95
2y

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"ANALISIS DEL RENDIMIENTO DEL SISTEMA DE COLAS DE LA SUPERCOMPUTADORA CRAY Y-MP"

T E S I S

Que para obtener el título de

INGENIERO EN COMPUTACION

P r e s e n t a n

CARLOS PATIÑO LARA
MARIA DEL PILAR VALDIVIA SOTO

DIRECTOR: DR. ENRIQUE DALTABUIT GODAS

México D. F. 1996

Dedico este trabajo a mi MADRE con todo mi amor.

Pensando siempre en ti mamá . . .

Pilar

A mis padres Lydia y Guillermo, a quienes debo todo lo que soy hoy y siempre y agradezco infinitamente el haberme educado dentro del placer de entender las cosas.

Carlos

AGRADECIMIENTOS

Agradezco a mi padre Andrés Valdivia Valencia† por sus enseñanzas y consejos que siempre permanecen en mi mente y viven conmigo.

Gracias a mi madre Ma. del Pilar Soto de Valdivia por tanto amor, por haberme brindado la oportunidad de estudiar.

A mis hermanos Lina, Andrés† y Enrique.

A mi hermano Rodolfo por su confianza y apoyo.

En especial a mi hermano Alberto mi gran amigo, quien siempre me ha dado su ayuda incondicional, su atención y cariño.

Pilar

A la memoria de mi abuelita Irene, quien siempre me alentó a seguir adelante y me enseñó la importancia de concluir una carrera universitaria.

A mi abuelita Chelo, por su cariño y por sus ganas de disfrutar la vida.

A mi hermano Israel por su apoyo y comprensión en los momentos difíciles de la carrera. Espero que esto te motive a que tú también les des está satisfacción a nuestros padres.

Muy especialmente a mi hermano Guillermo quien sin su apoyo, comprensión y ayuda incondicional no me hubiera sido posible concluir la carrera.

Carlos

A la máxima casa de estudios, la Universidad Nacional Autónoma de México por habernos abierto sus puertas.

A la Facultad de Ingeniería por siempre nuestra casa, y a sus maestros, por la formación profesional que nos han legado.

A la Dirección General de Servicios de Cómputo Académico, en donde complementamos nuestra carrera universitaria.

De manera muy especial al Dr. Enrique Daltabuit Godas, porque sin su valiosa dirección, apoyo y comprensión, no hubiera sido posible concluir este trabajo. Porque gente como usted es la que necesita el país. Con gran admiración y respeto hacia usted . . . ¡ Gracias !

Pilar y Carlos

CONTENIDO

CONVENCIONES UTILIZADAS

Itálica	Se utiliza para nombres de comandos y utilerías de UNIX, argumentos, directorios y nombres de archivos, así como para hacer enfasis en nuevos términos y conceptos, cuando son mencionados por primera vez.
Itálica Negrita	Se utiliza para señalar aquellas palabras que forman parte del glosario de terminas.
`Constante Normal`	Se utiliza para mostrar la salida de comandos, en los ejemplos, y para mostrar el contenido de algunos archivos.
`Constante Negrita`	Se utiliza para nombres de comandos, en los ejemplos, o para texto que debe ser tecleado literalmente por el usuario.
`Constante Itálica`	Se utiliza para mostrar fragmentos de código y en algunos ejemplos para denotar variables cuyo valor debe ser sustituido. (Por ejemplo la variable *`filename`*, debe ser sustituida por el nombre de archivo correspondiente).

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

La Supercomputadora CRAY de la UNAM
Descripción del Problema
Propuesta de Solución
Objetivo del Trabajo de Tesis
Contenido del Trabajo de Tesis

El surgimiento de las computadoras, así como el de muchos de los grandes inventos de que ahora gozamos, obedeció a la necesidad del hombre por resolver problemas en un menor tiempo y con un menor esfuerzo. Así como también a la necesidad de resolver problemas que no se podían solucionar con las herramientas que existían. Y es por lo mismo que constantemente se buscan mejoras a lo ya existente, ya que la reducción de tiempo y esfuerzo, permite aumentar la productividad, lo cual puede producir ganancias económicas.

Desde la 1ra computadora que existió hasta los más poderosos sistemas computacionales de nuestros días, se ha perseguido afanosamente el abatir costos en materiales para su fabricación y a su vez, lograr el más alto nivel de integración y producción.

En esta lucha permanente, se han creado nuevas tecnologías, que obedecen a la demanda de eficiencia y efectividad que se requiere en los diversos campos de la ciencia y la tecnología. Pero es preciso considerar, que no siempre es necesario el invertir más dinero para lograr la eficiencia requerida, sino saber explotar el equipo con que se cuenta de una manera óptima.

En ocasiones el problema se origina en la compra del equipo, en donde no se determina y analiza el objetivo para el cual se realizará la compra, y dejándose llevar por anuncios publicitarios, se toman en cuenta factores tales como el precio, la apariencia, cifras de velocidades en los diferentes dispositivos, etc., que si bien, son factores determinantes, no son ni los únicos ni los más importantes.

Es por esto que se ha dedicado un especial estudio al rendimiento de los sistemas, en donde se puede determinar si en realidad las características del mismo son insuficientes para las aplicaciones y los usuarios que se tienen, o si el error es que no se están usando correctamente.

El estudio del rendimiento de los sistemas computacionales, es un aspecto sumamente importante y que en muchas ocasiones se pasa por alto, ya sea por

el desconocimiento de su importancia, o por no contar con personal capacitado en esta área. La complejidad del análisis del rendimiento de un sistema, es proporcional a la complejidad del mismo. Además, entre más sofisticado sea el sistema, se requiere mayor estudio de su rendimiento y es más difícil lograr que todos sus componentes trabajen en su punto óptimo. Tal es el caso de las Supercomputadoras.

El término "Supercomputadora" es atribuido a la computadora con más poder de cómputo en cierto momento. Lo anterior implica que solamente un modelo de computadora podría tener este calificativo en un momento dado. En la práctica, un número reducido de computadoras de diferentes fabricantes tiene el honor de agruparse dentro de la clase de supercomputadoras. Y no es posible decir que un modelo de computadora es la más poderosa. Ya que el poder de una computadora no es lineal. Muy a menudo las diferentes arquitecturas de cada tipo de máquina son un factor significante en determinar cuál supercomputadora es la mejor para determinada aplicación, ya que se ha demostrado que el poder de las supercomputadoras actuales es dependiente de la aplicación.

Las supercomputadoras han existido desde la invención de las computadoras. Son máquinas muy complejas para construirse y por lo mismo, son muy costosas. Estas computadoras son capaces de funcionar como computadoras de propósito general, o correr aplicaciones de negocios si es necesario. Sin embargo el diseño, producción y explotación de las supercomputadoras es un negocio muy serio que responde principalmente a la demanda militar y científica. Ya que lo que mejor saben hacer, es resolver problemas en los que se tienen que calcular muchos números, con alta precisión y exactitud.

El poder de una computadora científica es medido por su velocidad al calcular números de punto flotante, almacenamiento en memoria principal, la precisión y rango de los números que calculan y el ancho de banda de la información que manejan.

Una supercomputadora no es una simple máquina monolítica. Es un gran número de estructuras independientes unidas en una arquitectura. Tales estructuras son unidades funcionales escalares y vectoriales, memoria central, ***buffers*** de instrucción y subsistema de Entrada/Salida (E/S). Ambas, la independencia de las estructuras y la habilidad de interactuar durante el procesamiento, proveen de un paralelismo substancial en todo el sistema.

El objetivo de todo diseñador de computadoras, es producir una computadora que realice cálculos de manera más barata que sus rivales. Como los grandes sistemas son muy complejos y consisten de un número de estructuras unidas, el éxito de cada arquitectura es determinado por qué tan bien realiza su objetivo. Esto depende en gran parte de la tecnología usada para construir la máquina y la confiabilidad de sus componentes electrónicos. Otros factores incluyen si las diversas estructuras de la computadora están integradas lo suficientemente para producir un sistema balanceado, y si el paralelismo esta disponible en la

arquitectura del sistema, pudiendo ser explotado fácilmente durante la solución de grandes problemas.

Las arquitecturas desbalanceadas pagan grandes precios en el rendimiento siempre que alguna aplicación provoca un cuello de botella en el sistema. La tarea de los administradores del sistema, de los ingenieros de software y de los programadores de aplicaciones es no rebasar las fronteras de los cuellos de botella y maximizar la explotación de los elementos paralelos en el hardware.

1.1. LA SUPERCOMPUTADORA CRAY DE LA UNAM

En Septiembre de 1989, la Coordinación de la Investigación Científica, en colaboración con la Dirección General de Servicios de Cómputo Académico (DGSCA), organizó el Primer Seminario sobre Aplicaciones de Supercómputo. Lo anterior debido a la inquietud que existía de que la Universidad Nacional Autónoma de México (UNAM) incorporara al Supercómputo como una más de las herramientas a disposición de la comunidad académica.

En dicho seminario se mostró la evidencia de que los diversos grupos de investigación tenían los elementos necesarios para requerir la puesta en operación de una supercomputadora. La adquisición de una supercomputadora estimularía cambios cualitativos en áreas de investigación ya establecidas y permitiría realizar investigaciones novedosas en las diferentes áreas.

A finales de 1989 se formó la Comisión de Supercómputo, la cual tendría la misión de buscar diferentes propuestas y analizarlas para escoger la mejor, y realizar la adquisición de una supercomputadora. Para lo anterior, dicha comisión visitó centros de Supercómputo, realizando ***benchmarks*** y analizando las principales características de los equipos considerados, además de las cotizaciones propuestas.

Finalmente la Comisión de Supercómputo consideró que la propuesta de la empresa CRAY Research Inc., era la mejor debido a que ofrecía un equipo muy probado, con una base de usuarios extensa y con los mejores paquetes y programas disponibles en el mercado. La propuesta elegida incluía como condición para la adquisición de la supercomputadora, el establecimiento de un laboratorio de investigación y de una red de alta velocidad que permitiera la comunicación adecuada entre el laboratorio y la supercomputadora.

La Universidad Nacional Autónoma de México, a través de la Dirección General de Servicios de Computo Académico, adquirió una supercomputadora CRAY modelo Y-MP. Dicha computadora entró en operación en noviembre de 1991, junto con el Laboratorio de Visualización.

Características de la CRAY

Físicamente, la supercomputadora CRAY de la UNAM, consta del gabinete principal (*mainframe*) en el cuál se ubican los cuatro procesadores con los que cuenta la máquina y el dispositivo de estado solido (SSD), además de un gabinete para el sistema de refrigeración, otro para el sistema de E/S y cuatro gabinetes más donde se almacenan los discos. El gabinete principal de la supercomputadora mide 1.90 m de altura y pesa 2,450 Kg, ocupando un área de 1.5 m^2.

Características de Hardware

A continuación se listan las características principales del hardware de la supercomputadora:

- Cuenta con cuatro procesadores diseñados por CRAY Research Inc. capaces de trabajar en paralelo, así como realizar operaciones escalares y vectoriales. Cada procesador, teóricamente puede realizar 166 MIPS (166 Millones de Instrucciones Por Segundo), por lo que haciendo trabajar a los cuatro procesadores simultáneamente e independientemente a toda su capacidad (lo que no es una condición cotidiana de trabajo) se podrían alcanzar 664 MIPS. Por otro lado, en modo de operación vectorial, cada procesador puede alcanzar un rendimiento pico de 333 MFlops (333 Millones de operaciones de punto Flotante por segundo) para totalizar 1,332 MFlops con los cuatro procesadores trabajando.

- La memoria central de la supercomputadora es de 64 MW (MegaWords, donde una Palabra en CRAY es igual a 64 bits). Cabe mencionar que en la literatura referente a supercomputadoras, las unidades de almacenamiento en memoria se expresan en MegaWords y no en MegaBytes como podría suponerse.

- Cuenta con un Dispositivo de Estado Sólido (SSD) de alto rendimiento para almacenamiento temporal de datos, con una capacidad de 128 Mega-palabras, y una velocidad de transferencia con la memoria central de 1,000 MB/s a través de canales especiales.

- Cuenta con un subsistema de entrada y salida (IOS), encargado de transferir los datos entre la memoria central, y los diferentes periféricos así como también hacia el dispositivo de almacenamiento temporal SSD, todo esto mediante un ***buffer*** de memoria de 4 MW y 10 canales de E/S (I/O), 4 de ellos de baja velocidad (6 MB/s), 4 de mediana velocidad (100 MB/s) y 2 de alta velocidad (1,000 MB/s).

- Posee una capacidad en disco de 38.4 GB, repartidos en 8 unidades de disco modelo DD-41, con una velocidad de transferencia de 9.6 MB/s y tiempo de acceso promedio de 16 ms, fabricados por CRAY Research Inc.

- Cuenta con una unidad para 4 cartuchos marca Storagetek, con una capacidad de almacenamiento de hasta 200 MB por cartucho.

- Tiene un robot con capacidad de 800 cartuchos que se utiliza para la migración de datos que no han sido utilizados.

- Para conectar a la supercomputadora con la red universitaria, existen 2 ruteadores con interfaces Ethernet y FDDI cada uno.

- Cuenta también con un sistema de refrigeración, encargado de mantener a los procesadores y unidades de disco a temperatura adecuada, por medio de un líquido llamado *fluorinert*, el cuál se distribuye, en los dispositivos a través de un sistema de conducción.

La supercomputadora, además de las características en hardware arriba presentadas, cuenta con dispositivos dedicados a un tipo muy especifico de calculo o acción, como son los registros vectoriales, escalares y de dirección, así como también las unidades funcionales de suma, multiplicación y aproximación reciproca, lo que permite a la supercomputadora realizar en algunos casos varias partes de un mismo cálculo casi de manera simultánea. De lo anterior, se desprende la gran capacidad de la supercomputadora para realizar operaciones aritméticas a gran escala y a muy altas velocidades.

Caracteristicas de Software

El sistema operativo de la supercomputadora CRAY es UNICOS versión 8.0.3, el cuál esta basado en UNIX ***System V*** Release 4 desarrollado por los laboratorios AT&T, además de contar con extensiones Berkeley Release 4.3.

El sistema operativo UNICOS, cuenta con capacidades de multiproceso, además de contar con un sistema de archivos distribuido en diferentes unidades físicas.

En cuanto a los compiladores con los que cuenta la supercomputadora CRAY, se encuentran instalados los compiladores de lenguaje "C" versión 4.0.3.20, FORTRAN 77 versión 6.0.4.0 y el compilador de FORTRAN 90 versión 1.0.3.4, con capacidades de optimización y vectorización automática de código.

Existe también instalada en la supercomputadora, la biblioteca científica, la cuál se puede definir como un conjunto de rutinas en su mayoría enfocadas a la realización de cálculos numéricos, a las que es posible acceder desde cualquier programa. Estas rutinas están altamente optimizadas en todos los sentidos y el

código que las conforma fue previamente compilado e incluso reescrito en lenguaje ensamblador.

1.2. DESCRIPCIÓN DEL PROBLEMA

Por su naturaleza, es necesario en todo sistema UNIX, llevar una contabilidad del uso de recursos, ya que es información indispensable para que el administrador del sistema pueda analizar el rendimiento del mismo. El objetivo del análisis del uso de recursos es poder realizar un afinamiento, logrando que cada uno de los usuarios perciba el sistema como suyo, como si contara con recursos para él solo. Esto es algo que muy pocos administradores pueden lograr, y para ello se requiere de un análisis minucioso, después del cual, es posible adoptar ciertas medidas y/o cambiar ciertos parámetros de configuración.

La información necesaria para realizar el análisis de recursos, se concentra en archivos que la contabilidad de los sistemas UNIX genera automáticamente, siempre que esté activada. Todas las versiones del sistema operativo UNIX cuentan con un sistema de contabilidad, y UNICOS, el sistema operativo de la Supercomputadora CRAY, además de la contabilidad estándar, cuenta con un sistema de contabilidad diseñado especialmente para adaptarse a la arquitectura de un sistema CRAY, llamado CSA (CRAY System Accounting). Éste proporciona otras facilidades que no están disponibles en la contabilidad estándar, como la contabilidad por trabajo, por dispositivo, por ***daemons***, para el sistema de colas NQS, para el subsitema de cintas, etc.

La contabilidad genera archivos (y reportes) muy grandes y difíciles de analizar a simple vista. Estos archivos se componen generalmente de columnas de números que representan uso de recursos por proceso, registros de conexiones, de sesión, monitoreo del uso de disco y la carga para sesiones específicas. Estas medidas se encuentran en unidades diversas y se interpretan de diferentes maneras. Debido a lo anterior, se han creado múltiples herramientas, que procesan los archivos generados por la contabilidad y presentan la información de una manera fácil de entender.

En el Departamento de Administración de Supercómputo de la DGSCA, se emplean diversas utilerías para monitorear al sistema y analizar el rendimiento del mismo. Los administradores constantemente llevan a cabo tareas que les permiten observar el comportamiento del sistema de manera interactiva, además de realizar reportes periódicos de uso de recursos.

Una de las presentaciones más accesibles para analizar el rendimiento del sistema, la constituyen las gráficas. Es mucho más sensible a la vista, observar algún comportamiento de manera gráfica, que intentarlo por medio de gigantescas columnas de números. Es por ésto que en su mayoría, las herramientas existentes para análisis de rendimiento se componen de gráficas.

Por supuesto que los reportes en modo texto también tienen una gran importancia y son incluidos en gran parte de estas herramientas.

Algunas de las herramientas que se utilizan en el Departamento de Supercómputo son: el *systop*, que proporciona información por comando, indicando el usuario que lo ejecutó y el procentaje de CPU utilizado; y el *sam*, que es un monitor de la actividad del sistema.

El motivo por el cual se han tenido que crear herramientas propias, es porque muchas de las ya existentes no contemplan todas las características de la supercomputadora. Por lo que los científicos de CRAY se han preocupado por la creación de utilerías para el estudio del rendimiento.

Algunas otras han sido creadas en el Departamento de Supercómputo, adaptándose a los requerimientos que ahí existen. Para los reportes periódicos antes mencionados, se utilizan datos generados por el SAR (System Activity Report). El cual toma lecturas periódicas del uso global de recursos. Estos datos son procesados y generan gráficas que se utilizan para realizar cambios a la configuración del sistema con objeto de lograr un mejor rendimiento. Esto es mejor ya que presentan exactamente lo que el grupo de administración del Departamento necesita, y además se tiene la posibilidad de hacer cambios en el momento que se requiera. Estas herramientas fueron realizadas para efectuar un análisis previo a este trabajo de tesis, y de ellas se hablará más adelante.

Con la utilización de las herramientas que se han mencionado, la Supercomputadora de la UNAM, ha sido analizada y afinada, logrando un nivel de rendimiento de gran aceptación. Desafortunadamente, para el análisis del sistema de colas, no existe una utilería creada que pueda utilizarse en el Departamento de Supercómputo de la UNAM. Los estudios de rendimiento de este sistema tienen que llevarse a cabo analizando los datos de contabilidad directamente. Lo cual toma mucho tiempo y requiere de mayor esfuerzo.

Este sistema de colas con que cuenta la CRAY, es llamado NQS (Network Queuing System), y prevé la congestión difiriendo los trabajos hasta que el sistema tiene tiempo para ejecutarlos. El sistema de colas es muy útil, pero requiere de más conocimiento por parte de los usuarios.

El NQS permite a los usuarios someter peticiones de trabajos en *lotes* y encolarlos de acuerdo a sus características. Que son principalmente, el tiempo estimado de uso de CPU y la cantidad de memoria que utilizarán. Pero con frecuencia los usuarios no someten sus trabajos en la cola adecuada o someten más trabajos de los que está permitido al mismo tiempo. Lo anterior ya sea por ignorancia o por mal hábito. Esto ocasiona problemas de rendimiento, ya que algunos recursos se desperdician considerablemente.

Además, debido a los constantes cambios que sufre la población de usuarios de CRAY, los procesos que se someten en NQS, a menudo varían en tamaño y en

tiempo de ejecución. Por esto, se necesita realizar cambios a la configuración del sistema de colas. Es decir, analizar qué colas deben eliminarse y cuáles añadirse. Es necesario analizar cómo se ha comportado el sistema y qué características tendrán los nuevos procesos que se ejecutarán en NQS o cuáles tipos ya no se seguirán ejecutando.

El tiempo de análisis del sistema de colas es muy grande, ya que consiste en revisar reportes generados periódicamente. Estos reportes son obtenidos mediante la ejecución de comandos de UNICOS que toman la entrada de archivos de contabilidad, y los transforman en reportes diarios. Debido a ésto es muy difícil analizar el comportamiento en un periodo de tiempo grande, ya que hay que generar tantos reportes como días quieran analizarse. Otra desventaja es que el tamaño de los archivos de donde se lee la información es muy grande, por lo que es necesario almacenar esos archivos en cinta y cuando se requiere información de algunos de ellos, éstos tienen que ser leídos y bajados a disco, lo cual carga al sistema, consume disco, y sobre todo, es costoso en tiempo.

Otra fuente de información importante para el análisis del rendimiento del sistema de colas, son las anotaciones diarias hechas al monitorear el sistema interactivamente. Pero esta información es muy general, y con ella pueden resolverse solo una parte de los problemas existentes.

La demanda de uso de la Supercomputadora de la UNAM es cada vez mayor. Es por eso que el Departamento de Administración de Supercómputo de la DGSCA, busca constantemente nuevas alternativas que permitan mejorar el servicio en todos los aspectos y uno de los más importantes es el de lograr un buen rendimiento del sistema de colas NQS.

1.3. PROPUESTA DE SOLUCIÓN

La solución a la problemática existente, fue la de realizar un sistema que pudiera manipular la información proveniente de los archivos de contabilidad de la CRAY, convirtiéndola en gráficas que muestren y den respuesta a las preguntas que un administrador se hace cuando quiere valorar el comportamiento del NQS o desea reconfigurarlo. La carga generada por esta herramienta debe ser mínima, pero debe realizar la tarea de respaldar la información que necesite de manera automática. Además la información respaldada, no debe consumir mucho disco.

La solución propuesta constituye la realización de un monitor que permita al administrador del sistema, desplegar gráficas que muestren el comportamiento del sistema de colas. En base a estas, el administrador puede detectar si alguien esta haciendo mal uso del sistema de colas o si existen cuellos de botella provocados por fallas en la configuración.

El administrador podrá observar el comportamiento del NQS de acuerdo al período de tiempo que más le convenga. Ya que se contará con datos históricos, que estarán disponibles cuando así se requiera.

1.4. OBJETIVO DEL TRABAJO DE TESIS

Desarrollo de una herramienta gráfica para el análisis del rendimiento del Sistema de Colas de la Supercomputadora CRAY Y-MP, permitiendo el monitoreo y la visualización del sistema en forma sencilla, rápida y confiable.

1.5. CONTENIDO DEL TRABAJO DE TESIS

Este trabajo de tesis se compone de 6 capítulos. En los cuales se describen los antecedentes, conocimientos y pasos realizados, necesarios para llevar a cabo la elaboración del monitor gráfico.

A continuación se describe el contenido de los capítulos:

Capítulo 1, *Introducción*, tiene como finalidad orientar al lector y justificar en términos generales el trabajo de tesis. Así como proporcionar antecedentes del sistema a analizar.

Capítulo 2, *Monitoreo del Sistema*, proporciona la teoría necesaria para que el administrador pueda conocer su sistema, aspecto fundamental para que se pueda realizar un análisis de rendimiento. Además permite conocer de manera particular a la supercomputadora CRAY de la UNAM.

Capítulo 3, *El Sistema de Colas NQS*, proporciona los conceptos necesarios para conocer el sistema NQS. Así como la problemática ocasionada por un sistema NQS mal configurado o mal utilizado.

Capítulo 4, *El Sistema de Contabilidad de la CRAY CSA*, explica las características y funcionamiento del Sistema de Contabilidad de la CRAY.

Capítulo 5, *Análisis del Rendimiento*, explica la metodología utilizada para realizar el análisis del rendimiento de un Sistema. Así como la teoría necesaria para la comprensión, diseño y realización de un monitor de rendimiento. Además discute la importancia de la presentación de los datos.

Capítulo 6, *Desarrollo del Monitor Gráfico de NQS*, describe paso a paso el análisis, diseño, implementación y uso del monitor gráfico de NQS.

Apéndice, muestra el uso del monitor de NQS, mediante un caso practico de análisis del comportamiento del sistema de colas, durante un periodo dado.

Glosario de Términos, define algunos de los términos técnicos mas utilizados, a lo largo de este trabajo.

2. MONITOREO DEL SISTEMA

Comunidades de Usuarios
Control de Procesos
El Calendarizador
Sumisión de Trabajos en Horario de Bajo Uso
Limites de Uso de CPU
Herramientas para el Monitoreo del Sistema

El monitoreo del sistema es un factor de gran importancia para todo administrador, ya que es así como puede conocer más a fondo el sistema y reconocer los estados del mismo y las causas que los provocan. Es importante señalar que el monitoreo no debe comenzar cuando se han empezado a notar anomalías en el sistema o cuando se han recibido quejas de algunos usuarios. El monitoreo debe ser una de las principales tareas cotidianas del administrador, ya que es así como se pueden prevenir posibles problemas antes de que éstos se susciten. En la mayoría de las ocasiones es mucho más fácil resolver los problemas antes de que se presenten, que una vez que se encuentran en el sistema.

En algunas ocasiones, las rutinas de monitoreo diarias no pueden prevenir al administrador de posibles problemas de rendimiento, pero le arrojan información de que algo anormal esta sucediendo. El sistema parece estar muy lento, o los trabajos toman más tiempo en ejecutarse de lo que deberían. Entonces se detecta que el rendimiento del sistema es pobre y se deben considerar formas de mejorar la situación, sobre todo si el rendimiento cae repentinamente.

Además la información que se recopila sirve como antecedente cuando se tiene que reconfigurar el sistema. Tal fue el caso de la CRAY cuando se compraron 4 discos más. Ya que gracias al monitoreo realizado cuando solo contaba con 4 discos, se pudo distribuir de la mejor manera la carga de trabajo entre los 8 discos actuales.

Lo mismo sucedió cuando se decidió ampliar la memoria a 64 Megawords en Febrero de 1994. Originalmente, la supercomputadora contaría con 2 procesadores y 32 Megawords de memoria. CRAY Research Inc., decidió que regalaría otros 2 procesadores, lo cual fue aceptado por la UNAM. Pero la configuración inicial de la CRAY no era adecuada para un sistema con 4 procesadores. Esto pudo constatarse por medio de los reportes mensuales que se realizaban. Se observaba que a pesar de que los procesadores no se estaban utilizando al máximo, los procesos de usuarios sufrían grandes tiempos de

espera. En la Figura 2-1 se muestra el porcentaje de uso de cada uno de los CPU´s de la CRAY en 1993, el cual en promedio fue menor al 40% durante todo el año.

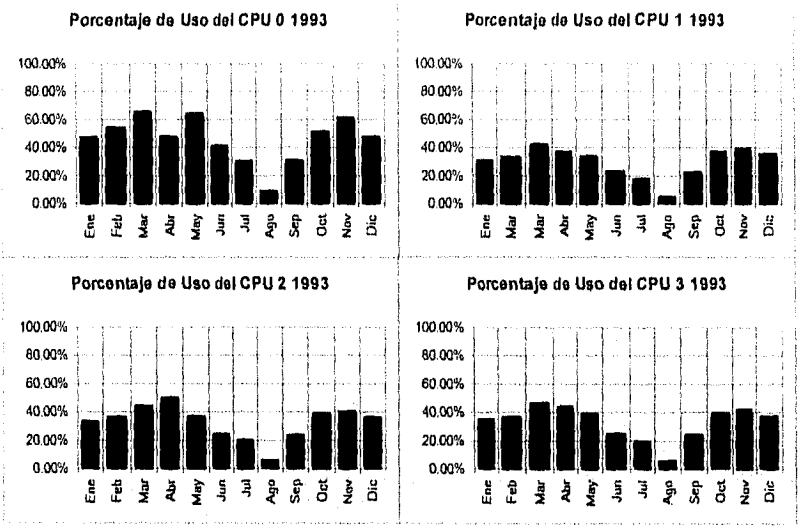


FIGURA 2-1. PORCENTAJE DE USO DE CPU EN 1993.

El problema era claro, no se tenía la memoria suficiente y los procesos tenían que esperar a que se liberara memoria para poder ser ejecutados, a pesar de que no se estuvieran usando los 4 procesadores. Debido a lo anterior, CRAY donó 32 Megawords de memoria adicional. Lo que dió como resultado un sistema con 64 Megawords. El incremento de memoria provocó un cambio inmediato en el porcentaje de uso de CPU. Esto puede observarse en la Figura 2-2, la cual muestra que los 4 CPU´s siempre están por arriba del 80% de uso.

Las gráficas mostradas en las Figuras 2-1 y 2-2 son el resultado de un sistema de monitoreo que mediante la utilería SAR genera estadísticas de uso de CPU, disco, memoria y dispositivo de estado sólido. Este sistema está instalado en una máquina UNIX, en la cual diariamente la supercomputadora almacena en forma remota los datos correspondientes al uso de cada recurso, y fué uno de los trabajos realizados que anteceden a este trabajo de tesis.

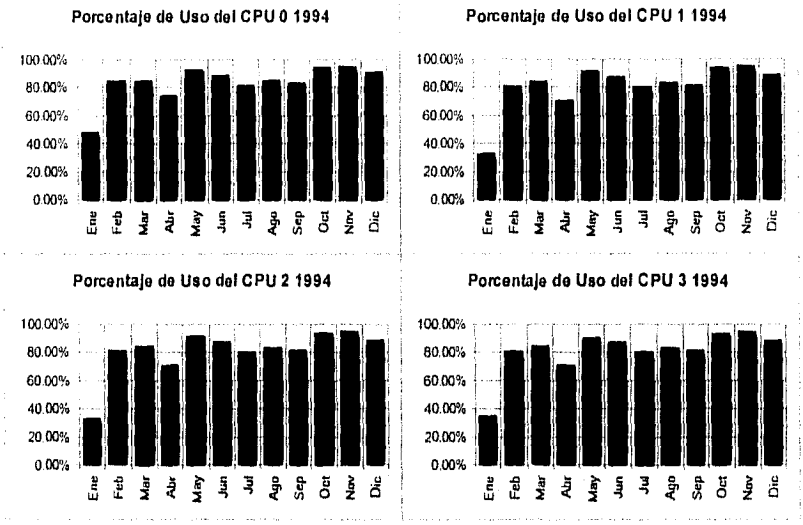


FIGURA 2-2. PORCENTAJE DE USO DE CPU EN 1994.

2.1. COMUNIDADES DE USUARIOS

Un factor que contribuye a la carga general del sistema es la comunidad de usuarios, que ejecutan sus trabajos en el sistema. Es de gran importancia para el rendimiento del sistema y para los administradores de sistemas UNIX, el mantener una buena comunicación con los usuarios, no solo para saber cómo cargan el sistema, sino para conocer sus opiniones sobre el comportamiento del mismo, lo cual es vital para el análisis de su rendimiento.

La administración es posible solo si se cuenta con la cooperación de los usuarios. Las limitaciones en requerimientos de memoria, tamaños de archivos, prioridades de trabajos, etc., son efectivos solo cuando cada uno de los usuarios coopera. Es más fácil obtener dicha cooperación cuando se logra que todos entiendan el problema y se den cuenta de que pueden ayudar a solucionarlo. Por ejemplo, ejecutando sus trabajos en lote. Es por ésto que la comunicación con los usuarios es considerada una muy buena herramienta.

En la CRAY se resuelven problemas de diversas áreas como física, matemáticas, química, medicina, ingeniería, ciencia de materiales, astronomía, biología, biotecnología, etc. Utilizando herramientas de alto nivel que resuelven ecuaciones, realizan cálculos por elemento finito, análisis molecular, análisis de esfuerzos, cálculo de efectos ambientales, planeación de cirugía, simulación del nacimiento de una estrella, entre otros. Todos ellos provenientes de una comunidad de usuarios que crece día con día. La CRAY tiene alrededor de 400

usuarios que pertenecen a diferentes grupos y atiende en promedio a 70 usuarios diariamente.

En ocasiones, cuando la población de usuarios es muy numerosa, se puede recurrir a la agrupación de usuarios de acuerdo a las aplicaciones que utilizan. Aunque el conocimiento no es personal, se tiene idea de cuáles grupos de usuarios o cuáles aplicaciones pueden causar problemas al sistema. Puesto que cuando se tienen problemas de rendimiento, no es suficiente saber qué componente es el que esta sobrecargado, debemos también conocer cuál es la causa de esta sobrecarga antes de tomar alguna decisión para optimizar.

Si existe este nexo entre el administrador y sus usuarios, será más fácil para el administrador encontrar la configuración que sea aceptable tanto para usuarios como para el administrador, es decir, llegar a un acuerdo de rendimiento. Es importante señalar que un acuerdo de rendimiento es una forma de estructurar la búsqueda de requerimientos para optimizar el rendimiento antes de que el sistema falle.

La naturaleza de los usuarios del sistema tiene un gran efecto en el rendimiento del mismo y también lo tiene en las tácticas a usar para mejorar el rendimiento. Una clasificación general de los tipos de usuarios podría ser la siguiente:

a) Usuarios que ejecutan un gran número de trabajos relativamente pequeños. Por ejemplo, usuarios que utilizan la mayor parte del tiempo editando o ejecutando utilerías de UNIX.

b) Usuarios que ejecutan un número pequeño de trabajos relativamente grandes, por ejemplo, usuarios que ejecutan grandes programas de simulación con enormes archivos de datos.

c) Usuarios que ejecutan un número pequeño de trabajos que usan intensivamente el CPU, que no requieren de muchas operaciones de E/S, pero sí de mucha memoria y tiempo de CPU. Los desarrolladores de programas caen dentro de esta categoría.

Los tres grupos antes mencionados pueden causar problemas. Varias docenas de usuarios ejecutando el comando *grep* y accesando sistemas de archivos remotos, pueden estorbar más que unos cuantos usuarios accesando archivos de gigabytes. Sin embargo, los tipos de problemas causados por estos tres tipos de usuarios no son iguales, por ejemplo, establecer colas para ejecución en lote puede ser útil para reducir la contención entre grandes trabajos, los cuales pueden ser corridos en horas de bajo uso, pero no ayudaría si el problema se da por usuarios tecleando en editores de texto o leyendo su correo.

La supercomputadora CRAY tiene diversidad de usuarios y por tanto de aplicaciones. La mayoría de los usuarios caen dentro de los 2 últimos grupos, ya que la máquina está especialmente afinada para ejecutar trabajos en lote.

Cuando se otorga una clave en la supercomputadora, se les hacen recomendaciones a los usuarios. Dos de ellas son: el evitar el uso de editores de texto y utilizar el Sistema de Colas NQS. Se les explica que es mejor editar sus programas en algún otro equipo, ya que para ello no se necesita de la potencia de una supercomputadora. Después transferir sus archivos a la CRAY y ejecutarlos en ella. En cuanto al sistema de colas, si ejecutan ahí sus trabajos, utilizarán mejor los recursos, colaborando con la distribución de la carga en las 24 horas del día, evitando así cuellos de botella en horas de alto uso.

Programas de usuarios

Mejorar el rendimiento de un sistema no significa que un solo programa corra más rápido, sino que todo el sistema trabaje mejor. Esto trae como consecuencia que todos los programas corran más rápido. Se debe estructurar el sistema para maximizar su tiempo de respuesta total. Logrando que sus recursos sean usados eficientemente y distribuidos claramente para todos los usuarios. Se debe descubrir qué parte del sistema esta sobrecargada, entender cómo reducir la carga, y saber a cuál parte se le tiene que hacer una actualización.

La clave para que un sistema trabaje con un alto rendimiento es que este bien configurado y se use bien. Cuando los usuarios están disgustados porque sus programas tardan mucho en correr, tenemos el primer síntoma de que algo anda mal en el sistema. Si se trata de un usuario en particular puede ser que su programa no este bien hecho o que no sepa usar los recursos[1]. O quizá sea él quien este encontrando justamente el punto débil del sistema.

En cualquiera de las situaciones, se le debe poner atención a las quejas de un usuario en particular. Si se trata de un programa mal hecho o no optimizado, el grupo de atención a usuarios podrá ayudarlo y mostrarle las utilerías de optimización existentes. Cuando se trata de un cuello de botella encontrado por ese programa en particular, se deberá analizar el problema y realizar las actividades necesarias para que no cause otros problemas futuros y el usuario quede satisfecho.

Existe una utilería de UNIX que reporta el tiempo requerido para ejecutar un programa, indicando el tiempo total de ejecución, el tiempo de sistema y el tiempo de usuario. Por ejemplo:

```
% /bin/time application
      4.8 real      0.5 user      0.7 sys
```

Este reporte muestra que el programa corrió en 4.8 segundos de tiempo total. Este es el tiempo real medido por un usuario sentado en su terminal con un cronómetro. El tiempo que el sistema empleó trabajando en el programa es más

[1] El mejoramiento del rendimiento de un programa en especial, no será tratado aquí, ya que ésto generalmente depende de la arquitectura del sistema en particular y de modificar los algoritmos entre otras cosas.

pequeño. Utilizó 0.5 segundos de tiempo en estado de usuario y cerca de 0.7 segundos en tiempo en estado de sistema. El tiempo total de CPU utilizado, fue de 1.2 segundos, solamente 1/4 del tiempo total.

El resto del tiempo fue utilizado llevando a cabo operaciones de E/S, que el comando *time* no reporta. El manejo de las operaciones de E/S requiere cierto cómputo, el cual se atribuye al tiempo del sistema. Pero el tiempo empleado en interfaces de red y controladores de terminales no se contabiliza. La mayoría del tiempo es utilizado en correr trabajos de otros usuarios.

Cuando un usuario ejecuta un programa en NQS, necesita estimar el tiempo de CPU del programa. Con el comando *time* se puede obtener este dato.

Muchos componentes diferentes contribuyen a el tiempo total de corrida de un programa. Es importante entender el rol que ellos juegan cuando se trata de resolver un problema de rendimiento. Dichos componentes se describen en la Tabla 2-1.

TABLA 2-1. COMPONENTES QUE INTERVIENEN EN LA EJECUCIÓN DE UN PROGRAMA.

TÉRMINO EN INGLÉS	TÉRMINO EN ESPAÑOL	DESCRIPCIÓN	CÓMO SE OBTIENE
User-state CPU time	Tiempo de CPU en estado de usuario	Se refiere al tiempo total de CPU utilizado para ejecutar el proceso en estado de usuario. Incluye el tiempo utilizado ejecutando funciones de biblioteca pero excluye el tiempo utilizado en ejecuciones de llamadas al sistema. El tiempo en estado de usuario, está bajo el control del programador. El programador debe conocer cuales rutinas de biblioteca son más eficientes y cuáles no y debe conocer cómo usar el *prof* y el *gprof* para encontrar en dónde emplean más tiempo su programas.	Es mostrado en la salida del reporte que genera el comando *time*.
I/O time	Tiempo de E/S	Se refiere al tiempo que utiliza el subsistema de E/S atendiendo las peticiones de E/S para cada trabajo. Bajo UNIX, el tiempo de E/S es difícil de medir. Sin embargo hay algunas herramientas para determinar si el sistema de E/S está sobrecargado y para realizar algunas consideraciones de configuración que puedan ayudar a disminuir los problemas de carga.	Por medio del comando *ja*. Por medio de los reportes que genera la contabilidad del sistema.
Network time	Tiempo de red	Se refiere al tiempo que utiliza el subsistema de E/S atendiendo peticiones de red. Esta es una subcategoría del tiempo de E/S y depende críticamente de la configuración y el uso.	Por medio del comando *ja*. Por medio de los archivos de contabilidad del sistema.

