys a los servidores designados para obtener la información almacenados en sus bases de datos, también conocidas como mapas. Los servidores NIS pueden ser subdivididos en masters o esclavos. Un servidor masters mantiene las copias reales de los mapas NIS. Un servidor esclavo mantiene copias de las bases de datos NIS, las cuales recibe del servidor maestro.

Existe cuatro tipos de topologías NIS principalmente:

- Un dominio único con un servidor maestro, sin servidores esclavos, y con uno o más clientes.

- Un dominio único con un servidor maestro, con uno o varios servidores esclavos y uno o más clientes.

- Múltiples dominios, cada uno con su propio servidor maestro, sin servidores esclavos, y con uno o más clientes.

- Múltiples dominios, cada uno con su propio servidor maestro, con uno o varios servidores esclavos y con uno o más clientes.

Una configuración apropiada necesita al menos de un servidor NIS y de uno o más clientes. Para instalar un servidor hace falta configurar el demonio *ypserv* e identificar

los distintos ficheros que NIS va a distribuir a los programas clientes. Para instalar un cliente hace falta configurar los programas *ypbind, ypwhich, ypcat, yppoll y ypmatch*. El mas importante es *ypbind*, una vez que este se ejecute en el sistema ya podremos decir que tenemos un cliente NIS.

2.4.2.- Configurando un servidor NIS.

Configurar un servidor NIS involucra los siguientes pasos:

- Establecer el nombre del dominio NIS.
- Configurar e inicializar el demonio servidor, *ypserv*.
- Inicializar los mapas NIS.
- Inicializar el demonio de password NIS.
- Inicializar el demonio de transferencia si se usan servidores maestros.
- Modificar el procreso de inicializacion del sistema para incluir los demonios NIs cada vez que se reinicia el PC.

Establecer el nombre del dominio NIS.

Utilizamos el comando 'nisdomainname':

nisdomainname *dominionis*

Donde dominionis es el nombre del dominio que queremos asignar. Este método solo hace una asignación temporal, más tarde explicaremos como hacerlo de forma permanente, de modo que sobreviva a un reinicio de la computadora.

Configurar e inicializar el demonio servidor, *ypserv.*

Después de haber asignado el nombre del dominio, el siguiente paso es configurar y arrancar el demonio *ypserv*. Los archivos de configuración para el demonio son:

- /var/yp/securenets
- /etc/ypserv.conf

Estos ficheros contienen opciones de configuración, llamadas lineas de opción, para el *ypser*. También contiene información de acceso, llamada reglas de acceso, usadas por los demonio ypserv y ypxfrd. Los valores por defecto para el fichero /etc/ypserv.conf y válidos para la mayor parte de las configuraciones es la siguiente:

/etc/ypser.conf

```
dns:no
*:shadow.byname:port:yes
*:passwd.adjunct.byname:port:yes
*:*:none
```

Las líneas de opción tienen el siguiente formato:

option:[yes|no]

option: puede ser *dns* o *xfr_check_port*. *dns* controla si el servidor NIS mira en DNS para host que no estén listados en los mapas hosts. *xfr_check_port* controla si el demonio *ypserv* corre en un puerto inferior al 1024, también llamados puertos privilegiados.

Las reglas de acceso siguen el siguiente formato:

host:map:security:mangle[:field]

host: dirección IP.
map: el nombre del mapa.
security: el tipo de seguridad a utilizar, tiene varias posibilidades:
-none : permite acceder al mapa al host al menos que mangle este a yes, en tal caso es acceso es denegado.
-port : permite acceder si la conexión se hace a través de un puerto privilegiado.
-deny : deniega el acceso al host.
-des : requiere autentificación DES.
mangle: indica el tipo de puerto a usar. *yes* indica puerto privilegiado.

El fichero de configuración más importante es /var/yp/securenets que define los números de red y las mascaras de red que definen las listas de host a los que se les permite acceder a los mapas del servidor NIS. Contiene una entrada por línea del siguiente formato:

m.m.m.m n.n.n.n

donde m.m.m.m es la máscara y n.n.n.n es el numero de red. Por ejemplo:

255.255.255.255 127.0.0.1
255.255.255.0 192.168.0.0

Para la primera línea estaríamos permitiendo el acceso al servidor NIS a la IP 127.0.0.1. Para la segunda línea estaríamos permitiéndolo para todas las IP que estén entre 192.168.0.1 hasta 192.168.0.255. El resto de hosts tendrían denegado el acceso.

Antes de inicializar el servidor hay que asegurarse de que esté activo el demonio *portmapper*. Para asegurarse de que el demonio portmapper esta ejecutándose hay que ejecutar la siguiente línea:

/etc/rc.d/init.d/portmap status

En caso de no estar ejecutándose habrá que lanzarlo con:

/etc/rc.d/init.d/portmap start

Una vez activo el portmap, hay que pasar a ejecutar el ypserv:

```
# /etc/rc.d/init.d/ypserv start
```

Y para comprobar que realmente esta activo:

```
# rpcinfo –u localhost ypserv
```

Inicializar los mapas NIS

Una vez inicializado el servidor NIS hay que crear las bases de datos. El comando utilizado es *ypinit* el cual construye un conjunto de mapas NIS que son guardados en el subdirectorio /var/yp. Son creados obteniendo información de las ficheros de bases de datos que están en /etc. Estos ficheros son:

```
/etc/passwd
/etc/group
/etc/hosts
/etc/networks
/etc/services
/etc/protocols
/etc/netgroup
/etc/rpc
```

Para crear las bases de datos NIS, hay que ejecutar el siguiente comando:

```
# /usr/lib/yp/ypinit –m
```

La opción –m indica que se están creando mapas para el servidor maestro.

Inicializar el demonio de password NIS.

Este demonio tiene informados tanto a los clientes como a los servidores esclavos de los cambio efectuados en cuanto a usuarios se refiere. Se llama *yppasswdd* y maneja los cambio sobre los passwords actualizando toda información que depende de ellos. Para inicializarlo basta ejecutar el siguiente comando:

```
# /etc/rc.d/init.d/yppasswdd start
```

Hay que tener en cuenta que a los usuarios NIS no se les tiene permitido cambiar su nombre en la sesión. Para permitirlo habría que inicializar el demonio con las siguientes opciones:

```
yppasswdd –e chfn
```

Inicializar el demonio de transferencia. Para topologías con servidores esclavo.

El demonio *ypxfrd* se usa para agilizar las transferencias de mapas NIS entre el servidor maestro y los servidores esclavos. Para inicializarlo basta con ejecutar el siguiente comando:

/etc/rc.d/init.d/ypxfrd start

Modificar el proceso de inicialización del sistema para incluir los demonios NIS cada vez que se reinicia el PC.

Una vez configurado el servidor NIS, hay que preparar el sistema para que estos cambios sean persistentes y no los perdamos cuando reiniciemos la máquina. Para esto es necesario imponer la ejecución de los demonios ypserv, yppasswdd y ypxfrd (si se usan servidores esclavos) cada vez que se encienda el sistema.

El primer paso es guardar el nombre del dominio NIS. Para esto hay que añadir la siguiente línea en /etc/sysconfig/network:

NISDOMAIN = nisdomainname

Donde nisdomainname es el nombre que hemos elegido.

El siguiente paso es ejecutar una herramienta de configuración del sistema que posee Red Hat llamada *serviceconf*. Para ellos desde el terminal nos logamos como root y tecleamos *serviceconf*. Seguidamente aparecerá una lista de servicios junto a una casilla de selección. Para nuestro caso seleccionamos los demonios del NIS (ypserv, yppasswdd y ypxfrd), guardamos los cambio y cerramos la aplicación.

2.4.3.- Configurando un cliente NIS.