TABLA 2-1. COMPONENTES QUE INTERVIENEN EN LA EJECUCIÓN DE UN PROGRAMA. (CONTINUACIÓN)

TÉRMINO EN INGLÉS	TÉRMINO EN ESPAÑOL	DESCRIPCIÓN	CÓMO SE OBTIENE
System-state CPU time	Tiempo de CPU en estado de sistema	Se refiere al tiempo total de CPU en estado de sistema utilizado para correr el programa. Es decir, el tiempo que utiliza en la ejecución de código del ***kernel*** para la ejecución del programa. Incluye el tiempo empleado en ejecutar llamadas al sistema y en llevar a cabo funciones de administración relacionadas con el programa. La distinción entre el tiempo usado en una rutina de biblioteca y el usado en servicios del sistema, es importante y muy a menudo se presta a confusión. Por ejemplo, una llamada al *strcpy*, se ejecuta en tiempo de usuario totalmente. Una llamada al *printf* o el *fork*, es mucho más compleja, ya que requieren de servicios del ***kernel*** de UNIX, por lo que se utiliza tiempo de sistema. Todas las rutinas de E/S requieren servicios del ***kernel***. El tiempo en estado del sistema esta parcialmente bajo el control del programador. Sin embargo, los programadores no pueden cambiar el tiempo que se usará en cualquier llamada al sistema, solo pueden reescribir sus programas para usar más eficientemente las llamadas al sistema.	Es mostrado en la salida del reporte que genera el comando *time*.

TABLA 2-1. COMPONENTES QUE INTERVIENEN EN LA EJECUCIÓN DE UN PROGRAMA. (CONTINUACIÓN)

TÉRMINO EN INGLÉS	TÉRMINO EN ESPAÑOL	DESCRIPCIÓN	CÓMO SE OBTIENE
Time spent running other progrmas.	Tiempo empleado corriendo otros programas	Se refiere al tiempo utilizado para correr procesos ajenos al proceso en cuestión. Conforme crece la carga del sistema, el CPU emplea menos tiempo trabajando en un proceso dado. Esto incrementa el tiempo total (*elapsed time*) requerido para ejecutar el trabajo. Esto es molesto pero salvo algunos problemas con el E/S o memoria virtual, hay muy poco que pueda hacerse al respecto.	Por medio del comando *ja*. Por medio de los archivos de contabilidad del sistema. Realizando estimaciones con ayuda del comando *time*.
Virtual memory performance.	Rendimiento de memoria virtual	Este es el aspecto más complejo del rendimiento del sistema. Puesto que idealmente todos los trabajos deben permanecer en memoria física todo el tiempo. En la práctica esto es imposible, aún en un sistema que cuenta con mucha memoria. Cuando la memoria física esta totalmente ocupada, el sistema operativo mueve partes de los trabajos a disco, lo cual libera memoria para los trabajos que necesita comenzar a ejecutar. Este intercambio toma tiempo y más aún cuando los trabajos tienen grandes requerimientos de memoria.	Por medio del comando *ja*. Por medio de los archivos de contabilidad del sistema.

2.2. CONTROL DE PROCESOS

El objetivo de esta sección, es mostrar como el administrador del sistema puede monitorear los procesos para observar que es lo que esta haciendo el sistema. En base a ello, tener la posibilidad de terminar la ejecución de algún proceso que esté causando problemas serios. Desde luego, ésto debe hacerse con sumo cuidado y notificando al dueño del proceso lo que está sucediendo.

Al conjunto de programas que se encuentran corriendo en un sistema UNIX en un momento determinado, se les denomina procesos. UNIX da la impresión de

que muchas actividades se están realizando al mismo tiempo, pero solo un proceso es el que se ejecuta en un momento dado[2] en un procesador. La sensación de tener programas corriendo concurrentemente, se debe al método llamado *time-slicing*, mediante el cual UNIX intercambia los procesos que se están ejecutando, en intervalos regulares y cortos de tiempo. El proceso activo cambia tan rápido, que hace aparentar que todos los procesos se encuentran corriendo al mismo tiempo, cuando en realidad cada proceso se ejecuta solo en un pequeño porcentaje de tiempo.

Debido a que la CRAY cuenta con 4 procesadores, más de un proceso puede ser ejecutado al mismo tiempo. A esto se le conoce como procesamiento en paralelo. Cuando se paraleliza un programa, partes de él se ejecutan al mismo tiempo en los procesadores.

Definición de proceso

Un proceso, es un solo programa corriendo en un espacio de direcciones y que tiene asociado un conjunto de estructuras de datos dentro del ***kernel***. Un proceso es diferente de un trabajo o de un comando. Estos últimos pueden estar compuestos por un conjunto de procesos que juntos, realizan una tarea especifica.[3]

El espacio de direcciones de un proceso no es otra cosa más que una sección de memoria que el ***kernel*** ha reservado para el uso del proceso. La cual contiene el código del programa, las variables que utiliza el proceso, el stack del proceso y cualquier otra información necesaria para el ***kernel*** mientras el proceso esta corriendo.

Las estructuras de datos internas del ***kernel***, contienen información utilizada para manejar el proceso. A continuación se lista la información más importante contenida en dichas estructuras:

- El mapa del espacio de direcciones del proceso.
- El estado actual del proceso.
- La ***prioridad*** de ejecución del proceso.
- El informe de uso de recursos del proceso.
- El dueño del proceso.

Es importante señalar que estrictamente el espacio de direcciones no se ubica únicamente en una sección de memoria, ya que en sistemas con memoria virtual

[2] Estrictamente hablando, esto es cierto sólo para computadoras con un solo procesador.
[3] Es importante señalar, que la mayoría de los autores utilizan las palabras proceso y trabajo indistintamente.

puede darse el caso de que el espacio de direcciones de un proceso se encuentre en disco y no necesariamente en la memoria física.

Tipos de procesos

Los procesos en UNIX, pueden ser divididos principalmente dentro de las siguientes categorías:

Procesos Interactivos

Los procesos interactivos, son aquellos que son iniciados y controlados dentro de una sesión con el sistema. Estos procesos pueden ser ejecutados en primer plano (*foreground*) o en segundo plano (*background*). Un proceso ejecutado en primer plano, es aquel en el cual se ejecuta el proceso y necesariamente hay que esperar la salida del mismo para iniciar la ejecución de uno nuevo. Cuando se ejecuta un proceso en segundo plano, se permite utilizar la terminal inmediatamente después de que se ha lanzado el proceso sin tener que esperar la salida del mismo.

Procesos en lote

Los procesos en lote, son aquellos cuya ejecución no está asociada con una sesión de trabajo. Es decir, dichos procesos son sometidos a una cola que se encargará de ejecutarlos secuencialmente en el momento más adecuado según la carga del sistema y a las prioridades definidas por el sistema de colas de la máquina en particular. Sistemas como SunOs y ***System V***, poseen un comando llamado *batch* que realmente es bastante primitivo. Pero el sistema NQS que es el que se utiliza en la CRAY, es mucho más eficiente. Fue desarrollado por la NASA y es utilizado en sistemas que requieren de alto rendimiento.

Daemons

Los ***deamons*** son procesos normalmente inicializados al momento de levantar el sistema. Se ejecutan en segundo plano, esperando a que otro proceso requiera de sus servicios. Siguiendo con la filosofía de modularidad de los sistemas UNIX, los ***deamons*** son procesos y no parte del ***kernel***, como se pudiera suponer.

Atributos de un proceso

A continuación, en la Tabla 2-2 se explican los principales atributos de un proceso.

TABLA 2-2. ATRIBUTOS DE UN PROCESO.

ATRIBUTO	DESCRIPCION
PID	El PID (Process IDentification) es el número que el ***kernel*** utiliza para identificar al proceso. Los PID's, son asignados en el orden en que los procesos son creados. Si el ***kernel*** alcanza el número máximo de PID's, inicializa el contador no utilizando aquellos PID's que aún se encuentran en uso.
PPID	El PPID (Parent Process IDentification) es el PID del proceso padre. El proceso padre es aquel proceso que creó al nuevo proceso.
UID	El UID, contiene el número del usuario que creó el proceso. Dicho usuario, es considerado el dueño del proceso y es la única persona además del superusuario, que puede cambiar los parámetros del mismo. Todos los recursos que consuma el proceso son contabilizados a este usuario.
GID	El GID, es el número de identificación del grupo del proceso. Cuando un proceso es creado, el GID del proceso toma el GID del proceso padre.
Prioridad	La ***prioridad*** de un proceso determina el momento en el que se ejecutará un proceso. Cuando el ***kernel*** tiene recursos para correr otro proceso, elige aquel con la más alta ***prioridad***. Cabe mencionar que la ***prioridad*** de un proceso, no es lo único que determina cuándo se ejecutará el mismo, ya que el ***calendarizador*** del ***kernel***, toma en cuenta tanto la ***prioridad***, como la cantidad de tiempo de CPU que se le ha asignado recientemente y el tiempo que ha estado en espera mientras otros procesos han estado corriendo.
Control de la Terminal	En caso de ausencia de redireccionamientos, la mayoría de los procesos, tienen asociado el control de la terminal que determina la salida estándar, la entrada estándar y la salida de errores del proceso.

Monitoreo de procesos mediante el comando ps

El comando *ps*, es la herramienta principal del administrador del sistema para monitorear procesos, ya que proporciona información completa acerca de la actividad del sistema. El comando *ps*, puede ser utilizado para obtener información acerca de un proceso como puede ser el PID, el UID, la ***prioridad***, el control de la terminal, la cantidad de memoria que está utilizando, el tiempo de CPU que ha utilizado y hasta el estado del mismo.

A continuación se muestra un reporte típico generado con el comando *ps* en UNICOS el sistema operativo de la CRAY.

```
$ ps -ef
    UID    PID   PPID  C     STIME      TTY  TIME  CMD
   root      0      0  0  09:36:35        ?  0:00  sched
   root      1      0  0  09:36:35        ?  0:02  /etc/init
    ...
   gull   7997      1 10  09:49:32   ttyp3  0:04  csh
 martin  12923  11324  9  10:19:49   ttyp5 56:12  f77 -o foo foo.F
 chavez  16725  16652 15  17:02:43   ttyp6 10:04
g94<Hg.dat>Hg.log
     ng  17026  17012 14  17:23:12 console  0:19  vi benzene,txt
```

Como se ha mencionado, es ocasiones es necesario eliminar procesos. Estos pueden ser ***daemons*** que ya no se necesitan, procesos de usuarios que están causando problemas de rendimiento, programas que han bloqueado alguna terminal, etc. Mediante el comando *ps* se puede obtener el identificador del proceso (PID), que se utiliza cuando se realiza la eliminación de algún proceso.

Otra utilidad es poder conocer la prioridad con la que se ejecuta un proceso determinado o conocer su ***valor nice***.

2.3. EL CALENDARIZADOR

Controlando la ***prioridad*** en el ***calendarizador***, es posible forzar al CPU a emplear más de su tiempo de proceso en algunos trabajos específicos. Por lo que es posible utilizar esta opción para dar más ***prioridad*** sobre trabajos que son críticos, previniendo al sistema de utilizar tiempo en trabajos que no son importantes. Si el sistema soporta una gran comunidad de usuarios, algunos de los cuales corren trabajos muy grandes, entonces sería conveniente darles una menor ***prioridad*** para mantener satisfechos a la mayoría de los usuarios.

Sin embargo, si el sistema está dedicado a tareas particulares y soporta otro tipo de usuarios solo por cortesía, los trabajos dedicados deben tener mayor ***prioridad***, no importando su tamaño.

Por otra parte, si un trabajo está haciendo un uso intensivo de memoria, es mejor darle una ***prioridad*** muy alta, para que termine pronto y los demás trabajos

puedan continuar, en lugar de estarlo retardando por más tiempo. Desde luego, esta solución es solamente apropiada si los trabajos que causan los problemas de memoria son relativamente raros. Si todos los trabajos están en el mismo caso y se les da una ***prioridad*** alta a todos, se puede tener una falla en la operación del sistema debido a la imposibilidad de resolver las necesidades de recursos de todos ellos.

Prioridades del calendarizador

El comando *nice* sirve para modificar la ***prioridad*** de un proceso. Un proceso con un ***valor nice*** alto, corre a una ***prioridad*** muy baja, y viceversa.

Como una regla general, el reducir la ***prioridad*** de E/S de un trabajo, no cambiaría mucho las cosas. El sistema recompensa aumentando su ***prioridad,*** a aquellos trabajos que han gastado mucho de su tiempo esperando por una petición de E/S. Pero reducir la ***prioridad*** de uso de CPU, puede tener un efecto significante. Compilaciones, aplicaciones que realizan muchos cálculos matemáticos, o programas por el estilo, son buenos candidatos para aplicarles el comando *nice*.

Los niveles nice que existen en UNICOS, van desde el 0 hasta el 39, siendo el valor ***default*** el 20. El número 39 corresponde a la más baja ***prioridad*** y el 0 a la más alta.

Para someter un trabajo con ***prioridad*** baja, debe ser modificado con el comando *nice*. Si se desea especificar un ***valor nice*** determinado, se utiliza la siguiente línea:

```
% nice +/-n comando
```

Donde *n* es el nivel de ***prioridad*** deseado. Cualquier usuario puede modificar su ***valor nice*** para someter sus trabajos a una menor ***prioridad*** con un valor de *n* entre 0 y 19, ya que el valor de *n* es relativo al nivel de ***default*** (20), es decir un *nice* +6 corresponderá al nivel 26. Solo el superusuario puede modificar la ***prioridad*** de los trabajos para que éstos se ejecuten con mayor ***prioridad*** pudiendo utilizar valores de *n* negativos, con los cuales se llega a obtener el total del tiempo de CPU.

Si se utiliza este comando sin especificar ningún ***valor nice,*** se le asignará al proceso el nivel 24, ya que el valor *n* por ***default*** es el 4. Ejemplo:

```
% nice comando
```

2.4. SUMISIÓN DE TRABAJOS EN HORARIO DE BAJO USO

Un aspecto muy importante y que muchos de los usuarios olvidan, es el hecho de ejecutar los trabajos que no requieren interacción, durante la noche o los fines de semana. Haciendo uso de estas horas, se puede aumentar considerablemente el tiempo de respuesta del sistema. La dificultad para lograr ésto, radica en que el administrador debe concientizar a los usuarios de la importancia de hacer uso del sistema en las horas de bajo uso.

La manera de realizar lo anterior es por medio de alguna de las opciones que se listan a continuación.

El comando *at*

Permite al usuario correr trabajos a una hora arbitraria, lo cual se logra de la siguiente manera:

```
% at options time scriptfile
```

Si no se desea hacer un ***script***, los comandos pueden ser tecleados en la terminal, indicando la terminación con Ctrl-D:

```
% at options time
Comando 1
Comando 2
...
Ctrl-D
```

El tiempo (*time*) se especifica con cuatro dígitos, representando la hora en formato de 24 horas. Por ejemplo, 0130 representa la 1:30 am. y 1400 representa las 2:00 pm. También se pueden utilizar abreviaturas como 1am, 130pm, etc.

Bajo ***System V***, existe la opción *-l* para el *at*, con la cual se listan los trabajos que se encuentran en la cola de ejecución, y con *at -r*, se borran los trabajos incorrectos.

Un aspecto muy importante es que los usuarios analicen cómo está la cola de ejecución antes de someter algún trabajo. Lo anterior debido a que es posible que a todos los usuarios se les ocurra someter sus trabajos a la media noche, lo cual podría causar problemas. El administrador debe interactuar con sus usuarios para que puedan usar el sistema más eficientemente. Lo cual es una tarea muy difícil para el administrador y que tiene que estar realizando constantemente.

Colas en Lote

La ejecución en lote es una manera efectiva de realizar mucho trabajo, especialmente en ambientes de producción orientada. Una ***cola en lote*** es una de las mejores formas para asegurar que una computadora permanecerá activa durante horas.

Usando el comando *at*, se tiene mucha actividad en horas tipicas, a las 12:00 a.m. o a la 1:00 a.m., etc., mientras que en una ***cola en lote***, mantendrá al sistema ejecutando trabajos siempre que haya trabajo que hacer.

En ***System V.4*** y SunOS, existe el comando *batch*. El sistema tiene un ***cola en lote*** que ejecuta trabajos en el orden en el que fueron introducidos a la cola. Lo cual se hace con la siguiente línea:

```
% batch
Comando 1
Comando 2
Ctrl-D
```

También se puede usar un ***script***:

```
% batch script -name
```

Para eliminar trabajos de la cola de ejecución, se usa el comando *atq* y *atrm* para Sun OS y *at -l* y *at -r* para ***System V.4***.

Sistemas de Colas en lote

Algunos sistemas UNIX grandes, tienen un sistema de colas más avanzado, que prevé la congestión difiriendo los trabajos hasta que el sistema tiene tiempo para ejecutarlos.

El NQS es un sistema más sofisticado, se utiliza principalmente por fabricantes de supercomputadoras, particularmente Convex y CRAY. Desafortunadamente, se requieren muchas de las cosas que a la mayoría de los usuarios les molesta hacer. Cuenta diversas opciones y un complejo sistema de prioridades y permisos. NQS es muy útil cuando se desea poner a trabajar a una supercomputadora las 24 horas del día. Acerca del sistema de colas NQS para una supercomputadora CRAY, se hablará detalladamente en el siguiente capítulo.

2.5. LIMITES DE USO DE CPU

En ocasiones, a pesar de los esfuerzos del administrador por convencer a sus usuarios de que sometan sus trabajos en *lote*, hay algunos que ejecutan grandes trabajos en horas de alto uso, no tomando en cuenta las recomendaciones hechas por el administrador. La respuesta a lo anterior, es establecer límites de uso de CPU, para ello existe en C ***shell*** el comando *limit*, el cual establece el tiempo máximo de CPU de que un usuario puede disponer. A continuación se ilustra el uso de dicho comando:

```
% limit -h CPUtime time
```

Si no se utiliza la opción *-h*, se establece un límite suave, si se utiliza, entonces se establece un límite duro. La diferencia entre estos dos tipos de límites es que cuando se trata de un límite suave, es posible incrementar el límite antes de que el programa muera, lo cual no se permite cuando se trata de un límite duro. Un límite suave nunca debe ser más grande que un límite duro. El comando utilizado para eliminar los límites es el *unlimit*.

Se pueden utilizar las notaciones siguientes para establecer un límite:

hoursh	Límite en horas.
minutesm	Límite en minutos.
seconds	Límite en segundos, es el ***default***.
minutes:seconds	Límite en minutos y segundos.

2.6. HERRAMIENTAS PARA EL MONITOREO DEL SISTEMA

Existen muchas herramientas disponibles para diagnosticar el comportamiento del sistema. Algunas de ellas para el diseño y desarrollo de programas, las cuales dependen del modelo de máquinas sobre las cuales se trabaja. Otras herramientas también diseñadas para arquitecturas específicas, las cuales dependen de las interfaces de usuario disponibles para desplegar reportes entre otras cosas. Es importante tomar en cuenta el impacto que causarán sobre los programas de usuarios y los requerimientos de las herramientas que se usarán.

El sistema ofrece herramientas de propósito general que pueden ser utilizadas para obtener información acerca del estado de procesos de usuario, uso de disco, de CPU, de memoria, etc. Las más importantes son listadas a continuación.

cron

Ejecuta comandos específicos en intervalos regulares. Estrictamente hablando, el *cron* no tiene nada que ver con el rendimiento, sin embargo, se utiliza para

recabar información de uso de recursos y comportamiento del sistema de manera regular, la cual puede ser analizada posteriormente. También sirve para realizar tareas regularmente que ayuden al mejoramiento del rendimiento como la limpieza de sistemas de archivos.

uptime

Reporta el promedio de carga del sistema. Puede ser útil para la detección de anomalías, y para la planeación.

ps

Reporta información de los procesos que están corriendo en el sistema.

iostat

Proporciona información acerca del uso del disco, así como información acerca del uso de CPU.

sar

Proporciona información sobre el uso de recursos, por ejemplo el uso de CPU y el porcentaje de ***idle time***. El *sar* produce un número increíble de reportes útiles.

sa

Genera un conjunto de reportes que muestran qué comandos han sido ejecutados y qué recursos de CPU han utilizado.

prdaily

Genera un conjunto de reportes diarios de contabilidad.

perfmeter

Genera un reporte que puede describir diferentes estadísticas.

2.6.1. MONITOREO DEL PROMEDIO DE CARGA DEL SISTEMA

El promedio de carga del sistema es una forma conveniente de resumir la actividad del sistema. Es la primera estadística que se debe observar cuando el rendimiento se siente pobre. UNIX define la carga promedio del sistema como el número promedio de procesos en la cola de ejecución del ***kernel***, durante un intervalo. Aunque un proceso es un conjunto sencillo de instrucciones, la mayoría de los programas se ejecutan como un simple proceso, pero algunos se dividen en varios procesos. Un proceso se encuentra en la cola de ejecución si:

- No está en espera de un evento externo, por ejemplo cuando espera a que alguien introduzca un caracter por medio del teclado.
- No ha realizado alguna llamada al *wait*.
- No ha sido detenido con el uso de CTRL-Z.

Si bien, el promedio de carga nos ayuda a establecer la actividad del sistema, existen dos principales razones para que esta medida no de una imagen fiel de la carga del sistema y son:

- El promedio de carga cuenta como si se estuvieran ejecutando, a todos los trabajos que se encuentran esperando por una petición de E/S. Esto incluye los procesos que están esperando por operaciones de disco bajo Network File System (***NFS***). Si el servidor de ***NFS*** no responde, el proceso puede esperar durante horas.
- El promedio de carga no contabiliza por ***prioridad*** en el ***calendarizador***. No hace diferencia entre los trabajos que han sido establecidos con la menor ***prioridad*** (por medio del comando *nice*) y por lo tanto no consumen tanto CPU y los que están corriendo en la más alta ***prioridad***.

El comando *uptime* reporta el promedio de carga del sistema produciendo el siguiente reporte:

```
% uptime
mysystem up 6 days, 1:27, load average: 1.34, 2.18, 2.51
```

Este reporte muestra el nombre del sistema y cuánto tiempo ha estado ejecutando. También reporta los tres promedios de carga del último minuto, los últimos 5 minutos y los últimos 15 minutos, respectivamente. Es muy útil para notar si la carga está aumentado o disminuyendo.

El comando *ruptime* produce un reporte similar para cada ***host*** en la red local, si se tiene ***TCP/IP***. Además proporciona los sistemas remotos que están abajo e informa cuánto tiempo han estado así.

El comando *rup host* es otra variante del *uptime*, ya que provee un reporte *uptime* del ***host*** especificado.

Estos dos últimos comandos, requieren que el sistema remoto ejecute un ***daemon*** llamado *rwhod*, para poder ser usados, el cual puede ser un problema de rendimiento por sí mismo, ya que como se ha mencionado, los ***daemons*** cargan al sistema.

Como una regla, un promedio de carga mayor de 10 es extremadamente alto para un mainframe o algún otro sistema de computo de gran tamaño. Un promedio de carga entre 4 y 7 es ligeramente pesado y es pertinente pedir a los usuarios que ejecuten sus trabajos más grandes por la noche. Si el promedio de carga es menor que 3 no hay por que preocuparse.

Si estamos hablando de una estación de trabajo, los límites anteriores bajan considerablemente, ya que la mayoría de las estaciones de trabajo están diseñadas para un solo usuario y no toleran promedios de carga de trabajo mayores a 1 o 2.

Si se está usando un ***System V***, pero no se tienen las herramientas ***TCP/IP*** antes mencionadas o se ha decidido no usarlas, se puede utilizar el *sar* con la opción *-q*, el cual es equivalente al comando *uptime*:

```
% sar -q
ora ora 3.2 2 i386 05/21/90
  00:00:01     runq-sz     %runocc     swpq-sz     %swpocc
  01:00:03       1.2          45          0           0
  02:00:02       1.5          70          .4          15
  03:00:01       2.1          95          1.1         53
```

Los campos del reporte anterior se describen en la Tabla 2-3:

TABLA 2-3. DESCRIPCIÓN DE CAMPOS DEL COMANDO *sar*.

CAMPO	DESCRIPCIÓN
runq-sz	La longitud promedio de la cola de ejecución durante el intervalo en cuestión. La cola de ejecución lista los trabajos que están en memoria y son ejecutables, no incluye aquellos que esperan por una petición de E/S o que están dormidos.
%runocc	El porcentaje de tiempo que la cola de ejecución esta ocupada.
swpq-sz	La longitud promedio de la cola de swap durante el intervalo en cuestión. Esta cola lista los trabajos que están listos para correr pero que no pueden porque han sido enviados al ***área de swap***.
%swpocc	El porcentaje de tiempo que la cola de swap esta ocupada.

2.6.2. MONITOREO DE PROCESOS

Si el sistema tiene un promedio alto de carga de trabajo, se deben analizar los trabajos que se están ejecutando; para esto se tiene el comando *ps* cuyos reportes ayudan a diferenciar entre problemas de memoria virtual, problemas de E/S y otros problemas. En estos reportes se podrá observar cuáles trabajos y cuáles usuarios son responsables de la carga del sistema y determinar si es necesario o no reconfigurar el mismo.

El comando *ps* tiene muchas opciones, muchas de las cuales corresponden a diferentes formatos de salida, por lo que no se mencionarán todas las opciones de manera exhaustiva, pero sí las más útiles y cómo analizarlas.

El comando ps

Este comando se utiliza mucho, para realizar el monitoreo del sistema. En ocasiones los usuarios dejan procesos que no necesitan y se olvidan de ellos. Por lo que el administrador debe estar al pendiente de cuáles procesos se están ejecutando y quién es el dueño de dichos procesos. Por medio de este comando se obtiene información importante acerca del proceso. Además es posible obtener los criterios necesarios para decidir si algún proceso es factible de eliminar, o en su defecto hablar con el dueño y requerirle de más información.

Con el comando *ps* se puede saber el estado del proceso. Por ejemplo si es un proceso del sistema y siempre está en memoria. En este caso sería conveniente saber si el sistema en verdad necesita de dicho proceso. También es posible

saber si algún proceso está actualmente ejecutándose en el procesador, o si está en estado de espera para completar una operación de E/S de la terminal. En ocasiones hay procesos que terminan su ejecución pero que no han sido borrados de las tablas de procesos, a los cuales se les conoce como procesos ***zombie***.

Cuando se requiere eliminar algún proceso, es necesario ejecutar el comando *kill*. Este comando dará por terminada la ejecución del proceso cuyo identificador haya sido dado como parámetro. Para lo anterior, es necesario ejecutar previamente el comando *ps*. Este muestra el número de identificación del proceso, y del proceso padre del mismo.

También por medio del *ps* es posible saber qué ***valor nice*** tiene el proceso, el cual es usado para establecer prioridades, o directamente se puede obtener el nivel de prioridad del proceso.

A continuación se muestra un ejemplo:

```
% ps -el
 F   S  UID  PID  PPID  C PRI  NI   ADDR   SZ WCHAN  TTY   TIME    COMD
19   S    0    0     0  0   0  20 40256c    2 d0158  ?     0:01   sched
10   S    0    1     0  0  39  20 40256c   15 e0000  ?     6:30   init
19   S    0    2     0  0   0  20 40256c    0 d007a  ?     0:00   vhand
19   S    0    3     0  0  19  20 40256c    0 d06af  ?     4:31   getty
10   S    0  107     0  0  28  20 40256c   29 d1020  cons  0:01    csh
```

En el reporte generado por el *ps*, los procesos están ordenados por edad. Por ejemplo para el comando *sched*, por el número hexadecimal indicado en el 1er campo, podemos saber que se trata de un proceso del sistema que siempre está cargado en memoria. Por la S en el segundo campo, sabemos que está esperando para completar algún evento. Mientras que el signo de interrogación en el último campo, indica que no tiene ninguna terminal asociada.

Algunas soluciones rápidas

Se puede decir mucho acerca de un sistema, con solo observar los comandos que se están ejecutando. Es posible identificar los procesos que están consumiendo más tiempo de CPU y memoria. A continuación se listan algunas soluciones rápidas para cuando se han detectado problemas comunes:

- Ocasionalmente se puede observar que el sistema esta obstruido por muchos trabajos idénticos creados por el mismo usuario. Dependiendo de la situación, sería bueno terminar algunos de estos trabajos utilizando el comando *kill*.

- Si existe algún proceso que tenga acumulado mucho tiempo sin justificación, esto quizá indique que este proceso se encuentra en un ciclo infinito.

- Si algún proceso que no se considera muy importante está consumiendo un gran porcentaje de CPU, se necesitará reasignarle una ***prioridad***, si se está usando ***BSD*** se forzará al trabajo a ser ejecutado con una ***prioridad*** menor, esto es posible con el comando `renice 4 -p` *pid*, donde *pid* es el número identificador del proceso, y 4 es un valor típico de baja ***prioridad***. En ***System V*** no existe el comando *renice*, por lo que la única alternativa es matar al proceso con el comando *kill*.

- Si algún proceso está consumiendo un gran e inesperado porcentaje de memoria del sistema, se debe consultar al usuario y preguntar si algo ha cambiado desde la última vez que lo ejecutó y si es necesario, matar al proceso. No debe usarse el comando *renice* con programas que tienen problemas de memoria, bajar la ***prioridad*** a este tipo de programas causa que los problemas de memoria duren más.

- De manera general, solo se debe tener cuidado con aquellos trabajos que están bloqueados por alguna operación de E/S en disco, ya que los procesos que están dormidos o esperan a algún evento externo no afectan al rendimiento del sistema

2.6.3. MONITOREO DEL TIEMPO OCIOSO

Los datos proporcionados por el comando *ps*, son muy específicos, en ocasiones no se requiere información acerca de cada proceso, sino del sistema total. Existen algunas otras formas de obtener información acerca del uso de CPU de una manera global. El comando *iostat* o el *vmstat*, para UNIX ***BSD***, o *sar -u* para ***System V***, dan una imagen rápida de la utilización de CPU.

La herramienta *iostat* proporciona una medida del ***idle time*** del CPU. El ***idle time***, es el tiempo que gasta el procesador sin hacer nada, esperando que un evento externo ocurra. Un reporte típico del *iostat* sería el siguiente:

```
% iostat 5
      tty             dk0             dk1             CPU
tin   tout   bps   tps   msps   bps   tps   msps   us   ni   sy   id
  1     32    16     2    0.0    14     1    0.0   14    1   10   74
  0      0    17     2    0.0    11     1    0.0    1   47   13   40
  0      0    87     7    0.0    12     1    0.0    0   96    4    0
  0      0    36     3    0.0     0     0    0.0    0   97    3    0
  0      7    28     2    0.0     0     0    0.0    1   97    1    0
```

El primer argumento, 5, le indica al *iostat* que imprima estadísticas cada cinco segundos. Se debe ignorar la primera línea del reporte, la cual intenta reportar promedios estadísticos desde que el sistema fue levantado. Las columnas más importantes son las descritas a continuación en la Tabla 2-4:

TABLA 2-4. DESCRIPCIÓN DE CAMPOS DEL COMANDO *iostat.*

CAMPO	DESCRIPCIÓN
us	El porcentaje de tiempo que el sistema utilizó en el estado de usuario ejecutando procesos a una ***prioridad default*** o mayor.
ni	El porcentaje de tiempo que el sistema utilizó en el estado de usuario ejecutando procesos como una ***prioridad*** baja.
sy	El porcentaje de tiempo que el sistema utilizó en estado de sistema.
id	El porcentaje de tiempo que el sistema utilizó en ***idle time***.

Si combinamos la información obtenida de los comandos anteriores con la información acerca del promedio de carga del sistema, podremos darnos una idea de lo que esta pasando con nuestro sistema.

Idealmente se desea que el ***idle time*** sea bajo, ya que ésto revelaría que el sistema se está utilizando casi en toda su capacidad. Por lo anterior se puede esperar un ***idle time*** alto cuando el promedio de carga del sistema es bajo.

En ***System V***, el comando *sar -u* proporciona información general acerca de la actividad del sistema, a continuación se muestra un reporte típico:

```
% sar -u
ora ora 3.2 2. i386 05/21/90
  00:00:01      %usr      %sys      %wio      %idle
  01:00:03        25         6         2         67
  02:00:02        22         6         3         69
  03:00:01        20         2         2         72
```

Para cada período de tiempo, *sar* reporta el porcentaje de tiempo utilizado en cuatro estados que se describen en la Tabla 2-5.

TABLA 2-5. ESTADOS DEL SISTEMA QUE REPORTA EL COMANDO *sar*.

ESTADO	DESCRIPCIÓN
usr	El monto de tiempo utilizado ejecutando código en estado de usuario.
sys	El monto de tiempo utilizado ejecutando código en estado de sistema.
wio	El monto de tiempo utilizado esperando por un bloque de E/S.
idle	El monto de tiempo que el CPU estuvo ocioso.

2.6.4. LA CONTABILIDAD

Los sistemas UNIX, en general, proveen un ***proceso*** de contabilidad, el cual graba estadísticas acerca de cada ***proceso*** que se corre, incluyendo su UID. Además, también guarda registros de la imagen de los procesos y de los recursos del sistema, tales como memoria, tiempo de CPU, operaciones de E/S, etc.

El ***proceso*** de contabilidad es diseñado para registrar el uso de recursos del sistema. Los datos colectados pueden ser usados para cobrar a los usuarios el uso de los recursos, o bien para monitorear el rendimiento del sistema.

Los sistemas de contabilidad en ***BSD*** y ***System V*** son diferentes, pero se basan en similares datos de entrada. Sin embargo, a pesar de que la información ordenada que recogen es en esencia idéntica, los métodos de salida y los formatos son muy diferentes.

Como con todos los sistemas de contabilidad, el software de contabilidad de UNIX, produce una pequeña, pero detectable carga al sistema. En ***BSD***, la contabilidad es usualmente habilitada en los sistemas nuevos, pero puede ser deshabilitada si se desea. Para el ***System V***, la contabilidad es inicialmente deshabilitada y debe ser puesta a funcionar por el administrador del sistema.

Con las utilerías de contabilidad pueden determinarse aspectos importantes para cualquier administrador como la carga de trabajo, problemas de rendimiento, parámetros de afinación de aplicaciones y parámetros de seguridad.

En cualquier sistema operativo, las herramientas de contabilidad proporcionan una parte considerable para la optimización del rendimiento. El sistema de contabilidad proporciona una forma fácil de recabar información importante, ya

que cualquier persona interesada en el rendimiento necesita saber qué es lo que esta haciendo el sistema.

Debemos saber qué aplicaciones ejecutan los usuarios, el tiempo que se tardan en ejecutar sus aplicaciones, qué tiempo se tarda el sistema en ejecutar utilerías generales y qué usuarios utilizan más los recursos del sistema. Una vez que sabemos qué programas están consumiendo el tiempo y la memoria, entonces podemos iniciar una estrategia de afinación.

La contabilidad estándar de los sistemas UNIX varía de un sistema a otro, pero en general proporciona reportes similares sobre los recursos del sistema. A continuación se listan algunos de los tipos de reportes que genera la contabilidad estándar de los sistemas UNIX:

- Actividad de terminales: Reporta el uso para cada terminal.
- Reporte de uso: Reporta la utilización del sistema por usuario.
- Lista de comandos: Reporta el número de veces que un comando ha sido utilizado en un período determinado, así como el uso de CPU que ha tomado cada comando.
- Lista de comandos mensual: Igual que el reporte anterior excepto que éste se realiza por mes.
- Último acceso al sistema: Reporta la última vez que cada usuario entró al sistema.

Utilerías de contabilidad

El sistema de contabilidad en ***BSD***, es habilitado por el comando *accton*, el cual se especifica en el ***script*** de inicialización */etc/rc*. En dicho ***script*** deberá aparecer la siguiente línea:

```
/etc/accton /usr/adm/acct
```

El parámetro */usr/adm/acct* especificado en la línea anterior, es el archivo en el cual la contabilidad depositará toda la información recolectada.

Después de haber ejecutado el comando *accton* el sistema llevará a cabo la contabilidad si se cumplen las siguientes condiciones:

- El archivo */usr/adm/acct* debe existir.
- El sistema de archivos en el cual reside el subdirectorio */usr/adm* debe estar por debajo del 95% de su capacidad de uso.

Cabe mencionar que el archivo */usr/adm/acct* crece indefinidamente, pudiendo ocasionar que el sistema de archivos en que se encuentra, rebase el 95% de su capacidad deteniendo el proceso de contabilidad. Para evitar lo anterior, puede ejecutarse el siguiente comando:

```
# cp /dev/null /usr/adm/acct
```

El comando anterior trunca el archivo de contabilidad destruyendo todos los datos existentes.

El sistema de contabilidad en ***System V***, es también habilitado por el *accton*; sin embargo, el *accton* no se ejecuta directamente. Se ejecuta el ***script*** */usr/lib/acct/startup*. Este comando debe ser parte del ***script*** de inicialización */etc/init.d/acct:*

```
/bin/su -adm -c /usr/lib/acct/startup
```

Este comando asume que *adm* es un nombre de usuario válido en el sistema, por lo tanto debe existir esta cuenta.

Para deshabilitar la contabilidad, se debe ejecutar el ***script*** */usr/lib/acct/shutacct*, que se debe ejecutar también siempre que se realice un ***shutdown***.

Para que se lleve a cabo la contabilidad, un número de ***scripts*** en ***shell*** generan reportes de contabilidad, los cuales son usualmente ejecutados por medio de la utilería *cron*. Los comandos más importantes se muestran en la siguiente tabla:

TABLA 2-6. COMANDOS MAS IMPORTANTES DE LA CONTABILIDAD ESTÁNDAR.

ESTADO	DESCRIPCIÓN
runacct	Genera un conjunto de 5 reportes diarios. Los cuales son puestos en el directorio */usr/adm/acct/sum*, en un archivo llamado *rprtmmdd*, donde *mm* indica el mes y *dd* indica el día.
dodisk	Genera estadísticas de uso de disco, las cuales son sumarizadas por el *runacct*.
ckpacct	Previene al archivo de contabilidad principal */usr/adm/acct/pacct*de crecer excesivamente, suspende la contabilidad si hay menos de 500 bloques libres en el sistema de archivos.

TABLA 2-6. COMANDOS MAS IMPORTANTES DE LA CONTABILIDAD ESTÁNDAR. (CONTINUACIÓN)

ESTADO	DESCRIPCIÓN
monacct	Convierte todos los reportes diarios de contabilidad en un reporte mensual. El cual es puesto en el directorio */usr/adm/acct/fiscal*, en el archivo fiscrpt*nn*, donde *nn* indica el mes.
prdaily	Obtiene las estadísticas de contabilidad para los datos actuales y las envía a la salida estándar.

Para realizar la afinación, los reportes más importantes son los realizados por el *runacct* y el *prdaily*. Sin embargo, el *monacct* prevendrá que los reportes diarios crezcan excesivamente. Una forma de automatizar la contabilidad es añadiendo las siguientes líneas al archivo *crontab* del usuario *root*.

```
# genera los reportes de contabilidad diariamente (1 am)
0  1  *  *  *  /usr/lib/acct/runacct
# actualiza las estadisticas de uso de disco semanalmente (2 am
# Lunes)
0  2  *  *  1  /usr/lib/acct/dodisk
# Checa el tamaño del archivo de contabilidad cada hora (a la
# media)
030  *  *  *  *  /usr/lib/acct/ckpacct
# Genera los sumarios mensuales (2 am el primer dia del mes)
0  2  1  *  *  /usr/lib/acct/monacct
```

Reportes de contabilidad

Para obtener un conjunto de reportes de contabilidad, se puede observar directamente a los archivos que están en el directorio */usr/adm/acct/sum*, o usar el comando *prdaily*.

```
% usr/lib/acct/prdaily mmdd
```

donde *mmdd* indica el mes y día para el cual se desea el reporte. El reporte es enviado a la salida estándar y si se omite el día y el mes, *prdaily* genera los resultados de el día actual.

A diferencia de ***BSD***, la contabilidad en ***System V*** produce siempre cinco reportes, y no hay posibilidad de seleccionar solo uno de ellos. Dichos reportes serán descritos en la Tabla 2-7 que se muestra continuación:

TABLA 2-7. REPORTES DE CONTABILIDAD PARA *System V*.

REPORTE	DESCRIPCIÓN
Actividad de terminales	Reporta el uso de cada terminal
Reporte de uso	Reporta el uso del sistema por usuario. El reporte sumariza el monto total de tiempo de CPU, bloques de disco, etc.
Sumario de comandos	Reporta el número de veces que cada comando fue usado y proporciona estadísticas acerca del uso de CPU.
Sumario mensual de comandos.	Lo mismo que el reporte previo, pero sumariza desde que el comando *monacct* fue ejecutado por última vez.
Último acceso al sistema	Reporta la última vez que cada usuario entró al sistema.

2.6.5. RECOLECCIÓN DE DATOS MEDIANTE EL *SAR*

Para ***System V***, el monitoreo del rendimiento esta basado en la utilería *sar*, la cual lee e interpreta los datos acerca de la actividad del sistema. Los archivos de datos son generados por el comando *sadc*, el cual colecta periódicamente estadísticas y las guarda para análisis posteriores. No hay ninguna equivalencia al comando *sar* en ***BSD***.

El *sadc* normalmente se ejecuta una vez cada hora, además de hacerlo cada vez que el sistema se levanta. Los archivos de datos generados por el *sadc* son puestos en el directorio */usr/adm/sa*, y nombrados *sadd*, donde *dd* es la fecha del dia actual.

Para que el comando *sadc* se ejecute al momento de levantar el sistema, el archivo */etc/init.d/perf*, debe contener la siguiente línea:

```
su sys -c "/usr/lib/sa/sadc /usr/adm/sa/sadc /usr/adm/sa/sa `date +%d`"
```

Para ejecutar el *sadc* periódicamente, se usa el ***script*** *sa1*, el cual puede ser ejecutado mediante el *cron* periódicamente. Si se desea llevar a cabo la colección de estadísticas cada hora, debe existir la siguiente línea de comandos en el archivo *crontab* de *root*:

```
0  *  *  *  *  /usr/lib/sa/sa1
```

Donde el nombre de el archivo *crontab* puede ser */usr/spool/cron/crontabs/sys* o */usr/spool/cron/crontabs/root*, dependiendo de la configuración del sistema.

El comando *sa1* utiliza dos argumentos para su ejecución: el período de ejecución dado en segundos, para realizar el muestreo y el número de muestras a tomar. Por ejemplo, la siguiente línea en el archivo de *crontab*, le indica al proceso *cron* que ejecute *sa1* cada hora y a su vez los argumentos de *sa1* indican que deben tomarse 3 muestras, una cada 20 minutos (1200 segundos)

```
0  *  *  *  *  /usr/lib/sa/sa1 1200 3
```

Operación básica del sar

El comando *sar*, puede ser utilizado en diferentes formas, para producir diferentes reportes. A continuación se listan tres formas básicas:

```
$ sar -opciones intervalo número_de_muestras
$ sar -opciones -f archivo
$ timex -s programa
```

El primero de los comandos, reporta la actividad del sistema, según el *intervalo* y el *número_de_muestras*, y no requiere del comando *sadc*. Si se omite el argumento *número_de_muestras*, el comando se ejecutará indefinidamente.

El segundo comando, colecta los datos a través del comando *sadc*, interpreta los mismos e imprime un reporte. El argumento *archivo* le indica al *sar* que archivo utilizar; si se omite este argumento, *sar* tomará la información del archivo correspondiente al día actual, es decir del archivo */usr/adm/sa/sadd*, donde *dd* indica el día del mes.

El tercer comando, *timex*, es el comando *sar* disfrazado. Este comando ejecuta el *programa* que se le indica como argumento, y colecta los datos mientras se ejecuta dicho programa, para después imprimir todos los reportes generados por el *sar* cuando el programa termina. Este comando es muy útil para detectar aquellos programas que se consideran sospechosos de degradar el rendimiento del sistema.

En los dos primeros ejemplos, el argumento *opciones*, es una secuencia de letras que controlan los tipos de reportes que el comando *sar* producirá.

El comando *sar* es utilizado en el Departamento de Supercómputo para obtener el uso de recursos de la supercomputadora. Se tiene un conjunto de ***shells*** y programas en *awk* que se ejecutan automáticamente mediante el comando *cron*. Estos ***scripts*** utilizan los datos generados por el *sar*, los procesan y se obtienen 5 reportes para el análisis de los siguientes recursos:

- Uso de CPU
- Uso de Memoria
- Uso de Swap
- Uso de Disco
- Uso del SSD

La realización de este monitor de recursos, fue antecedente a la realización del monitor del Sistema de Colas NQS. En un inicio se pretendía realizar un monitor de recursos de la CRAY más sofisticado, que tuviera una interfaz gráfica y mostrara el comportamiento de los diferentes recursos del sistema. Debido a lo anterior se empezó realizando un grupo de ***scripts*** que procesaban la información proveniente de la utilería *sar*. Dichos ***scripts*** daban como resultado archivos de texto a partir de los cuales se realizaban gráficas en Excel (como las mostradas en las Figuras 2-1 y 2-2).

Se pudo observar que no era necesario realizar una interfaz gráfica para ese monitor, ya que una vez hechas las gráficas en Excel, se puede actualizar la gráfica cada mes con la nueva información. Lo cual es muy sencillo y soluciona el problema de la visualización de los resultados.

Este tipo de monitoreo se comenzó a llevar a cabo desde 1993. Como se tenía respaldada información del *sar* desde 1992, ésta se rescató y se generaron los reportes para dicho año. Con lo anterior solo se cubrió una parte de la información necesaria para el análisis del rendimiento, ya que el *sar* no reporta el uso de NQS. Por otro lado, se requiere poder observar en cualquier periodo de tiempo, el comportamiento del sistema de colas. El cual cuenta con un promedio de 16 tipos de colas, en las que intervienen factores tales como: el usuario, el complejo de memoria y el complejo de tiempo. El número de gráficas que se pueden necesitar durante el análisis del NQS es variable y depende del comportamiento del periodo, cola o complejo que se esté analizando. Por lo anterior, no es posible automatizar esta tarea de la misma forma que se utilizó para generar los reportes mensuales hechos con los datos proporcionados por el *sar*.

3. EL SISTEMA DE COLAS NQS

Funcionamiento de NQS
Conceptos y Terminología
Usuarios de NQS
Administración de NQS
Configuración de NQS

El sistema de colas en red de UNICOS (*NQS*), es un subsistema de procesamiento en lote de UNICOS que fue originalmente desarrollado por Sterling Software para la NASA/AMES. Permite ejecutar programas escritos en ***shell*** de UNICOS en lote. Significa que después de someter un trabajo a NQS, es posible continuar trabajando en la misma terminal sin tener que esperar a que el programa termine su ejecución. O bien puede terminarse la sesión, sin que esto interrumpa la ejecución del programa. Otro aspecto importante y que es el objetivo principal de NQS es que ejecuta las peticiones de usuarios optimizando la utilización de recursos. Un programa que se coloca en una cola de NQS también llamado [1]***petición en lote***, no se ejecuta inmediatamente después de haber sido enviado, se ejecuta en cuanto hay recursos disponibles para su ejecución de acuerdo a prioridades establecidas y al número de colas y procesos encolados.

Cuando un programa en NQS termina su ejecución, tanto la salida estándar como la salida estándar de errores que el programa produce, son almacenados en el espacio en disco designado al usuario que sometió el trabajo. El ***shell*** utilizado para ejecutar dichos programas, puede ser aquel definido por el administrador de NQS, el mismo al ***shell*** de entrada al sistema del usuario o el ***shell*** actual de la línea de comandos cuando se realiza la petición. Lo anterior depende de lo especificado al momento de configurar el sistema de colas. La diferencia principal entre un ***script*** de ***shell*** convencional y uno escrito para NQS, es la posibilidad de incluir opciones de NQS a manera de comentarios al inicio del programa.

[1] Por simplificación, a partir de este punto y hasta el final de este trabajo, utilizaremos solo la palabra petición o peticiones para hacer referencia a las peticiones en lote.

Características principales de NQS

- Es posible someter un ***script*** de comandos de UNICOS a NQS para su ejecución en ***lote*** local o remotamente.

- Se pueden definir límites en cuanto al uso de recursos del sistema al momento de someter las peticiones. Tales como: el tiempo máximo de CPU que se le debe asignar al programa, el tiempo mínimo que debe esperar para ser ejecutado o el espacio máximo de memoria que podrá utilizar la petición.

- Es posible observar el estado de las peticiones sometidas a NQS y tener cierto control sobre ellas.

- Es posible especificarle a NQS en donde se almacenará la salida producida por una petición sometida.

- Se puede suspender en cualquier momento la ejecución de una petición sometida. Y salvar una "fotografía" (*checkpoint*) del estado de la petición, para reanudar su ejecución a partir del punto en que se suspendió la ejecución. Lo anterior se realiza automáticamente si el sistema de colas (NQS) o el sistema completo son dados de baja y reinicializados nuevamente.

- Cuenta con una interfaz de administración y una interfaz de usuario.

3.1. FUNCIONAMIENTO DE NQS

En NQS existen diferentes colas en las cuales pueden haber procesos encolados. Estas colas son definidas por el administrador del sistema NQS. Bajo este esquema, los procesos de usuarios se forman y esperan "turno" para ser ejecutados. La cola en la que se formarán y el momento de ser ejecutados dependen de diversos atributos que serán explicados más adelante. Cuando un usuario somete un ***script*** de comandos de UNICOS a NQS, se crea una copia del archivo que es la que finalmente se mandará a ejecución. Lo cuál permite que el usuario modifique el archivo original sin afectar la ejecución de la petición ya realizada.

NQS cuenta con dos tipos de colas, *colas de transición (tipo pipe)* y *colas en lote*. Las colas de transición, son las encargadas de mandar las peticiones de una cola a otra. Es decir, mandan las peticiones a las colas destino definidas por el administrador de NQS, las cuales pueden ser colas en lote o incluso de transición y pueden estar definidas tanto en el sistema local como en uno remoto. Las colas en lote, como su nombre lo dice, solo aceptan y procesan ***peticiones en lote***. Es decir son aquellas en las cuales UNICOS ejecuta las peticiones dependiendo de los límites en el uso de los recursos del sistema.

Estos límites son establecidos por el administrador de NQS al momento de definir las colas y pueden ser: el máximo tiempo de CPU por petición y el tamaño máximo de memoria que puede utilizar una petición.

Normalmente, todas las peticiones son enviadas a una cola de transición, (aunque en algunos sistemas es posible someter la petición directamente a una cola en lote) que se encarga de mandar la petición a una de la colas en lote. Ahí permanecerá hasta que UNICOS tenga los suficientes recursos disponibles para ejecutar la petición. Este proceso se muestra en la Figura 3-1.

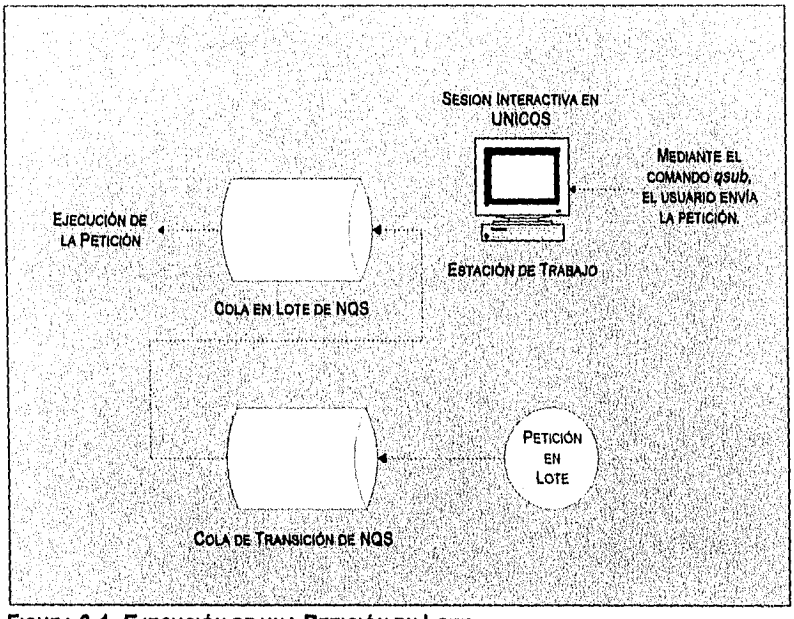


FIGURA 3-1. EJECUCIÓN DE UNA PETICIÓN EN LOTE.

Selección de la cola que NQS asigna a una petición

Cuando una petición es sometida a NQS, la cola a la cual se enviará la petición se selecciona de acuerdo a los siguientes criterios:

- Si al momento de someter la petición no se especifica el nombre de la cola en la que se desea se ejecute la petición, NQS selecciona aquella definida en el ambiente del usuario que lanzo la petición, o en su defecto utiliza la cola definida como ***default*** por el administrador de NQS.

- Si al momento de someter la petición se especificó el nombre de la cola en la que se desea se ejecute la petición, NQS manda la petición a dicha cola, siempre y cuando la cola que se especifique permita la recepción de peticiones en forma directa.

3.2. CONCEPTOS Y TERMINOLOGÍA

Las ***peticiones en lote*** para NQS y las colas definidas en el sistema, son la base principal de la operación de NQS. Por lo que es importante conocer cada uno de los atributos que componen tanto a las peticiones, como a las colas mismas del sistema y que se describen a continuación.

Atributos de las Peticiones

- Límites de recursos.

  Son un conjunto de límites impuestos a la petición o a ***procesos*** individuales de la petición para la utilización de un recurso. Algunos límites típicos son el tamaño máximo del archivo, el tiempo de CPU y el tamaño máximo de la memoria utilizada. El usuario puede establecer estos límites al momento de mandar la petición, o bien NQS establece estos límites implícitamente basandose en la cola que ejecutará la petición.

- Prioridad Intracola.

  La prioridad intracola, determina el orden a seguir para ejecutar cada una de las peticiones en una cola dada. Después de que una petición es aceptada en una cola en lote, el calendarizador de NQS calcula periódicamente por medio de un algoritmo, la prioridad intracola para cada trabajo encolado, cambiando el orden en el que los trabajos se seleccionarán, ya que la petición con la prioridad intracola más alta, es la elegida para iniciar su ejecución. Entendiendose por elegida, a aquella petición que mejor se ajuste a los diferentes límites de la cola y desde luego al momento en que el proceso fue enviado a ejecución .

- Valor Nice.

  El ***valor nice***, es la prioridad de ejecución de los ***procesos*** que componen una petición. Este pude ser establecido explícitamente al momento de enviar la petición, o implícitamente si no se especifica. Disminuyendo este valor, aumentaría la ***prioridad*** de ejecución del ***proceso***. Aunque no es recomendable hacerlo, debido a que ésto puede causar serios problemas al calendarizador de ***procesos*** del CPU.

- ***Shell*** de ejecución.

  Es el ***shell*** que se utilizará para ejecutar la petición. La elección del ***shell*** de ejecución para cada una de las peticiones, depende de si fue o no explícitamente especificado en la petición. Cuando no se especifica el ***shell*** de ejecución en la petición, se utiliza el establecido por NQS dependiendo de la forma en que se configuró, como se muestra en la Tabla 3-1.

TABLA 3-1. CONFIGURACIÓN DEL *Shell* DE EJECUCIÓN PARA LAS PETICIONES.

COMANDO	DESCRIPCIÓN
set shell_strategy fixed = path	Especifica el ***path*** completo del ***shell*** por ***default*** a utilizarse para ejecutar las peticiones.
set shell_strategy free	Especifica que el ***shell*** a utilizarse para ejecutar las peticiones, será definido en el ***script*** de la petición o en su defecto utilizara el ***shell*** de entrada al sistema.
set shell_strategy fixed login	Especifica que el ***shell*** a utilizarse para ejecutar las peticiones, será el ***shell*** de la línea de comandos en donde se envió a ejecución el programa.

Si el administrador de NQS, no establece ninguno de los anteriores al momento de configurar el sistema, el valor ***default*** sería: *set shell_strategy free*. Es importante mencionar que si se utilza el ***shell*** de entrada del usuario, no se podrán reinicializar los programas con estas características en caso de que se de de baja el sistema, ya que no podrá determinar cuál es el ***shell***.

Atributos de las Colas

- Restricciones de acceso.

  Para cada cola, el acceso puede ser restringido o no restringido. Esto siginifica que si una cola esta restringida, solo las peticiones de aquellos usuarios definidos en la lista de acceso para la cola en cuestión serán

aceptadas en dicha cola. Por el contrario si la cola no esta restringida, cualquier petición será aceptada.

- Prioridad Entrecolas

  Cada cola definida en el sistema, tiene una prioridad entrecolas asociada, que indica el orden en el cual las colas de cada tipo serán atendidas para procesar la siguiente petición. La cola con la prioridad entrecola más alta, será la que se atienda primero.

- Límite de grupo por cola

  Cada cola en lote, tiene asociado un límite de peticiones por grupo. El cual controla el número máximo de ***procesos*** que los usuarios pertenecientes a un grupo determinado pueden correr concurrentemente en la cola en cuestión.

- Límite de memoria por cola

  Cada cola en lote, tiene asociado un límite de uso de memoria. Este especifica el monto máximo de memoria que puede ser reservado por los ***procesos*** que se encuentran corriendo en la cola en un momento dado.

- Límite de ***procesos*** corriendo por cola

  Cada cola definida en el sistema, tiene asociado un límite de ***procesos*** corriendo por cola. Este establece el número máximo de ***procesos*** que pueden estar corriendo en la cola en un momento dado.

- Límite de usuario por cola

  Cada cola en lote, tiene asociado un límite de usuario. El cual determina el número máximo de ***procesos*** por usuario que pueden estar corriendo en la cola en un momento dado.

Complejos de Colas

Un complejo de colas, es un conjunto de colas en lote agrupadas por el administrador de NQS para facilitar la administración del mismo. Cada complejo, tiene un conjunto de atributos asociados, mediante los cuales es posible controlar por ejemplo, el número total de ***procesos*** corriendo concurrentemente, o bien la cantidad máxima de memoria que pueden apartar los ***procesos*** pertenecientes a todas las colas que forman el complejo. Así como las peticiones y las colas tienen atributos asociados, los complejos de colas también cuentan con atributos. Los cuales se deben considerar al momento de configurar el sistema de NQS. Dichos atributos, se describen a continuación:

- Límite de grupo por complejo de colas

  Cada complejo de colas, tiene asociado un límite de peticiones por grupo. Este controla el número máximo de ***procesos*** que pueden correr concurrentemente en las colas miembros del complejo, para todos los usuarios pertenecientes a un grupo determinado.

- Límite de memoria por complejo de colas

  Cada complejo de colas, tiene asociado un límite de uso de memoria, que especifica el monto máximo de memoria que puede ser apartado por todos los ***procesos*** que se encuentran corriendo concurrentemente en las colas miembros del complejo.

- Límite de ***procesos*** corriendo por complejo de colas

  Cada complejo de colas, tiene asociado un límite de ***procesos*** corriendo por complejo de colas, que establece el número máximo de ***procesos*** que pueden estar corriendo concurrentemente en las colas miembros del complejo.

- Límite de usuario por complejo de colas

  Cada complejo de colas, tiene asociado un límite de usuario por complejo de colas. El cual determina el número máximo de ***procesos*** por usuario que pueden estar corriendo concurrentemente en las colas miembros del complejo.

Límites Globales

Además de los límites por cola y por complejos de colas, también existen los límites globales, cuya función principal es la de restringir la carga de trabajo total controlada por NQS, generada por los ***procesos*** corriendo concurrentemente en el sistema de colas.

A continuación se describen cada uno de estos límites:

- Límite global de peticiones

  Este límite representa el número máximo de peticiones, que pueden correr concurrentemente bajo el control de NQS.

- Límite global de grupo

  Indica el número máximo de peticiones, que pueden correr concurrentemente, todos los usuarios pertenecientes a un mismo grupo.

- Límite global de memoria

  Es el monto total de memoria, que puede ser apartado, por todos los ***procesos*** que se encuentran corriendo concurrentemente bajo el control de NQS.

- Límite global de usuario

  Es el número máximo de ***procesos*** por usuario, que pueden correr concurrentemente.

Estados de las Colas

Para cada cola en lote definida en el sistema, existen diferentes estados de las colas. Los cuales sirven para darnos idea de lo que esta pasando con cada una de las colas en un determinado momento. Dichos estados se definen de acuerdo a las dos siguientes propiedades:

1) La capacidad de la cola para aceptar peticiones como se describe en la Tabla 3-2.

TABLA 3-2. ESTADOS DE LAS COLAS ACEPTANDO PETICIONES.

ESTADO	DESCRIPCIÓN
CLOSED	El ***deamon*** de NQS no esta corriendo en el sistema local.
DISABLED	La cola se encuentra deshabilitada y no esta aceptando peticiones.
ENABLED	La cola se encuentra habilitada y aceptando peticiones.

2) La capacidad de la cola de liberar peticiones listas para correr como se describe en la Tabla 3-3.

TABLA 3-3. ESTADOS DE LAS COLAS LIBERANDO PETICIONES.

ESTADO	DESCRIPCIÓN
INACTIVE	Las peticiones en espera, pueden empezar a ejecutarse, pero ninguna petición lo esta haciendo
RUNNING	Las peticiones en espera, pueden empezar a ejecutarse, y algunas peticiones lo están haciendo.
SHUTDOWN	NQS no esta corriendo en el sistema en donde está definida la cola.
STOPPED	Las peticiones en espera, no pueden empezar a ejecutarse, y ninguna petición lo este haciendo.
STOPPING	Peticiones nuevas no pueden empezar a ejecutarse, aunque existen algunos peticiones completando su ejecución.

3.3. USUARIOS DE NQS.

En la Tabla 3-4 se muestran los principales [2]comandos de usuario, para interactuar con NQS, así como una breve descripción de los mismos.

TABLA 3-4. PRINCIPALES COMANDOS DE USUARIO DE NQS.

COMANDO	DESCRIPCIÓN
qchkpnt	Realiza un checkpoint de una petición dada.
qdel	Borra una petición hecha a NQS.

[2] Consultar el manual UNICOS User Commands Reference Manual, para una descripción completa de estos comandos.

TABLA 3-4. PRINCIPALES COMANDOS DE USUARIO DE NQS. (CONTINUACIÓN)

COMANDO	DESCRIPCIÓN
qlimit	Muestra los límites y el ***shell*** de ejecución por ***default***, de NQS.
qstat	Muestra información, acerca del estado de las colas, peticiones y complejos de colas.
qsub	Manda un trabajo a ejecutarse en NQS.

Parecería fácil que los usuarios aprendieran a usar el sistema de colas NQS. Pero el aspecto más importante para que un usuario lo utilice óptimamente, es el saber definir qué recursos necesitará su programa. Por ejemplo, si un programa de usuario utiliza 3 MW de memoria y se ejecuta en un tiempo aproximado de 24 horas. El usuario deberá utilizar la cola de 24 hr 4 MW, que es la que más se adecúa a los recursos que requerirá su programa. El problema real que suele presentarse en [3]Sirio, es que el usuario por ignorancia o simplemente por no calcular la cantidad de memoria que utilizará su proceso, envía su programa a la cola de 24 hr 14 MW. Lo que ocasiona que hasta 24 horas se desperdicien 11 MW de memoria, como se ilustra en la Figura 3-2.

[3]Sirio es el nombre nombre del host de la Supercomputadora Cray de la UNAM

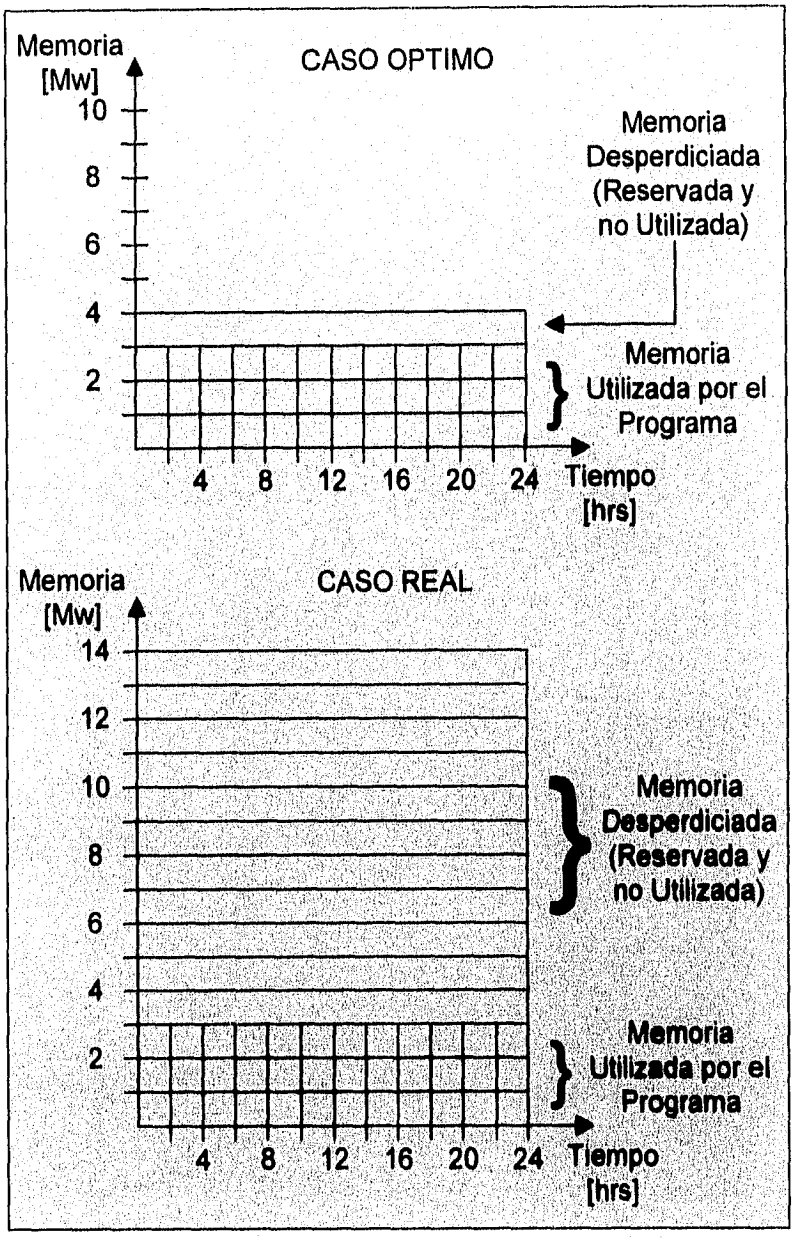


FIGURA 3-2. EJEMPLO DE MALA UTILIZACIÓN DE UNA COLA DE NQS.

Otro aspecto importante es que los usuarios estén consientes de que existen límites de procesos por usuario. En la CRAY de la UNAM no se estableció ni el límite de usuario por cola ni el límite de usuario por complejo. Lo anterior debido a que el límite global de usuario es 2. Esto quiere decir que solo 2 procesos de un usuario determinado pueden estarse ejecutando al mismo tiempo en cualquiera de las colas. El problema principal aquí, es que algunos usuarios envían más de 2 peticiones y bloquean las colas. Por ejemplo, supongamos que el usuario6 envía 5 procesos; 3 a la cola de 4 hr, 14 MW; y 2 a la cola de 4 hr, 4 MW, como se muestra en el caso real dentro de la Figura 3-3. Por el límite global, solo 2 de sus peticiones podrán ejecutarse al mismo tiempo. Y a pesar de que las colas no hayan llegado a su límite de procesos por cola, y haya más peticiones de otros usuarios esperando ser ejecutadas, éstas intentarán ejecutar los procesos del usuario que envió los 5 procesos y esperarán hasta que alguno de sus procesos termine. Esto provocará que el sistema se subutilice, ya que a pesar de que hay recursos para ello, no se están ejecutando los trabajos que podrían ser ejecutados. Además los dueños de los otros procesos, se quejarán de que sufren grandes tiempos de espera. Los cuales como puede observarse, son innecesarios. Si el usuario6 conociera esta limitante, y solo mandara el Proceso1 y el Proceso4, se utilizarían mucho mejor los recursos como se muestra en el caso óptimo ilustrado en la Figura 3-3.

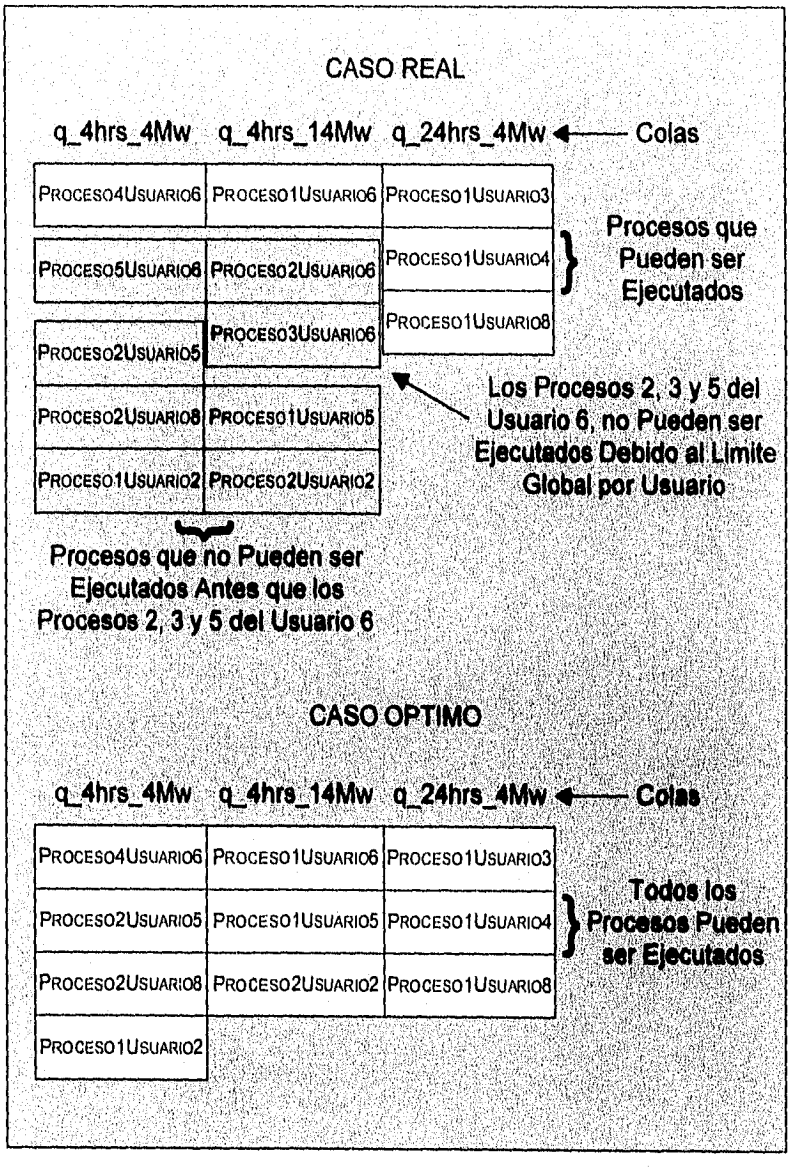


FIGURA 3-3. EJEMPLO DEL LIMITE GLOBAL DE PROCESOS POR USUARIO DE NQS.

3.4. ADMINISTRACION DE NQS.

La administración de NQS, juega una parte fundamental dentro de la administración total del sistema. Esto debido a que la gran mayoría de los recursos del sistema, son utilizados por las peticiones que los usuarios someten a NQS. Por esto el rendimiento del sistema depende en gran medida de una buena administración y configuración del sistema de colas.

El comando *qmgr*, es una interfaz mediante la cual es posible configurar y controlar al susbsistema NQS. Esta interfaz trabaja a través de un conjunto de comandos y cuenta con ayuda en línea para cada uno de ellos. Existen dos niveles de acceso a la interfaz *qmgr* que se muestran a continuación en la Tabla 3-5.

TABLA 3-5. NIVELES DE ACCESO AL COMANDO *qmgr*.

NIVEL	DESCRIPCIÓN
Administrador	Acceso a todos los comandos de *qmgr*, para monitorear, controlar y configurar NQS.
Operador	Acceso solo a un subconjunto de comandos de *qmgr*, para controlar y monitorear las colas y peticiones.

La siguiente Tabla 3-6 muestra los principales [4]comandos para la administración de NQS así como una breve descripción de los mismos.

TABLA 3-6. COMANDOS PRINCIPALES PARA LA ADMINISTRACIÓN DE NQS.

COMANDO	DESCRIPCIÓN
qmgr	Interfaz para configurar y administrar el sistema de colas de UNICOS.

[4] Consultar el manual UNICOS Administrator Commands Reference Manual, para una descripción completa de estos comandos.

TABLA 3-6. COMANDOS PRINCIPALES PARA LA ADMINISTRACIÓN DE NQS. (CONTINUACIÓN).

COMANDO	DESCRIPCIÓN
qstart	Inicializa las variables de ambiente de NQS, así como la ejecución del mismo.
qstop	Da de baja al sistema de colas NQS, de una forma ordenada.

Archivo de registro de NQS.

NQS cuenta con un archivo de registro, en el que periódicamente almacena, mensajes de actividad de NQS en formato de texto. El nombre y el lugar de este archivo, se especifican mediante el comando *set log_file* de la interfaz *qmgr*.

NQS permite controlar el nivel de detalle de la información que se almacenará en el archivo de registro. Con el fin de permitirle al administrador establecer un nivel de detalle más alto, en caso de que se le presente algún problema y necesite analizar con mayor detalle la actividad del sistema de colas. En cuyo caso es posible utilizar el comando *set debug* de la interfaz *qmgr*.

Contabilidad de NQS.

La información de contabilidad que produce NQS, es parte del ***deamon*** de contabilidad del sistema. Básicamente, esta información abarca la iniciación de ejecución y terminación de las peticiones, así como algunas condiciones como el reiniciar la ejecución de un programa ya sea desde el inicio o en el punto que se quedó. Los comandos *set accounting on* y *set acounting off*, de la interfaz *qmgr*, habilitan y deshabilitan respectivamente, la contabilidad de NQS.

En el siguiente capítulo se analizarán los datos que proporciona la contabilidad acerca del NQS. Ya que la parte fundamental de esta tesis es analizar el sistema NQS por medio de los datos generados por la contabilidad.

3.5. CONFIGURACION DE NQS.

Es importante señalar, que la correcta configuración de NQS permite utilizar los recursos del sistema de una manera más eficiente. También cabe mencionar que no es fácil determinar la mejor configuración para un sistema dado si no se

cuentan con las herramientas necesarias para analizar que es lo que esta haciendo el sistema.

Este punto es quizá el más importante de todo este trabajo. Ya que precisamente la configuración de NQS fue la causa por la que se realizó. La supercomputadora de la UNAM está especialmente afinada para ejecutar trabajos en lote. Por lo que su rendimiento general depende en gran parte de la configuración del sistema NQS.

Por ejemplo, en Sirio se tienen definidos 18 tipos de colas distribuidas en 6 complejos de tiempo y 5 complejos de memoria. Los complejos de tiempo definidos son: 1 minuto, 15 minutos, 4 horas, 24 horas, 48 horas y 96 horas. Los complejos de memoria definidos son: 4 Megawords, 14 Megawords, 20 Megawords, 28 Megawords y 56 Megawords. Las colas definidas resultan de la combinación de estos complejos. Por ejemplo existe una cola de 24 horas y 4 Megawords. Su nombre, el cual se estableció al momento de crear la cola es q_24h_4Mw. Todas las colas de Sirio tienen un nombre como el anterior. El cual se compone utilizando los nombres de complejos de tiempo y memoria. En la Figura 3-4, se muestra la parte de una salida típica del comando *qstat*, mediante la cual se pueden observar las colas definidas en el sistema.

```
----------------------------
NQS 1.1 BATCH QUEUE SUMMARY
----------------------------
QUEUE NAME              LIM  TOT ENA STS QUE RUN  WAI HLD ARR EXI
----------------------- ---  --- --- --- --- ---  --- --- --- ---
q_1min_4Mw               10    0 yes  on   0   0    0   0   0   0
q_1min_28Mw               2    0 yes  on   0   0    0   0   0   0
q_15min_4Mw              10    0 yes  on   0   0    0   0   0   0
q_15min_14Mw              3    0 yes  on   0   0    0   0   0   0
q_15min_20Mw              2    1 yes  on   0   1    0   0   0   0
q_15min_28Mw              1    0 yes  on   0   0    0   0   0   0
q_15min_56Mw              1    0 yes  on   0   0    0   0   0   0
q_4h_4MW                 10    0 yes  on   0   0    0   0   0   0
q_4h_14Mw                 3    9 yes  on   6   3    0   0   0   0
q_4h_20Mw                 2    0 yes  on   0   0    0   0   0   0
q_4h_28Mw                 1    0 yes  on   0   0    0   0   0   0
q_24h_4Mw                10    9 yes  on   6   3    0   0   0   0
q_24h_14Mw                2    3 yes  on   2   1    0   0   0   0
q_24h_20Mw                1    1 yes  on   1   0    0   0   0   0
q_24h_28Mw                1    1 yes  on   1   0    0   0   0   0
q_48h_28Mw                1    0 yes  on   0   0    0   0   0   0
q_48h_56Mw                1    7 yes  on   6   1    0   0   0   0
q_96h_28Mw                1    1 yes  on   1   0    0   0   0   0
----------------------- ---  --- --- --- --- ---  --- --- --- ---
sirio                    15   32              23   9    0   0   0   0
```

FIGURA 3-4. COLAS DEFINIDAS EN LA CRAY DE LA UNAM.

El siguiente complejo de memoria que existe después del de 4 MW, es el de 14 MW. Esto significa que cualquier proceso que requiera entre 4 y 14 MW de

memoria, se ejecutará en una cola que pertenezca al complejo de 14 MW. Si no existiera el complejo de 14 Mw, y el siguiente fuera el de 20 Mw. Todos los trabajos que utilizaran entre 4 y 20 Mw se ejecutarían en una cola que perteneciera al complejo de 20 Mw. Si un trabajo se ejecuta en 24 horas y requiriere de 10 MW, estaría desperdiciando mucha memoria. Ya que de los 20 Mw reservados, 10 estarían sin usarse hasta 24 horas. Si al administrador del sistema de colas notara que existe un gran número de procesos en esta situación, la decisión inmediata sería crear un nuevo complejo que midiera un poco más de los 10 Mw. Lo anterior para que los procesos intermedios se pudieran ejecutar ahí y se optimizara el uso de memoria. Como se puede observar es de suma importancia observar el comportamiento del sistema para conocer los requerimientos típicos de las peticiones de usuarios.

Otro aspecto muy importante es la prioridad entrecolas. Por ejemplo, si se diera una prioridad mayor a los trabajos que pertenecen a complejos de tiempo grandes, se acumularían gran cantidad de trabajos pequeños esperando ser atendidos. Esto sería muy peligroso, ya que se podría llegar al límite global de peticiones en lote, lo cual bloquearía el sistema NQS y habría usuarios que tendrían que esperar mucho tiempo para que sus trabajos de 1 minuto o 15 minutos fueran ejecutados.

Otro parámetro importante es el límite de procesos ejecutándose por cola. En la CRAY de la UNAM se tiene una cola de 1 min y 4 Mw, la cual tiene establecido un límite de procesos ejecutándose por cola de 10. Este límite se estableció después de haber analizado que 10 procesos utilizando 4 Mw durante 1 minuto, no significaban ningún problema para el sistema total. Pero para el caso de la cola de 15 minutos y 56 Mw, su límite de procesos es 1. Lo anterior debido a que la capacidad de la memoria es de 64 Mw. Pero no solo se debe observar la capacidad de la memoria. Por ejemplo si se diera un límite de 3 procesos a la cola q_24h_14Mw, no habría aparentemente tantos problemas albergando a los 4 procesos al mismo tiempo, pero no quedaría espacio suficiente para procesos de otras colas.

Es posible que algún usuario envíe un proceso a la cola q_48h_28Mw, y solo utilice 5 Mw. Podría pensarse que el usuario no sabe utilzar bien el NQS, ya que está desperdiciando 43 Mw de memoria durante 2 días. Pero debido a que no existe una cola con menos memoria que pertenezca al complejo de 48 horas, el usuario no tiene otra opción. Así que si este tipo de procesos fuera frecuente, la solución sería crear una cola de 48 horas, que ocupara menos memoria, quizá 14 Mw.

Uno de los parámetros más importantes es el límite global de memoria. Este límite debe ser lo suficientemente grande como para soportar la mayor cantidad de proceso. Y tiene que ser lo suficientemente pequeño como para no causar que el sistema falle por falta de memoria.