Configurar un cliente NIS involucra los siguientes pasos:

- Establecer el nombre del dominio NIS.
- Configurar e inicializar el demonio cliente, *ypbind*.
- Testear el demonio cliente.
- Configurar los ficheros de inicio que usa NIS.
- Reinicia el cliente.

Establecer el nombre del dominio NIS.

El primer paso es configurar el nombre del dominio NIS. Se hace del mismo modo que para el servidor.

nisdomainname *nisdomain*

Configurar e inicializar el demonio cliente, *ypbind*.

El demonio *ypbind* utiliza el fichero de configuración /etc/yp.conf el cual especifica que servidores NIS usar y como localizarlos. Las entradas de este fichero siguen el siguiente formato:

[domain *nisdomain*] ypserver *nisserverip*

nisserverip : indica la IP del servidor NIS
nisdomain: nombre del domino NIS

Al igual que para el servidor necesitamos que este ejecutando el demonio portmap. Y ya solo queda activar el demonio ypbind:

/etc/rc.d/init.d/ypbind start

Ahora falta editar el archivo /etc/hosts.conf para habilitar la posibilidad de que el NIS busque por nombres de host mofidificando la siguiente línea:

order hosts,nis,bind

Finalmente también hay que editar el fichero /etc/nsswitch.conf modificando las siguiente líneas.

passwd: files nis
shadow: files nis
group: files nis
hosts: files nis

Configurar los ficheros de inicio que usa NIS.

El primer paso es guardar el nombre del dominio NIS. Para esto hay que añadir la siguiente línea en /etc/sysconfig/network:

NISDOMAIN = nisdomainname

Donde nisdomainname es el nombre que hemos elegido.

2.4.4.- Seguridad en NIS.

En general los problemas son los mismo que se describían para el NFS, heredados del uso del portmapper que compromete la seguridad del sistema. Además como se transmite información acerca de passwords de usuarios se recomienda no usar NIS a través de internet a menos que se use una transmisión encriptada ya sea usando SSH o encapsulamiento de IP's. La principal medida de seguridad para NIS es limitar el acceso a los mapas NIS a través del fichero /var/yp/securenets.

2.5.- Network File System (NFS).

2.5.1.- ¿ Qué es Network File System (NFS) ?

NFS es el servicio más comunmente usado para permitir accesos locales a sistemas de ficheros o discos remotos bajo redes Linux. Usa una arquitectura básica cliente/servidor. El servidor consta de los discos fisicos que contienen los sistemas de ficheros a exportar y distintos demonios para hacer a estos visibles a la red exterior. Los clientes NFS solo montan dichos sistemas de ficheros exportados.

El uso mas común del servicio NFS es el de centralizar el almacenamiento de las cuentas de usuario de una red garantizando que un usuario pueda acceder desde cualquier puesto a su directorio de una forma controlada y segura.

La principal ventaja de NFS es que se centraliza la administración, mejorando el mantenimiento de la red. Por ejemplo hacer el backup de un sistema de ficheros almacenado en el servidor es mucho mas sencillo que andar haciendo un backup de cada nodo de la red en particular. También facilita la administración del acceso a usuarios, pudiéndoles asignar un espacio de almacenamiento concreto, limitándoles el acceso según los permisos, etc..

Como desventaja tiene que al ser un servicio distribuido es sensible a las congestiones de red, y que la utilización de discos de alto almacenamiento degrada en la eficiencia del servicio. El tener centralizados todos los datos da lugar a que si se produce un fallo tanto de aplicación como de las unidades físicas del servidor hace que toda la red quede inutilizable.

2.5.2.- Configurando un servidor NFS

La configuración del servidor se puede dividir en cuatro fases:

- Diseño
- Implementación
- Testing
- Monitorización

Diseño: el diseño de un servidor NFS involucra:

- Seleccionar un sistema de ficheros a exportar.
- Elegir los usuarios (o hosts) a los cuales se les va ser permitidos exportar los sistemas de ficheros.
- Seleccionar un convenio de nombres para facilitar el mantenimiento de la red.

Buenos candidatos como sistemas de ficheros a exportar son aquellos que se comparten por todos los usuarios de una red como pueden ser los directorios de trabajo de proyecto o directorios compartidos(/home). También son candidatos aquellos sistemas estaticos que no necesitan ser replicados en cada máquina (/usr).

- Usa /home/username para montar los directorios home.

- Usa los mismos path names tanto en el servidor como en el cliente.
- Solo se pueden exportar sistemas de ficheros locales, no se pueden exportar sistemas q a su vez están montados como NFS.
- El subdirectorio de un sistema de ficheros exportado no puede ser exportado individualmente al menos que el subdirectorio resida en un disco físico distinto al del directorio padre.

Archivos clave, comandos y demonios:

- Archivos de configuración y de estado:
  - /etc/exports : contiene la lista de los sistemas de ficheros y las opciones a exportar (que los clientes podrán montar). Cada línea tiene el siguiente formato:

    dir host(options) [host(options)] …

    dir : directorio o sistema de ficheros a exportar
    host : especifíca uno o mas hosts a los que les está permitido montar *dir*. Puede ser especificado como un nombre simple, como un grupo netgroup NIS, como un grupo de host.
    Opciones :
    - rw : exporta el sistema de ficheros como lectura-escritura.
    - ro : exporta el sistema como solo lectura.
    - …

    Ejemplo : /var/tmp 192.168.0.1 (rw)

  - /var/lib/nfs/rmtab : en el se añade una línea cada vez que se recibe una petición de montaje por un nodo externo a partir del demonio *rpc.mountd*. Cuando se recibe una petición para desmontar un sistema se elimina el registro correspondiente.
  - /var/lib/nfs/xtab : en el se añade una línea cada vez que se recibe una petición de montaje en la que se informa de las opciones de exportación del sistema de ficheros.
  - /etc/hosts.allow : indica los servicios permitidos para unos específicos hosts. Las líneas tiene el siguiente formato:

    daemon: host_list [host_list]

    daemon : demonio del servicio a permitir.
    host_list : lista de uno o mas hosts a los que se les va a permitir el acceso al servicio definido por daemon.

  - /etc/host.deny : indican los servicios que se deniegan a todos los hosts que no estén definidos en *hosts.allow.*

    Ejemplo de entradas en el fichero host.deny:
    portmap : ALL
    lockd: ALL
    mountd:ALL

rquotad:ALL
statd: ALL

- Demonios:
  - rpc.portmap : habilita a los clientes NFS el descubrir la disponibilidad de los servicios NFS dados por el servidor
  - rpc.mountd : procesa las peticiones de montaje pedidas por los clientes NFS
  - rpc.nfsd : ofrece todos los servicios NFS excepto el bloqueo de ficheros y la administración de las cuotas de almacenamiento.
  - rpc.statd : implementa el cierre NFS cuando el sistema sufre algun fallo.
  - rpc.lockd
  - rpc.rquotad : ofrece información acerca de los sistemas NFS exportados a los clientes.
- Scripts y comandos
  - /etc/rc.d/init.d/portmap : inicializa el demonio *portmap*.
  - /etc/rc.d/init.d/nfslock : inicializa los servicios *lokd* y *statd*.
  - /etc/rc.d/init.d/nfs : script que inicializa todos los demonios NFS de servidor anteriormente definidos.
  - nfsstat : muestra información del estado del subsistema NFS.
  - showmount : ofrece información acerca de los clientes y sistemas de ficheros que se han montado.
  - rpcinfo –p : muestra por pantalla los demonios NFS que están ejecutandose.
  - exportfs : permite manipular la lista de exportaciones sin tener que editar */etc/exports.*

  Para activar el servidor NFS es necesario activar los tres scripts utilizando el comando start :

  /etc/rc.d/init.d/portmap start
  /etc/rc.d/init.d/nfs start
  /etc/rc.d/init.d/nfslock start

  Para desactivar el servidor se utiliza el commando stop:

  /etc/rc.d/init.d/portmap stop
  /etc/rc.d/init.d/nfs stop
  /etc/rc.d/init.d/nfslock stop

2.5.3.- Configurando un cliente NFS

Para configurar el cliente NFS se necesita solo tres pasos: disponer de los demonios *statd* y *lockd*, añadir las entradas para exportar NFS en el fichero /etc/fstab, y montar los sistemas de ficheros con el comando *mount*.

1 paso: añadir líneas al fichero */etc/fstab* para indicar las opciones y sistemas a impotar. Las lineas deben tener el suiguiente formato:

nombre_servidor:sistema_ficheros_a_importar nombre_sist_fich_en_local NFS opciones

Las opciones más comunes para importar sistemas de ficheros via NFS son:

rsize=8192 : tamaño del buffer de lectura
wsize=8192 : tamaño del buffer de escritura
hard : permite que las operaciones NFS que fallen se reintenten antes de decir que el servidor no responde
intr : permiten el uso de señales para interrumpir operaciones NFS fallidas.
nolock
0
0

2 paso: inicializar el demonio portmap

/etc/rc.d/init.d/portmap start

3 paso: montar los sistemas de ficheros que se quieren importar

mount /home

2.5.4.- Seguridad en NFS

NFS sufre de algunos problemas de seguridad que le hacen potencialmente inseguro para ser utilizado a través de internet o redes no controladas. En este apartado se mostrarán los riesgos específicos de un servidor y un cliente NFS y se propondrán remedios para minimizar la exposición de la red a estos riesgos.

SERVICIO	PUERTO	PROTOCOLOS
Portmap	111	TCP,UDP
Nfsd	2049	TCP,UDP
Mountd	variable	TCP,UDP
Lockd	variable	TCP,UDP
Statd	variable	TCP,UDP
Rquodat	variable	UDP

Una de las principales debilidades de NFS es el fichero /etc/exports. Si alguien es capaz de tomar una direccion listada en /etc/exports todos los puntos de montaje quedan accesibles. Otra debilidad es el control de acceso al sistema de ficheros de linux que se exporta una vez que un cliente monta un sistema, los permisos de usuarios normales y de grupo en los ficheros toman el control de acceso.

La primera línea de defensa contra estas dos debilidades consiste en usar un ‘control de acceso host’ para limitar el acceso a los servicios del sistema, particularmente al ‘portmapper’. De forma más general se puede usar un aplicación firewall usando ‘netfilter’, este es mas fuerte que el nivel de seguridad de los demonios NFS. Para configurarlo eficientemente es necesario especificar todos los puertos y

servicios que usa NFS para aplicarles un filtro de paquetes. En la siguiente tabla se muestran un listado de los puertos y protocolos utilizados por los demonios NFS.

Hay que fijarse en que *mountd, lockd, statd, y rquodat* no tiene asociado un puerto en concreto y a que es asignados aleatoriamente por el *portmapper*. La mejor forma de direccionar esta variabilidad es asignar a cada demonio un puerto específico usando la opción –p y luego aplicar el filtrado de paquetes a estos puertos.

Consideraciones de seguridad para el servidor NFS.

- Usar siempre la opción root_squash en el fichero /etc/exports. Esta opción hace que los usuarios root's de los clientes no tengan acceso a los ficheros que a los que solo el usuario root del servidor tenga acceso. Por defecto esta opción es activada por el NFS, pero no esta de más indicarla.
- Otra de las cosas que hace NFS para mantener la seguridad del servidor es añadir la opción *secure* a los ficheros exportador para que lo tenga en cuenta el servicio *mountd*. La opción *secure* previene de que un malevolente usuario no root inicialice un dialogo NFS por un puerto no privilegiado para usarlo como punto de lanzamiento para ataques.

Consideraciones de seguridad para los clientes NFS.

- En el cliente tenemos que deshabilitar los programas SUID (set UID) en los sistemas de ficheros montados usando la opción *nosuid*. Esta opción previene que el usuario root del servidor pueda crear un programa SUID en el sistema de ficheros exportados, ya que podría logarse en el cliente como usuario normal, y luego usando el programa UID de root podría a llegar logarse como root en el cliente.
- A veces también se puede usar la opción *noexec* para no permitir la ejecución de programas, pero a veces es ineficaz ya que lo que se necesita exportar del servidor son scripts y demás programas.

A pesar de todo lo contado en este apartado NFS es muy complejo, y un sistema nada trivial, por lo que se seguirán descubriendo nuevos bugs.

2.5.5.- NFS en el cluster de PlayStation 2.

2.5.5.1 Servidor

- Configuración del fichero /etc/exports

- En la distribución Red Hat es muy sencillo activar el servicio de NFS. Solo hay que activarlo y el sistema por si solo configura todos los scripts de arranque. Desde el terminal ejecutar el comando '**setup**' y dentro de la lista de opciones seleccionar '**system services**'. Ahora solo falta marcar la opción 'nfs' y el sistema hará el resto.

2.5.5.2 Clientes

- Configuración del fichero /etc/fstab

2.6.- Sistema de Colas.

2.6.1- Instalación en el Front end.

Para la instalación del sistema de colas en el servidor será necesario obtener el paquete correspondiente a la distribución de linux sobre la que se esté trabajando. En el caso de trabajar con Red Hat linux 9.0 el paquete correspondiente se encuentra disponible en los cds de la distribución original. El nombre del paquete es *openpbs-2.3pl2-1.i386.rpm.* Para obtener el paquete correspondiente para el resto de distribuciones se puede encontrar bien en los cds de la distribución que se esté usando o bien en la dirección web www.rpmfind.net.

La instalación del paquete se realiza como es habitual. Si se está trabajando desde el escritorio bastará con hacer doble clic en el paquete y si se está trabajando en modo consola bastará con escribir el siguiente comando desde el shell:

Shell> rpm –i [ruta-fichero\] openpbs-2.3pl2-1.i386.rpm

Si la instalación del paquete transcurre sin problemas, ya se debería tener instalado el sistema de colas en el servidor. Para comprobar que la instalación se ha realizado correctamente se debe comprobar que al reiniciar el sistema se lanzen en el arranque los 3 demonios propios del PBS: *pbs_mom, pbs_server y pbs_sched.* En el apartado configuración del servidor se explicará que son estos demonios y que hacer con ellos. También debe comprobarse que se han creado los archivos y carpetas necesarios para el PBS. Debe por tanto comprobarse que se han creado las siguientes carpetas en el sistema a través de las cuales posteriormente se accederá a los ficheros de configuración del sistema de colas:

/usr/spool/PBS : Archivos de configuración.
/usr/pbs: Ejecutables.

Si no ha ocurrido ningún problema ya se encuentra instalado el sistema de colas en el servidor y se puede entonces saltar al segundo apartado *Instalacion en las ps2* para continuar el proceso.

Es muy posible que al realizar la instalación del paquete aparezca un error en la comprobación de dependencias. Dicho error de dependencias indicará que se necesita el paquete libtcl8.0. En caso de no tener instalado dicho paquete o tener una versión anterior bastará con instalarlo y realizar de nuevo el proceso. En el caso de tener instalado dicho paquete en una versión posterior a la versión 8.0 surge un problema puesto que la instalación del paquete PBS busca que se encuentre ya instalado el paquete libtcl8.0 pero no comprueba que existan versiones de ese mismo paquete posteriores. Por lo tanto en caso de tener instalado el paquete libtcl8.1 o superior no nos dejará continuar la instalación. Este problema es un claro bug del paquete actual openpbs que se supone será corregido en futuras versiones del mismo.