Como puede observarse, los ejemplos anteriores son solo algunos de los problemas que pueden presentarse. Una configuración buena se obtiene logrando la combinación óptima de todos los parámetros que se han descrito en este capítulo. Como es de suponerse, el esfuerzo requerido para configurar un sistema NQS es muy grande. Se podría pensar que esta configuración solo se realiza una vez. Y cuando ha quedado lo mejor posible, el administrador no tiene que preocuparse más. Pero en un sistema como el de la UNAM, en donde las aplicaciones y los usuarios cambian de un momento a otro, se debe analizar la configuración constantemente si se quiere mantener un buen rendimiento.

La configuración de NQS consta de las 2 siguientes partes:

- Definir los ***hosts*** de NQS en la red.
- Configurar NQS en el ***host*** local

La primera parte de la configuración, es una tarea que se realiza siempre y cuando NQS vaya a interactuar con un ***host*** remoto. Debido a que la configuración del sistema de colas de la supercomputadora CRAY que tiene la UNAM, es totalmente local, no se analizará la definición de ***hosts*** en la red.

Configuración de NQS en el *host* local.

La configuración NQS en el ***host*** local, se realiza mediante la interfaz *qmgr*. Ahí se establecen los valores de los parámetros de NQS. La mayoría de ellos, por ***default***, tienen un valor establecido. La configuración de NQS se realiza solo una vez, al momento en que se ejecuta por primera vez. Dicha configuración permanece en el sistema aunque este sea reinicializado.

Archivos de configuración

La Tabla 3-7, muestra los archivos de configuración de NQS, así como una breve descripción de los mismos.

TABLA 3-7. ARCHIVOS DE CONFIGURACIÓN DE NQS.

ARCHIVO	DESCRIPCIÓN
/usr/src/net/include/nqs.h	Contiene parámetros de configuración del sistema.

TABLA 3-7. ARCHIVOS DE CONFIGURACIÓN DE NQS. (CONTINUACIÓN)

ARCHIVO	DESCRIPCIÓN
/usr/src/net/include/nqsdeflim.h	Contiene límites de recursos definidos por NQS. Para la mayoría de los parámetros, un valor de 0 significa que el uso del recurso es ilimitado.
/usr/src/net/nqs.mh	Contiene parámetros para el *nmakefile*, que sirve para la instalación de NQS.

Procedimiento de Configuración de NQS

Este apartado pretende presentar de manera general, los pasos a seguir para configurar el sistema de colas NQS. Se considera que es la primera vez que se configura en el sistema y que el administrador se encuentra trabajando con la interfaz *qmgr*.

1.- Determinar quien será el operador y quien el administrador de NQS, mediante el comando *add manager*.

2.- Definir las colas en lote del sistema, los límites de recursos para las peticiones y la prioridad intracola asociada con cada cola definida. Opcionalmente se pueden especificar límites de colas. Todo lo anterior, mediante los comandos *create batch_queue*, y *set*.

3.- Especificar las colas de transición y sus respectivos destinos.

4.- Determinar el archivo de registro a utilizarse, mediante el comando *set log_file*.

5.- Activar o desactivar la generación de la contabilidad de NQS, mediante el comando *set accounting*.

6.- Establecer los límites globales deseados, mediante el comando *set global*.

7.- Determinar el nivel de detalle de la información a almacenarse en el archivo de registro de NQS, mediante el comando *set debug*.

8.- Especificar el ***shell*** de ejecución para las ***peticiones en lote***, mediante el comando *set shell_strategy*.

9.- Definir el directorio en el cuál NQS almacenará los archivos generados al realizar un ***checkpoint*** o al dar de baja NQS. Un ***checkpoint***, es una llamada al sistema que se encarga de salvar el avance de un ***proceso*** hasta el momento en que se ejecutó el ***chekpoint***. Lo cual se utiliza para tener la opción de volverlo a ejecutar a partir de donde se hizo el último ***checkpoint*** sin tener que ejecutarlo nuevamente desde el principio. Dichos archivos contienen la imagen de los ***procesos*** que estaban corriendo al momento de ocurrir cualquiera de las acciones anteriores. El comando mediante el cual se especifica este directorio es: *set checkpoint_directory*.

10.- Definir los complejos de colas, así como establecer sus límites asociados, mediante los comandos *create complex* y *set complex*.

11.- Establecer la cola ***default***, para ejecutar aquellas peticiones en las que el usuario no especifique en qué cola desea que se ejecute su petición. Lo anterior se lleva a cabo mediante el comando *set default batch_request*. Típicamente la cola ***default*** es una cola de transición, configurada con múltiples colas en lote como destino. Lo cual tiene la finalidad de que la cola de transición mande las peticiones a la cola que más se ajuste a los recursos demandados por las mismas.

12.- Definir las características del ***daemon*** *afdaemon*. Mediante la utilización del comando *set quickfile*.

13.- Determinar los parámetros mediante los cuales se establece la prioridad entrecola. Lo cual se realiza mediante los siguientes comandos: *set sched_factor share*, *set sched_factor cpu*, *set sched_factor memory*, y *set sched_factor time*.

Consideraciones a seguir al configurar NQS

A continuación se presenta una serie de puntos que se deben considerar, al momento de configurar NQS.

- Al realizar un ***script*** para NQS, es conveniente comentar cada operación que se realice. Ya que los comandos de NQS no producen ningún tipo de respuesta en la salida estándar.

- Evitar configurar colas de transición, cuyo destino sea otras colas de transición, directa o indirectamente.

- Cuando una petición es enviada de una cola de transición a una cola destino, el orden a seguir para elegir la cola destino, depende del orden que se siguió cuando se configuraron las colas en lote. Por lo tanto es importante configurar las colas destino de tal forma que cada petición que un usuario envíe y que no especifique en qué cola desea que se ejecute dicha

petición, sea encolada en aquella cola que mejor se ajuste a las necesidades de la petición.

- Al momento de definir el directorio donde se almacenarán los archivos cada vez que se realice un ***checkpoint***, se debe considerar, que este directorio se encuentre en un sistema de archivos con el espacio suficiente. Ya que ahí se almacenarán las imágenes de los ***procesos*** que se encuentran corriendo en ese momento.

4. EL SISTEMA DE CONTABILIDAD DE LA CRAY CSA

Conceptos y Terminología del CSA
Localización de Archivos y Directorios
Reportes del CSA
Operación Diaria del CSA
Configuración del CSA
Procesamiento de Datos del CSA
Contabilidad de Daemons

El sistema operativo UNICOS, soporta dos tipos de contabilidad:

- La contabilidad estándar de UNIX ***System V***.
- El sistema de contabilidad de CRAY Research, el CSA

El sistema de contabilidad de CRAY fue diseñado para adaptarse a los requerimientos únicos de contabilidad de los sistemas CRAY Research. Al igual que la contabilidad estándar de UNIX, CSA provee de métodos para colectar la utilización de recursos por ***proceso***, el registro de conexiones de sesión, el monitoreo del uso de disco y para cargar cuotas a cuentas específicas. El CSA también proporciona otras facilidades que no están disponibles en la contabilidad estándar, y se listan a continuación:

- Contabilidad por trabajo.
- Contabilidad por dispositivo.
- Contabilidad por ***daemons***, para el Sistema de Colas NQS, para el UNICOS Station Call Procesor (USCP), y para el subsistema de cintas.
- Contabilidad de disco por ID.
- Contabilidad para periodos arbitrarios.
- Un flexible sistema de unidades de cobro (SBU).
- Un solo archivo conteniendo todos los datos de contabilidad de un período.
- Interfaz de formateo front-end.
- Respaldo o almacenamiento de datos de contabilidad.

Los parámetros de configuración necesarios para el CSA, se localizan en el archivo llamado */etc/config/acct_config*.

4.1. CONCEPTOS Y TERMINOLOGIA DEL CSA

Contabilidad diaria

De manera diferente a la contabilidad estándar, la contabilidad del CSA puede correr tantas veces como sea necesario durante un día. Sin embargo, esta acción se identifica como contabilidad diaria o del día.

Contabilidad periódica

De manera similar a la contabilidad estándar, el CSA proporciona una contabilidad periódica. Sin embargo, le permite a los administradores del sistema especificar los periodos de tiempo para los cuales se correrá la contabilidad acumulada. Aunque es común que sea corrida una vez al mes, es posible determinar que se corra más de una vez cada mes.

Datos reciclables

Por ***default***, los datos de contabilidad de sesiones activas son reciclados hasta que la sesión termina. CSA reporta solamente datos para sesiones terminadas, a menos que el *csarun* sea ejecutado con la opción *-A*. *csarun* pone los datos reciclables dentro de archivos de datos en el directorio */usr/adm/acct/day*. Estos archivos de datos son nombrados poniéndoles el prefijo 0.

Sesión

CSA organiza los datos de contabilidad por sesiones y tiempos de boot y entonces pone los datos dentro de un archivo de registro de sesión llamado *SESSION record*. Para los datos que no provienen de trabajos de NQS, un *SESSION* consiste de todos los datos de contabilidad para un identificador de trabajo durante un período de boot.

Este archivo es el que utiliza el monitor de NQS. La información contenida en este archivo esta en formato especial y solo puede ser manipulada mediante comandos de UNICOS.

Un *SESSION* para trabajos de NQS consiste de datos de contabilidad para todos los ID's de trabajos asociados con el identificador de secuencia de trabajos NQS.

Los trabajos de NQS pueden abarcar múltiples períodos de boot. Si un trabajo es reiniciado, éste tiene asociado el mismo job ID durante todos los períodos de boot en los cuales corra. Si se re-ejecutan trabajos de NQS , éstos tienen múltiples identificadores.

Uptime/boot period

Es un periodo delineado por los tiempos de alta y baja del sistema, el cual se encuentra en el archivo */etc/csainfo*. El *csaboots* es un comando que escribe a este archivo durante el proceso de inicialización del sistema.

4.2. LOCALIZACION DE ARCHIVOS Y DIRECTORIOS

En esta sección se describirán los archivos y directorios utilizados en el CSA. La estructura del directorio */usr/adm/acct* que se muestra en la Figura 4-1, es creada para que sea fácil encontrar los archivos de datos y reportes del CSA.

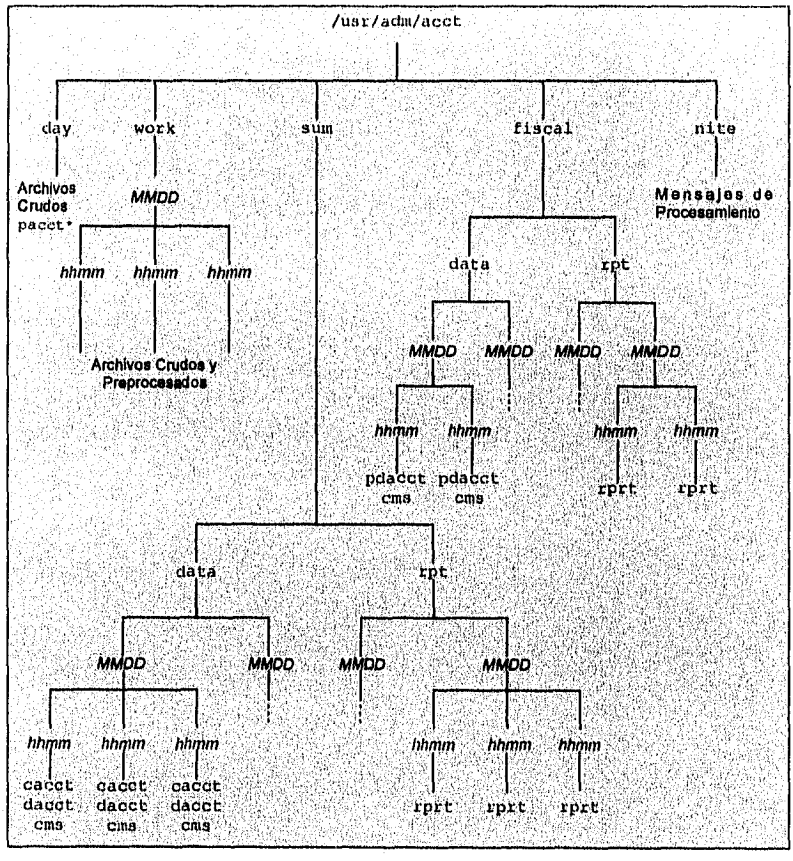


FIGURA 4-1. ESTRUCTURA DE DIRECTORIOS DE *usr/adm/acct*.

La estructura */tmp* es también usada mientras el *csarun* esta corriendo. La Figura 4-2 ilustra dicha estructura de directorios.

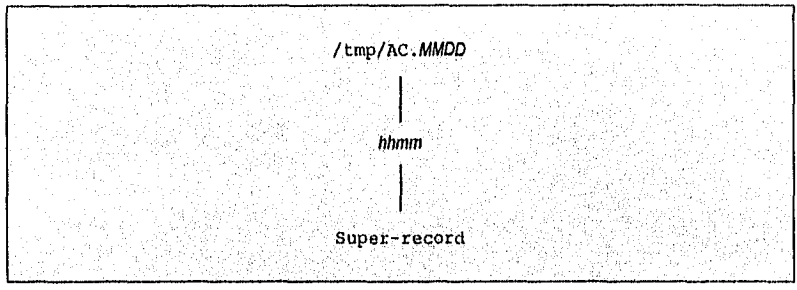


FIGURA 4-2. ESTRUCTURA DE DIRECTORIOS DE */tmp*.

Es importante hacer notar que solo los archivos *pacct** del directorio */usr/adm/acct/day* pueden ser leidos por todos los usuarios. Es permitido para todos los usuarios examinar los archivos *pacct** usando el comando *acctcom*, todos los demás directorios y archivos contenidos en el directorio */usr/adm/acct* son accesibles solo por root y los usuarios del grupo de contabilidad adm.

Las siguientes abreviaturas se usarán para representar nombres de archivos o directorios, y tienen los significados que se indican en la Tabla 4-1.

TABLA 4-1. ABREVIATURAS PARA ARCHIVOS Y DIRECTORIOS.

ABREVIATURA	DEFINICIÓN
MMDD	Mes, día
hhmm	Hora, minutos

Directorios de Contabilidad

Los directorios mostrados en la Tabla 4-2 se usan tanto para la contabilidad estándar como para el CSA.

TABLA 4-2. DIRECTORIOS DE LA CONTABILIDAD.

DIRECTORIO	DIRECTORIO
/usr/lib/acct	Contiene todos los procedimientos en ***shell*** y programas en C, necesarios para correr la contabilidad estándar o el CSA.
/usr/adm/acct/day	Contiene los registros de contabilidad por ***proceso*** y todos los archivos de datos sin procesar. Los archivos de contabilidad de ***daemons*** del CSA son almacenados en este directorio.
/usr/adm/acct/nite	Contiene los mensajes obtenidos durante el procesamiento.
/usr/adm/acct/sum	Contiene los archivos de sumario acumulativos, actualizados por el *runacct* o el *csarun*. En este directorio, se encuentran los archivos */pt/MMDD/hhmm/rprt* que contienen una variedad de reportes diarios del CSA.
/usr/adm/acct/fiscal	Contiene archivos periódicos de contabilidad creados por el *monacct* (***System V***) o por el *csaperiod* (CSA). En este directorio, los archivos */pt/MMDD/hhmm/rprt* contienen una variedad de reportes periódicos para el CSA.
/usr/adm/acct/work	Contiene los archivos temporales usados por los procedimientos de contabilidad diarios.

Archivos de Contabilidad

En la siguiente Tabla 4-3, se muestran los archivos de contabilidad especialmente importantes.

TABLA 4-3. PRINCIPALES ARCHIVOS DE LA CONTABILIDAD.

ARCHIVO	DESCRIPCIÓN
/etc/config/acct_config	Contiene el archivo de configuración de la contabilidad.

TABLA 4-3. PRINCIPALES ARCHIVOS DE LA CONTABILIDAD. (CONTINUACIÓN)

ARCHIVO	DESCRIPCIÓN
/etc/wtmp	Contiene los registros de entradas y salidas del sistema para los usuarios.
/etc/csainfo	Contiene el tiempo de boot escrito a este archivo por el */etc/csaboots*.
/usr/adm/acct/day/pacct	Contiene archivos de datos escritos por el ***kernel*** de UNICOS.

El archivo de configuración de la Contabilidad

Los parámetros configurables para la contabilidad son localizados en el archivo */etc/config/acct_config*, el cual puede ser accesado usando el menú del sistema, seleccionando las siguientes opciones:

Configure System ==>
Accounting Configuration ==>
Edit accounting configuration

Cuando se entra al menú del sistema, se trabaja con una copia del archivo de configuración (*/etc/install/cfdb/acctng_config*).

Los cambios que se desean hacer a la configuración de la contabilidad, se realizan estableciendo los valores para las variables del ***shell*** localizadas en el */etc/config/acct_config*, algunos cambios frecuentes a la configuración se listan a continuación:

- Habilitar el sistema de unidades de cobro, (SBU's) para los datos de contabilidad por ***proceso***. En este caso se pueden establecer algunas variables, por ejemplo, los factores de peso *P_BASIC, P_TIME y P_STIME*.
- Establecer cargos por los trabajos de NQS, con las variables *NQS_TERM_xxx*.
- Modificar el sistema de archivos en el cual reside el directorio */usr/adm/acct*, ésto se logran modificando la variable *ACCT_FS*.
- Trabajar con un archivo de contabilidad alterno, se modifica la variable *ACCTCONFIG*.

- Compilar una lista de usuarios a los cuales se les enviará un ***e-mail*** cuando un error es detectado, por ***default*** estos usuarios son root y adm, y la variable es *MAIL_LIST*

- Cambiar el número mínimo de bloques libres necesitados para correr el *csarun* o el *csaperiod*, el ***default*** es 500 bloques libres y la variable en cuestión es *MAIL_BLKS*.

Scripts en shell y binarios en C

El directorio */usr/lib/acct* contiene todos los programas y ***scripts*** usados tanto por la contabilidad estándar como por los paquetes del CSA. El único programa del CSA que no se localiza ahí es el */etc/csaboots*, el cual registra los tiempos de boot cuando el sistema es reinicializado. Todos los programas usados únicamente por el CSA, empiezan con la cadena *csa*.

Archivos de datos no procesados

Tanto la contabilidad estándar como el CSA, localizan los archivos de datos de contabilidad no procesados en el directorio */usr/adm/acct/day*. El uso de este directorio simplifica la búsqueda de los archivos actuales de contabilidad. En la Tabla 4-4, se muestra la localización de los archivos sin procesar de contabilidad.

TABLA 4-4. LOCALIZACIÓN DE LOS ARCHIVOS DE LA CONTABILIDAD ANTES DE SER PROCESADOS.

ARCHIVO	DESCRIPCIÓN
/usr/adm/acct/day/dtmp	Datos de contabilidad de disco.
*/usr/adm/acct/day/nqacct**	Datos de contabilidad del ***daemon*** NQS.
*/usr/adm/acct/day/pacct**	Datos de contabilidad por ***proceso***.
*/usr/adm/acct/day/tpacct**	Datos de contabilidad del ***daemon*** de cinta.
*/usr/adm/acct/day/usacct**	Datos de contabilidad del ***daemon*** [1]USCP.

1 UNICOS Station Call Processor.

TABLA 4-4. LOCALIZACIÓN DE LOS ARCHIVOS DE LA CONTABILIDAD ANTES DE SER PROCESADOS. (CONTINUACIÓN)

ARCHIVO	DESCRIPCIÓN
/etc/csainfo	Tiempos de [2]*boot*.
/etc/wtmp	Datos del tiempo de conexión.

En los casos en los que los archivos de contabilidad que están en el directorio */usr/adm/acct/day* tengan el sufijo 0, significa que éstos contienen datos de sesiones que no se completaron durante el periodo de contabilidad previo.

Archivos de datos siendo procesados

Diariamente, al inicializarse la contabilidad, el CSA mueve los archivos sin procesar del directorio */usr/adm/acct/day* al apropiado directorio */usr/adm/acct/work/MMDD/hhmm*. Estos archivos en el directorio *work*, contienen la infirmación que se describe en la siguiente Tabla 4-5.

TABLA 4-5. DESCRIPCIÓN DE LOS ARCHIVOS DE LA CONTABILIDAD SIENDO PROCESADOS.

ARCHIVO	DESCRIPCIÓN
Ever.tmp	Archivo work de verificación de datos.
*Pctime**	Datos preprocesados de tiempo de conexión.
*Pnqacct**	Datos preprocesados de NQS.
*Puptime**	Tiempos en los que el sistema esta trabajando.
Rctime0	Datos de conexión que serán reciclados en el siguiente periodo de contabilidad.

2 Periodo de arranque.

TABLA 4-5. DESCRIPCIÓN DE LOS ARCHIVOS DE LA CONTABILIDAD SIENDO PROCESADOS. (CONTINUACIÓN)

ARCHIVO	DESCRIPCIÓN
Rnqacct0	Datos de NQS que serán reciclados en el siguiente período de contabilidad.
Rpacct0	Datos por ***proceso*** que serán reciclados en el siguiente período de contabilidad.
Ruptime0	Tiempos en los que el sistema esta trabajando que serán reciclados en el siguiente período de contabilidad.
Rusacct0	Datos de USCP que serán reciclados en el siguiente período de contabilidad.
*Wctime**	Datos sin procesar de tiempo de conexión.
Wdisktacct	Datos de contabilidad de disco, en formato *cacct.h.*
Wdtmp	Datos de contabilidad de disco, obtenidos con el comando *diskusg* o *acctdusg.*
*Wnqacct**	Datos de contabilidad sin procesar de NQS.
*Wpacct**	Datos de contabilidad sin procesar por ***proceso.***
*Wtpacct**	Datos de contabilidad sin procesar de cinta.
*Wusacct**	Datos de contabilidad sin procesar de USCP.
Wwtmp	Datos de contabilidad sin procesar de tiempo de conexión.

Archivos de datos procesados

En la Tabla 4-6, se describen los archivos de la contabilidad ya procesados.

TABLA 4-6. DESCRIPCIÓN DE LOS ARCHIVOS DE LA CONTABILIDAD YA PROCESADOS.

ARCHIVO	DESCRIPCIÓN
/tmp/AC.MMDD/hhmm/Super-record	Archivo de registro de sesión. Por lo general este archivo es borrado, después de que se usa por el CSA.
/usr/adm/acct/fiscal/data/MMDD/hhmm/pdacct	Datos periódicos consolidados.
/usr/adm/acct/fiscal/data/MMDD/hhmm/cms	Datos perlódicos de uso de comandos.
/usr/adm/acct/sum/data/MMDD/hhmm/cacct	Datos diarios consolidados, este archivo es borrado por el *csaperiod*, si se utiliza con la opción *-r*.
/usr/adm/acct/sum/data/MMDD/hhmm/cms	Datos diarios del uso de comandos, este archivo es borrado por el *csaperiod*, si se utiliza con la opción *-r*.
/usr/adm/acct/sum/data/MMDD/hhmm/dacct	Datos diarios del uso de disco, este archivo es borrado por el *csaperiod*, si se utiliza con la opción *-r*.

4.3. REPORTES DEL CSA

CSA genera reportes diarios y periódicos, la siguiente Tabla 4-7, muestra en dónde se localizan estos reportes:

TABLA 4-7. LOCALIZACIÓN DE LOS REPORTES DIARIOS GENERADOS POR EL CSA.

ARCHIVO	DESCRIPCIÓN
/usr/adm/acct/fiscal/rpt/MMDD/hhmm/rprt	Reporte periódico de contabilidad.
/usr/adm/acct/sum/rpt/MMDD/hhmm/rprt	Reporte diario de contabilidad.

El reporte diario de contabilidad se compone de diferentes reportes, que se describen a continuación en la Tabla 4-8.

TABLA 4-8. REPORTES DIARIOS GENERADOS POR EL CSA.

TIPO DE REPORTE	DESCRIPCIÓN
Daily Report	Información del tiempo de conexión.
Daily Usage Report	Información del uso de recursos por usuario.
Unfinished Jobs	Información de trabajos no terminados. Este reporte no es parte de la versión de UNIX System V. Es único para UNICOS.
Daily Command Summary	Este reporte muestra los comandos más usados. Ayuda a identificar las áreas del sistema que utilizan la mayor cantidad de recursos.
Last Login	Proporciona la fecha en que una cuenta de usuario en particular fue usada por última vez.
Daemon Usage	Proporciona información de los ***daemons*** NQS, tape y USCP.

Este último reporte es obtenido por medio del comando **csadrep**. Este reporte es usado para la obtención de las estadísticas que se utilizan en el monitor de NQS.

A continuación se muestra la parte de este reporte, donde se encuantra alguna de la información utilizada por el monitor.

NQS Queue Report

Queue Name	Number of Jobs	CPU Time (sec)	(%)	Used USCF	Used Tapes	Ave Queue Wait (sec)	
q_24h_14Mw	5	20039	(84)	0	0	0.20	
q_24h_4Mw	1	1	(0)	0	0	0.00	
q_15min_4Mw	5	1012	(1)	0	0		0.40
q_4h_4Mw	3	12387	(9)	0	0	0.33	
q_4h_14Mw	14	4496	(3)	0	0	0.21	
q_15min_14Mw	5	1035	(1)	0	0	0.20	
q_4h_20Mw	2	1824	(1)	0	0		0.50
q_1min_4Mw	1	29	(0)	0	0		1.00
q_24h_28Mw	1	2206	(2)	0	0	1.00	
Totals	37	143029	(100)	0	0	0.30	

Queue:	q_24h_14Mw
Average CPU Time:	24007.900
Largest CPU Time:	75633.981
Smallest CPU Time:	3.587
Average Job Memory Hiwater (K-words) :	6958
Largest Job Memory Hiwater (K-words) :	9047
Smallest Job Memory Hiwater (K-words):	407
Jobs submitted from sirio.cray.unam :	5

Queue: q_24h_14Mw

User Name	Request Name	CPU Time (sec)	Memory HiWat (KW)	Machine Name	Sequence Number
flw	el	3.587	407	sirio.cray.unam	5916
rso	tauq.1	9346.486	9047	sirio.cray.unam	5902
rso	tauG.4	75633.981	9047	sirio.cray.unam	5811
tss	el	10967.542	8145	sirio.cray.unam	5883
flw	el	24087.903	8145	sirio.cray.unam	5881

User ID	Number of Jobs	CPU Time (sec)
flw	2	24091.490
rso	2	84980.467
tss	1	10967.542
Total	5	120039.499

Como puede observase en la primera parte se muestra la información general de las colas que fueron utilizadas en el período correspondiente. Después tenemos la información detallada de la cola q_24h_14Mw. La cual está dividida en tres bloques. De la misma forma que para esta cola, se muestra la información para cada una de las colas que se utilizaron en el período.

4.4. OPERACION DIARIA DEL CSA

Cuando UNICOS esta corriendo en ***modo multiusuario***, la contabilidad por lo general se comporta de la manera que se describe a continuación. Sin embargo, en cada máquina es posible adaptar la contabilidad de acuerdo a ciertos requerimientos, por lo cual es posible que algunos puntos no coincidan con un sistema en particular.

1. El tiempo de arranque es escrito en el archivo */etc/csainfo*. Esto se realiza cada vez que el sistema se inicializa, el comando que lo efectúa es el */etc/csaboots*, el cual es invocado a su vez por el comando *rc* durante la inicialización del sistema.

2. El ***proceso*** de contabilidad es habilitado. Cuando el sistema cambia a ***modo multiusuario***, el ***script*** */usr/lib/acct/startup* es llamado por el */etc/rc* y lleva a cabo las siguientes funciones:

   - Escribe un registro *acctg on* al */etc/wtmp*, el programa que lo escribe es el *acctwtmp*.

   - Habilita el ***proceso*** de contabilidad con la siguiente línea de comando:

     /usr/lib/acct/turnacct on

     El comando *turnacct* llama al programa *accton* con el argumento */usr/lib/acct/day/pacct*.

   - Remueve los *lock* files y salva los archivos *pacct* y *wtmp*.

   El comando */usr/lib/acct/remove* es invocado para limpiar y salvar los archivos *pacct* y *wtmp* en el directorio */usr/adm/acct/sum*.

3. Por ***default***, la contabilidad para los ***daemons*** de NQS, cinta y USCP es habilitada en el subsistema de CSA por el ***script*** */usr/lib/acct/startup*.

4. La cantidad de espacio de disco usado por cada usuario es determinada periódicamente. El comando */usr/lib/acct/dodisk* es ejecutado periódicamente por el comando *cron* para generar una imagen del espacio de disco usado por cada usuario. El *dodisk* debe ser ejecutado cuando mucho una vez por cada vez que el */usr/lib/acct/csarun* es ejecutado, ya que ejecuciones múltiples del *dodisk* en un mismo periodo de contabilidad, sobreescriben la salida previa del *dodisk*.

5. Se crea un archivo de cuota. Cuando se decide cargar cobros a algunos usuarios, se debe ejecutar el */usr/lib/acct/chargefee*. En cada periodo de contabilidad, el archivo de cuotas es puesto dentro de los registros consolidados por el */usr/lib/acct/csaperiod*.

6. La contabilidad diaria se ejecuta. En un momento específico durante el día, el *csarun* es ejecutado por el *cron* para procesar las datos de contabilidad actuales. La salida del *csarun* es un archivo diario de contabilidad consolidado y un reporte en ASCII.

7. La contabilidad periódica se ejecuta. En un momento específico durante el día, o en ciertos días del mes, el comando */usr/lib/acct/csaperiod* es ejecutado por el *cron* para procesar los datos de contabilidad consolidados de los periodos previos. La salida de *csaperiod* es un archivo periódico de contabilidad consolidado y un archivo ASCII.

8. La contabilidad es deshabilitada. Cuando el sistema es dado de baja exitosamente, el ***script*** */usr/lib/acct/shutacct* es ejecutado por el */etc/shutdown*. *shutacct* escribe un registro "*acct off*" al */etc/wtmp*. Este llama al */usr/lib/acct/turnacct* y al */usr/lib/acct/turndacc* para deshabilitar la contabilidad por ***proceso*** y la contabilidad de ***daemons***.

4.5. CONFIGURACION DEL CSA

Como se ha mencionado, el CSA puede ser configurado para los requerimientos específicos de un sistema, a continuación se describirá brevemente cómo se configura el CSA. El CSA puede ser ejecutado por una persona con permisos de super-usuario o por usuarios que tengan permisos *acct* y pertenezcan al grupo adm.

1. Cambiar los factores de peso ***default*** del Sistema de Unidades de Cobranza (SBU), si es necesario. Por ***default***, no se calculan las unidades de cobranza. Esto se modifica en el archivo de configuración */etc/config/acct_config*.

2. Modificar los parámetros que sean necesarios en el archivo */etc/config/acct_config* el cual contiene todos los parámetros de configuración del sistema de contabilidad.

3. Si el *csaboots* no es invocado por el *rc*, añadir las siguientes líneas a el ***script*** *rc* en la sección para el nivel de ejecución de multiusuario para registrar el tiempo de arranque. El *csaboots* debe ser ejecutado antes del */usr/lib/acct/startup*.

```
if [ -x /etc/csaboots ]
then
   /etc/csaboots -v >> $RC_LOG
fi
```

4. Añadir la siguiente línea al */etc/inittab* para registrar los tiempos de arranque del sistema.

```
cs::sysinit:/etc/csaboots -v > /dev/console 2>&1
```

5. En caso de no existir, añadir la siguiente línea al */etc/shutdown* para apagar el ***proceso*** de contabilidad por ***proceso*** y de ***daemons*** antes de que el sistema sea dado de baja:

   /usr/lib/acct/shutacct

6. Si se requiere la contabilidad de ***daemons***, se debe habilitar al momento de inicializar el sistema, realizando los siguientes pasos:

   - Asegurarse de que las variables en el */etc/config/acct_config* para los subsistemas para los cuales se desea habilitar la contabilidad de ***daemons*** están puestas en *on*. Las variables son *NQS_START*, *TAPE_STAR* y *USCP_START*.

   - La contabilidad para NQS y Tape debe también ser habilitada por el ***daemon*** asociado. Usar el comando *qmgr set accounting on* para encender la contabilidad de NQS y el *tpdaemon -c* para cinta. USCP no necesita ser habilitado por el ***daemon*** USCP.

7. Para la mayoría de las instalaciones, se deben poner entradas como la que se muestra a continuación en el */usr/spool/cron/crontabs/root*, para que el *cron* ejecute automáticamente la contabilidad:

```
0 4 * * 1-6 /usr/lib/acct/csarun 2> /usr/adm/acct/nite/fd2log
0 3 * * 1-6 /usr/lib/acct/dodisk -a -v 2> /usr/adm/acct/nite/dk2log
```

8. Periódicamente checar el tamaño de los archivos *acct*. Se deben poner entradas similares a las siguientes en el */usr/spool/cron/crontabs/root*:

```
0 * * * * /usr/lib/acct/ckdacct nqs tape uscp
0 * * * * /usr/lib/acct/ckpacct
```

9. Para correr periódicamente la contabilidad, una entrada similar a la siguiente debe ser puesta en el */usr/spool/cron/crontabs/root*.

```
15 5 1 * * /usr/lib/acct/csaperiod -r 2> /usr/adm/acct/nite/pd2log
```

   Este comando genera un reporte periódico sobre todos los archivos de datos consolidados encontrados en */usr/adm/acct/sum/data/** y después borra los archivos.

10. Actualizar el archivo de días festivos. Este archivo contiene la tabla de tiempo selecto y no selecto para la contabilidad del sistema, la cual debe ser editada para reflejar la calendarización de días festivos para el sistema durante el año.

4.6. PROCESAMIENTO DE DATOS DEL CSA

A continuación se describen los pasos que el CSA realiza para procesar los datos.

1. Generar los archivos de contabilidad no procesados (*raw*).- Diferentes ***daemons*** y ***procesos*** del sistema, escriben a los archivos de contabilidad no procesados, estos archivos incluyen *pacct*, *nqacct*, *usacct*, *tpacct*, *wtmp*, y *csainfo*.

2. Crear un archivo de cuotas.- En algunos sistemas se desea cargar cuotas a ciertos usuarios, lo cual puede hacerse con el comando *chargefee*, el cual crea un archivo de cuotas que es procesado por el comando *csaaddc*.

3. Producir las estadísticas de uso de disco.- El ***script*** *dodisk* permite a los sistemas tomar imágenes del uso de disco. *dodisk* no reporta uso dinámico, solamente el uso en un momento determinado, cuando el comando es ejecutado. El uso de disco es procesado por el *csaaddc*.

4. Preprocesar los archivos no procesados seleccionados.- Generalmente, un archivo de datos que debe ser preprocesado, contiene múltiples registros para una sesión. Estos registros están distribuidos en el archivo, y el procesamiento de los registros a menudo depende de otros eventos que son registrados en el archivo de contabilidad, por ejemplo, el *system boot*. El preprocesador colapsa información acerca de una sesión en un registro de salida.

   Los datos de contabilidad de NQS y de tiempo de conexión son preprocesados por los comandos *csanqs* y *csaline* respectivamente.

5. Organizar los datos de contabilidad.- El *csabuild* organiza los datos no procesados y los datos preprocesados por sesiones y tiempos de *boot*. Con excepción de las estadísticas de uso de disco y cuotas, el archivo de salida del *csabuild* contiene todos los datos de contabilidad disponibles de cada sesión.

6. Reciclar la información acerca de las sesiones incompletas.- Los datos de contabilidad de sesiones incompletas, son salvados y procesados nuevamente durante el siguiente período de contabilidad. Esta información es reciclada hasta que la sesión se completa o hasta que es intervenida manualmente. Los datos de contabilidad para sesiones incompletas se reportan durante cada período de contabilidad.

7. Generar el reporte de uso de ***daemons***, el cual es añadido al reporte diario. El *csadrep* da como salida información acerca de ***procesos*** interactivos, NQS, cinta y uso de USCP.

8. Convertir el archivo de registro de sesión en un formato *front-end*.

9. Generar los datos de uso de comandos. Esta información es obtenida por el *acctcms* y se reporta en los reportes diarios y periódicos.

10. Consolidar el archivo *SESSION record*. Estos archivos son demasiado grandes por lo que generalmente no se mantienen en disco mucho tiempo. El CSA consolida los datos y guarda la versión condensada en disco. Los reportes de contabilidad se obtienen de los datos consolidados. La consolidación es hecha por el *csacon*.

11. Generar un reporte de contabilidad basado en los datos consolidados. Los reportes son generados con el *csacrep*.

12. Crear el reporte de contabilidad diaria, el cual incluye:

    - Estadísticas de tiempo de conexión.
    - Estadísticas de uso de disco.
    - Información de sesiones incompletas.
    - Sumario de comandos.
    - Reporte de contabilidad consolidado.
    - Información acerca del último acceso al sistema.
    - Reporte de uso de ***daemons***.

13. Generar los datos periódicos de contabilidad. El *csaaddc* une los archivos condensados creados en el paso 10 en un solo archivo, el resultado contiene información de varios períodos de contabilidad.

14. Generar los datos periódicos del uso de comandos. *acctcms* une los datos del uso de comandos de múltiples períodos de contabilidad, que fueron creados en el paso 9. Se crea tanto un archivo ASCII como un archivo binario.

15. Producir un reporte de contabilidad periódico. El *csacrep* es usado para generar un reporte basado en un archivo periódico de contabilidad.

Los pasos del 4 al 12 son llevados a cabo durante cada período de contabilidad por el *csarun*. La contabilidad periódica, de los pasos 13 al 15, es inicializada por el comando *csaperiod*. La contabilidad diaria y la periódica, así como la generación de cuotas y el uso de disco, pueden ser calendarizados por el *cron* para ejecutarse regularmente.

4.7. CONTABILIDAD DE DAEMONS

La información de la contabilidad para los ***daemons*** NQS, tape y USCP, es escrita a los archivos *nqacct*, *tpacct* y *usacct*, respectivamente, en el directorio */usr/adm/acct/day*.

En la mayoría de los casos, la contabilidad de los ***daemons*** debe ser habilitada tanto por el subsistema CSA como por el ***daemon***. Ya se ha explicado cómo habilitar la contabilidad de ***daemons*** al momento de levantar el sistema, pero también es posible hacerlo después de que el sistema se ha levantado.

Se puede habilitar la contabilidad para un ***daemon*** específico con el comando *turndacct*. Además para el NQS y cinta, se debe habilitar la contabilidad usando el comando *qmgr set accounting on* y *tpdaemon -c* respectivamente.

La contabilidad de los ***daemons*** es deshabilitada por el comando *shutacct* al momento de dar de baja al sistema y también se puede deshabilitar sin dar de baja el sistema, usando el comando *trundacct off*. y el ***daemon***. Por ejemplo, para deshabilitar la contabilidad para NQS, se ejecuta el siguiente comando:

```
# /usr/lib/acct/turndacct off nqs
```

5. ANALISIS DEL RENDIMIENTO

Metodología de Análisis
Selección de las Técnicas
Selección de las Métricas
Selección de la Carga de trabajo
Teoría de Monitoreo
Monitoreo por medio de la contabilidad
Presentación de los datos.

Para los usuarios el rendimiento significa: "Cuánto tiempo tarda el sistema en ejecutar mi trabajo". Para un administrador, tener un buen rendimiento significa ser capaz de obtener el máximo provecho de la configuración que se tiene, por medio del correcto afinamiento del mismo. Aunque cada usuario en lo individual sufra una ligera degradación en su tiempo de respuesta.

En todo sistema, los usuarios, los administradores y los diseñadores, están interesados en la evaluación del rendimiento, ya que su meta es obtener el más alto rendimiento al más bajo costo. La evaluación del rendimiento es requerida no solo cuando el sistema está operando, sino en cualquiera de los estados de su ciclo de vida: diseño, fabricación, compra/venta, operación y actualización de hardware o software.

- En su etapa de diseño y fabricación.- Cuando un diseñador de sistemas computacionales compara varias opciones de diseño y encuentra la mejor. Debe evaluar el rendimiento de cada una de sus opciones y así decidir cuál usará.

- En su etapa de compra/venta.- Cuando alguna institución desea adquirir un equipo. El futuro administrador o grupo de administradores comparan varios sistemas. Deciden cual de ellos será el mejor para el grupo de aplicaciones para las cuales será adquirido el sistema.