Para solucionar este problema no queda otro remedio que engañar al sistema realizando un enlace simbólico (con el nombre que busca el openpbs) apuntando

realmente al libtcl8.X que tengamos instalado. Por lo tanto se debe escribir en el shell la siguiente orden:

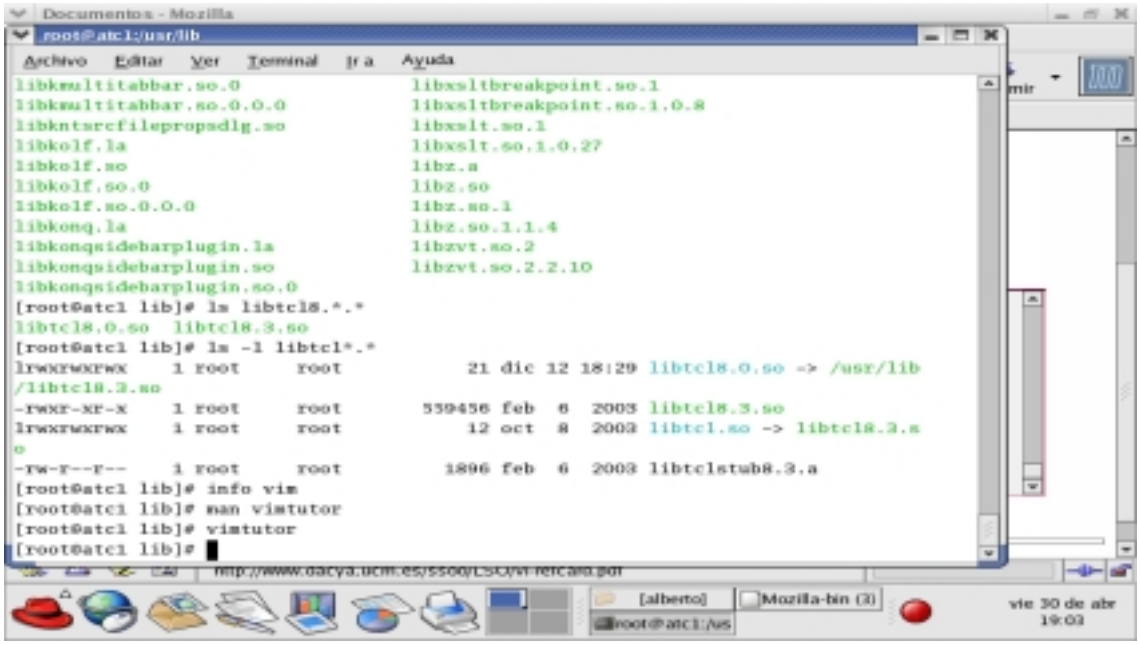
Shell> ln –s /usr/lib/libtcl8.0.so /usr/lib/libtcl8.X.so

Esta orden crea un enlace simbólico de nombre libtcl8.0 (el buscado por el openpbs al comprobar dependencias) apuntando al libtcl que tengamos ya instalado. Donde se indica libtcl8.X deberá escribirse en vez de X el número correspondiente a nuestra versión previamente instalada.

Una vez hecho esto para instalar el paquete finalmente se debe escribir la siguiente orden desde el shell:

Shell> rpm –i --nodeps openpbs-2.3pl2-1.i386.rpm

Con la cual se manda instalar el paquete sin realizar comprobaciones de dependencias para que el enlace simbólico creado sirva de puente y engañe al sistema en la instalación.



2.6.2.- Instalación en las ps2.

La instalación del PBS en las ps2 es más engorrosa que en el front end. El primer problema que aparece es que no existe ningún paquete OpenPbs para la distribución BlackRhino, por lo que no queda más remedio que compilar directamente el código

fuente en las ps2. Para ello es necesario obtener dicho código fuente, el cual sólo se encuentra disponible a través de la web oficial del OpenPBS www.openpbs.org.

Para poder acceder al código fuente es necesario darse de alta en la web y obtener un registro. El tiempo medio para la validación de una petición de registro en la web y la posterior obtención del código de acceso y usuario a través del correo electrónico es aproximadamente de una semana.

Una vez se tenga un usuario y un código de acceso se puede realizar la descarga del código fuente y de los scripts de configuración necesarios. Para ello basta escribir desde el shell de la ps2 la siguiente orden:

Shell ps2> wget --http-user=Nombre_Usuario --http-passwd='Contraseña' http://www.openpbs.org/UserArea/downloads/Openpbs_2_3_16.tar.gz

para luego descomprimir el fichero que acabamos de bajar en la carpeta que se desee:

[Directorio destino]Shell ps2>tar –zxvf [Ruta\]*Openpbs_2_3_16.tar.gz*

Una vez descomprimido el archivo se obtiene todo el código fuente del PBS y todos los scripts de configuración y compilación necesarios. Lo primero es ejecutar el script de configuración habitual antes de realizar cualquier compilación que determina si existen todos los compiladores y herramientas necesarias en el sistema y prepara el entorno para la instalación. Para ejecutarlo bastará con escribir desde el shell la siguiente orden:

Shell ps2> sh –xv ./configure --disable-server

en la cual se especifica mediante la opción --disable-server que sólo estamos interesados en la parte del cliente, por lo que sólo se generará el demonio necesario en el cliente (pbs_mom) y los archivos de configuración apropiados para el cliente.

Si no aparece ningún mensaje de error entonces ya se encuentra el entorno listo para realizar la compilación. Si falta algún compilador o algún componente por instalar véase el apéndice1 "Como bajar paquetes en Debian". Si por el contrario aparece un mensaje de error indicando que no se conoce la arquitectura, sistema operativo y/o fabricante del sistema entonces hay que realizar un paso adicional que se detalla en el siguiente párrafo.

El script de configuración llama a un script auxiliar (\builduties\config.sub) el cual se encarga de devolver en una cadena de texto el tipo de arquitectura, el fabricante y el sistema operativo correspondiente a la máquina donde se ejecuta de la forma "CPU-Fabricante-SistemaOperativo". Este script auxiliar puede ser ejecutado individualmente para comprobar que lo que determina es correcto respecto a nuestro sistema. El problema puede surgir cuando el script de configuración general llama al script auxiliar y comprueba con el sistema operativo que lo devuelto por este es correcto. Por alguna razón esta comprobación falla aún cuando el resultado devuelto por el script auxiliar es totalmente coherente con el entorno. La solución a este problema es editar el script configure para eliminar la llamada al script auxiliar y pasarle por consola la cadena correcta manualmente. Dichas líneas son las siguientes:

770: if canocical = '$srcdir/builduties/config.sub "$configuration"`;then:;else
771: }echo"configure:error:"config.sub failed on $configuration"" 1>&2;exit1;}
772: fi

Con estas líneas comentadas se elimina así el problema y para ejecutar ahora el configure basta con escribir:

Shell ps2> sh –xv ./configure --disable-server mips--linux*

Una vez hecho esto ya se encuentra el sistema listo para realizar la compilación. Basta escribir ahora las siguientes órdenes desde el shell:

Shell ps2> make

Shell ps2> su (en caso de no tener permisos de root)

Shell ps2> make install

En este punto ya se tiene generado el binario correspondiente al demonio *pbs_mom* y todos los archivos necesarios. Lo único que falta entonces es crear el script de arranque necesario para lanzar el demonio en el arranque. Para ello lo primero es crear el propio script de arranque en la carpeta /etc/init.d junto al resto de scripts del sistema ya existentes. Para ello se editará un nuevo fichero llamado pbs

Shell ps2> cd /etc/init.d

Shell ps2> vi pbs

y se escribirá en él lo siguiente:

```
case "$1" in
start)
  echo "Starting PBS daemons:"
   if [-x /usr/local/sbin/pbs_mom]; then
        echo –n "Starting pbs_mom:"
                start-stop-daemon --start --quiet --exec
                /usr/local/sbin/pbs_mom
        echo
   fi
;;

stop)
   echo "Shuting down PBS:"
   if [-x /usr/local/sbin/pbs_mom]; then
        echo –n "Stoping pbs_mom: "
                start-stop-deamon --stop --quiet --exec
                /usr/local/sbin/pbs_mom
        echo
   fi
;;
```

```
status)
        status pbs_mom
;;
restart)
    echo "Restarting PBS:"
    start-stop-deamon --stop --quiet --exec
                /usr/local/sbin/pbs_mom
    start-stop-deamon --start --quiet --exec
                /usr/local/sbin/pbs_mom
    echo "done."
;;

*)
echo "usage: pbs {start | stop | restart | status }"
exit

esac.
```

se guarda y ya se tiene listo el script de arranque del pbs_mom.