- En su etapa de operación y actualización de hardware o software.- Cuando se determina qué tan bien esta el rendimiento y si se necesita realizar o no algunas mejoras al sistema. Las cuales pueden ser reconfigurando el software o realizando adquisiciones de hardware o software.

Una forma de ejemplificar lo mencionado, es lo que ha sucedido con la supercomputadora CRAY desde que se comenzó a planear su compra. Cuando

la UNAM decidió que era necesaria la adquisición de una supercomputadora, se formó la Comisión de Supercómputo. La cual realizaría el análisis en la etapa de venta/compra. Los integrantes de dicha comisión, tuvieron diversas presentaciones con los proveedores candidatos y visitas de representantes de centros de supercómputo. Su misión era la de elegir, de acuerdo a las necesidades de la UNAM, la mejor opción de compra.

Con el propósito de conocer mejor los posibles usuarios y proyectos de supercómputo en la UNAM, se emitió en marzo de 1990 una Convocatoria ofreciendo tiempo de supercómputo y gastos de viaje para el personal académico cuyos proyectos reunieran méritos académicos suficientes para trabajar en una supercomputadora. El objetivo era el de formar un ***benchmark*** tipo *Suite*. El cual es formado por programas que constituyen un grupo representativo del tipo de carga real que se tendrá en una computadora.

Con la colaboración de diversos proveedores fue posible conseguir que 10 proyectos integraran el Paquete de Pruebas con el cual se trabajaría. A este paquete se le añadió un programa que contiene el núcleo utilizado por los programas de proceso de imágenes y hace uso intensivo de la memoria central y del canal de entrada y salida, aspectos que no se encontraban cubiertos por los demás programas.

El conjunto de programas fue procesado en todas las computadoras a evaluar. Además se tomaron en cuenta aspectos de interacción, conectividad, facilidad de adaptación, optimización de programas y la calidad de la asistencia técnica de los proveedores.

También se consideraron los resultados de las pruebas realizadas con algunos otros ***benchmarks*** y las opiniones que los investigadores se formaron de cada máquina. Como ya se mencionó, la Comisión de Supercómputo fue la encargada de evaluar el rendimiento en la etapa de venta/compra. Lo que llevó a la resolución de que la oferta hecha por CRAY Research Inc. era la más apropiada.

La máquina se compraría con dos procesadores, 32 Megawords de memoria principal y 4 discos que en total tendrían una capacidad de 19 gigabytes. Pero CRAY decidió que daría como regalo los otros dos procesadores con que cuenta actualmente la supercomputadora.

Lo anterior propició que los primeros 2 años, se tuvieran reportes con un rendimiento bajo. Se observaba que los 4 CPU's no estaban utilizándose al máximo. Sin embargo los procesos que ejecutaban los usuarios, sufrían grandes tiempos de espera. Entre otras causas, lo que motivaba esta problemática era que la memoria central no se había comprado para un sistema con 4 procesadores, sino con 2. Aunque los procesadores estaban listos para ejecutar, la memoria no permitía albergar procesos para que éstos pudieran ser ejecutados.

En 1994 se duplicó la memoria, con lo que el rendimiento del sistema mejoró considerablemente. En los reportes mensuales se puede observar como el porcentaje de uso de CPU se mantiene casi siempre arriba del 80%. Además en 1993 se obtuvieron como donativo otros 4 discos, pues la población de usuarios y las aplicaciones habían crecido. Lo anterior se convirtió en una decisión de actualización de hardware en la etapa de operación del ciclo de vida de un sistema computacional.

A pesar de las actualizaciones de hardware realizadas, se sabe que los procesos de usuarios ejecutados en NQS, continúan con problemas en los tiempos de espera. Por los análisis realizados, se conoce que las causas pueden ser dos principalmente: el mal uso del sistema de colas por parte de los usuarios o que la configuración del sistema de colas no sea la apropiada.

Se analiza constantemente la configuración y se trata de que sea la mejor. Por otra parte el grupo de atención a usuarios ayuda a los usuarios para que aprendan a utilizar lo mejor posible el sistema. Pero el constante cambio de usuarios y aplicaciones, exige que el sistema se analice constantemente, permitiendo detectar lo más pronto posible a los usuarios con problemas y los cambios necesarios a la configuración del sistema NQS.

Es por ésto que se decidió que una herramienta de gran utilidad y que se necesita urgentemente es un monitor gráfico de NQS. El cual pueda proporcionar la información con la premura necesaria y ayude a realizar predicciones, antes de que algunas fallas se conviertan en problemas que puedan dañar severamente el rendimiento de la CRAY.

5.1. METODOLOGÍA DE ANÁLISIS

Las aplicaciones de computadoras son variadas y los usuarios son muy diferentes entre sí, además un mismo usuario cambia conforme va adquiriendo madurez en el uso del sistema. Por lo que no es posible tener un estándar de medición de rendimiento ni de técnicas a utilizar para ello.

La etapa de análisis dentro del ciclo de vida de los sistemas para la cual ha sido realizada esta tesis, es la etapa de operación y actualización. En donde intervienen el administrador y los usuarios. De aquí en adelante cualquier referencia a rendimiento será en esta etapa, a menos de que se indique lo contrario.

La evaluación del rendimiento permite saber si el sistema está funcionando óptimamente, es decir si los recursos se están utilizando de la mejor manera. Además, en caso de descubrir que no es así, se obtiene información importante para atacar el problema, en el mejor de los casos, antes de que éste se presente.

Para mejorar el rendimiento, se pueden realizar diferentes acciones: reconfigurar el ***kernel***, modificar hábitos de uso del sistema, adquirir nuevos productos de software que carguen menos el sistema o adquirir nuevo productos de hardware.

Como se ha mencionado, la evaluación exitosa del rendimiento de un sistema no se puede llevar a cabo en forma mecánica. Se requiere de un conocimiento a fondo del sistema, así como de una selección cuidadosa de la metodología, carga de trabajo y herramientas a utilizar para efectuar el análisis.

Para poder lograr que un sistema tenga un buen rendimiento, se propone seguir la siguiente metodología:

a) Conocer los principales objetivos del sistema._ El administrador debe tener presente en todo momento, el motivo por el cual fue adquirido el sistema, es decir, cuál es su principal función, ya que ésta deberá tener la prioridad más alta para uso de recursos.

b) Conocer las aplicaciones del sistema._ El administrador debe tener conocimiento de qué tipo de aplicaciones se utilizan y cómo es el consumo de recursos para cada aplicación.

c) Conocer la población de usuarios._ El administrador debe familiarizarse y tener el mayor contacto posible con sus usuarios. Saber qué aplicaciones ejecutan y cómo cargan el sistema, es decir, qué ***procesos*** realizan. Esto resulta muy difícil, y en ocasiones hasta imposible. La solución es agruparlos por características de uso, monitorear constantemente el sistema observando quienes se encuentran en sesión, qué están haciendo y cómo se está comportando el sistema.

d) Conocer los requerimientos de rendimiento del sistema._ El administrador debe conocer la premura de que el sistema entregue resultados. Pueden existir diferentes rangos dentro de los cuales caigan las diferentes aplicaciones que se ejecutan en el sistema de acuerdo a prioridades preestablecidas.

e) Determinar los puntos críticos del sistema._ Es la localización de los estados en los cuales el sistema muestra tanto el mejor como el peor comportamiento.

f) Afinar el sistema._ Consiste en realizar las acciones convenientes para que el sistema se comporte óptimamente. Estas acciones son determinadas en base a los resultados obtenidos en el análisis, y pueden ser modificaciones a la configuración actual del sistema, o bien aplicación de políticas de uso del mismo.

g) Determinar el número y tamaño de los componentes a adquirir._ En muchas ocasiones, la causa por la que se tiene un rendimiento bajo, es que las

aplicaciones y los usuarios han crecido y la demanda de recursos es tal, que con la configuración actual no es posible mantener el rendimiento en un estado aceptable. Se concluye con bases firmes que es necesario adquirir nuevos componentes.

h) Predecir el rendimiento para cargas futuras._ Se refiere a realizar estimaciones de crecimiento del sistema o de actualizaciones futuras. Es conveniente tener en cuenta lo que sucederá con el sistema en un futuro para poder predecir la carga de trabajo que se tendrá y buscar alternativas para mantener el buen rendimiento.

Una vez logrado el conocimiento del sistema, se determinan los puntos críticos. Este punto parece ser sencillo, pero en realidad es la parte medular del análisis. Ya que de no seleccionar cuidadosamente la carga de trabajo y no contar con las herramientas adecuadas, el análisis puede arrojar información errónea. La cual lejos de ayudarnos a realizar un buen afinamiento, nos puede conducir a terribles fallas en la operación del sistema o compra de nuevos componentes.

Las siguientes acciones, son la metodología propuesta para encontrar los puntos críticos de un sistema.

1. Seleccionar las técnicas apropiadas para la evaluación. Existen tres técnicas:

   - Medición
   - Simulación
   - Modelado Analítico

2. Seleccionar las métricas de rendimiento. Una *métrica* es un criterio para evaluar el rendimiento de un sistema, por ejemplo: el tiempo de respuesta, la velocidad de transferencia, las transacciones por segundo (TPS), etc.

3. Seleccionar la carga de trabajo del sistema. Constituida por las peticiones que harán trabajar al sistema durante el análisis. La carga de trabajo puede ser real o sintética. La real, es la que producen los usuarios cotidianamente, cuando el sistema está operando. La sintética es una carga predeterminada.

4. Medir el rendimiento correctamente. Para medir en forma correcta el rendimiento, se necesitan dos herramientas cuando se ha elegido que la carga será sintética:

   - Herramienta para cargar el sistema.- Como se ha mencionado, la carga puede ser generada o usar la carga real del sistema. Un generador de carga conocido es el RTE (Remote Terminal Emulator), el cual emula a varios usuarios.

- Herramienta para medir los resultados.- A la herramienta para medir resultados se le conoce como monitor. Los monitores pueden ser de Hardware, Software, Firmware o Híbridos.

En caso de que se haya dispuesto trabajar con la carga real, sólo se necesita una herramienta que es el monitor.

5. Usar técnicas estadísticas propias para comparar diferentes alternativas. En ocasiones, se realiza la misma medición repetidas veces y los resultados pueden ser diferentes en cada ocasión. Las técnicas estadísticas nos ayudan a utilizar los datos entregados por esta serie de medidas.

6. Desarrollar mediciones y experimentos de simulación en su caso. Las simulaciones son muy útilles cuando se desea observar cómo reacciona el sistema ante variaciones a un factor determinado. Mediante simulaciones es posible separar efectos de factores en forma individual.

   Se debe tener especial cuidado en la selección de semillas y algoritmos para la generación de números aleatorios, y en la duración de la simulación; así como en el análisis de la simulación.

7. Si se utiliza un modelado analítico, realizar el modelado correctamente. Al diseñar un modelo analítico, se debe contar con expertos que tengan los elementos suficientes para realizar un modelo lo más apegado a la realidad. En el que se hagan el menor número de suposiciones posible.

5.2. SELECCION DE LAS TÉCNICAS

A continuación se listan los criterios más importantes para efectuar la selección apropiada de la técnica a utilizar:

1) Estado del sistema.- Esto se refiere a si es un sistema nuevo o de no serlo, si hay información previa. La técnica de medición no es recomendable cuando no se tienen antecedentes de rendimiento del sistema. Ya que para que una medición tenga sentido, ésta debe ser comparada con otra por lo que se requiere de información histórica.

2) Tiempo requerido.- Es el tiempo necesario para la evaluación. En situaciones en las que es de gran urgencia presentar resultados de evaluación, el modelado analítico es probablemente la única opción. Las simulaciones toman gran tiempo. Las mediciones toman, por lo general, menos tiempo que las simulaciones pero más que el modelado analítico.

3) Herramientas disponibles.- Las herramientas con las que se cuenta para realizar el análisis, incluyen herramientas proporcionadas por el sistema, lenguajes de simulación e instrumentos y herramientas de medición. Por

ejemplo, la principal herramienta para el modelado, sería el analista que cuenta con los conocimientos, la experiencia y la habilidad necesaria para realizar el modelado.

4) Nivel de confiabilidad.- Esta es otra importante consideración que se refiere al nivel de cercanía a la realidad que tiene cada una de las posibles técnicas. Las simulaciones pueden incorporar más detalles y requerir menos suposiciones que el modelado analítico y, por lo tanto, acercarse más a la realidad. Las mediciones, a pesar de que parecen ser más reales, no siempre son las más seguras, ya que los resultados dependen de muchos parámetros de ambiente, tales como la configuración actual, el tipo de carga, y el tiempo que se utilice para realizar la medición. Estos factores en ocasiones pueden ser únicos para el momento en que se realiza el experimento. Además debe considerarse la precisión y exactitud del instrumento o herramienta con el que se realiza la medición. Por lo tanto el nivel de confiabilidad de las mediciones puede variar desde un nivel muy alto, hasta la ausencia de seguridad.

5) Costo.- Las mediciones pueden ser las más costosas cuando se realizan por medio de hardware, ya que requieren equipo real, instrumentos y tiempo. Para las mediciones por medio de software, el principal instrumento sería un monitor de software, que es más barato que uno de hardware. Aunque también se consideran costosas, debido al tiempo que toma el realizarlas y al costo que implica almacenarlas. La simulación ocupa el segundo lugar en cuanto a costos, mientras que para el modelado analítico solo se necesita papel y lápiz.

6) Factibilidad.- Este aspecto se refiere a si es vendible o no el análisis. Es mucho más fácil convencer si el análisis se basa en mediciones reales. Mucha gente no confía en resultados analíticos simplemente porque no entiende la técnica o el resultado final. En ocasiones es mejor usar más de una técnica simultáneamente.

5.3. SELECCION DE LAS METRICAS

Para cada estudio de rendimiento, se debe seleccionar un conjunto de criterios de rendimiento o métricas. Lo primero que debe hacerse es preparar una lista de los servicios ofrecidos por el sistema. Para después, determinar las métricas para cada servicio o para los servicios que sea importante medir. Dichos servicios pueden ser clasificados dentro de tres categorías:

1) El sistema realizó el servicio correctamente,
2) El sistema realizó el servicio incorrectamente o
3) El sistema no realizó el servicio.

1) Si el sistema realizó el servicio correctamente, su rendimiento es medido por el tiempo que le tomó realizarlo, la rapidez a la cual el servicio es llevado a cabo y los recursos consumidos durante la realización del servicio. Para ello se utilizan tres métricas relacionadas con: el tiempo, la rapidez y los recursos. Las cuales se conocen como:

   - Tiempo de respuesta.
   - Productividad.
   - Utilización.

   El recurso con la más alta utilización es llamado "cuello de botella". Las optimizaciones de rendimiento para este recurso son las que reditúan más en el rendimiento.

2) Si el sistema realizó el servicio incorrectamente, se dice que ha ocurrido un error. Es aconsejable clasificar los errores y determinar las probabilidades de cada clase de errores.

3) Si el sistema no realizó el servicio, se dice que ha fallado, que no está operando o que no esta disponible. Una vez más es aconsejable clasificar los modos de falla y determinar las probabilidades de cada clase.

   Las métricas asociadas con los tres tipos de servicios enumerados son conocidas como:

   - Exito.
   - Confiabilidad
   - Disponibilidad.

Durante la realización de estudios de rendimiento, suele cometerse el error de obtener solo valores medios. Estos valores son muy importantes, pero no debe pasarse por alto el efecto de la variabilidad.

Por ejemplo los reportes mensuales de uso de recursos de la CRAY, se generan solo si el sistema realizó el servicio correctamente. Se usan métricas de utilización. Existen reportes que muestran valores máximos y mínimos de uso. Es posible observar el comportamiento en horas de alto uso y en horas de bajo uso. Pudiendo así, observar cómo varía el uso durante un día normal y cómo se comporta el sistema en los fines de semana o días festivos.

Las métricas de utilización que se emplean en estos reportes son:

- Porcentaje de uso de CPU
- Bloques de uso de disco
- Bloques de uso de SSD
- Porcentaje de uso de memoria
- Porcentaje de uso de swap

Para el monitor de NQS que se propuso realizar, se usan métricas de tiempo de respuesta y de utilización. Por ejemplo:

- Métricas de Utilización.

  Para medir la memoria, se utilizan Megawords.
  Para medir el uso de CPU, se utilizan segundos.
  Para medir el uso de alguna cola, se utiliza el número de procesos encolados.

- Métricas de Tiempo de Respuesta.

  Para medir el tiempo de espera en las colas, se utilizan segundos.

La utilidad de estas métricas, se observa cuando se interrelacionan unas con otras. Por ejemplo si una cola tiene grandes tiempos de espera, y se observa que esa cola no atiende gran número de procesos. Se podría pensar que un mismo usuario lanzó varios procesos al mismo tiempo. Lo cual obstruye la cola, ya que un usuario solo puede tener 2 procesos ejecutándose al mismo tiempo. Pero por otro lado la cola puede ejecutar 4 procesos al mismo tiempo. Entonces a pesar de que por límite de procesos la cola podría ejecutar los dos siguientes procesos encolados, no puede hacerlo por el límite de procesos por usuario.

Este problema se presenta porque el usuario no sabe que existe un límite de procesos por usuario. O en ocasiones no le importa perjudicar el rendimiento y simplemente manda a ejecución todos sus procesos.

También por medio de la métrica de utilización de memoria, se podrá observar si los complejos de memoria son adecuados. Por ejemplo supongamos que se tiene un complejo de memoria de 4 MW y el siguiente complejo hacia arriba es de 20 MW. Si se observa que en la cola de 20 MW se ejecutan muchos procesos que ocupan entre 5 y 15 MW. Se podrá concluir que es necesario crear una cola intermedia de 15 MW.

En sistemas compartidos por muchos usuarios, dos tipos de métricas de rendimiento deben ser consideradas:

- Las individuales
- Las globales.

Las individuales reflejan la utilización de cada usuario, mientras que las globales, reflejan la utilización del sistema total. La utilización de recursos, la confiabilidad y la disponibilidad son métricas globales, mientras que el tiempo de respuesta puede ser medido tanto globalmente como individualmente.

5.4. SELECCION DE LA CARGA DE TRABAJO

La selección de la carga de trabajo es una de las partes más importantes en cualquier proyecto de evaluación de rendimiento. Puesto que es posible llegar a conclusiones erróneas si la carga de trabajo no fue debidamente seleccionada. Como muchos otros aspectos de la evaluación del rendimiento, la selección adecuada de la carga de trabajo requiere de muchas consideraciones por parte del analista, lo que es parte del arte de la evaluación del rendimiento que se adquiere con la experiencia.

La carga de trabajo puede ser clasificada en dos tipos:

- *Carga de trabajo real.*- Es aquella que se observa cuando el sistema se encuentra en operación normal. Por lo cual es difícil de controlar y por consiguiente imposible de repetir en el sistema y mucho menos en sistemas ajenos. De ahí que la carga de trabajo real no sea utilizada para comparar el comportamiento de diferentes sistemas en un estudio de rendimiento. Pero es valida cuando se desea mejorar el rendimiento de un sistema en particular.

- *Carga de trabajo sintética.*- Sus características deben ser similares a la carga de trabajo real, puede ser repetida de una manera controlada, por lo que puede ser desarrollada y aplicada en la comparación de sistemas en un estudio de rendimiento.

Existe un curso llamado "System Performance Evaluation and Tuning" (Evaluación del Rendimiento y Afinación de Sistemas), que es impartido por CRAY Research Inc. El cual realiza un estudio de rendimiento en 2 máquinas. Ambas máquinas se someten a una carga de trabajo sintética, prefabricada por un especialista de CRAY. Estas máquinas son máquinas dedicadas que se utilizan solo para la impartición de cursos y realización de pruebas. Es por esto que pueden ser comparadas y fácilmente se puede determinar cuál de las 2 configuraciones es mejor para una determinada carga sintética.

Por el contrario, la carga que se utilizará para medir el rendimiento del sistema de colas de la CRAY de la UNAM, es la carga real. Lo anterior debido a que no se trata de comparar a esta máquina con otras, sino de analizar cómo es su comportamiento real a lo largo del tiempo y cómo reacciona el sistema ante un cambio de configuración en el sistema de colas.

Comúnmente, los siguientes tipos de carga de trabajo han sido utilizados para comparar sistemas computacionales:

1. La instrucción suma
2. Combinación de instrucciones
3. Kernels
4. Programas sintéticos

5. Benchmarks

La instrucción suma

Históricamente, cuando aparecieron los primeros sistemas computacionales, los procesadores eran los componentes más caros y más utilizados dentro de un sistema, por lo cuál el rendimiento de un sistema era sinónimo del rendimiento de su procesador. Las primeras computadoras tenían un conjunto de instrucciones muy pequeño, siendo la más común de las instrucciones *la instrucción suma*, así que en los primeros estudios de rendimiento la computadora que realizaba más rápido la instrucción suma, era considerada como la de mejor rendimiento, de ahí que la instrucción suma era la única carga de trabajo y el tiempo de ejecución de la instrucción suma era la única métrica de rendimiento.

Combinación de instrucciones

Con el crecimiento del número y la complejidad de las instrucciones disponibles, la instrucción suma empezó a ser insuficiente para comparar sistemas, por lo que hubo la necesidad de contar con una carga de trabajo más detallada. Esto llevo a los expertos a utilizar lo que se conoce como *combinación de instrucciones*, que consiste en seleccionar una subconjunto de instrucciones acompañadas cada una de ellas con su respectiva frecuencia de uso, que se utiliza como factor de peso para cada instrucción. Así se puede obtener un promedio general de tiempo de ejecución de instrucción por sistema y así compararlo con otros sistemas.

En la actualidad, este tipo de carga de trabajo no es muy utilizada, ya que el tiempo de ejecución de las instrucciones en los sistemas modernos es muy variable, debido a la diferencia de arquitecturas entre los sistemas. El tiempo de ejecución depende directamente de algunos aspectos como pueden ser los diferentes modos de direccionamiento, el ***buffer cache*** y la interacción con otros dispositivos en la transferencia de datos.

Kernels

Debido a la alta variabilidad del tiempo de ejecución de las instrucciones en los sistemas, surgieron los [1]*kernels*, que son considerados una generalización de la combinación de instrucciones. Un kernel es la función o servicio proporcionado por el procesador con mayor uso; es decir a partir de una lista de funciones del procesador, se elige aquella que se utiliza más frecuentemente y se utiliza como carga de trabajo, denominándose a dicha función como kernel. En algunos

[1] No confundir con el kernel de UNIX, ya que simplemente es un nombre para identificar a este tipo de carga de trabajo.

sistemas especializados se puede identificar un conjunto de operaciones comunes, por ejemplo la operación de invertir una matriz, y en base a este kernel, comparar diferentes procesadores.

La principal desventaja de los kernels, es que tipicamente no interactuan con dispositivos de E/S, por lo que el rendimiento del kernel no refleja el rendimiento total del sistema.

Programas sintéticos

Como consecuencia de que los kernels no realizan operaciones con dispositivos de E/S y debido a que estas operaciones empezaron a tomar importancia dentro de las cargas de trabajo, surgieron los *programas sintéticos*, que consisten en ciclos de instrucciones que realizan peticiones de E/S.

Dentro de las principales ventajas de este tipo de programas, se encuentran la baja complejidad para desarrollarlos y su portabilidad a otros sistemas.

Benchmarks

En forma general, el termino ***benchmark*** está estrechamente relacionado con la carga de trabajo. Los kernels y programas sintéticos por ejemplo, son comúnmente llamados ***benchmarks***. Algunos autores definen al proceso de comparar dos o mas sistemas como *benchmarking*. Y a la carga de trabajo utilizada en la comparación como ***benchmarks***. En forma particular, podemos decir que los ***benchmarks***, son programas diseñados especialmente para cargar al sistema y usar recursos de forma determinada. De esta manera podemos probar los componentes del sistema, controlando el uso de cada uno de ellos.

Diseñar ***benchmarks***, es tan complicado como describir los síntomas de un rendimiento pobre. El ***benchmark*** apropiado para cualquier situación depende del problema que nuestro sistema tenga. Desarrollar buenos ***benchmarks*** es absolutamente crucial en la afinación de un sistema.

Si se desean comparar sistemas y estos serán utilizados para una aplicación en particular, un conjunto representativo de funciones para dicha aplicación es la carga de trabajo ideal que se debe utilizar De ahí que los ***benchmarks***, se describen de acuerdo a las funciones que realizan.

No siempre es necesario que el administrador construya sus ***benchmarks.*** Existen ***benchmarks*** comerciales que pueden adaptarse a las necesidades del análisis en cuestión y que han sido probados en diferentes sistemas. Por mencionar algunos ejemplos, existe el *LINPAK*, desarrollado por Jack Dongarra en 1983 y que consiste en un conjunto de programas para resolver sistemas densos de ecuaciones lineales. O bien el *kernel sieve*, que ha sido utilizado para

comparar microprocesadores, computadoras personales y lenguajes de alto nivel. Este ***benchmark*** esta basado en el algoritmo de Eratosthenes utilizado para encontrar todos los números primos que existen, hasta un número dado.

El ***benchmark*** de *LINPAK*, fue utilizado para la evaluación de las propuestas cuando la Comisión de Supercómputo decidió qué supercomputadora comprar para la UNAM. En esa ocasión también se utilizó el *benchmark suite*, el cual es formado por un paquete de programas que puede considerarse representativo del tipo de carga real que la máquina a evaluar tendría. Ese paquete de pruebas fue procesado en todas las supercomputadoras evaluadas. Además de estos 2 tipos de ***benchmarks***, se utilizó una carga sintética que procesaría imágenes y haría uso intensivo de la memoria principal y del canal de entrada/salida.

5.5. TEORIA DE MONITOREO

Una vez que se ha seleccionado la carga de trabajo a utilizar, se necesita la herramienta para medir los resultados de la carga de trabajo: un monitor. El cual es una herramienta que permite observar las actividades de un sistema. En general los monitores observan el rendimiento de los sistemas, colectando estadísticas, procesando datos y desplegando resultados, algunos identifican las áreas con problemas y proponen soluciones.

Existen diferentes personas interesadas en la realización de monitoreo, por ejemplo un programador de sistemas, el cual puede usar un monitor para encontrar los segmentos más frecuentemente usados de su software y optimizar su rendimiento. O bien un administrador de sistemas, quien puede usar un monitor para medir la utilización de recursos y encontrar cuellos de botella, caracterizar la carga de trabajo, o para realizar afinaciones al sistema ajustando los parámetros necesarios.

Para poder realizar un estudio de rendimiento y afinación es necesario monitorear el sistema. Para elegir el monitor adecuado es importante familiarizarse con los términos que se utilizan para los monitores. Los más importantes se describen a continuación en la Tabla 5-1:

TABLA 5-1. TERMINOLOGÍA DE MONITORES.

TÉRMINO	DESCRIPCIÓN
Evento	Un cambio en el estado del sistema, que debe ser observado o que dispara alguna acción del monitor. Por ejemplo, cuando se inicia una búsqueda en un disco.
Registro de eventos	Bitácora de eventos que contiene información importante acerca de los mismos. Puede incluir: la hora del evento, el tipo de evento, el tiempo que transcurrió al sucederse el evento.
Consumo de recursos	Los recursos del sistema que emplea un monitor. Por ejemplo, cuando un monitor realiza procesos pesados y hace gran uso del CPU, se dice que tiene un alto consumo de recursos. Uno de los parámetros de diseñó más importantes de los monitores es reducir el consumo de recursos.
Dominio	El conjunto de actividades observables por el monitor. Por ejemplo, si se observa la actividad del CPU, el número de discos que se observarán, las terminales, etc.
Frecuencia de observación	La máxima frecuencia de eventos que un monitor puede observar correctamente.
Resolución	La fineza con la que se observa la información.
Ancho de banda	El número de bits de información registrados sobre un evento. Determina el almacenamiento requerido para registrar los eventos.

5.5.1 MONITORES DISTRIBUIDOS

Muchos de los sistemas computacionales actuales son distribuidos y consisten de diversos componentes de hardware y software que trabajan juntos, separadamente y concurrentemente. Monitorear un sistema distribuido es más difícil que monitorear un sistema centralizado.

Un monitor por sí mismo es distribuido ya que consiste en varios componentes que trabajan tanto en forma separada como concurrentemente. La manera más fácil de entender los componentes de un monitor es dividiendo sus funciones en

capas. Cada capa hace uso de los servicios proporcionados por las capas más bajas y extiende los servicios disponibles a las capas de niveles más altos.

Estrictamente hablando, un monitor puede estar formado por cero o más componentes de cada capa, siempre y cuando cuente con por lo menos un componente de alguna capa. Frecuentemente se construyen monitores que solo se componen de elementos pertenecientes a una capa. Por lo que es muy común utilizar el término de "monitor distribuido" cuando se trata de un monitor más completo que cuenta con componentes de por lo menos 2 capas.

Puede existir una relación de muchos a muchos entre capas sucesivas. Por ejemplo un observador puede enviar datos a múltiples colectores. Y un colector puede obtener datos de múltiples observadores.

En la Figura 5-1, se ilustran las capas con las que puede contar un monitor distribuido y a continuación se explica cada una de ellas, así como su relación con el monitor de NQS objetivo de este trabajo de tesis.

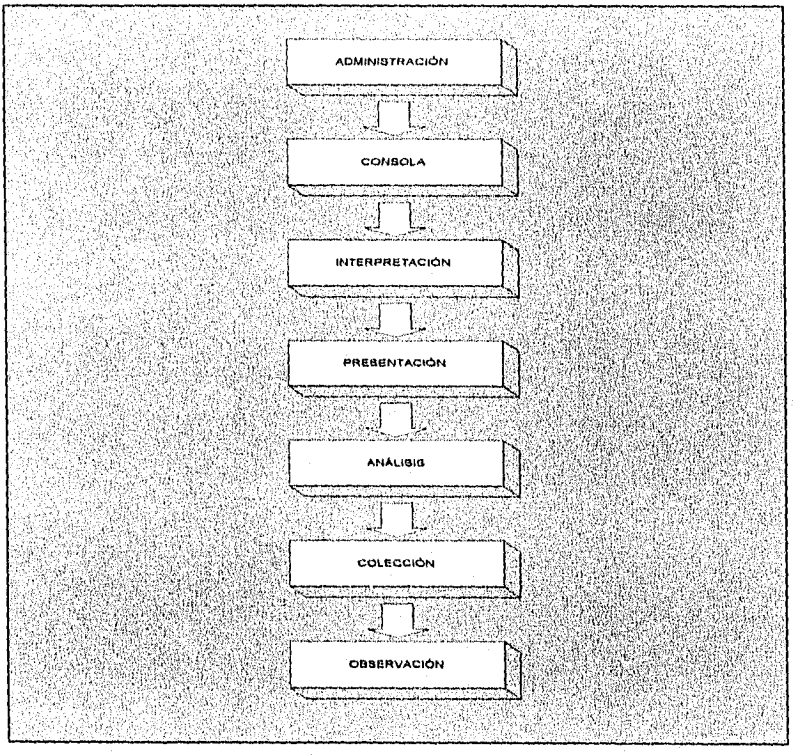


FIGURA 5-1. CAPAS DE UN MONITOR DISTRIBUIDO.

1. *Observación*: Esta capa recaba datos sin procesar de componentes individuales del sistema. Generalmente, cada componente puede tener un observador diseñado especialmente para él. Puede haber varios observadores localizados en diferentes subsistemas, cuando se trata de sistemas distribuidos.

   Para el monitor del sistema NQS se cuenta con un observador que no genera mucha carga debido a que está construido como parte del sistema operativo. Este observador es el sistema de contabilidad de la CRAY, el CSA. Debido a que el CSA registra información acerca del NQS, no fue necesario construir un observador. Esto hubiera representado más trabajo en la realización del monitor. Además es seguro que si se hubiera construido un observador, éste generaría más carga al sistema de la que puede generar la contabilidad.

   El SAR es un observador que se utiliza en la generación de reportes mensuales de uso de recursos, pero no es posible obtener información del sistema de colas mediante el SAR.

   Debido a que el monitor que se realizó está compuesto por varias capas, se dice que la contabilidad es el observador del monitor. Pero el CSA puede considerarse por sí mismo como un monitor, si se recuerda la definición de monitor en la que se especifica que puede tener cero o más componentes de cada capa.

2. *Colección*: Esta capa colecta los datos de los diferentes observadores. Es posible tener más de un colector en sistemas muy grandes.

   En CRAY existen comandos que obtienen información de los archivos de contabilidad. Para el caso particular del monitor gráfico de NQS, su colector fue construido en *bourne shell*. Utiliza el comando *csadrep*. Este comando que también es parte del sistema operativo, colecta la información acerca del sistema de colas NQS. Toma como entrada un archivo que genera la contabilidad diariamente y que se llama *SESSION*. Este archivo puede medir entre 20 y 40 megabytes. Cuando el *csadrep* colecta la información, genera un archivo de tamaño menor a 5 kilobytes.

3. *Análisis*: Esta capa analiza los datos recolectados por los colectores. Puede utilizar rutinas estadísticas para agrupar características. Realiza compresión de datos y obtención de resultados.

   Para el monitor de NQS, esta etapa fue construida utilizando programas en *bourne shell* y *awk*. Se divide principalmente en dos partes. En la primera se obtienen, promedios, sumatorias, modas y porcentajes. En la segunda parte, se preparan los datos obtenidos, para que la capa de presentación pueda tomarlos realizando el mínimo procesamiento. Los archivos que genera el análisis ocupan muy poco espacio. El consumo de CPU por parte del

analizador, no es considerable. Diariamente procesa los datos en horas de bajo uso.

Los archivos de contabilidad, permanecen cuando mucho 1 mes en el disco. Debido a su tamaño son respaldados en cinta y eliminados del disco. Los archivos que genera el colector no tienen que ser respaldados, ya que el analizador los utiliza inmediatamente que se han creado y enseguida los elimina. El monitor de NQS tiene un consumo de disco muy bajo, ya que los archivos que necesitan permanecer en disco son muy pequeños.

4. *Presentación*: Este componente del monitor tiene que ver con la interfaz con el usuario. Puede producir reportes, gráficas o alarmas.

   La capa de presentación del monitor de NQS se realizó en lenguaje *bourne shell*, *awk* y C, utilizando las bibliotecas de OSF/Motif y el graficador llamado *xgraph*, el cual permite presentar la información por medio de gráficas de barras.

5. *Interpretación*: Se refiere al sentido que toman los datos en la presentación. La realizan generalmente analistas que después de observar la presentación, realizan conclusiones.

   La interpretación será realizada por el administrador del sistema de colas de la CRAY de la UNAM.

6. *Consola*: Este componente proporciona una interfaz para controlar los parámetros y estados del sistema. Estrictamente hablando, la consola no es una parte del monitor. Sin embargo, el monitoreo y control de funciones, son muy a menudo usadas juntas.

   La CRAY cuenta con una herramienta de configuración llamada Menú del Sistema, en la cual es posible modificar los parametros de configuración.

7. *Administración*: La entidad que toma la decisión para realizar los cambios a los parámetros del sistema o configuraciones, basandose en la interpretación del monitor de rendimiento. El administrador implementa su decisión utilizando una consola. Un administrador de software existe solamente en monitores inteligentes con facilidades de control automatizadas.

5.5.2. CLASIFICACIÓN DE MONITORES

Los monitores son clasificados en base a un número de características, tales como el nivel de implementación, el mecanismo de activación y la habilidad para desplegar los resultados. Estas tres clasificaciones pueden ser usadas para caracterizar un monitor.

- De acuerdo al Nivel de implementación:

  - Monitores de Software.
  - Monitores de Hardware.
  - Monitores de Firmware.
  - Monitores Híbridos.

- De acuerdo al mecanismo de activación.

  - Monitores por evento.- Los cuales se activan cuando ocurre un evento. Cabe mencionar que si el evento ocurre muy a menudo, puede ocasionarse un consumo de recursos alto.

    La contabilidad es un monitor por evento. Después de que se ejecuta un proceso, se registra la información referente al uso de recursos del mismo.

  - Monitores de muestreo.- Se activan cada determinado intervalo de tiempo, independientemente de los eventos que puedan ocurrir. Son ideales para observar eventos frecuentes.

    La utilería *sar* en un monitor de muestreo. Cada determinado intervalo de tiempo, recolecta información del estado del sistema.

- De acuerdo a la forma de desplegarlos:

  - Monitores en línea.- Despliegan el estado del sistema continuamente, o en intervalos, pero de manera interactiva. Muestran lo que esta sucediendo en el momento de ejecutarlos.

    En la CRAY es utilizado el *SysTop*, un monitor en línea que muestra el comando que se está ejecutando y el porcentaje de uso de CPU que esta utilizando, separado por procesador.

  - Monitores en batch.- Colectan los datos que pueden ser analizados más tarde usando un programa de análisis.

    El monitor de NQS es un monitor en batch. Su observador y su colector van reuniendo información que es utilizada más tarde por el analizador.

Monitores de Software

Se usan para monitorear al Sistema Operativo y software de alto nivel, como redes o bases de datos. Varias instrucciones son ejecutadas en cada activación.

En comparación con los monitores de hardware, los de software tienen una frecuencia de observación menor, menos resolución y más alto consumo de recursos.

A continuación se muestran algunos parámetros de diseño importantes para realizar el diseño de un monitor de software:

1. *Mecanismos de activación.*- La primera decisión que el diseñador de un monitor de software tiene que tomar es cómo activar la rutina de colección de datos, para ello existen tres mecanismos:

    a) *Instrucción Trap.*- Consiste en implementar el software con instrucciones *trap* en puntos apropiados en el código. La instrucción *trap* es un mecanismo de interrupción de software que transfiere el control a una rutina de colección de datos. El monitor mide tiempos totales de ejecución para varios servicios del Sistema Operativo usando instrucciones *trap* al principio y al final del código del servicio. El efecto de una instrucción *trap* es muy similar al de una llamada a una subrutina. De hecho, si no se dispone de una instrucción *trap*, se usa una llamada a una subrutina.

    b) *Modo de registro de eventos.*- Consiste en cambiar el procesador a un modo de recolección de datos. La ejecución del programa es interrumpida después de cada instrucción y el control se pasa a una rutina de colección de datos. Tiene un muy alto consumo de recursos, se usa para monitorear aplicaciones donde el tiempo entre eventos no tiene que ser medido.

    c) *Reloj de interrupción.*- Se usa un servicio de reloj de interrupción proporcionado por el Sistema Operativo para transferir el control a una rutina de colección de datos a intervalos de tiempo determinados. Este mecanismo llamado muestreo, es especialmente bueno para eventos frecuentes ya que el consumo de recursos es independiente de la frecuencia del evento.

2. *Tamaño del Buffer.*- La mayoría de los monitores de software registran datos en ***buffers*** para después enviar la información a cinta o a disco. El tamaño de los ***buffers*** debe ser grande para que se disminuya la frecuencia de escritura a almacenamiento secundario. Pero debe ser pequeño para que el tiempo perdido en una operación de escritura no sea muy grande y no se perciba el uso de memoria del monitor. Si el ***buffer*** es muy grande, reservará mucha memoria que tal vez no sea utilizada inmediatamente. Por lo tanto el tamaño óptimo de un ***buffer***, está en función del la frecuencia de observación, el ancho de banda y la frecuencia de envío a almacenamiento secundario.

3. *Número de buffers.*- Los ***buffers*** son usualmente organizados en forma de anillo. Realizan los procesos de registro de datos hacia el ***buffer*** y escritura de datos del ***buffer*** al almacenamiento secundario en forma paralela. Se van limpiando y llenando automáticamente. Por lo que se requiere un mínimo de 2 ***buffers*** para la operación continúa y simultánea. De lo contrario, el monitoreo tendría que ser detenido mientras se escribe a disco, para poder continuar con el registro de datos.

4. *Desbordamiento de buffers.*- A pesar de que existan varios ***buffers***, es posible que todos se llenen. Se puede sobreescribir uno ya usado o dejar de registrar datos hasta que uno se libere, pero en ambos casos se pierde información.

5. Compresión de datos y análisis.- Es posible que el monitor procese datos conforme los va observando, esto reduce en gran parte la demanda de almacenamiento pero aumenta el consumo de recursos.

   El monitor de NQS no procesa los datos conforme los va observando. Los almacena durante todo el día y al final del día los colecta y los procesa. Pero tampoco espera meses para procesarlos. No afecta al almacenamiento secundario, ya que solo es necesario mantener los datos un día. No afecta el consumo de CPU, ya que el procesamiento no es muy frecuente. Además realiza el análisis y la compresión de datos en horas de bajo uso del sistema.

6. *Mecanismo de activación y desactivación.*- La mayoría de los monitores tienen un mecanismo para habilitar o deshabilitar el monitoreo. Es muy conveniente tener la posibilidad de deshabilitarlo cuando no se requiere, ya que el tenerlo activado produce consumo de recursos.

7. *Lenguaje.*- La mayoría de los monitores están escritos en un lenguaje de programación de bajo nivel, como el ensamblador y el C. Tienen la finalidad de mantener el consumo de recursos al mínimo. El colector de un monitor de software, es por lo regular, parte del Sistema Operativo.

   Como se ha mencionado, el monitor de NQS cuenta con un colector construido en Bourne Shell que ejecuta comandos que son parte del sistema operativo y que están realizados en lenguaje C.

8. *Prioridad.*- Si el monitor se ejecuta asíncronamente, su prioridad debe ser baja para que las operaciones clave del sistema sean afectadas lo menos posible. Sin embargo, si se requiere registrar la hora de los eventos, la prioridad debe ser alta, para que el retraso en su ejecución no cause un desvío significante en las horas registradas.

9. *Monitoreo de eventos anormales.*- Un monitor debe ser capaz de observar tanto eventos normales como anormales. Por lo regular, los anormales

tienen mayor prioridad que los normales, ésto debido a que los anormales ocurren con menos frecuencia que los normales y causan menor consumo de recursos. Un monitor de eventos anormales además de presentar reportes de texto y gráficas, contaría con alarmas para dar aviso de que algo anormal esta sucediendo.

Monitores de Hardware

Un monitor de hardware consiste de equipo separado que se conecta al sistema para monitorearlo. Casi no se consumen recursos del sistema. Este tipo de monitores generalmente tienen un consumo de recursos bajo, en comparación con los de software y su frecuencia de observación es alta.

Existen algunos monitores de hardware de propósito general en el mercado, que pueden tener los siguientes elementos:

1. *Impedancias*: Se usan pruebas de alta impedancia para observar señales de puntos específicos en el hardware del sistema.

2. *Contadores*: Son incrementados cuando ocurre un evento en particular.

3. *Elementos lógicos*: Las señales obtenidas de las pruebas, pueden ser combinadas usando compuertas lógicas como el AND, el OR, etc. A su vez, las combinaciones pueden ser usadas para indicar eventos que incrementan a los contadores.

4. *Comparadores*: Pueden ser usados para comparar contadores o valores de señales con valores preestablecidos.

5. *Hardware de Mapeo*: Proporciona histogramas de cantidades observadas para que sean procesadas. Consiste en varios comparadores y contadores.

6. *Reloj*: Usado para registrar horas o para activar operaciones de muestreo.

7. *Cinta/Disco*: La mayoría de los monitores tienen dispositivos para almacenar datos.

Los monitores de hardware han evolucionado a lo largo de varias generaciones de desarrollo. Originalmente, los monitores contenían alambrados de control lógico. La siguiente generación tenía hardware de mapeo con memoria y comparadores. Hoy en día los monitores son inteligentes, son programables y contienen sus propio procesador, memoria y dispositivos de E/S.

Comparación entre monitores de hardware y de software

Resulta difícil determinar, teniendo un problema dado, cuál tipo de monitor será el más conveniente. A continuación se muestran algunas consideraciones que son determinantes para decidir cuál tipo de monitor utilizar:

1. *Dominio.*- Es considerar qué necesita ser medido. Es decir, con cuál de los 2 tipos se puede medir.

- Monitores de Hardware.- Pueden medir señales eléctricas en los busses y registrarlas de manera segura incluso a muy altas velocidades. Sin embargo, les resulta difícil determinar información a más alto nivel, como longitudes de colas o usuarios actuales, a menos que la información esté disponible en un registro de hardware.

- Monitores de Software.- Pueden determinar fácilmente la información de alto nivel. No pueden observar eventos a bajo nivel, como el tiempo de fetch de una instrucción.

2. *Frecuencia de observación.*- Es la frecuencia a la cual los eventos tienen que ser observados.

- Monitores de Hardware.- Pueden registrar eventos a frecuencias más altas que los monitores de software..

- Monitores de Software.- Requieren cientos de instrucciones por observación. No pueden ser usados si el tiempo entre eventos es pequeño.

3. *Resolución de tiempo.*

- Monitores de Hardware.-Tienen un reloj separado y pueden proporcionar una resolución del orden de los nanosegundos.

- Monitores de Software.- Usan el reloj del sistema, el cual típicamente tiene una resolución del orden de los milisegundos.

4. *La experiencia del analista de rendimiento.*

- Monitores de Hardware.- Requieren conocimiento íntimo del hardware del sistema.

- Monitores de Software.- Requieren conocimiento íntimo del software del sistema.

5. *Capacidad de registro de datos.*

- Monitores de Hardware.- No afectan el consumo de recursos. Son recomendables si no se tiene almacenamiento secundario suficiente en el sistema, ya que este tipo de monitores pueden contar con sus propios dispositivos de almacenamiento.

- Monitores de Software.- Directamente se afecta el consumo de recursos.

6. *Ancho de Banda.*

   - Monitores de Hardware.- Tienen diversas pruebas que pueden registrar eventos simultáneos.

   - Monitores de Software.- Es secuencial, por lo que no puede registrar eventos simultáneos, a menos que sea software distribuido.

7. *Consumo de recursos.*

   - Monitores de Hardware.- Consume pocos recursos del sistema, o en el mejor de los casos, no los consume.

   - Monitores de Software.- Consumen recursos que deberían estar disponibles para los usuarios.

8. *Portabilidad.*

   - Monitores de Hardware.- La mayoría de los monitores de hardware son independientes de la arquitectura y del Sistema Operativo.

   - Monitores de Software.- Son desarrollados específicamente para un hardware y un software en particular.

9. *Disponibilidad.*

   - Monitores de Hardware.- Un monitor de éste tipo, no depende del funcionamiento del sistema que está analizando. Si éste se encuentra funcionando mal, el monitor puede servir para hacer un rastreo o análisis de falla del sistema.

   - Monitores de Software.- Pueden no estar funcionando correctamente, si el sistema observado falla. El monitor detendrá su ejecución en el caso de que el sistema este fuera de operación.

10. *Errores.*

- Monitores de Hardware.- A pesar de revisar completamente un monitor de hardware, es posible olvidar alguna prueba o conectarlas a los puntos equivocados.

- Monitores de Software.- Una vez que han sido revisados, es poco probable que se sucedan errores.

11. *Costo.*

    - Monitores de Hardware.- Se consideran de alto costo, ya que se requiere de equipo e instrumentación. Son mucho más costosos que los de software.

    - Monitores de Software.- Su costo es medio. Se requiere del tiempo que tomará realizarlo y las herramientas que se utilizarán para ello.

Monitores de Firmware y Monitores Híbridos.

Los monitores de ***firmware*** son implementados modificando el microcódigo del procesador. Son muy útiles para aplicaciones que caen entre las fronteras del monitoreo de hardware y el de software. Los monitores de ***firmware*** son en muchos casos, similares a los de software, sin embargo, los de ***firmware***, hacen una reducción de datos muy limitada, si es que la hacen. Son muy útiles en aplicaciones donde las consideraciones de tiempo impiden el uso de monitores de software y la inaccesibilidad de puntos de prueba impide el uso de monitores de hardware.

Los monitores de ***firmware*** han sido usados para monitoreo de redes donde las interfaces de red existentes, pueden ser fácilmente microprogramadas para monitorear todo el tráfico en la red.

Un monitor que usa combinaciones de software, hardware o ***firmware*** es un monitor híbrido. Los monitores de software tienen muy buena capacidad para reducción de datos, mientras que los de hardware tienen alta resolución, por lo que un híbrido que consiste de un componente como el recolector de datos de hardware y un componente de reducción de datos de software, obtiene los beneficios de ambos tipos de monitor.

Monitores de ejecución de programas

Los monitores de ejecución de programas también conocidos como optimizadores o analizadores. Son monitores de software diseñados para observar aplicaciones de software y ayudan a mejorar el rendimiento de los programas.

Hay muchas razones para monitorear la ejecución de un programa, algunas de las cuales se mencionan a continuación:

- *Tracing*: Para encontrar la ruta de ejecución de un programa.
- *Timing*: Para encontrar el tiempo empleado en los módulos del programa.
- *Tuning*: Para encontrar las secciónes del código que se ejecutan más frecuentemente o las que consumen más tiempo.
- *Chequeo de variables*: Para verificar las relaciones de las variables del programa.

Los programas que serán monitoreados y optimizados deben ser escogidos con base en los siguientes criterios:

1. El tiempo de respuesta: Algunos programas son muy críticos en cuanto al tiempo y es importante encontrar dónde se utiliza la mayor parte del tiempo para así poder mejorar la respuesta.
2. La frecuencia de uso: Los programas usados con una frecuencia muy alta deber ser optimizados primero.
3. El uso de recursos: Los programas que consumen gran porcentaje de los recursos deben ser optimizados.

Las tres técnicas más comunes para optimizar programas son:

1. La optimización del código: Consiste en encontrar la frecuencia dinámica de ejecución de los módulos del programa y optimizar el código de los módulos más frecuentemente usados.
2. La optimización de E/S: Consiste en unir las peticiones de E/S en grandes registros, cambiando el método de acceso a los archivos.
3. La optimización de la ***paginación***: Consiste en observar el patrón de referencia de páginas y reorganizar los segmentos del programa para que la actividad de ***paginación*** se minimize.

5.6. MONITOREO POR MEDIO DE LA CONTABILIDAD

La mayoría de los sistemas cuentan son un sistema de contabilidad donde se almacenan los recursos usados por los usuarios y sus procesos. Esta información es muy útil cuando se quiere analizar el rendimiento de un sistema.

Además gracias a esa información se pueden realizar cobros por uso de recursos a los usuarios.

La contabilidad puede ser clasificada como observador/colector de un monitor de software porque proporcionan información útil acerca del uso y rendimiento del sistema. Puede llegar a considerarse que la contabilidad es por sí misma un monitor de software. Antes de comenzar a desarrollar un monitor, es importante analizar las facilidades que incluye el sistema. A menudo no es necesaria la construcción de un observador/colector.

La ventaja principal de un registro de contabilidad es que ya ha sido implementado y forma parte del sistema. La mayoría de los datos se colectan durante la operación normal del sistema y el consumo de recursos es pequeño. Por lo que la colección de datos puede ser llevada a cabo por períodos de tiempo grandes. Otro aspecto importante es que los datos reflejan el uso real del sistema.

La desventaja principal de los registros de contabilidad es que no cuentan con programas de análisis y mucho menos con un despliegue apropiado. Lo más que pueden realizar, es convertir los registros en un formato legible o producir un sumario total.

5.7. PRESENTACIÓN DE LOS DATOS

Uno de los aspectos más importantes en cualquier estudio de evaluación de rendimiento, es la presentación de los resultados finales. Si partimos de la idea de que el objetivo principal de un estudio de rendimiento es ayudar en la toma de decisiones. Cualquier análisis cuyos resultados no pueden ser interpretados por los encargados de tomar las decisiones, es tan bueno como aquel análisis que nunca se lleva a cabo. Es responsabilidad del analista asegurarse que los resultados del análisis sean lo suficientemente claros y simples para los encargados de tomar las decisiones. Esto requiere del uso de las palabras, dibujos y gráficas adecuadas al momento de presentar los resultados finales del análisis. Gráficas de línea, de barra, de pie e histogramas, se emplean comúnmente en la presentación final de un estudio de rendimiento. Presentan algunas ventajas sobre explicaciones textuales de los resultados. En una gráfica es más fácil enfatizar o clarificar un punto determinado, reforzar una conclusión o bien resumir el resultado de un estudio.

Tipos de variables

Un factor importante que debe ser considerado al momento de elegir el tipo de gráfica a utilizar, es el tipo de variable a desplegar. Las variables pueden ser de dos tipos: *cualitativas* o *cuantitativas*. Las variables cualitativas, también llamadas variables *categóricas*, tienen estados, niveles o categorías. Su valor

esta definido por un conjunto de subclases mutuamente excluyentes. Dichas subclases normalmente se expresan en palabras y pueden ser ordenadas o no ordenadas. Por ejemplo la variable tipos de computadoras puede tener las siguientes tres subclases, supercomputadoras, minicomputadoras y microcomputadoras, y además esta ordenada. Las variables cuantitativas son aquellas cuyos valores posibles, están expresados numéricamente. Se dividen en dos tipos, *discretas* y *continuas*. Para una variable discreta, el número de valores que puede tomar puede ser finito o infinito, pero en cualquier caso el número puede ser contable. Una variable continúa puede tomar un número infinito e incontable de valores.

De lo anterior, es muy importante conocer el tipo de variable a graficar, ya que el tipo de gráfica a utilizar depende directamente de la misma. Si se desea mostrar la relación entre dos variables de tipo continuo, una gráfica de lineas es la adecuada. Una gráfica de barras es ideal cuando la variable independiente a graficar, es una variable discreta o cualitativa.

Recomendaciones para la elaboración de gráficas.

Las siguientes recomendaciones tratan de abarcar los casos más comunes en la elaboración de gráficas. Es importante señalar que son solo recomendaciones y no reglas.

1. *Requerir el mínimo esfuerzo del lector.* Uno de los parámetros más importantes que determinan la calidad de una gráfica, es el nivel de esfuerzo requerido para que el lector lea y entienda el contenido de la gráfica. Es decir que el lector capte el mensaje que se le quiere transmitir. Es importante elaborar gráficas sencillas y rápidas de entender, presentando los elementos necesarios para facilitar su comprensión. Por ejemplo en la siguiente Figura 5-2, se muestran dos gráficas de curvas. La primera de ellas (inciso a) etiqueta cada curva directamente en la gráfica, mientras que la segunda (Inciso b) lo hace mediante una caja de etiquetas a un lado de la gráfica. En el primer caso el lector requiere de un mayor esfuerzo para identificar cada una de las curvas, lo cual se complicaría aún más si aumentara el número de curvas en dicha gráfica.

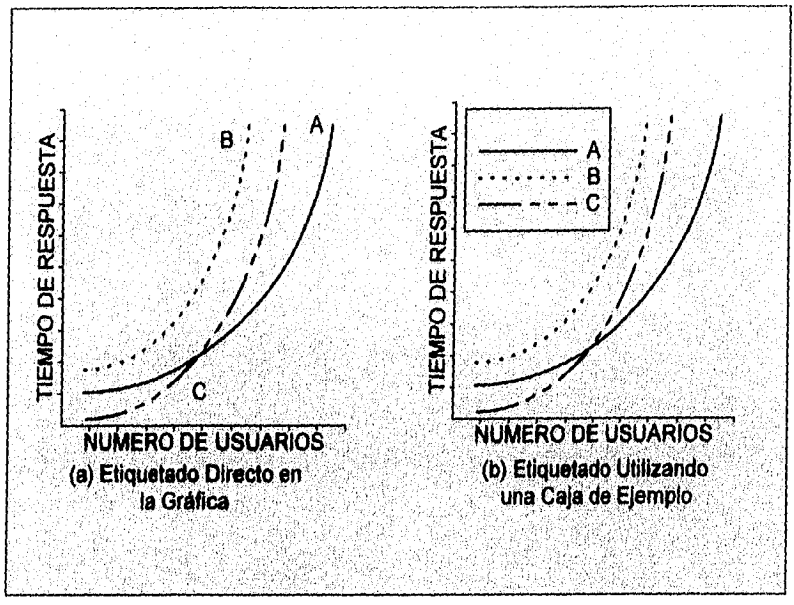


FIGURA 5-2. EJEMPLO DE ETIQUETADO DE GRÁFICAS.

2. *Maximizar la información*: La gráfica debe contener la información suficiente para que puedan ser interpretados los datos que en ella se presenten. Es decir, no debe faltar información, como etiquetas en los ejes incluyendo unidades, y debe omitirse el uso de símbolos que el lector deba recordar o bien consultar en algún otro documento.

3. *Minimizar el uso de tinta*: Presentar la mayor cantidad de información que sea posible, con la menor cantidad de tinta posible. La meta es maximizar la información en el menor espacio posible. Mucha información innecesaria en una gráfica puede hacerla ilegible e inentendible. Por ejemplo, la Figura 5-3 muestra dos gráficas de barras que despliegan la disponibilidad y no disponibilidad de un sistema. La no disponibilidad mostrada en la segunda gráfica (inciso b), es simplemente 1 menos la disponibilidad, dando como resultado una gráfica más informativa que la primera (inciso a).

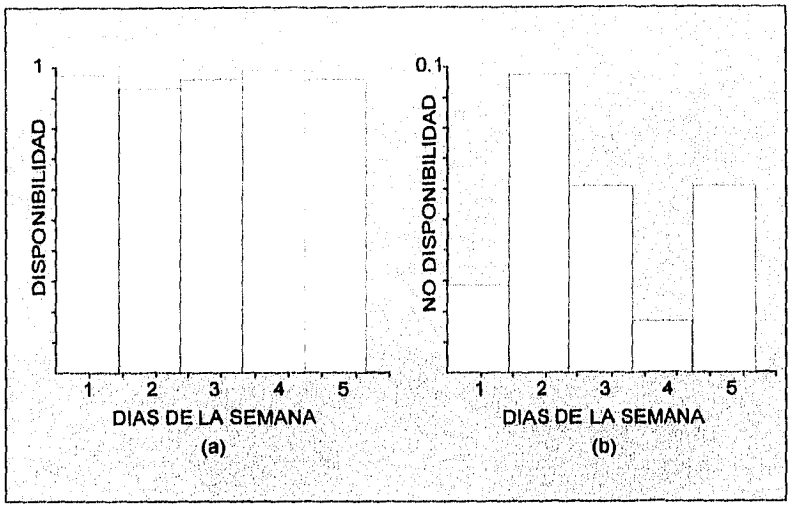


FIGURA 5-3. EJEMPLO DE PRESENTACIÓN DE LA INFORMACIÓN.

4. *Utilizar estándares ya aceptados*: Consiste en presentar la información como el lector la espera. La mayoría esta acostumbrada a que el origen de la gráfica sea en (0,0); que la variable independiente sea graficada en el eje de las abscisas y la dependiente en el de las ordenadas; que las escalas sean lineales y que crezcan de izquierda a derecha y de abajo hacia arriba, etc. El cambiar algunos de estos estándares esta permitido, pero indudablemente requerirá de mayor esfuerzo por parte del lector para interpretar la gráfica. Estos cambios deben hacerse solo cuando sea absolutamente necesario.

5. *Evitar Ambigüedades*: Para evitar ambigüedades, es necesario mostrar claramente las coordenadas de los ejes, las escalas de las divisiones, el origen e identificar individualmente las curvas y barras. La gráfica debe ser fácil de interpretar, por lo que no se deben graficar múltiples variables en la misma gráfica.

Errores comunes en la elaboración de gráficas.

En esta sección, se discuten algunos errores comunes en la elaboración de gráficas dentro de reportes de rendimiento. Es fácil encontrar gráficas con este tipo de errores, hasta en artículos presentados en publicaciones técnicas.

1. *Presentar demasiadas alternativas en una sola gráfica*: El objetivo de la presentación de los datos es facilitar la comprensión de los mismos al lector, por lo cual se deben evitar gráficas con demasiadas curvas, barras y otros

componentes. En forma general una gráfica de lineas, solo debe limitarse a 6 curvas, mientras que una de barras debe limitarse de acuerdo al espacio de graficación.

Esta limitante fue la que propició que el monitor de NQS tuviera algunas gráficas que solo se pueden mostrar mensualmente. En la mayoría de los casos, las gráficas pueden mostrar el comportamiento del sistema de colas en cualquier período de tiempo, desde un día hasta meses o años. Esto no es posible en los casos en los que la variable independiente es el tiempo. Lo anterior debido a que si se escogieran más de 30 días, no habría espacio suficiente para graficarlos.

2. *Presentar muchas variables en el eje de las ordenadas en una sola gráfica*: La Figura 5-4, muestra tres curvas, una para el tiempo de respuesta, otra para la utilización y una más para la velocidad de transferencia. Esta gráfica ahorra espacio, pero deja al lector la tarea de asociar cada curva con su escala apropiada. Presentar 3 gráficas diferentes, una para cada variable, es más claro y lógico en este caso.

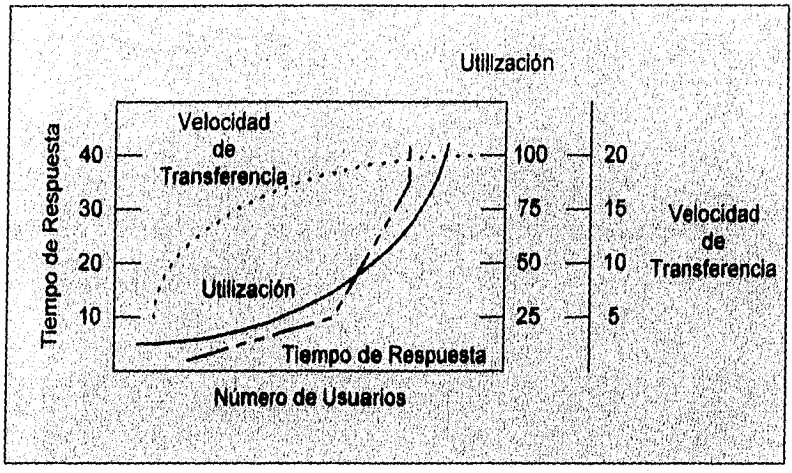


FIGURA 5-4. EJEMPLO DE GRAFICACIÓN DE MÁS DE UNA VARIABLE EN EL EJE DE LAS ORDENADAS.

3. *Utilizar símbolos en lugar de palabras clave*: La Figura 5-5, presenta dos gráficas. La primera de ellas (inciso a) es difícil de interpretar, ya que muestra símbolos sin ninguna explicación, mientras que la segunda (inciso b) utiliza palabras clave en la misma gráfica, facilitando así su interpretación.

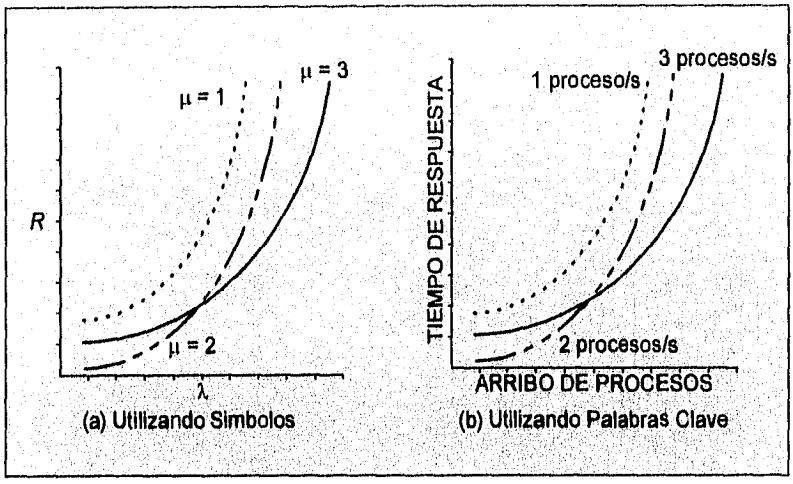


FIGURA 5-5. EJEMPLO DE GRAFICACIÓN UTILIZANDO SÍMBOLOS Y PALABRAS CLAVE.

4. *Colocar información extraña en la gráfica*: El objetivo de cualquier gráfica, es transmitir un mensaje en particular al lector. Cualquier información que represente un obstáculo para que el lector capte el mensaje, se considera como información extraña y debe ser eliminada de la gráfica.

5. *Seleccionar los rangos de las escalas inadecuadamente*: La mayoría de los graficadores disponibles seleccionan automáticamente las escalas de la gráfica basándose en el menor y el mayor de los datos a graficar, aunque en la práctica es muy común ajustar manualmente estos rangos.

6. *Utilizar gráficas de lineas en lugar de gráficas de barras*: Unir con lineas una serie de puntos sucesivos en una gráfica de lineas, significa que los valores intermedios pueden ser aproximados mediante interpolación. La Figura 5-6 muestra un ejemplo de este caso, ya que gráfica el rendimiento medido en MIPS para varios tipos de CPU's. En este caso debió haberse utilizado una gráfica de barras, ya que los valores intermedios entre los tipos de CPU's no tienen significado alguno.

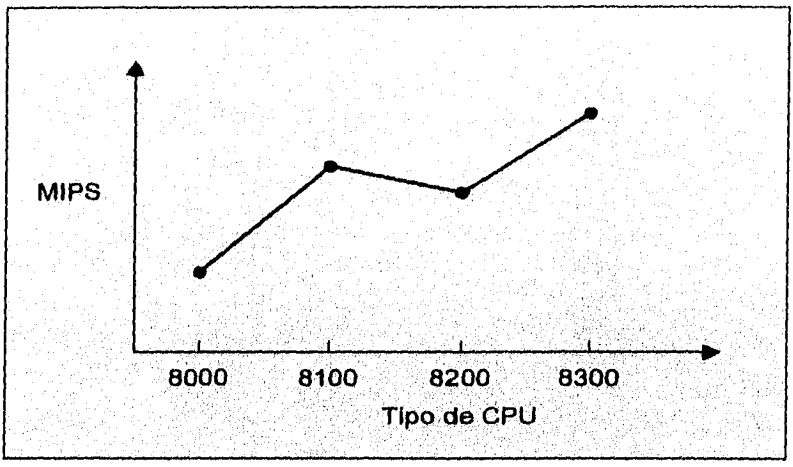


FIGURA 5-6. EJEMPLO DE MALA UTILIZACIÓN DE UNA GRÁFICA DE LINEAS.

6. DESARROLLO DEL MONITOR GRÁFICO DE NQS

Análisis de los Requerimientos del Software
Especificaciones del Diseño

El desarrollo del monitor gráfico de NQS, esta basado en el paradigma del ciclo de vida clásico de la ingeniería del software mostrado en la Figura 6-1. Algunas veces llamado "Modelo en cascada", el paradigma del ciclo de vida clásico, exige un enfoque sistemático, y secuencial del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento.

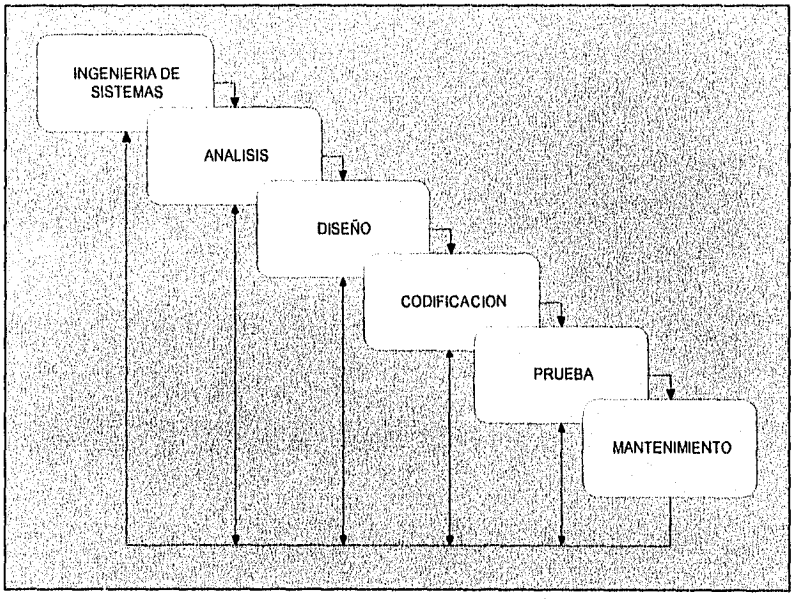


FIGURA 6-1. EL CICLO DE VIDA CLÁSICO.

6.1. ANÁLISIS DE LOS REQUERIMIENTOS DEL SOFTWARE.

6.1.1 INTRODUCCION.

Referencias del Sistema.

A finales de 1991 la Universidad Nacional Autónoma de México a través de la Dirección General de Servicios de Computo Académico, adquirió la supercomputadora CRAY, proporcionando a la comunidad científica universitaria una herramienta poderosa para el desarrollo de sus investigaciones.

A partir de entonces, el departamento de Administración de Supercomputo, ha realizado un gran esfuerzo tanto para satisfacer las necesidades de cada uno de los usuarios, como para difundir la cultura del supercomputo, con el fin de aprovechar al máximo este recurso.

Sin embargo, la extensa gama de aplicaciones que los usuarios ejecutan en la supercomputadora, a la par con el constante crecimiento en la población de usuarios de la misma, ha ocasionado que cada día sea mas complicado para los administradores del sistema, obtener el máximo rendimiento de este recurso. Por lo anterior, es necesario que el grupo de administradores cuente con las herramientas necesarias para poder analizar el comportamiento del sistema y en base a dicho análisis, tome las decisiones adecuadas para obtener el máximo provecho del mismo.

En la actualidad existen algunas herramientas para observar la actividad de la supercomputadora, pero ninguna de ellas se enfoca al análisis del Sistema de Colas NQS. Como sabemos, el Sistema de Colas NQS es parte fundamental en el rendimiento del sistema, ya que la mayoría de los trabajos de los usuarios son atendidos por dicho sistema de colas. De ahí que el propósito de este proyecto es el desarrollo de un monitor gráfico del Sistema de Colas NQS.

Objetivos.

- Desarrollar un monitor de software, que permita mostrar la información referente al uso del Sistema de Colas NQS de la supercomputadora CRAY, mediante la generación de gráficas de barras, en base a datos históricos generados y almacenados por el mismo monitor, de tal forma que estas gráficas puedan servir en la toma de decisiones para la correcta administración del sistema de colas.

- Desarrollar una interfaz gráfica amigable para el administrador del sistema, de tal forma que le sea fácil obtener y manipular la información que necesite.

Requerimientos del proyecto de software.

El monitor de NQS, mostrará la información necesaria para satisfacer los siguientes requerimientos:

- Observar el número de trabajos que se ejecutan en una determinada cola para un período dado.
- Observar el tiempo de espera promedio para una cola dada en un período determinado.
- Observar el número de trabajos que ejecutan los usuarios, en una cola determinada durante un período dado.
- Observar el promedio de memoria utilizada en una cola dada para un día determinado.
- Observar que tan eficientemente utilizan la memoria los usuarios en una cola dada, durante un período determinado.
- Observar el promedio de uso de CPU utilizado en una cola dada para un día determinado.
- Observar el número de trabajos que se ejecutan en un complejo de memoria dado, en un período determinado.
- Observar el número de trabajos que se ejecutan en un complejo de tiempo dado, en un período determinado.
- Observar el número de trabajos, el promedio de memoria utilizada y el promedio de uso de CPU, por un usuario dado en un período determinado.
- Además, el monitor de NQS, deberá satisfacer los siguientes requerimientos de rendimiento:
- Generar los datos históricos acerca del uso del sistema de colas para cada una de las gráficas a desplegar, y almacenar dichos datos de una manera eficiente, evitando así almacenar grandes archivos generados por la contabilidad.
- Procesar los datos para el desplegado de las gráficas, de la manera más óptima posible, con el objeto de generar la menor carga posible al sistema, al momento de utilizarlo.

Escenario de desarrollo

El monitor de NQS, está implementado en la supercomputadora CRAY Y-MP bajo el sistema operativo UNICOS versión 8.0.3.

Restricciones del proyecto de software

- El sistema no contempla la elaboración del graficador, ya que en la actualidad existen graficadores bastante completos que pueden realizar gráficas de barras con las características que se requieren.

- Aunque se tratará de desarrollar el sistema de tal forma que sea lo más portable posible, no se garantiza que el sistema funcione sin ningún cambio en una plataforma diferente al sistema CRAY que utilice el sistema de colas NQS. Sin embargo, los posibles cambios serán mínimos.

- La capa de Interpretación y Administración del monitor de NQS, no realizará funciones automáticas de reconfiguración a los parámetros del sistema de colas NQS. Esta consideración obedece a que en la toma de decisiones para algún cambio de configuración, cuyo objetivo es mejorar el rendimiento del sistema, tiene que ver en gran parte el criterio del administrador del mismo.

- El monitor de NQS no contará con la capa de Consola, debido a que el sistema operativo UNICOS cuenta con el menú del sistema para realizar cambios en la configuración del mismo.

6.1.2 DESCRIPCION DE LA INFORMACION

Para poder cumplir con los requerimientos antes mencionados, el monitor de NQS toma la información del sistema de contabilidad CSA del sistema operativo UNICOS, y como se ha mencionado anteriormente, una vez habilitado dicho sistema, diariamente se genera un archivo llamado *SESSION*, que resume el uso de recursos del sistema. Generalmente este archivo ocupa entre 20 y 40 MB de espacio en disco duro y debido al formato del archivo, y a que no es posible observar los archivos fuente que lo generan, solo es manipulable mediante comandos propios de UNICOS.

Hasta este punto esta es la única fuente de datos que se tiene acerca del uso de recursos del Sistema de Colas NQS. El resto de la información, se derivará del archivo de *SESSION* diario como se describe en el siguiente apartado.

Representación del flujo de la información.

A continuación se listan las transformaciones que deberá sufrir la información a través del sistema, partiendo del archivo *SESSION* diario:

- Del archivo *SESSION*, mediante comandos de UNICOS se generará un reporte en formato ASCII con todo lo referente al uso total del Sistema de Colas NQS.

- Del reporte de uso del Sistema de Colas NQS, se procesará la información, con el objeto de generar y almacenar los datos necesarios, para cada una de las gráficas que satisfacen los requerimientos antes presentados

- Ya generados los datos históricos, la siguiente transformación de la información, se presentará al momento de utilizar la interfaz gráfica para mostrar el comportamiento del sistema.

- Una vez observado el comportamiento del sistema, el administrador del mismo podrá tomar las decisiones correspondientes, con el objeto de mejorar el rendimiento del sistema de colas.

En la Figura 6-2, se muestra el flujo de la información de los datos.

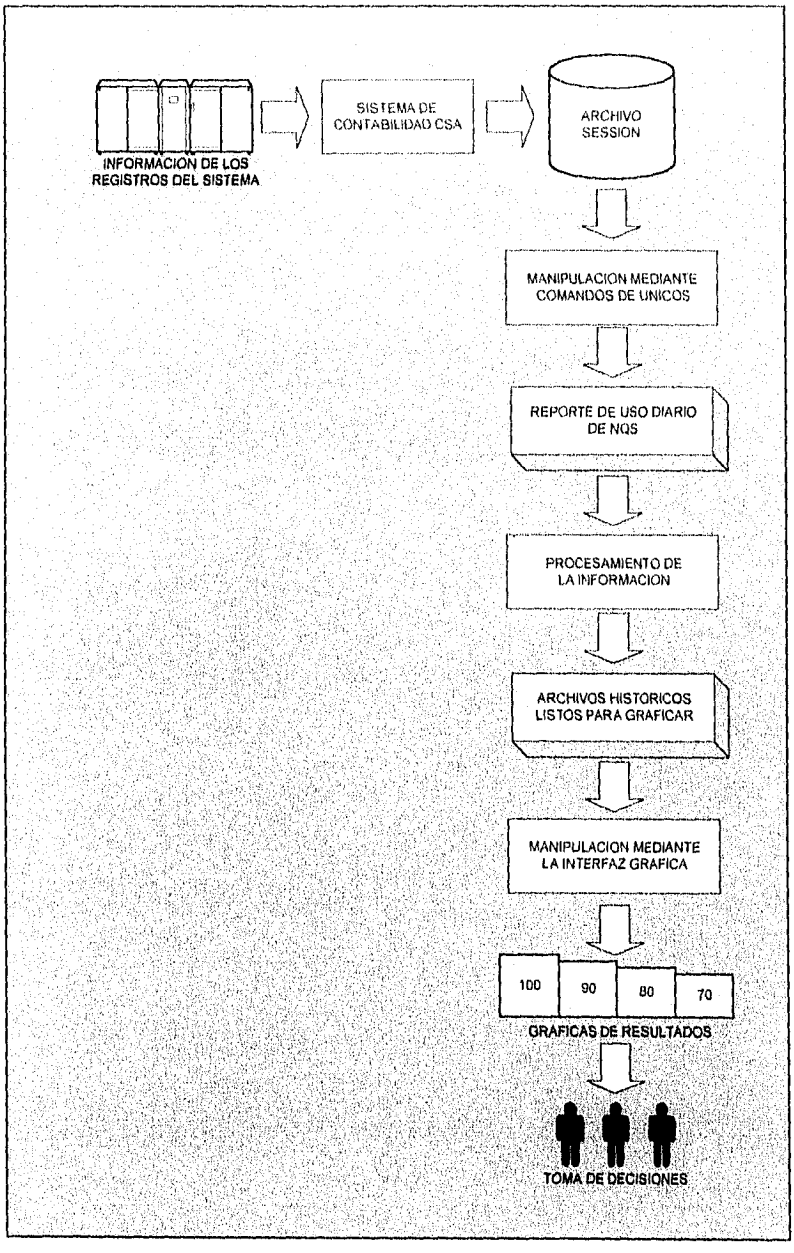


FIGURA 6-2. REPRESENTACIÓN DEL FLUJO DE LA INFORMACIÓN.

6.2. ESPECIFICACIONES DEL DISEÑO

6.2.1 AMBITO

Objetivos del sistema.

A partir de los objetivos planteados en el apartado de análisis de requerimientos, se tomó la decisión de desarrollar un monitor de software en batch, para el Sistema de Colas NQS con las siguientes capas:

- Capa de Observación, que recabe datos sin procesar del uso diario del sistema de colas.

- Capa de Colección, que como su nombre lo dice, colecte los datos del observador, acerca del uso diario del sistema de colas.

- Capa de Análisis, que procese los datos obtenidos por el colector para generar los datos históricos necesarios para obtener 19 tipos de gráficas de barras diferentes, que se explican mas adelante y que satisfacen los requerimientos del sistema. Además esta etapa se encarga de almacenar los archivos de datos de manera eficiente, confiable y con el formato adecuado para que estén listos para graficarse. Esto con el fin de evitar que el monitor de NQS sea una carga de procesamiento adicional al sistema, como se especifica en los requerimientos de rendimiento.

- Capa de Presentación, que a partir de los datos analizados, despliegue los diferentes tipos de gráficas de barras que muestren el comportamiento del Sistema de Colas NQS, utilizando el graficador *xgraph*, desarrollado en la Universidad de Berkeley. Además, esta capa contempla la interfaz gráfica amigable para el administrador del sistema, que facilita la operación del monitor de NQS.

- Capa de Interpretación, y Administración que a partir del conjunto de gráficas seleccionadas permita elaborar conclusiones acerca del comportamiento del sistema de colas y así mismo, tomar decisiones para mejorar su rendimiento.

En la Figura 6-3, se muestra la relación de cada una de las capas del monitor de NQS con la representación del flujo de la información expuesta en la Figura 6-2.