El siguiente y último paso es realizar el enlace simbólico correspondiente desde el nivel de arranque en el que nos encontremos o en el que deseemos que se lanze este demonio. Para ello en la carpeta /etc/rc.d/rcX.d (donde X es el nivel de arranque deseado) creamos un enlace simbólico de la siguiente forma:

Shell ps2> cd /etc/rcX.d

Shell ps2> ln –s S85pbs /etc/init.d/pbs

Una vez hecho esto ya se tiene todo instalado en la ps2. Bastará con reiniciar el sistema para comprobar que se lanza el demonio *pbs_mom* en el arranque sin problemas. Habrá que repetir todo este proceso en cada ps2 que forme parte del cluster o directamente replicar el disco duro sobre el que se ha hecho la primera instalación tantas veces como se desee.

2.6.3- Configuración del frontend.

Una vez que se tiene el sistema con todos los componentes instalados comienza el trabajo de configurar el PBS. Dicha configuración incluye preparar al servidor de acuerdo a lo requerido por el tipo de cluster a montar e implementar una política de planificación que dirija al sistema de colas.

Los 3 demonios que controlan el PBS son el *pbs_server, pbs_sched y el pbs_mom.* El pbs_server es el demonio que hace las tareas de servidor, el *pbs_sched* es el demonio que realiza la planificación, y por último el pbs_mom es el demonio que se ejecuta en los nodos o en el lado del cliente. Para el caso concreto de la creación del cluster de ps2, es razonable no lanzar el demonio pbs_mom en el front-end puesto que sólo es deseable que se ejecuten trabajos en las ps2, quedando el front-end sólo de

pantalla entre el usuario y las ps2. Para ello basta editar el script de arranque /etc/INIT.d/pbs y comentar las líneas que realicen la llamada a ejecución del pbs_mom en el front-end.

2.6.3.1.- Declaración de los nodos

Mediante PBS, la localización de los nodos que forman el cluster se realiza siempre desde el servidor. Cada nodo debe tener corriendo el demonio *pbs_mom (*existiendo siempre una copia del mismo por cada nodo que forme el cluster).

Cada nodo se puede declarar como nodo *exclusive o time-shared,* lo cual indica si un nodo no suelta un trabajo hasta que lo haya terminado (no admitiendo más trabajos a la vez) o si por el contrario se pueden enviar un número indeterminado de trabajos al dicho nodo, los cuales se ejecutarán virtualmente todos a la vez mediante una política round-robin o cualquier otra. Cualquier nodo en una de estas 2 modalidades debe ser declarado previamente en la lista de nodos situada en el fichero *PBS_HOME/ server_priv/nodes.* Existen 2 alternativas para realizar la declaración de nodos:

*** Editar directamente el archivo nodes:** En este archivo cada línea del mismo corresponde a una declaración de un nodo. La sintaxis utilizada para las declaraciones es la siguiente:

nombre_del_nodo[:tc] [propiedades] [np=numero_entero]

donde nombre_del_nodo indica el nombre a través del cual el servidor identificará al nodo; :tc indica, si se desea, que el nodo se va a declarar como nodo de tiempo compartido (time-shared); propiedades es una cadena de texto meramente informativa (ignorada por el sistema) que permite al usuario escribir propiedades del nodo según se desee; y finalmente np indica el número de procesadores virtuales que se desean especificar por nodo. Sino se especifica un numero para np el sistema asume que es siempre 1. Ejemplos de declaraciones serían:

nodo_1 mi_primer_nodo np=2
nodo_2:ts mi_primer_nodo_de_tiempo_compartido

*** Declarar los nodos a través del comando qmgr:** Mediante el comando qmgr se accede a un pequeño intérprete de órdenes del PBS que nos permite también declarar los nodos. La sintaxis utilizada es la siguiente:

qmgr: create node nombre_del_nodo [atributos=valores]

donde atributos y sus posibles valores asociados son:

Atributo	Valor
State	free, down, offline
Properties	Cualquier cadena alfanumérica de texto
Ntype	cluster, time-shared
Np	Número de procesadores virtuales

Esta lista de valores corresponden a estados que pueden ser determinados por el administrador pero existen también otros valores intermedios que sólo son de uso interno como por ejemplo el estado *busy* si un nodo time-shared ha superado su límite de trabajos aceptados, o *job-exlusive* si un nodo exclusive se encuentra realizando un trabajo. Notese que la especificación de un nodo exlusive se hace mediante el valor cluster en el atributo ntype.

Un ejemplo de declaración de nodos mediante qmgr sería por tanto:

Shell>qmgr
Qmgr> create node nodo1 np=2,ntype=cluster,properties="mi nodo favorito"

De la misma forma, qmgr permite la modificación de un nodo sin tener que eliminarlo y volverlo a declarar mediante la orden *set node* cuya sintaxis es:

Qmgr: *set node nombre_nodo [atributo=nuevo valor]*

De este modo si por ejemplo se desea modificar el atributo np del nodo creado anteriormente sería:

Qmgr: *set node nodo1 np=3*

Por último también existe la posibilidad de elimiar un nodo mediante la orden:

Qmgr: *delete node nombre_nodo*

2.6.3.2.- Determinación de donde se van a ejecutar los trabajos.

El donde se ejecuten los trabajos viene determinado por la interacción entre el planificador y el servidor a través del fichero nodes (PBS_HOME/server_priv/ nodes). Según exista este fichero o no existen diferentes criterios:

*** No existe el fichero nodes:** En caso de que no exista el fichero nodes el servidor sólo tiene conocimiento de su propio host, por lo tanto todos los trabajos sin especificación de host destino serán ejecutados en el propio host del servidor. Si por el contrario se especifica un host destino en la propia orden del trabajo se lanzará el trabajo en el host especificado.

*** Existe el fichero nodes:** En este caso entran en juego un conjunto de reglas que determinan donde se lanzará el trabajo:

1. Si se ha especificado un host destino en la orden del trabajo y dicho host se encuentra especificado en el fichero nodes como time-shared, entonces se lanzará el trabajo en dicho host.
2. Si se ha especificado un host destino en la orden del trabajo y no se encuentra dicho host en el fichero nodes especificado como time-shared o si se han especificado varios host en la

orden, entonces el servidor intenta colocar los trabajos en el nodo o nodos especificados en base al número de procesadores virtuales declarados para dichos nodo/s.

3. Si no se ha especificado una localización destino, entonces el servidor colocará el trabajo en el primer nodo disponible que corresponda con la especificación del trabajo lanzado.
4. Si existe un nodo configurado como nodo por defecto, entonces el trabajo se manda a ese nodo si se encuentra disponible o si está configurado como time-shared. Sino está disponible entonces se manda el trabajo al primer nodo disponible en el orden en que han sido declarados.
5. Sino existe ningún nodo por defecto y existe al menos un nodo declarado como time-shared, el trabajo se mandará a ese nodo.

2.6.3.3.- Puertos y direcciones de la red de trabajo.