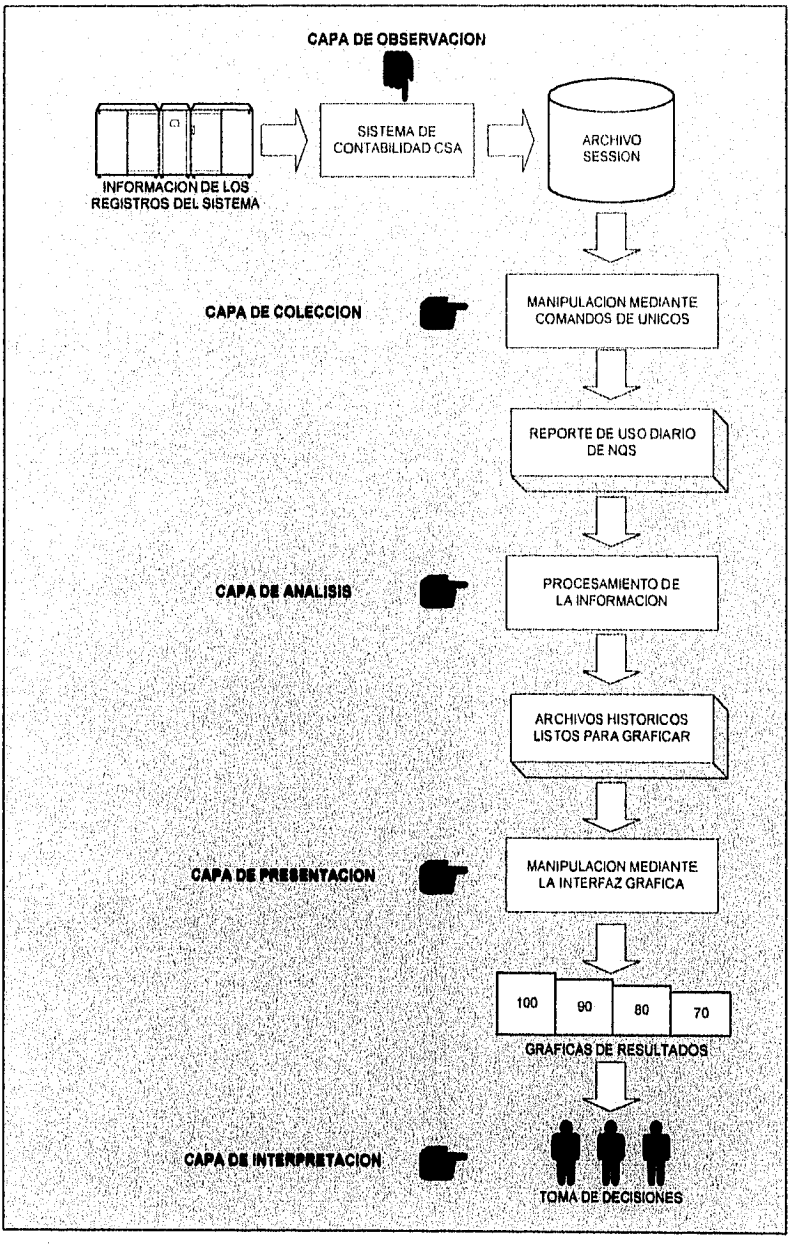


FIGURA 6-3. CAPAS DEL MONITOR DE NQS.

Recursos de Software

De acuerdo con los objetivos del sistema señalados en el apartado anterior, el software seleccionado como ambiente de desarrollo para implementar las capas de colección, análisis y presentación es el siguiente:

- Lenguaje "C" de UNICOS
- Protocolo X-Window con OSF/Motif
- Programación en Bourne Shell de UNICOS
- Programación en awk.

Para el desarrollo de la interfaz gráfica, se eligió utilizar lenguaje "C" junto con el medio ambiente del protocolo X-Window, utilizando las herramientas proporcionadas por OSF/Motif. Las siguientes razones justifican esta decisión:

- X-Window es un ambiente de desarrollo de interfaces gráficas, estándar, abierto y portable.
- X-Window es una interfaz gráfica fácil de utilizar.
- El desarrollo de interfaces gráficas en OSF/Motif se puede llevar a cabo mediante la utilización de lenguaje "C".
- Las utilerías de OSF/Motif incluyen la capacidad de manejo de ratón, creación de menús, de áreas de despliegue, de manejo de ventanas, etc.
- Las estaciones de trabajo, en su mayoría, proporcionan el ambiente X-Window y tienen la posibilidad de manejarlo con las herramientas de OSF/Motif.
- X-Window es un sistema gráfico que esta diseñado para funcionar en red, por lo que da la posibilidad de despliegues remotos con procesamiento local.

Recursos de Hardware

El monitor de NQS propuesto durante la etapa de análisis de requerimientos, está implementado en la supercomputadora CRAY Y-MP, con sistema operativo UNICOS versión 8.0.3.

Como se planteó en el apartado de análisis de requerimientos, el objetivo referente a la realización de una interfaz gráfica amigable para el administrador, exige el apoyo de al menos una computadora con capacidades de procesamiento y despliegue gráfico que se encuentre conectada a la red. Las

estaciones de trabajo son muy populares dentro de ambientes de red, además de que la mayoría de ellas, cuentan con las capacidades gráficas requeridas y con las configuraciones de software de las herramientas utilizadas (UNIX, Motif, TCP/IP).

Recursos Humanos

La realización e implementación de este sistema, requirió de amplios conocimientos en los siguientes temas:

- Programación en lenguaje C de UNICOS
- Programación en Bourne Shell
- Programación en awk
- Ambiente de desarrollo X-Window
- Programación OSF/Motif
- Administración del sistema UNICOS
- Administración del sistema de colas NQS
- Conocimiento del sistema de contabilidad estándar de sistemas UNIX y de la contabilidad propietaria de CRAY, CSA.
- Experiencia en evaluaciones de rendimiento para sistemas UNIX

Principales funciones del software

- Recabar la información correspondiente al uso diario del Sistema de Colas NQS.
- Manipular la información recabada para generar los diferentes tipos de gráficas que satisfacen los requerimientos del software. Así como también almacenar dicha información, de acuerdo a cada uno de los tipos de gráficas propuestos, cuyas características principales se explican en la Tabla 6-1:

TABLA 6-1 GRÁFICAS DELMONITOR DE NQS

NOMRE	DESCRIPCIÓN	EJES
Gráfica 1	Muestra el número de procesos sometidos en cada una de las colas definidas en el sistema para un período dado.	Colas vs No. Procesos
Gráfica 2	Muestra el tiempo de espera promedio que tuvieron los procesos sometidos en cada una de las colas definidas en el sistema para un período dado.	Colas vs Tiempo de Espera Promedio
Gráfica 2a	Gráfica por cola, que muestra el tiempo de espera promedio que tuvieron los procesos sometidos en dicha cola durante un mes determinado.	Días vs Tiempo de Espera Promedio
Gráfica 3	Gráfica por cola, que muestra que usuarios hicieron uso de la cola y cuantos procesos sometió cada usuario en un período dado.	Usuarios vs No. Procesos
Gráfica 4	Muestra el promedio de memoria utilizada que tuvieron los procesos sometidos en cada una de las colas definidas en el sistema para un día determinado.	Colas vs Promedio de Memoria Utilizada
Gráfica 4a	Gráfica por cola, que muestra el promedio de memoria utilizada que tuvieron los procesos sometidos en dicha cola, durante un mes determinado.	Días vs Promedio de Memoria Utilizada
Gráfica 5	Gráfica por cola, que muestra que usuarios hicieron uso de la cola y que tan eficientemente utilizaron la memoria los procesos que sometieron en un período dado.	Usuarios vs Eficiencia de Uso de Memoria

TABLA 6-1 GRÁFICAS DEL MONITOR DE NQS (CONTINUACIÓN)

NOMBRE	DESCRIPCIÓN	EJES
Grfáfica 5a	Gráfica por cola, que muestra que usuarios y cuantas veces hicieron mal uso de la cola en un periodo dado. Considerándose como mal uso de la cola, a aquellos procesos con una eficiencia de uso de memoria menor al 70 %.	Usuarios vs No. veces que usaron mal la cola
Gráfica 6	Muestra el promedio de uso de CPU, que tuvieron los procesos sometidos en cada una de las colas definidas en el sistema para un día determinado.	Colas vs Promedio de Uso de CPU
Gráfica 6a	Gráfica por cola, que muestra el promedio de uso de CPU, que tuvieron los procesos sometidos en dicha cola, durante un mes determinado.	Días vs Promedio de Uso de CPU
Gráfica 7	Muestra el número de procesos sometidos en cada uno de los complejos de memoria definidos en el sistema para un periodo dado.	Complejos de Memoria vs No. Procesos
Gráfica 7a	Gráfica por complejo de memoria, que muestra el número de procesos sometidos en dicho complejo de memoria, para un mes determinado.	Días vs No. Procesos
Gráfica 8	Muestra el número de procesos sometidos en cada uno de los complejos de tiempo definidos en el sistema para un periodo dado.	Complejos de Tiempo vs No. Procesos
Gráfica 8a	Gráfica por complejo de tiempo, que muestra el número de procesos sometidos en dicho complejo de tiempo, para un mes determinado.	Días vs No. Procesos

TABLA 6-1 GRÁFICAS DEL MONITOR DE NQS (CONTINUACIÓN)

NOMBRE	DESCRIPCIÓN	EJES
Gráfica 9	Gráfica por usuario, que muestra el número de procesos que sometió dicho usuario en cada una de las colas definidas en el sistema, para un período dado.	Colas vs No. de Procesos
Gráfica 10	Gráfica por usuario, que muestra que tan eficientemente utilizaron la memoria, los procesos que sometió dicho usuario en cada una de las colas definidas en el sistema, para un período dado.	Colas vs Eficiencia de Uso de Memoria
Gráfica 11	Gráfica por usuario, que muestra el promedio de uso de CPU, de los procesos que sometió dicho usuario en cada una de las colas definidas en el sistema, para un período dado.	Colas vs Promedio de Uso de CPU
Gráfica 12	Muestra la moda del promedio de eficiencia de uso de memoria, que tuvieron los procesos sometidos en cada una de las colas definidas en el sistema para un período dado.	Colas vs Promedio de Eficiencia de Uso de Memoria
Gráfica 13	Muestra la moda del promedio de uso de CPU, que tuvieron los procesos sometidos en cada una de las colas definidas en el sistema para un período dado.	Colas vs Promedio de Uso de CPU

- Mediante la interfaz gráfica, desplegar los tipos de gráficas requeridos por el administrador.
- Almacenar en memoria secundaria alguna gráfica determinada, generada por la interfaz gráfica del monitor.
- Desplegar una gráfica determinada, previamente generada por la interfaz gráfica del monitor.

6.2.2 DESCRIPCION DEL DISEÑO

Descripción Funcional de las Capas de Colección y Análisis.

La operación del monitor de NQS, inicia diariamente a las 23:30 hrs. a través de la herramienta *cron*, mediante la cual se ejecuta un programa en ***shell*** que realiza las siguientes actividades:

1. La capa de Colección, obtiene un reporte en formato ASCII, del uso del sistema de colas para el día actual, a partir del archivo *SESSION* generado por el sistema de contabilidad CSA como se muestra en la Figura 6-4:

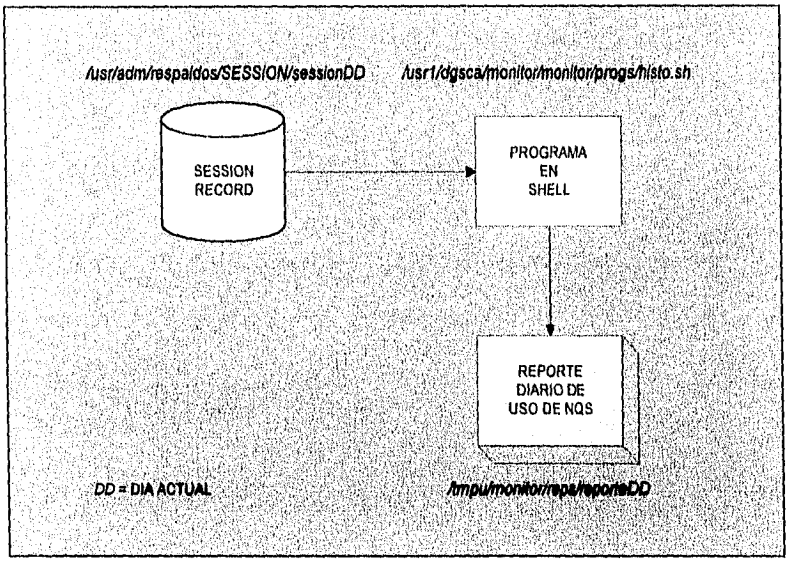


FIGURA 6-4. REPORTE DIARIO DE USO DE NQS.

2. Del reporte de uso diario de NQS generado en la capa de Colección, inicia la capa de Análisis, obteniendo la información acerca del uso de cada una de las colas definidas en el sistema, generando así un archivo para cada una de ellas, como se muestra en la Figura 6-5.

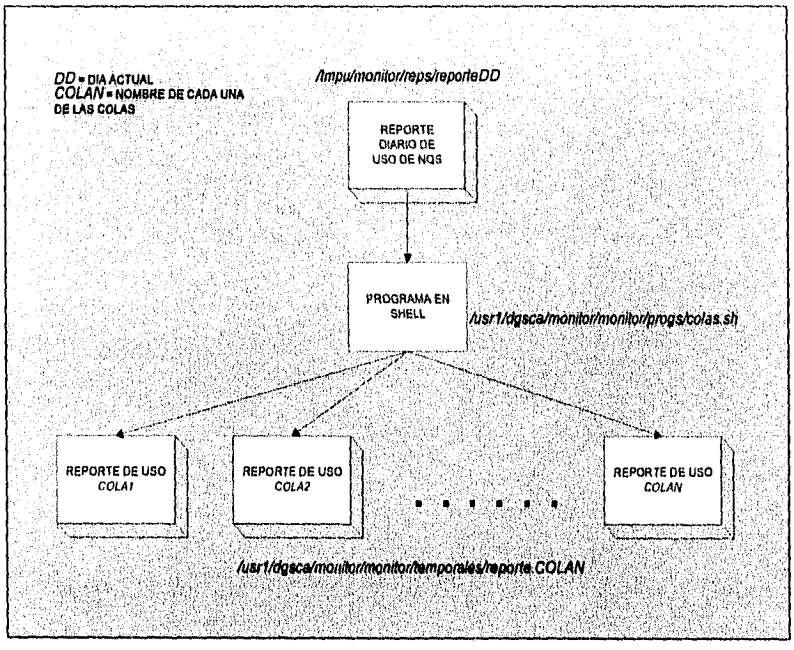


FIGURA 6-5. REPORTE PARA CADA COLA DEFINIDA EN EL SISTEMA.

3. De los reportes de uso para cada cola, procesa la información de acuerdo a cada tipo de gráfica y genera los archivos correspondientes a cada una de ellas, como se ilustra en la Figura 6-6.

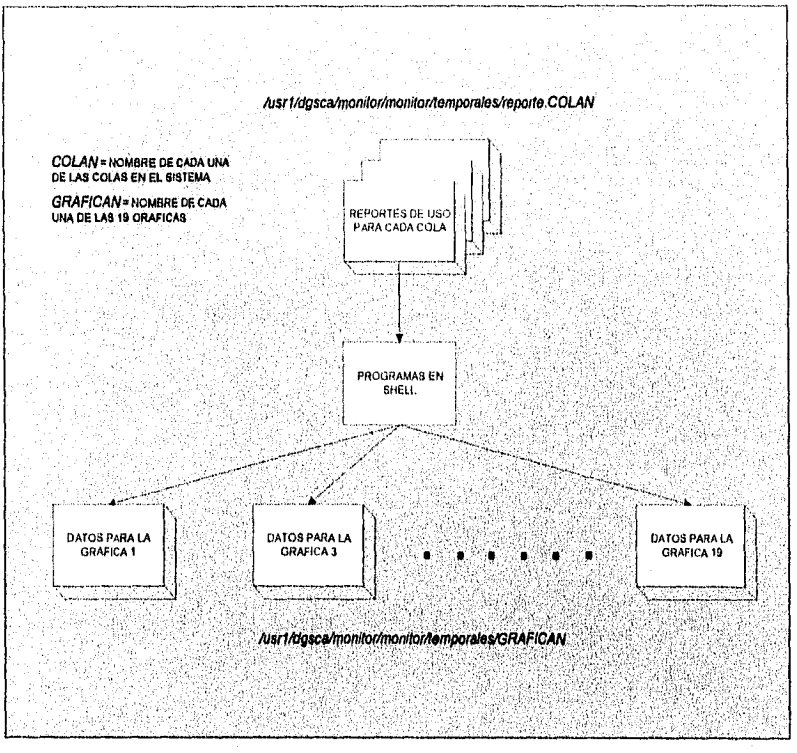


FIGURA 6-6. DATOS PARA CADA UNA DE LAS GRÁFICAS.

4. Finalmente, los datos generados para cada una de las gráficas, son convertidos al formato adecuado para el graficador *xgraph*, quedando listos para ser graficados y almacenados, de acuerdo a la fecha del día actual, y al tipo de gráfica como se muestra en la estructura de directorios de la Figura 6-7. En este punto es donde finaliza la capa de Análisis del monitor.

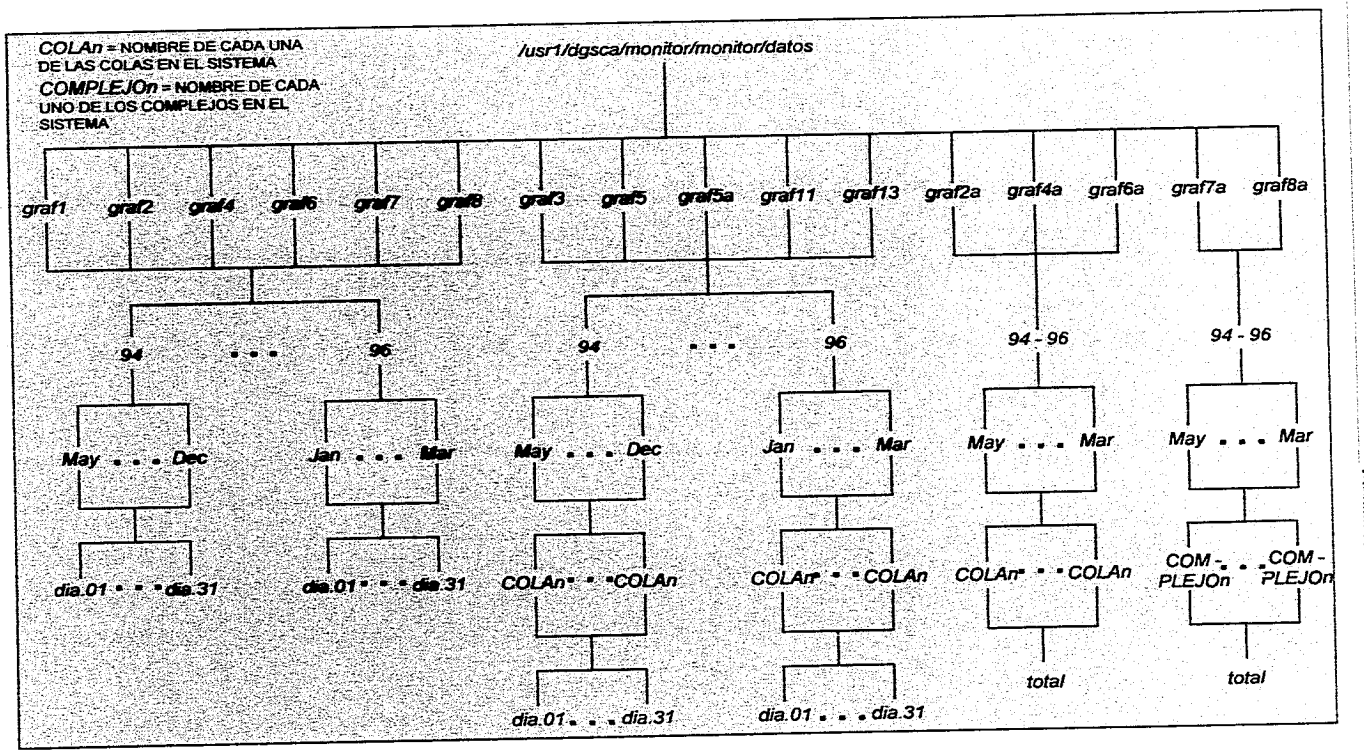


FIGURA 6-7. ESTRUCUTRA DE DIRECTORIOS DE LOS DATOS A GRAFICAR.

Cabe mencionar que solo se generan archivos históricos para 16 de las 19 graficas que componen el monitor. Lo anterior debido a que las graficas 9, 10 y 12 se generan a partir de los datos almacenados para las graficas 3, 5 y 5 respectivamente.

Descripción Funcional de la Capa de Presentación: Interfaz Gráfica.

Esta capa utiliza la información generada por la capa de análisis y la convierte a graficas. Para lo anterior, esta capa necesita interactuar con el usuario, pues es necesario que éste le proporcione parámetros de graficación tales como: periodo de análisis, cola, complejo o usuario a analizar y gráfica a desplegar. Con la información anterior, esta capa selecciona y procesa los archivos necesarios para alimentar al graficador *xgraph* y lo ejecuta.

Como ejecutar el Monitor Gráfico de NQS

Para ejecutar el Monitor Gráfico de NQS es necesario iniciar una sesión remota en CRAY desde una estación de trabajo. Los requerimientos para que el despliegue pueda ralizarse son los siguientes:

- El servidor de despliegue, es decir la estación de trabajo debe permitir el despliegue desde sirio. Para lo anterior se ejecuta el comando mostrado en la Figura 6-8.

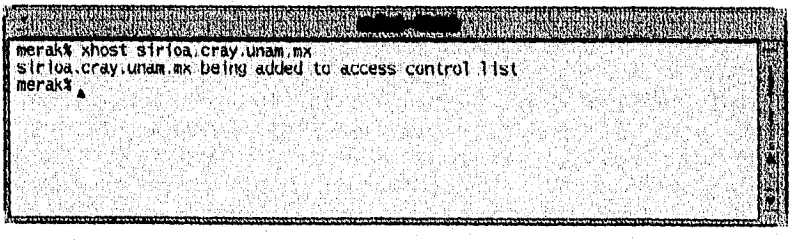


FIGURA 6-8. HABILITACION PARA ACEPTAR DESPLIEGUES DESDE SIRIO.

- Una vez permitido el despliegue se realiza la conexión a sirio y se debe establecer cuál será el servidor de despliegue. Para lo anterior se ejecutan los comandos mostrados en la Figura 6-9.

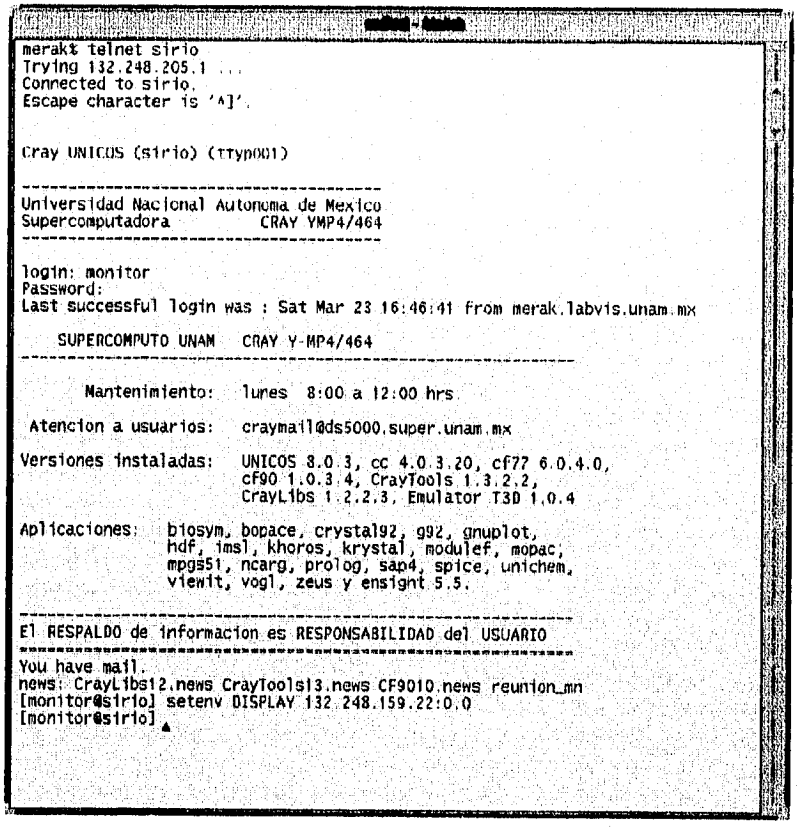


FIGURA 6-9. CONEXIÓN A SIRIO Y ESTABLECIMIENTO DE DESPLIEGUE.

El nombre con el que se ejecuta el Monitor es Monitor_NQS, con lo que se despliega la ventana principal.

Descripción de las ventanas que componen el monitor

La ventana principal cuenta con: la Barra de Menú, el Area de Información y el Botón de Graficar como se muestra en la Figura 6-10.

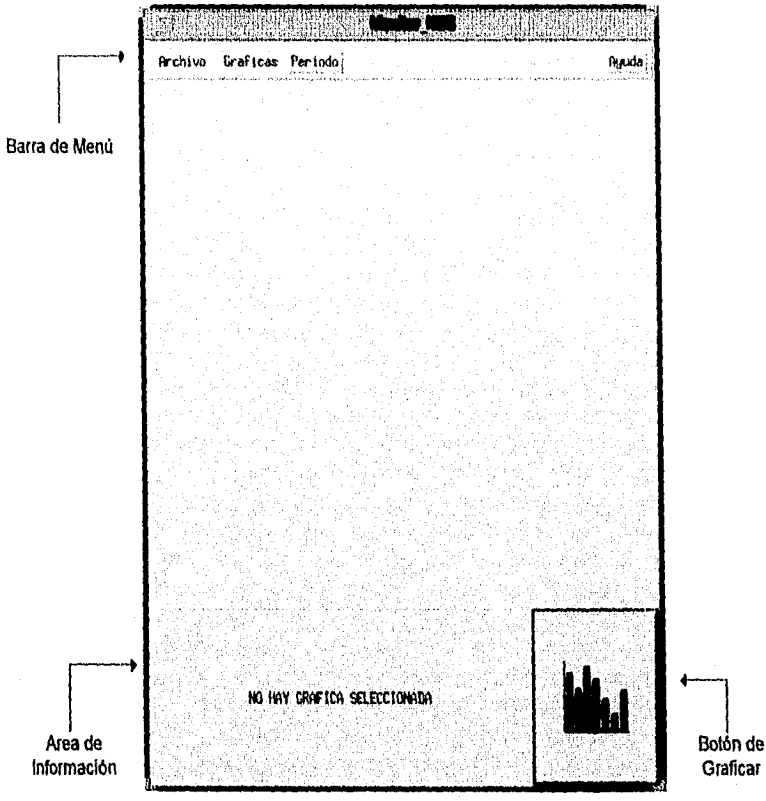


FIGURA 6-10. DESCRIPCIÓN DE LA VENTANA PRINCIPAL.

La Barra de Menú, presenta 4 opciones: Archivo, Graficar, Periodo y Ayuda que se describen a continuación:

- Archivo.- Permite que el usuario cargue o almacene un archivo generado por el monitor. También se utiliza cuando el usuario desea salir del sistema. Las opciones del menú de archivo se muestran en la Figura 6-11.

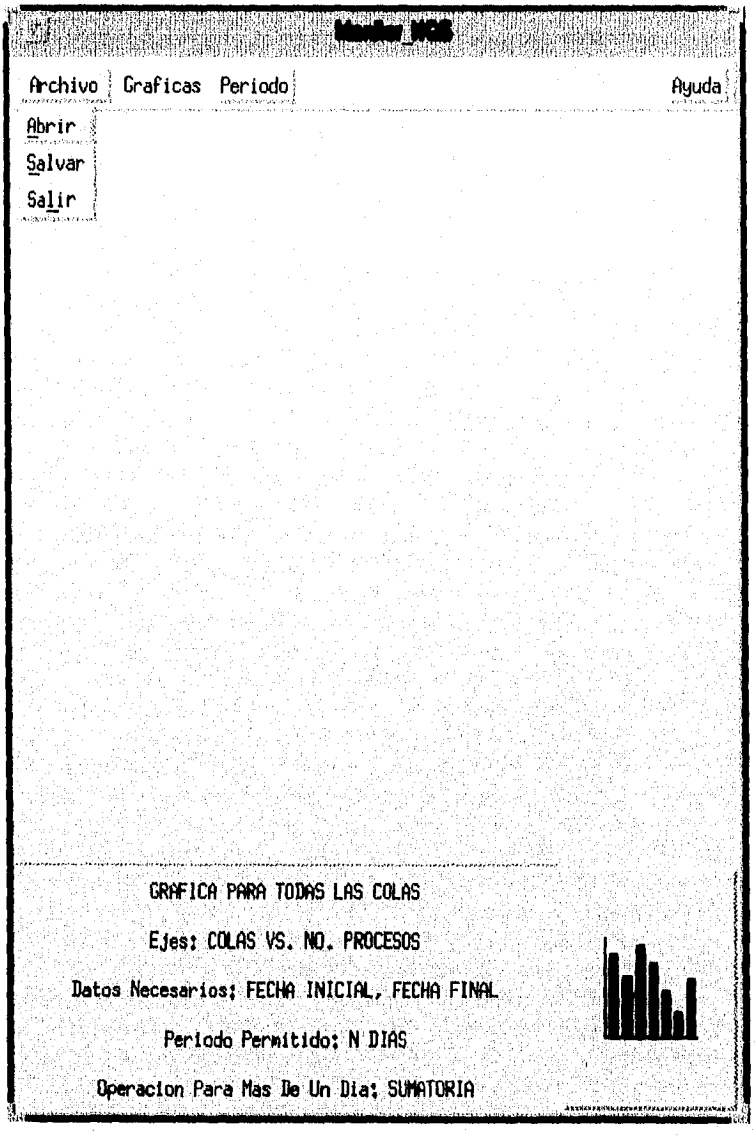


FIGURA 6-11. OPCIONES DEL MENU ARCHIVO.

Abrir.- Esta opción permite cargar los archivos necesarios para desplegar una gráfica previamente generada por el monitor, la cuál será mostrada al momento de oprimir el Botón de Graficar. El archivo que debe ser cargado

es aquel generado con la extension ".cmd", que a su vez llamará al archivo de datos del mismo nombre pero sin extensión, y desplegará la gráfica correspondiente.

Al seleccionar la opción de "Abrir archivo" del menú de "Archivo", se despliega la ventana que se muestra a continuación en la Figura 6-12.

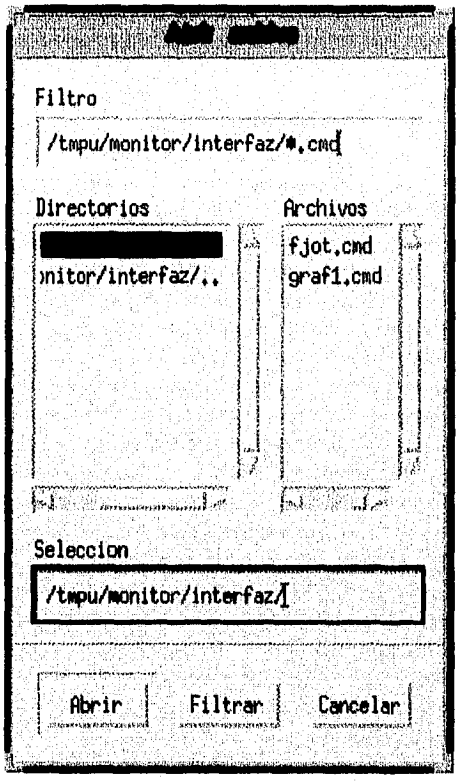


FIGURA 6-12. VENTANA PARA ABRIR ARCHIVOS.

Como puede observarse en la Figura 6-12, esta ventana proporciona la facilidad de seleccionar el directorio y el archivo que desea cargar. Así mismo permite filtrar los tipos de archivos que mostrará o teclear directamente la ruta y el nombre del archivo deseado.

Salvar.- Esta opción permite salvar los datos con lo que se generó la última gráfica, para ello almacenará dos archivos, uno con extension ".cmd" y otro con el mismo nombre pero sin extensión. El primero de ellos, sirve para almacenar información referente a la apariencia de la gráfica, como son las etiquetas de los ejes y el título de la gráfica. El segundo archivo, sin extensión, contiene únicamente los datos de la gráfica. Para que estos dos archivos sean almacenados, solo es necesario proporcionar el nombre sin extensión y automáticamente el monitor generará ambos archivos. De esta forma cuando se requiera desplegar la gráfica ya no será necesario indicar los parámetros de graficación ni realizar ningún procesamiento de archivos.

Esta ventana proporciona la facilidad de seleccionar el directorio deseado, mostrando el contenido del mismo. En la Figura 6-13 se muestra la ventana para salvar archivos.

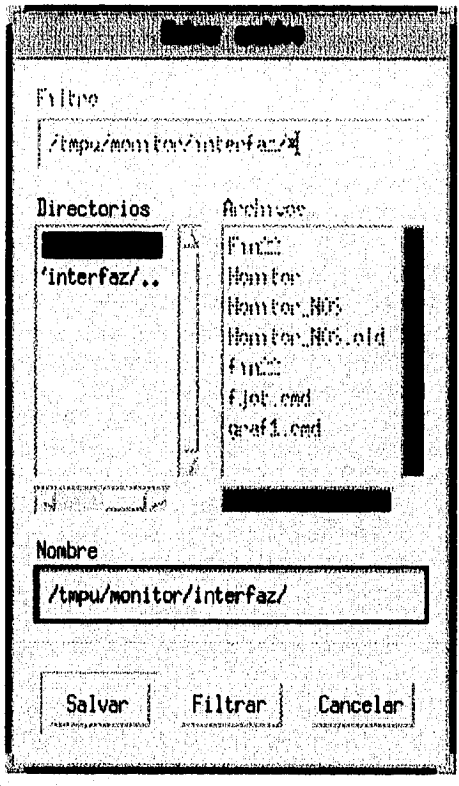


FIGURA 6-13. VENTANA PARA SALVAR ARCHIVOS.

Salir.- Esta opción permite terminar el programa.

- Graficas.- Sirve para elegir el tipo de gráfica que se desea generar.

  Al seleccionar esta opción se abre una ventana *pulldown* la cual indica qué es lo que se va a graficar en caso de ser seleccionado ese renglón. Los renglones son mutuamente excluyentes, lo cual significa que solo se puede generar una gráfica a la vez. Pero una vez generadas, se pueden tener varias gráficas abiertas.

  La Figura 6-14, muestra el menú pulldown de la opción "Graficas" de la barra de menú.

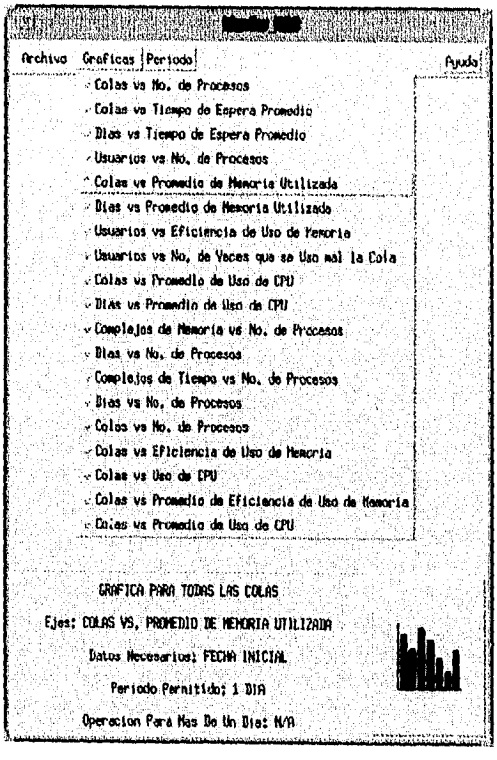


FIGURA 6-14. TIPOS DE GRAFICAS.

Para cada tipo de gráfica, existen ciertos parámetros de graficación, los cuales son mostrados en el área de "Información".

- Periodo.- Permite seleccionar el periodo a analizar. Al seleccionar esta opción se abre la ventana de "Periodo" en la cual se puede elegir año, mes y día para cada una de las 2 fechas que delimitan un periodo. Ambas fechas se muestran en esta ventana y se actualizan de acuerdo a la fecha seleccionada. La ventana "Periodo" se muestra en la Figura 6-15.

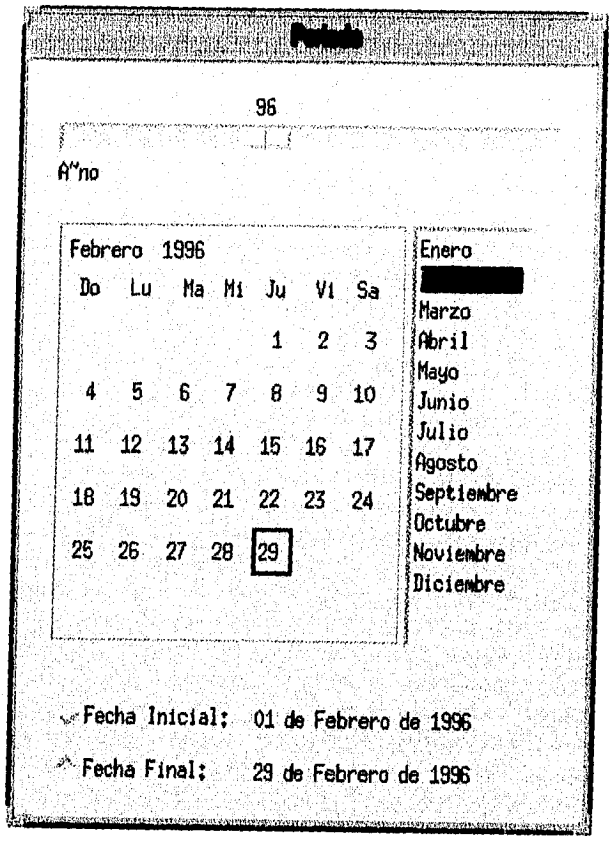


FIGURA 6-15. SELECCIÓN DEL PERIODO A GRAFICAR.

- Ayuda.- El Monitor cuenta con dos niveles de ayuda, el primer nivel es de ayuda rápida la cual se muestra el área de información. El segundo nivel de ayuda es más detallado y se muestra oprimiendo el botón de "Ayuda" de la barra de menú. En la Figura 6-16 se muestra un ejemplo de la ayuda.

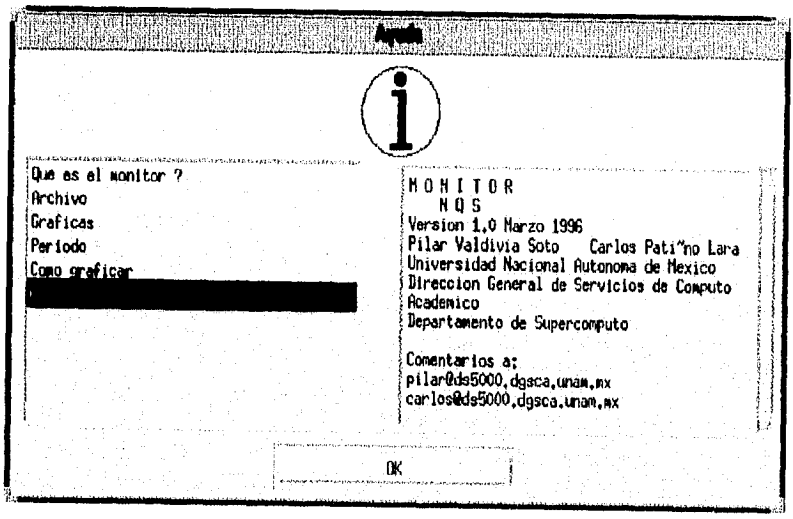


FIGURA 6-16. VENTANA DE AYUDA.

La ventana de "Ayuda" se compone de dos secciones una en la que se selecciona el tema del cual se requiere la información y la otra en la que dicha información es mostrada.

- Area de Información.- Se encuentra en la parte inferior de la ventana principal como se muestra en la Figura 6-10 y proporciona información adicional a la ventana pulldown asociada a la opción "Graficas" de la barra de menú. Ya que como se mencionó, al irse desplazando por los renglones de dicha ventana, ésta solo indica qué información se va a graficar, pero es necesario conocer algunos otros detalles como: si se trata de una gráfica por cola, por complejo, por usuario o general, o si es posible graficar cualquier periodo, etc. Estos parámetros de graficación son mostrados en el "Area de Información" y son descritos a continuación:

  - Gráfica general, por cola, por complejo o por usuario.

  - Ejes de la gráfica.- Qué información se presentará en el eje de las abscisas y qué en el eje de las ordenadas.

  - Datos Necesarios.- Qué parámetros adicionales hay que proporcionarle al Monitor para que pueda ser generada y desplegada la gráfica. Por ejemplo: fecha inicial, fecha final y cola. En ocasiones hay gráficas que se generan mensualmente por lo que solo basta proporcionar la fecha inicial, de la cual se tomará el mes a graficar. O puede ser que la gráfica

se genere por usuario, por lo que es necesario proporcionar la clave del usuario que se desea analizar.