El PBS hace uso de los nombres de hosts para determinar la localización de los trabajos y su identificación. Un sistema de colas PBS se reconoce por el nombre de host en el cual el servidor está corriendo (pbs_server). El nombre usado por los demonios es siempre el nombre canónico del host. Dicho nombre se obtiene por el campo *h_name* de la estructura de datos devuelta por la función de librería *gethostbyaddr().*

Los 3 demonios y todos los comandos harán uso del archivo /etc/services para identificar los puertos necesarios para la configuración. Estos puertos necesariamente necesitan estar por encima del número 1024. Por tanto debe añadirse el fichero /etc/services la siguiente información:

```
pbs          15001/tcp     #pbs_server
pbs_mom      15002/tcp     #desde/hacia el mom hasta el server
pbs_resmom   15003/tcp     #para peticiones de recursos del mom
pbs_resmom   15003/ucp    #para peticiones de recursos del mom
pbs_sched    15004/tcp     #pbs_sched (planificador)
```

Los números de puertos arriba especificados son los números por defecto de la última versión del PBS. Es recomendable no cambiarlos pero en el caso de hacerlo se deberá usar el mismo número consecuentemente en todos los sistemas relacionados.

2.6.3.4.- Arrancando el servidor.

La primera vez que se ejecute el servidor se le deberá indicar que cree una nueva base de datos. Para ello desde se ejecutará el pbs_server con la opción – t create de la forma:

Shell> [ruta\] pbs_server –t create

En el caso de que se encuentre ya el demonio en ejecución bastará con enviarle la señal del terminación de la forma:

Shell> kill -15 pid_proceso

y luego lanzar de nuevo la orden de ejecución con el –t create.

En este punto el servidor se encuentra en un estado inactivo y no se puede aún mandar ningún trabajo a ejecutarse en los nodos ni se puede interactuar con el planificador. Por lo tanto habrá que poner al servidor en el estado activo a través de la siguiente orden:

Shell> qmgr –c "set server scheduling=true"

2.6.3.5.- Arrancando el planificador.

El planificador no requiere ninguna configuración especial directamente. Bastará con que se lanze en el arranque el demonio correspondiente (pbs_sched)

2.6.3.6.- Configuración del servidor (pbs_server).

Existe un gran número de atributos configurables en el servidor, los cuales se pueden determinar mediante la orden set server (s s) del intérprete qmgr. La gran mayoría de los atributos de configuración son opcionales salvo excepciones como el atributo default_queue que indica cual es la cola por defecto (una concreta o ninguna). La clasificación de atributos configurables son:

*** Atributos requeridos:**

default_queue: Este atributo determina cual es la cola por defecto del sistema de colas a la cual se envían los trabajos para que el *pbs_sched* se encargue de su planificación. Si se quiere especificar una cola por defecto deberá estar creada antes. Para crear una cola se deberá ejecutar la orden create queue desde el qmgr de la forma:

qmgr: *c q nombre_cola queue_type=tipo_cola (explicado en la sección X.X.X)*

y para determinar la cola por defecto se deberá ejecutar la orden s s default_queue desde el qmgr de la forma:

qmgr: *s s default_queue=nombre_cola*

*** Atributos configurables opcionales:**

acl_hosts (importante): Determina la lista de hosts a los cuales se les puede enviar trabajos. Se pueden especificar nombres de hosts individuales o directamente especificar todos los hosts de un subdomonio de la forma *.subdomnio. Por ejemplo:

qmgr: *s s acl_hosts = consola1.subdominio.com*

qmgr: *s s acl_hosts = *.subdominio.com*

acl_host_enable: Permite al hosts servidor acceder a la lista de control

qmgr*: s s acl_host_enable=true/false*

default_node (importante): Determina el nodo por defecto en el cual se van a lanzar todos los trabajos en caso de no especificarse ninguno. La política de en que nodo se lanza un trabajo ya está explicada en el apartado X.3.2. Para especificar un nodo por defecto bastará con ejecutar la siguiente orden desde el qmgr:

qmgr*:s s default_node=nombre_nodo*

managers: Determina que usuarios tienen permisos de administración del sistema PBS. Para ello habrá que especificar el nombre de usuario seguido del nombre del subdominio en cuestión de la forma:

qmgr: *s s managers = yo@mi.domonio*
qmgr: *s s managers = pedro@*.dominio*

node_pack (importante): Determina el orden en el cual los múltiples nodos del cluster son asociados a trabajos. De nuevo viene determinado su efecto por las reglas explicadas en el apartado X.3.2. Si se especifica a *true* se repartirán los trabajos entre el menos número posible de nodos, si se especifica a *false* los trabajos en cambio se intentan repartir lo más posible entre los distintos nodos. Si finalmente se especifica a *unset* entonces se seguirá el orden en que los nodos fueron declarados al servidor.

query_other_jobs: Determina la posibilidad de acceder o no al estado de trabajos lanzados por otros usuarios. Si no se encuentra especificado o se encuentra a falso no se podrá acceder al estado de cualquier trabajo que no pertenezca al usuario en cuestión. Es recomendable configurarlo a *true.*

qmgr: *s s query_other_jobs = true*

resources_default: Establece los límites de recursos asignados a trabajos que fueron lanzados sin especificar ningún limite. Es importante especificar un valor por defecto para cualquier requerimiento de recursos que se pueda usar en la política de planificación. Para obtener información acerca de los recursos para el PBS de una distribución determinada vease la página del manual *pbs_resources_**.

Resources_max: Establece la máxima cantidad de recursos que pueden ser usados por un trabajo una vez que se encuentra en una cola. Este límite sólo se consulta sino se encuentra especificado el atributo *resources_default.*

2.6.3.7.- Configuración de las colas.

Existen 2 tipos de colas distintas definidas por el PBS, colas de enrutamiento (routing) y de ejecución (execution). Una cola de enrutamiento se usa para mover

trabajos a otras colas que pueden incluso existir en distintos servidores PBS. Este tipo de colas para el caso particular del cluster que nos ocupa no es importante. Las colas de ejecución son las colas en las que se encuentran los trabajos listos para ser ejecutados. Un trabajo permanece en la cola de ejecución hasta que se ha terminado de ejecutar.

Un servidor puede tener varias colas de los 2 tipos, pero siempre debe tener especificada una cola por defecto y normalmente será del tipo ejecución puesto que ningún trabajo encolado en una cola de enrutamiento puede ser lanzado a ejecución.

Existen distintos tipos de atributos para las colas parte de los cuales sólo son relativos a un tipo de colas y parte de los cuales son generales a los 2 tipos. Los atributos necesarios a tener en cuenta para la creación del cluster de ps2 son los siguientes:

*** Atributos requeridos por los 2 tipos de colas:**

queue_type: Indica si una cola es de tipo *routing* o de tipo *execution* y debe estar siempre especificado para cada cola. La especificación se realiza mediante la orden:

qmgr: *s s nombre_cola queue_type = execution/routing*

enabled: Determina si una cola se encuentra activa o no. La orden para especificarlo es:

qmgr: *s s nombre_cola enabled = true/false*

started: Este atributo determina si se encuentra a *true* que los trabajos en una cola sean procesados, por lo tanto debe ponerse a *true* siempre para cada cola.

qmgr: *s s nombre_cola started=true*

*** Atributos requeridos por las colas routing (sin relevancia para el tipo de cluster objetivo del proyecto):**

route_destinations: Indica las colas locales o de otros servidores a las cuales se deben mandar los trabajos encolados.

qmgr: *s q nombre_cola route_destination=nombrecola[@dominio]*

*** Atributos recomendados para los 2 tipos de colas:**

resources_max: Este atributo permite establecer el límite máximo de recursos que pueden consumir los trabajos encolados. Por lo tanto establece un filtro de trabajos en la cola. Sino se especifica un valor para un determinado recurso entonces no existe restricción para dicho recurso. Por ejemplo si se quiere especificar que ningún trabajo consuma más de 2 horas de tiempo de CPU se especificará el recurso límite de CPU a un máximo de 2 horas. De esta forma ningún trabajo que demande más de ese tiempo será aceptado en la cola.

resources_min: Equivalente al anterior pero especificando el límite mínimo de recursos.

***Atributos recomendados para las colas de ejecución:**

resources_default (importante): Define un conjunto de valores por defecto para los trabajos que no especifican límites de recursos. Es muy importante especificarlo puesto que sino cualquier trabajo que no indique explícitamente su demanda de recursos no será encolado muy posiblemente. Una especificación correcta de este atributo que permita a los trabajos que no especifican recursos ser encolados y posteriormente ejecutados es la siguiente:

qmgr: *s q nombre_cola resources_default.nodes=1*
qmgr: *s q nombre_cola resources_default.nodect=1*
qmgr: *s q nombre_cola resources_default.neednodes=1*

2.6.3.8.- Configuración de las políticas de planificación.

El fichero que determina la política de planificación que se aplica al sistema de colas se encuentra en la ruta */PBS_HOME/server_priv/sched_config.* El formato de dicho fichero es el siguiente:

opción: valor [valores asociados]

donde opción es cualquiera de las mostradas abajo; valor toma los valores true, false, un número o una cadena de texto; y finalmente el campo valores asociados que indica de la forma [prime | non_prime | all] su campo de acción. A continuación se muestran las opciones más habituales con sus valores por defecto entre llaves:

round_robin: Si este atributo se especifica a *true*, los trabajos se van ejecutando con una política de turno rotatorio. Si se pone a *false* se intentará ejecutar cada uno de los trabajos almacenados en la cola antes de pasar al siguiente. Los parámetros que controlan si un trabajo puede ser ejecutado son *resources_max, max_running, max_user_run* y *max_group_run.* {false all}

by_queue: Si se especifica a *true* los trabajos serán ejecutados desde la cola respectiva, si se pone a *false* se tratará todo el conjunto de trabajos del servidor como si estuviesen encolados en una sola cola global. {true all}

strict_fifo: Si se especifica a *true* los trabajos se ejecutarán en un orden FIFO estricto. Si este valor no esta especificado o esta a *false*, pueden existir trabajos en inanición permanente si existen siempre trabajos encolados que requieran menos recursos. {false all}

fair_share: Este atributo activa o desactiva el algoritmo de compartimiento justo. {false all}

load_balancing: Si se especifica a true si se desea que el planificador reparta los trabajos equilibradamente entre todos los nodos que se han declarado como time shared. {false all}

help_starving_jobs: Este atributo si se pone a *true* activa el sistema de ayuda del planificador para trabajos en inanición. Si un trabajo ha superado el tiempo máximo de espera indicado por el parámetro *starve_max* entonces el trabajo se considera en estado de inanición. Este parámetro de tiempo máximo debe ser establecido siempre.

2.6.3.9.- Declaración de los hosts que forman los nodos del cluster.

Los nombres de los hosts que forman los nodos del cluster deben ser declarados explícitamente al servidor. Para ello hay que editar el archivo PBS_HOME/mom_priv/ config y escribir una línea por cada nodo que queramos declarar de la forma:

$clienthost nombre_de_host

Si este fichero no está configurado el servidor no tendrá conocimiento de los nombres de host de los clientes por lo tanto no funcionará la comunicación entre nodos y servidor. Por ejemplo:

$clienthost consola1.cluster

Este mismo proceso debe ser realizado también en los clientes en el sentido inverso (explicado en la sección X.4.1)

2.6.4.- Configuración de las ps2.

El proceso de configuración de los nodos es bastante rápido puesto que la gran parte del trabajo de configuración se realiza desde el servidor.

2.6.4.1.- Declaración de los hosts que forman los nodos del cluster.

Cada nodo debe declarar explícitamente de que servidor es cliente. Para ello hay que editar el archivo PBS_HOME/mom_priv/ config y realizar la declaración del host del que se es cliente de la forma:

$clienthost nombre_de_host

Si este fichero no está configurado los nodos son serán capaces de comunicarse con el servidor. Por ejemplo:

$clienthost frontend.cluster

indica que el servidor es el que tiene por nombre de host frontend.cluster.

2.6.5.- Solución de problemas.

Pueden darse una gran variedad de problemas a la hora de hacer funcionar el sistema de colas, los cuales pueden darse por razones muy variadas que en muchos casos tienen que ver un desconocimiento del sistema operativo. Aquí se detallarán sólo los problemas más comunes y propiamente dichos de la configuración del PBS y no de cuestiones del sistema operativo.

2.6.5.1.- Los clientes no pueden contactar con el servidor.

Si cualquier comando desde el cliente que se ejecute es incapaz de contactar con el servidor existen varias posibles razones. Si el error devuelto es el 15034 "No server to connect" debe verificarse que efectivamente el servidor se encuentra activo y que la información del mismo es la adecuada. Para verificar dicha información se debe hacer lo siguiente:

Shell ps2> qmgr

Qmgr: *list server*

y se imprimirá por pantalla la información del servidor.

Si el error es del tipo 15007 "No permission" debe verificarse que en el servidor se han dado los permisos apropiados al hosts cliente (/etc/hosts.equiv). Si se han dado los permisos y aún así sigue apareciendo este error debe verificarse que en el Path del cliente se encuentra incluido el ejecutable *pbs_iff.*

2.6.5.2.- Los nodos se encuentran caídos.

Desde el servidor se puede consultar el estado de los nodos de 2 formas:

1. *Shell> pbsnodes –a*
2. *qmgr*
   list nodes @active

Si la información del nodo es la siguiente:

Node nodo1
State = down,state-unknown
Properties = "…."
Ntype = cluster

significa que el servidor no contactado nunca con el nodo desde que se arrancó el servicio. Debe chequearse que el demonio *pbs_mom* este corriendo en la ps2. Si se encuentra corriendo y aún así se tiene este problema, debe verificarse que el nombre del nodo especificado en el servidor (fichero nodes) se corresponde realmente con el nombre

real de host en la red. En caso de corresponderse puede tratarse de un problema en la red.

Si por el contrario la información del nodo es:

Node nodo1
State = down
Properties = "…."
Ntype = cluster

significa que el servidor ha podido contactar con la ps2 en el pasado en algún momento pero actualmente no es capaz de hacerlo. El servidor envía un ping a cada nodo periódicamente cada 10 minutos y es en ese momento cuando cambia el estado de los nodos según respondan o no. En este caso el problema seguramente sea de la red de conexión.

2.6.5.3.-No se recibe el resultado de los trabajos.

Si no se puede enviar al servidor el resultado de un trabajo, este se salva en un directorio especial PBS_HOME/undelivered y un mail se envía al usuario. Las causas típicas de este problema son:

* El host de destino no es fiable (trusted) y el usuario no posee un fichero .rhost.
* No se ha especificado un path correcto.
* El directorio en la especificación destino no es de escritura.
* El fichero .cshrc del usuario en el host destino genera salida cuando se ejecuta.
* El directorio spool en el host de ejecución no tiene los permisos adecuados (deben ser 1777).

2.6.5.4.-No se pueden ejecutar los trabajos.

Si se recibe el mensaje *"Job cannot be executed"* significa que el trabajo fue abortado por el mom cuando intento ejecutarlo en la ps2. La razón de esta cancelación se encuentra en el fichero .log del mom o en el fichero de salida de error estándar del usuario.

2.6.5.5.- No se reconoce el nombre del servidor.

Editar el fichero /usr/spool/PBS/server_name y escribir en él el nombre de host del servidor correcto.

2.7.- Librerías de paso de mensajes Mpi.

2.7.1.- Instalación en las ps2.

Al igual que el PBS no existe ningún paquete para la distribución BlackRhino, por lo que de nuevo habrá que bajarse el código fuente y compilarlo en la ps2. El código fuente se encuentra disponible en el ftp *ftp://ftp.mcs.anl.gov/pub/mpi/mpich.tar.gz.* Para descargarlo basta con escribir desde el shell la siguiente orden:

Shell ps2> wget ftp://ftp.mcs.anl.gov/pub/mpi/mpich.tar.gz.

y después basta con descomprimirlo en la carpeta que se desee:

[Directorio Destino]Shell ps2> tar –zxvf [Ruta\]mpich.tar.gz

Una vez hecho esto, basta con ejecutar los siguientes comandos desde el shell:

Shell ps2> ./configure –prefix = /usr/local/mpich-1.2.5

Shell ps2> ./make

y ya se tiene instalado todo lo relacionado con MPI en la ruta /usr/local/mpich-1.2.5. El ejecutable mpich encargado de lanzar un trabajo en pararelo se encuentra en la ruta /usr/local/mpich-1.2.5/util.

A parte del MPI propiamente dicho es necesario instalar el servicio rsh en las ps2 puesto que el MPI lo necesita para conectarse a las distintas máquinas a la hora de ejecutar trabajos en pararelo mediante paso de mensajes. Afortunadamente si existen paquetes para la distribución para el rsh por lo que sólo habrá que instalarlos normalmente (véase Apéndice1).

2.7.1.- Configuración.

2.7.1.1.- Actualización de permisos.

Es necesario establecer los permisos adecuados para que el mpich pueda conectarse a las diferentes máquinas a la hora de ejecutar trabajos en pararelo. Si no se configuran adecuadamente los permisos aparecerá el mensaje de error “Permission denied” en cuanto se intente ejecutar algo mediante el mpich. El fichero hosts.equiv tiene el siguiente aspecto:

Linea de hosts.equiv : (host | dirección IP) [usuario]

Por lo tanto para otorgar permisos basta con escribir por cada línea la dirección IP/nombre host de cada una de las otras ps2, o bien si se quiere ser un poco más estricto escribir la dirección IP/host seguida de un usuario determinado para darle sólo permisos a ese usuario en concreto. Así por ejemplo se puede escribir:

1: Consola1.cluster
2: Consola2.cluster root
3: 192.168.0.3

de forma que se da autorización al nombre de host consola1.cluster, a la dirección IP 192.168.0.3 y al administrador del host consola2.cluster.

2.7.1.2.- Determinación de dónde se ejecutan en paralelo los programas.

El donde se va a ejecutar un trabajo en paralelo viene determinado por el fichero *usr/local/mpich-1.2.5//util/machines/machines.LINUX*. En dicho fichero se especifican las direcciones IP o nombres de hosts de las máquinas donde se va realizar el trabajo en pararelo. Por defecto siempre que se quiera lanzar un trabajo paralelo en varias máquinas la primera de ellas es la propia máquina desde donde se lanza el trabajo salvo que se diga lo contrario. El resto de máquinas son las especificadas en el fichero machines.LINUX en el mismo orden en que se escriben. Una configuración de ejemplo podría ser:

```
# fichero machines.LINUX
192.168.0.2
192.168.0.3
192.168.0.4
```

en el cual se establece que si un trabajo paralelo se desea lanzar en 4 máquinas, la primera de ellas será la propia que lanza el trabajo, la segunda de ellas la correspondiente a la IP 192.168.0.2, la tercera la correspondiente a 192.168.0.3 y la última la correspondiente a 192.168.0.4. Si se desease lanzar el trabajo en más máquinas que las especificadas en el fichero machines.LINUX, se ejecutaría de nuevo el resto del trabajo restante volviendo a seguir el mismo orden del fichero como si fuese una cola circular. Por lo tanto si con la configuración anterior se quisiese lanzar un trabajo pararelo en 5 máquinas, la quinta máquina volvería a ser la correspondiente a la dirección IP 192.168.0.2. Un pequeño truco consiste en escribir la dirección IP de la máquina actual en el último lugar para en caso de superarse el límite de máquinas especificadas se siga repartiendo el trabajo en el mismo orden empezando por la propia máquina que lo lanza.

2.7.1.3.- Opciones de lanzamiento de un programa paralelo.

Para lanzar un programa en paralelo previamente compilado con la librería mpi, se debe realizar mediante el ejecutable mpich. La sintaxis del mpich es:

```
Shell ps2> mpich [opciones mpi] programa_paralelo.c [opciones programa]
```

La lista de opciones mpi determina las circunstancias en las que se lanza el trabajo. Las opciones más importantes son:

-arch<arquitectura>: Determina la arquitectura correspondiente, la cual se debe tener correspondencia en el fichero machines.arch en el directorio /usr/local/mpich-1.2.5/util/machines.

-h: Muestra la ayuda.

-machine<nombre de la máquina>: Usa el procedimiento de arranque para la máquina especificada.

-machinefile <fichero>: Especifica un nombre de fichero el cual se usará en vez del correspondiente machines.LINUX. Este fichero jugará exactamente el mismo papel que el machines.LINUX.

-np<número>: Especifica en cuantas máquinas se quiere distribuir el trabajo pararelo.

-nolocal: Especifica que no se quiere lanzar el trabajo en la máquina actual..

-t: No ejecuta nada, solamente se imprime lo que pasaría si se ejecutase.

-v: Opción verbose, escribe por pantalla comentarios relevantes.

3.- Bibliografía

"**EE user's manual**" Sony Computer Entertainment Inc. y "**VU user's manual**" Sony Computer Entertainment Inc.

"**Red Hat Linux 9: Red Hat Linux Referente Guide**", by Red Hat, Inc.

"**Red Hat Linux Networking and system Administration**", Terry Collings & Kurt Wall". M&T Books (2002).

"**Red Hat Linux Networking and system Administration**", Terry Collings & Kurt Wall". M&T Books (2002).

"**Red Hat Linux Networking and system Administration**", Terry Collings & Kurt Wall". M&T Books (2002).

Deconstructing PlayStation 2
http://xengamers.com/sections/features/5944/1

PS2: Aliased No More:
http://ps2.ign.com/articles/081/081661p1.html

The Playstation2 vs. the PC: a System-level Comparison of Two 3D Platforms
http://www.arstechnica.com/reviews/1q00/playstation2/ee-1.html

NFS-Root mini-HOW TO **from** http://www.tldp.org/HOTTO/mini/NFS-Root.html

Sound and Vision: A Technical Overview of the Emotion Engine:
The Linux NIS(YP)/NYS/NIS+ HOWTO from http://www.linux-nis.org/nis-howto/

Enlaces visitados en Internet:

http://arrakis.ncsa.uiuc.edu/ps2/index.php
http://blackrhino.xrhino.com
http://clusters.top500.org
http://playstation2-linux.com
http://ps2dev.sourceforge.net/
http://ps2linux.diabolus-ex-machina.com/wiki/index.php
http://www.beowulf.org
http://www.buyya.com/cluster
http://www.cesga.es
http://www.clustercomputing.org
http://www.clusterworldexpo.com/default.asp
http://www.hispacluster.org
http://www.playstation.com
http://www.unix.mcs.anl.gov/mpi
http://www.xrhino.com/

4.- Palabras clave.

Cluster, PlayStation2, BlackRhino, Emotion Engine, Nis, Nfs, Pbs, Mpi, Supercomputación, linux.

5.-Autorización.

Los alumnos Alberto Fernández Sánchez, Diego Arnáiz García y Javier Martín Esquifino de la Facultad de Informática de la Universidad Complutense de Madrid, autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado del proyecto que se menciona en este documento.

Fdo. Alberto Fernández Sánchez.

Fdo. Diego Arnáiz García.

Fdo. Javier Martín Esquifino.