- Periodo Permitido.- Existen gráficas que pueden ser generadas para n dias, o gráficas que solo pueden generarse para un día o para un mes.

- Operación para más de un día.- Cuando el periodo a graficar es mayor de un día es necesario que el usuario sepa si obtendrá sumatoria, promedio o moda.

- Botón de Graficar.- A la derecha del area de Información, se encuentra el Botón de Graficar como se muestra en la Figura 6-10. Mediante este botón, se le indica al monitor que genere y despliegue una gráfica, siempre y cuando los parámetros de graficación necesarios hayan sido establecidos o algún archivo haya sido cargado.

- Ventanas Adicionales.- Mediante estas ventanas el usuario indica al Monitor algunos parámetros de graficación que se requieren dependiendo de la gráfica que se desee mostrar.

Colas Existentes.- La Figura 6-17 muestra la ventana de selección de colas existentes, para que el usuario eliga cuál de ellas analizará.

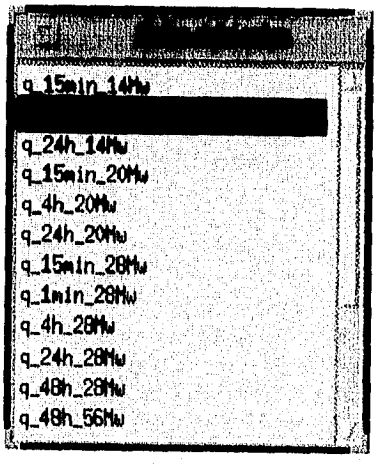


FIGURA 6-17. VENTANA DE SELECCIÓN DE COLAS.

Complejos de Tiempo.- La Figura 6-18 muestra la ventana de selección de complejos de tiempo existentes, para que el usuario eliga cuál de ellos analizará.

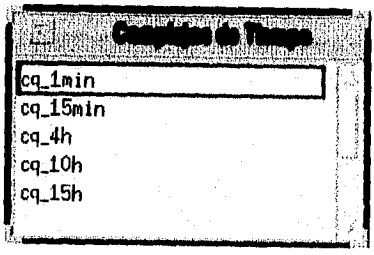


FIGURA 6-18. VENTANA DE SELECCIÓN DE COMPLEJOS DE TIEMPO.

Complejos de Memoria.- La Figura 6-19 muestra la vcentana de selección de complejos de memoria existentes para que el usuario eliga cuál de ellos analizará.

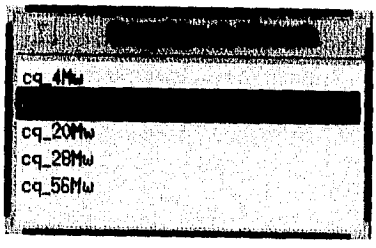


FIGURA 6-19. VENTANA DE SELECCIÓN DE COMPLEJOS DE MEMORIA.

Usuario.- Permite que el administrador, capture la clave del usuario que desea analizar. La Figura 6-20 muestra la ventana de captura de usuario.

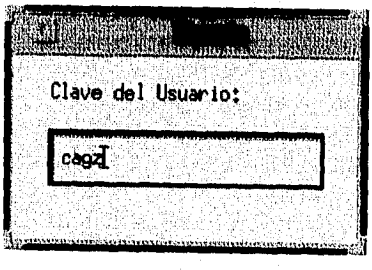


FIGURA 6-20. VENTANA DE CAPTURA DE USUARIO.

Descripción de las ventanas de gráficas generadas por el Monitor.

Estas ventanas muestran gráficas de barras, las cuales se muestran en diferentes colores y texturas para poder ser fácilmente distinguidas entre sí. Cada barra esta etiquetada de acuerdo a lo que se este graficando, por ejemplo puede representar una clave de usuario, un dia del mes, una cola, un complejo de memoria o un complejo de tiempo.

Cada gráfica muestra información adicional, dependiendo de lo que se está graficando, así como también indica el periodo que abarca cada gráfica generada.

Así mismo se muestra tanto en el eje de las ordenadas como en el de las abscisas el nombre y las unidades de la información graficada. La Figura 6-21, muestra un ejemplo de una gráfica generada por el monitor.

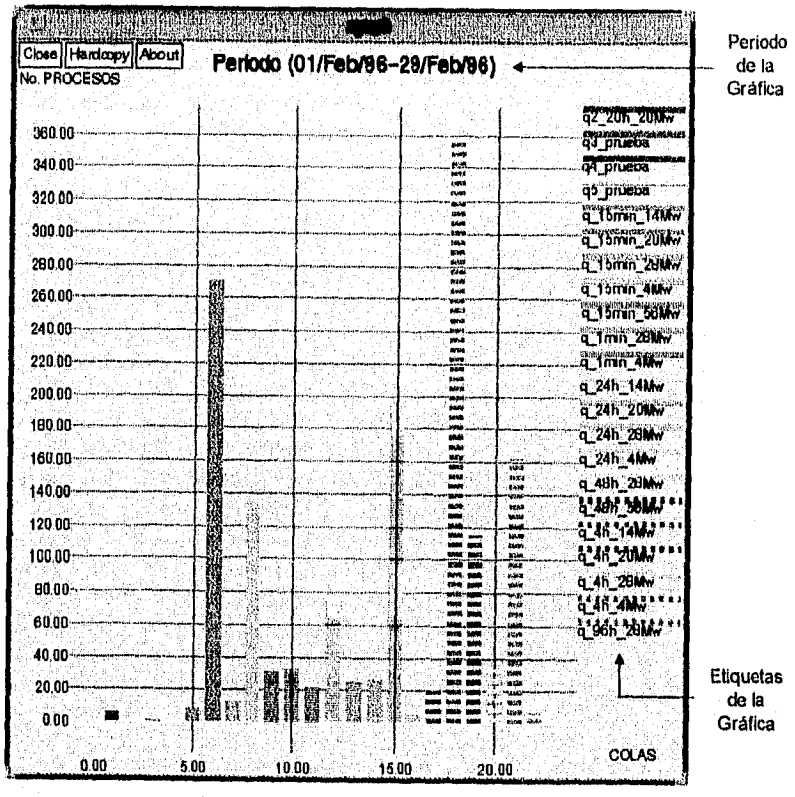


FIGURA 6-21. EJEMPLO DE GRÁFICA GENERADA POR EL MONITOR DE NQS.

CONCLUSIONES

Uno de los aspectos más importantes en la realización de este trabajo es el hecho de que se llevaron a cabo actividades en las que se pudieron aplicar diversos conocimientos. Antes de definir los alcances del proyecto se realizaron estudios de rendimiento y se desarrollaron algunos monitores menos sofisticados que fueron antecedentes al Monitor de NQS. Para lograr el objetivo propuesto se requirió del estudio de la administración y la contabilidad del sistema, hubo que realizar tareas de administración para recopilar la información requerida, se aplicó ingeniería de software, conocimientos de sistemas operativos, de rendimiento de sistemas, se realizó la programación de más de 40 programas entre shells, programas en awk y programas en C, estudio de teoría de monitores y desarrollo en ambientes gráficos entre otras cosas.

Otro punto de interés es que este trabajo de tesis en sus capítulos 2 y 5, sirve como base para el estudio del rendimiento de cualquier sistema. Si bien todo el trabajo se enfoca a la supercomputadora CRAY, las bases son las mismas para cualquier sistema, y dado que no existe mucha literatura acerca de este tema tan importante, es un buen punto de partida para iniciarse en el estudio y afinación de sistemas.

A lo largo del estudio de rendimiento que se realizó, se observó que el mantener a un sistema en su punto óptimo es algo que depende de diversos factores como se ha mencionado en los capítulos que a ello se refieren. Ya que es necesario tener un conocimiento muy profundo del sistema, su arquitectura, sus aplicaciones y sus usuarios. Lo cual es aún más difícil cuando las aplicaciones y los usuarios cambian con cierta frecuencia. En instituciones académicas la mayoría de los sistemas caen en este caso, ya que se busca explotar al máximo una inversión, es decir si se compró un equipo determinado, se pretende que pueda ser utilizado para la solución de diferentes problemas y aplicaciones. A diferencia de otros lugares en los que se busca la especialización, y en los que es posible tener un equipo dedicado para cada aplicación y con características especiales para ello. En ocasiones, quizá por falta de recursos económicos se intenta darle a los equipos versatilidad y por ende la administración y afinación de los mismos no puede ser la misma que se aplica bajo otras circunstancias.

Por lo regular, durante la evaluación y afinación de un sistema, es necesario realizar un análisis, desarrollar una hipótesis y hacer pruebas, y repetir estos pasos hasta encontrar puntos críticos, ya que no existen reglas o pasos a seguir que funcionen para cualquier equipo y que garanticen la afinación del mismo. De ahí que a lo largo de todo el trabajo de tesis, se recalque la importancia que tiene la experiencia del administrador y la información que éste pueda tener.

Además una recomendación sería utilizar el sistema como usuario no solo como administrador, ya que muchos usuarios se quejan por tiempos que realmente de acuerdo a su aplicación y a la carga del sistema son buenos, o algunos otros no se quejan hasta que el problema ya es muy grande.

A lo largo de este trabajo se ha mencionado que es primordial el contar con las herramientas adecuadas para el monitoreo de un sistema en particular, más aún cuando se trata de una Supercomputadora. Este proyecto se realizó para dar solución a un problema de rendimiento debido a que la compañía CRAY Research Inc. no proporcionó las herramientas de monitoreo necesarias para observar en su totalidad el comportamiento del sistema. La falta de un monitor del Sistema de Colas adecuado desde la puesta en operación de la Supercomputadora, provocó que en los primeros años, no se utilizaran sus recursos al máximo. Lo anterior debido a que los administradores empezaban a conocer la máquina y con las herramientas existentes era muy difícil observar el comportamiento del sistema NQS.

Como se mencionó en el capítulo 5, el estudio del rendimiento empieza desde el diseño de un sistema y desde que se ha pensado en adquirir un equipo. Cuando se realizó la compra de la CRAY, se pensó que el sistema ya contaba con las herramientas necesarias para evaluación de rendimiento y afinación, pero algunas de ellas no son prácticas o no existen. Es fundamental en la adquisición de un equipo tan costoso no siempre creerle al vendedor y realizar un análisis y revisión de todas las herramientas proporcionadas por el vendedor en la etapa de preventa y no en la de postventa. La compañía CRAY Research Inc., debería incluir dentro de sus políticas de venta de Supercomputadoras, la inclusión de un conjunto de herramientas de monitoreo, capaz de observar en su totalidad el comportamiento de sus sistemas.

En general, antes de tomar la decisión de realizar alguna herramienta, es aconsejable utilizar lo que ya se tiene, combinando la información que proporcionan las herramientas ya existentes. Cuando se han agotado al máximo estos recursos, entonces construir una herramienta propia pero no intentar realizarla toda, sino solo lo que falta para complementar lo ya existente.

En el caso del estudio del Sistema de Colas NQS, no existía herramienta alguna, sólo se contaba con la información arrojada por la contabilidad. El monitor de NQS es el primer sistema realizado para el análisis del Sistema de Colas NQS, no existe ninguna herramienta similar. La selección de la información presentada, el diseño de la interfaz gráfica y las gráficas presentadas son originales y no se trata de ninguna adaptación a algo ya existente. Este sistema se encuentra ya en producción y es la manera más fácil de analizar al Sistema de Colas NQS de manera rápida y confiable.

En cuanto a los resultados obtenidos, se puede concluir que el monitor de NQS desarrollado, cumple con las expectativas y requerimientos establecidos, tanto

en el objetivo del tema de tesis como en la teoría de monitores presentada en la metodología de análisis de rendimiento para sistemas computacionales.

Finalmente el objetivo de la elaboración de este trabajo de tesis, que surgió como una necesidad para la resolución de un problema, se fue convirtiendo durante el desarrollo del mismo, en una necesidad para los autores de concluir satisfactoriamente, con un proyecto que permitió consolidar los conocimientos adquiridos y que constituye una aportación de gran importancia para su desarrollo profesional.

APÉNDICE

Caso Práctico de Uso del Monitor de NQS

El objetivo de este apéndice, es el de mostrar el uso del monitor de NQS, mediante una sesión típica para observar el comportamiento del sistema de colas durante un período determinado.

Para no perder la continuidad durante el caso práctico de uso del monitor, y facilitar la visualización de las gráficas que se mencionan, éstas se encuentran al final del Apéndice, ordenadas de acuerdo al número de cada una de ellas.

El caso práctico del monitor de NQS que aquí se presenta, proporciona información acerca del uso del sistema de colas para los siguientes tres rubros.

- Utilización de memoria
- Utilización de CPU
- Utilización de complejos de memoria y complejos de tiempo

Es importante señalar, que en la primera parte donde se analiza la utilización de la memoria, se incluye el análisis de aspectos generales de utilización del sistema de colas, como son: el número de procesos que se ejecutaron durante el periodo, tiempos de espera promedio de las colas y análisis de utilización del sistema para un usuario en particular.

Para este caso práctico, se tomó como periodo de análisis el mes de Febrero de 1996, por lo cual para generar las gráficas presentadas, se eligió como fecha inicial el día 1ro. de Febrero de 1996 y fecha final el 29 de Febrero del mismo año. Es importante señalar, que la numeración de las gráficas que se presentan, coincide con la definición de las gráficas mostrada en la Tabla 6-1 del capitulo correspondiente al Desarrollo del Sistema.

Para iniciar el análisis de la utilización de memoria, la Gráfica 1 es un buen punto de partida ya que muestra el número total de procesos que fueron sometidos al sistema de colas NQS. Como se puede observar en esta gráfica, la cola que más demanda tuvo fue la de 4h y 14 Mw ya que muestra 359 trabajos durante los 30 días. Partiendo del resultado de la Gráfica 1, es posible analizar la cola que más demanda tuvo durante el periodo.

VER GRÁFICA 1

La Gráfica 2 muestra el tiempo de espera promedio que tuvieron cada una de las colas durante el periodo. Esta gráfica sirve para observar si por ejemplo la cola que mas se utilizó tuvo o no grandes tiempos de espera. Es importante señalar que como se trata de un periodo mayor a un día los valores graficados son la moda para cada una de las colas. En esta gráfica destacan los aproximadamente 18,500 segundos (aprox. 5hrs) de espera que tuvieron los trabajos sometidos en la cola 24hrs y 28 Mw, lo que quiere decir que el tiempo de espera promedio más frecuente durante el periodo, osciló en este valor. Cabe mencionar, que aunque esta cola registró tiempos de espera muy altos, es de las que menos se utilizó durante el periodo. Por el contrario la cola más popular, q_4h_14Mw registra uno de los tiempos de espera promedio más bajos de aproximadamente 1000 segundos (aprox. 16 min.).

VER GRÁFICA 2

Siguiendo con el análisis de la cola más popular, en la Gráfica 2a, se muestra el tiempo de espera promedio que tuvo la cola q_4h_14Mw en cada uno de los días del mes. Como se puede apreciar a pesar de que en la Gráfica 2 se registró un tiempo de espera promedio aproximado a los 1000 segundos durante todo el mes, en la Gráfica 2a hay días en los que el tiempo de espera promedio estuvo bastante más alto. Esta gráfica ayuda al administrador a determinar que días la cola tuvo tiempos de espera altos y en base a ello estudiar mas a fondo esos días, para poder determinar cuál fue la causa de dichos tiempos de espera.

VER GRÁFICA 2a

Hasta este punto, sabemos que la cola q_4h_14Mw tuvo días con tiempos de espera altos, así que a continuación se analizará que usuarios utilizaron la cola y posteriormente que tan bien o mal la utilizaron. La Gráfica 3 muestra el número de procesos por usuario sometidos en la cola de 4h y 14Mw. Como se puede observar los usuarios que mas procesos sometieron son *wsh*, *cagz* y *jacr*.

VER GRÁFICA 3

Los datos presentados en la Gráfica 3 se pueden comparar con los de la Gráfica 5, en la que se muestra que tan eficientemente los usuarios utilizaron la memoria en la cola de 4h y 14Mw. Como se puede observar en la Gráfica 5 el usuario *wsh* quien fue el que sometió más procesos a la cola durante el periodo, es el que tiene uno de los promedios de eficiencia de uso de memoria más bajos (menos del 10%). En este caso valdría la pena recomendarle al usuario *wsh* que utilice la cola de 4h y 4Mw ya que en cada uno de los 59 procesos que sometió durante el periodo, desperdicio más de 10 Mw de memoria.

VER GRÁFICA 5

Continuando con el análisis de la memoria, la Gráfica 5a nos dice cuántas veces los usuarios hicieron mal uso de la cola; considerando como mal uso a aquellos procesos que utilizaron la memoria en un porcentaje menor al 70%. En esta gráfica se observa que el único usuario que utilizó la cola y que no aparece en la gráfica, debido a que en todos sus procesos utilizó bien la memoria es el usuario *ian*. Lo anterior justifica que en la Gráfica 5 sea el único usuario que rebasó en promedio el 70% de eficiencia ya que registró una eficiencia mayor al 95%. Por otro lado esta gráfica comparada con la Gráfica 3 confirma que todos los procesos sometidos por el usuario *wsh* desperdiciaron memoria.

VER GRÁFICA 5a

En este punto, cuando se ha identificado que un usuario esta haciendo mal uso de una cola, las gráficas por usuario 9, 10 y 11 son de gran utilidad, ya que es importante saber la utilización que un usuario determinado le ha dado al sistema de colas en general.

A continuación se analizará como ha utilizado el sistema de colas, el usuario *wsh*. La Gráfica 9 nos muestra el número de procesos que el usuario *wsh* sometió en el sistema de colas durante el periodo. Como se puede observar, solamente utilizó dos colas: q_4h_14Mw (59 procesos) y q_1min_4Mw (2 procesos).

VER GRÁFICA 9

En este punto sabemos que el usuario wsh, utilizó mal la cola de 4h y 14Mw, pero no sabemos como utilizó la cola de 1min y 4Mw. De lo anterior, mediante la Gráfica 10 es posible saber qué tan eficientemente utilizó la memoria en todas las colas en las que sometió algún trabajo. Se puede observar que en ambas colas utilizadas tuvo una eficiencia de uso de memoria menor al 10%.

VER GRÁFICA 10

Continuando con el análisis del usuario *wsh*, en la ultima gráfica por usuario, (Gráfica 11) se puede observar que el usuario *wsh* tuvo un promedio de uso de CPU de aproximadamente 14,500 segundos (4 hrs. aprox.) para la cola de 4h y 14Mw. Lo cual indica que el complejo de tiempo sí fue bien seleccionado por el usuario y por lo tanto confirma que el desperdicio de memoria que tuvo durante la ejecución de sus trabajos en dicha cola, fue casi durante las 4 horas que consumió por proceso.

VER GRÁFICA 11

Después del análisis efectuado al usuario *wsh*, cabe mencionar la importancia que tiene el poder analizar más de una gráfica a la vez, ya que si solo se analizará la Gráfica 10 para saber cómo un usuario utilizó la memoria, podríamos llegar a conclusiones erróneas. Lo anterior debido a que sin analizar

las Gráficas 9 y 11, no es posible saber cuántos trabajos fueron ejecutados en cada cola, y cuál fue el promedio de uso de CPU por cola. De lo anterior, no se podría concluir que el problema realmente se presenta en el uso de la cola q_4h_14Mw y no en la de q_1min_4Mw, ya que no es lo mismo que 59 procesos desperdicien 10 Mw de memoria durante aproximadamente 4 horas que 2 procesos desperdicien 3.5 Mw durante aproximadamente 1 minuto.

Otra gráfica que sirve para analizar el uso de la memoria es la Gráfica 4, que muestra el promedio de memoria utilizada por cola para un día determinado, en este caso el 1ro de Febrero. El objetivo de esta gráfica es proporcionar al administrador diariamente, un panorama general de utilización de memoria por cola. En caso de observar algún problema, el administrador puede generar otras gráficas que muestren información más detallada, por ejemplo la Gráfica 12, la cual muestra el porcentaje de eficiencia de uso de memoria para todas las colas durante un periodo determinado.

VER GRÁFICAS 4 Y 12

En la Gráfica 4, se observa que el 1ro de Febrero, la cola más utilizada (4h y 14Mw) tuvo un promedio de memoria utilizada de aproximadamente 2.8Mw, si observamos la Gráfica 12, que muestra la moda de su eficiencia de uso de memoria durante el periodo, se puede observar que apenas alcanza el 10%, lo que confirma que no solo se utilizó mal el 1ro de Febrero sino se ha estado subutilizando durante todo el mes. La Gráfica 4a complementaría la información que se tiene hasta ahora acerca del uso de memoria de la cola de 4h y 14Mw, ya que tiene como objetivo mostrar qué tan eficientemente se utilizó la memoria en una cola día con día, durante un mes determinado. Se puede observar que en la cola de 4h y 14Mw, 14 días sus promedios estuvieron por debajo de los 4Mw. Por otro lado los días restantes que rebasaron los 4Mw, nunca estuvieron cerca de los 14 Mw. De lo anterior quizá convendría crear una cola de 4h y 8Mw, en la que bien hubieran podido correr todos los procesos enviados a la cola de 4h y 14Mw, durante el periodo analizado.

VER GRÁFICA 4a

En esta parte del caso práctico, termina el análisis de la utilización de memoria e inicia el análisis de utilización de CPU. Si bien es cierto que el uso de memoria es más crítico que el uso del CPU, es importante saber si los complejos de tiempo definidos para las colas son los adecuados. Lo anterior debido a que quizá la causa de que muchos procesos desperdicien memoria obedece a la falta de una cola con el tiempo de CPU adecuado para dichos procesos.

De manera similar a la Gráfica 4, la Gráfica 6 que muestra el promedio de uso de CPU por cola para un día determinado, (en este caso el 1ro de Febrero) tiene como objetivo proporcionar al administrador diariamente, un panorama general de utilización de uso de CPU por cola. En caso de observar algún problema, el administrador puede generar otras gráficas que muestren información más

detallada, por ejemplo la Gráfica 13, la cual muestra el promedio de uso de CPU para todas las colas durante un periodo determinado.

VER GRÁFICAS 6 Y 13

Continuando con el análisis de utilización de CPU, la Gráfica 6a muestra el promedio de uso de CPU que tuvieron los procesos sometidos en una cola durante un mes determinado. Se puede observar que en la cola de 24h y 14Mw, en el mes de Febrero solo dos días se utilizó por encima del límite de tiempo de CPU, establecido por la cola de 4h y 14Mw que es la inferior inmediata. Si se observa nuevamente la Gráfica 12 se puede apreciar que durante el periodo, la cola de 24h y 14Mw tuvo un promedio de eficiencia de memoria de apenas del 10% (1.4Mw) lo que indica que algunos de los procesos que se ejecutaron en esta cola pudieron haber sido ejecutados en la cola de 4h y 14 Mw.

VER GRÁFICAS 6a y 12

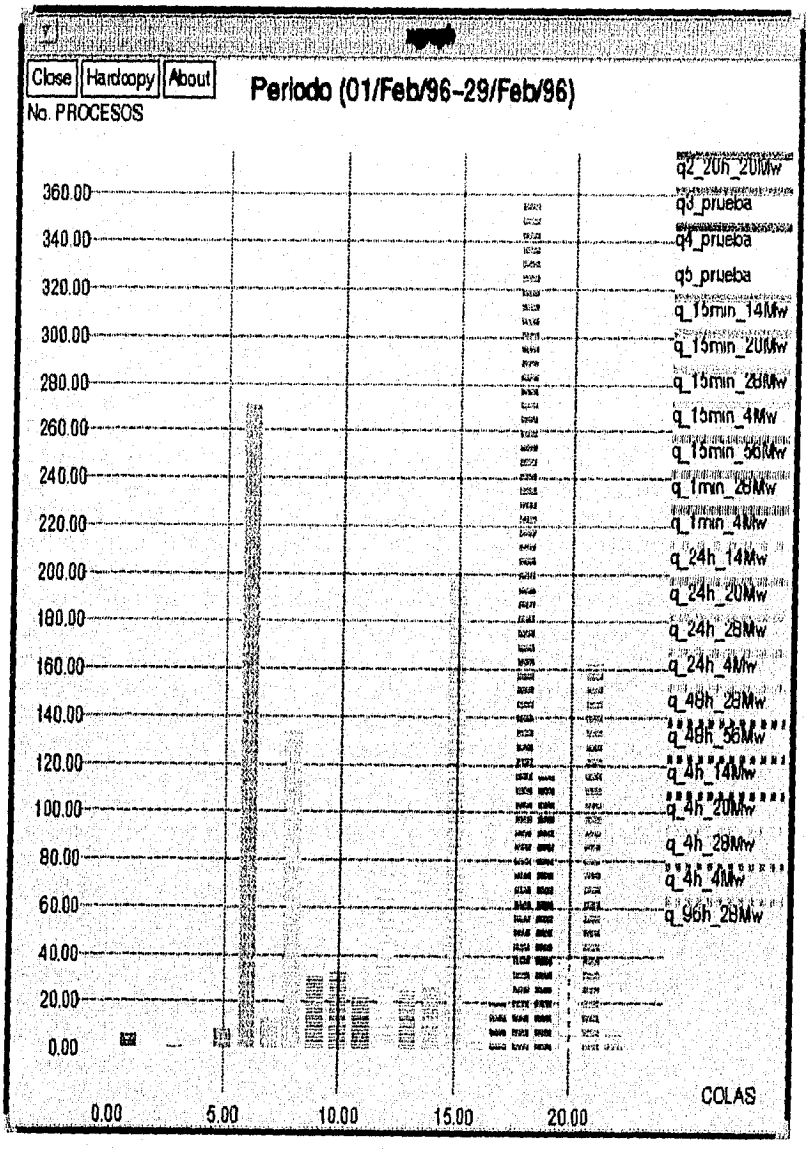
En esta última parte del caso práctico, se muestra la información acerca del uso del sistema de colas a partir de los complejos. Las gráficas 7, 7a, 8 y 8a muestran información acerca de los complejos de memoria y complejos de tiempo. El objetivo de estas gráficas es mostrar el uso de cada complejo, apoyando al administrador al momento de decidir si crear o no un nuevo complejo que pueda facilitar la administración del Sistema de Colas NQS.

Las siguientes Gráficas 7 y 7a muestran el número de procesos ejecutados para cada complejo de memoria en un periodo determinado y el número de procesos ejecutados para un complejo de memoria dado durante el mes de Febrero respectivamente. Como se puede observar, el complejo de memoria más popular es el de 4Mw, ya que atendió a más de 500 procesos durante el periodo.

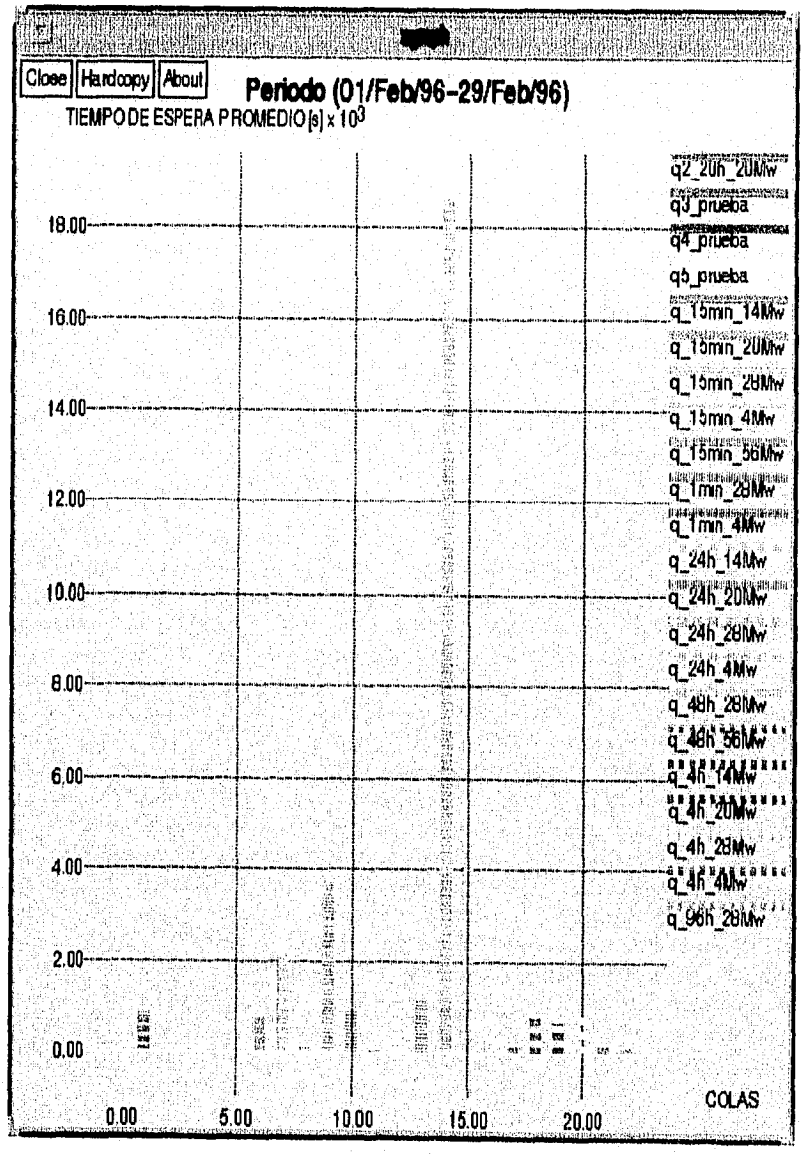
VER GRÁFICAS 7 Y 7A

Finalmente, las siguientes Gráficas 8 y 8a muestran el número de procesos ejecutados para cada complejo de tiempo en un periodo determinado y el número de procesos ejecutados para un complejo de tiempo dado durante el mes de Febrero respectivamente. Como se puede observar, el complejo de tiempo más popular durante el periodo, fue el de 15 min, ya que atendió a casi 700 procesos durante el periodo.

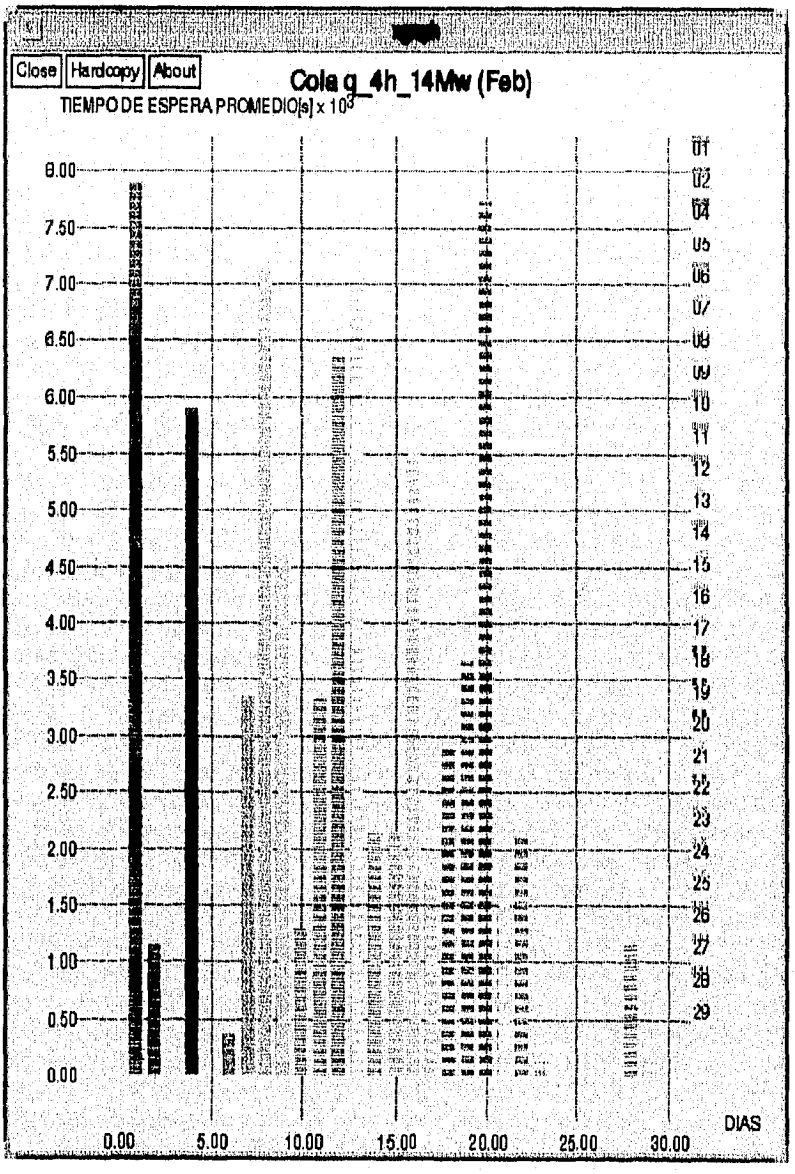
VER GRÁFICAS 8 Y 8A


Close
Hardcopy
About
Periodo (01/Feb/96-29/Feb/96)
No. PROCESOS
360.00
340.00
320.00
300.00
280.00
260.00
240.00
220.00
200.00
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
0.00
5.00
10.00
15.00
20.00
COLAS
q2_20h_20Mw
q3_prueba
q4_prueba
q5_prueba
q_15min_14Mw
q_15min_20Mw
q_15min_28Mw
q_15min_4Mw
q_15min_56Mw
q_1min_28Mw
q_1min_4Mw
q_24h_14Mw
q_24h_20Mw
q_24h_28Mw
q_24h_4Mw
q_48h_28Mw
q_48h_56Mw
q_4h_14Mw
q_4h_20Mw
q_4h_28Mw
q_4h_4Mw
q_96h_28Mw

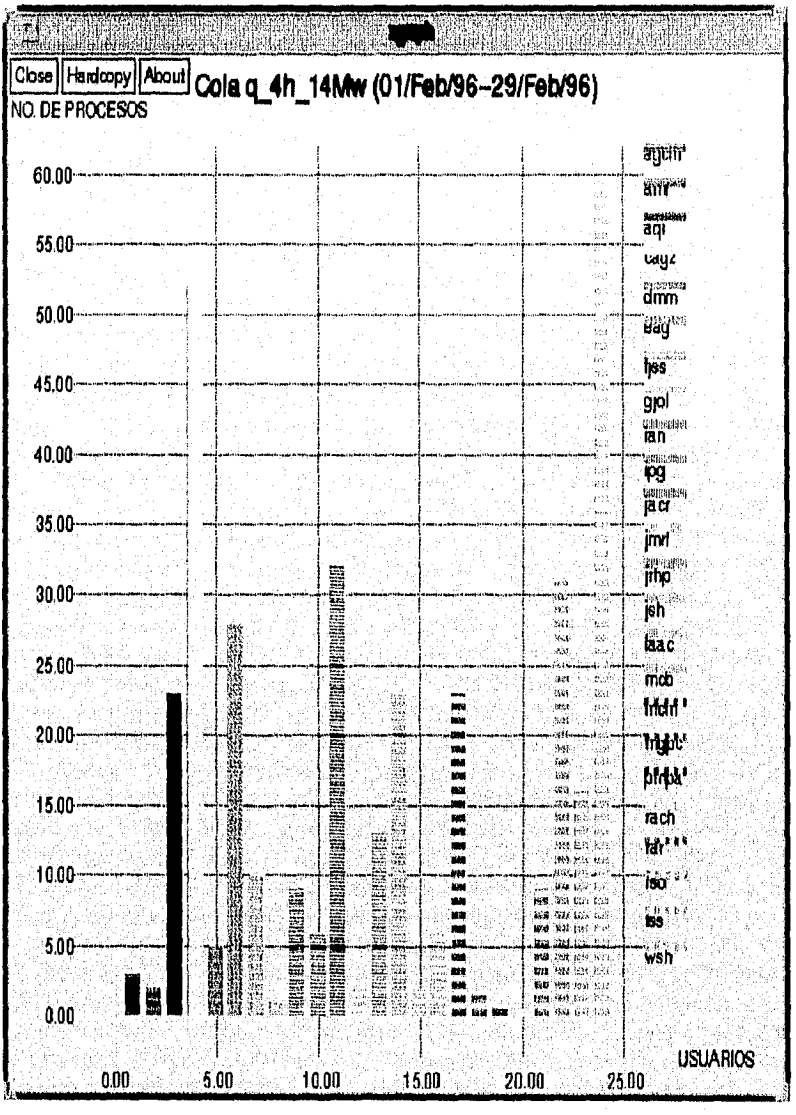
GRÁFICA 1.


Close
Hardcopy
About
Periodo (01/Feb/96-29/Feb/96)
TIEMPO DE ESPERA PROMEDIO [s] x 10^3
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
0.00
5.00
10.00
15.00
20.00
COLAS
q2_20h_20Mw
q3_prueba
q4_prueba
q5_prueba
q_15min_14Mw
q_15min_20Mw
q_15min_28Mw
q_15min_4Mw
q_15min_56Mw
q_1min_28Mw
q_1min_4Mw
q_24h_14Mw
q_24h_20Mw
q_24h_28Mw
q_24h_4Mw
q_48h_28Mw
q_48h_56Mw
q_4h_14Mw
q_4h_20Mw
q_4h_28Mw
q_4h_4Mw
q_96h_28Mw

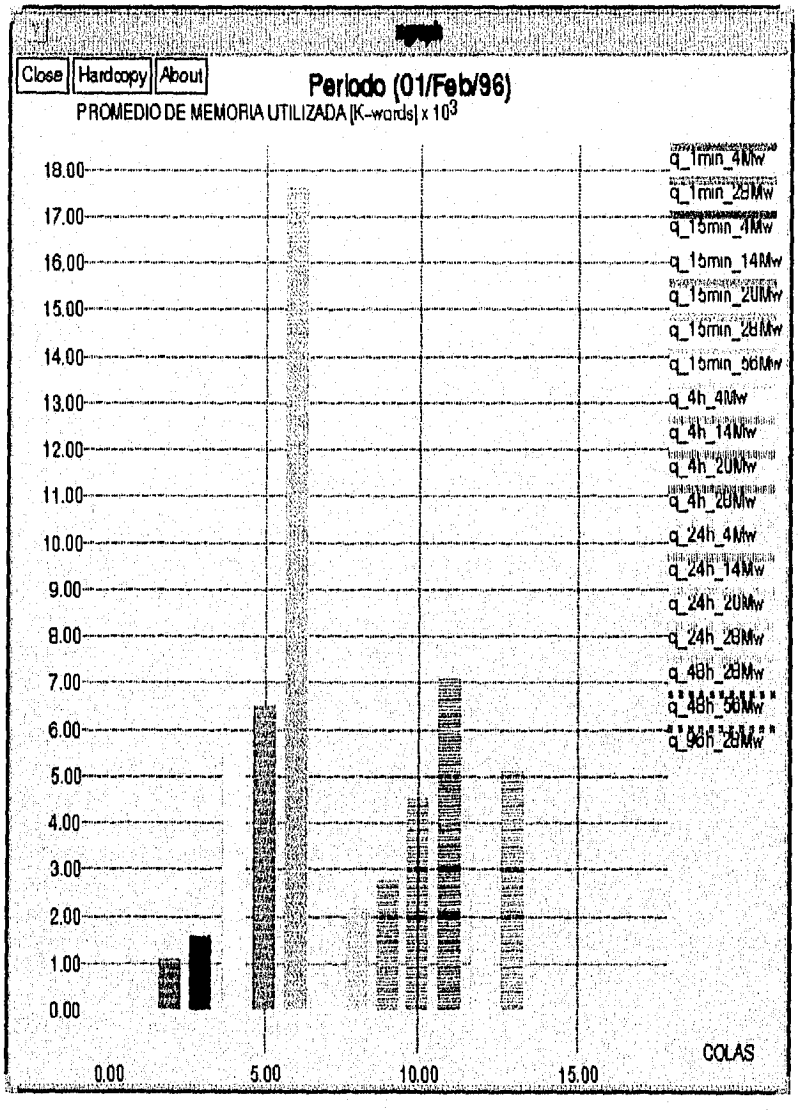
GRÁFICA 2.


Close
Hardcopy
About
Cola q_4h_14Mw (Feb)
TIEMPO DE ESPERA PROMEDIO[s] x 10^3
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
5.00
10.00
15.00
20.00
25.00
30.00
DIAS

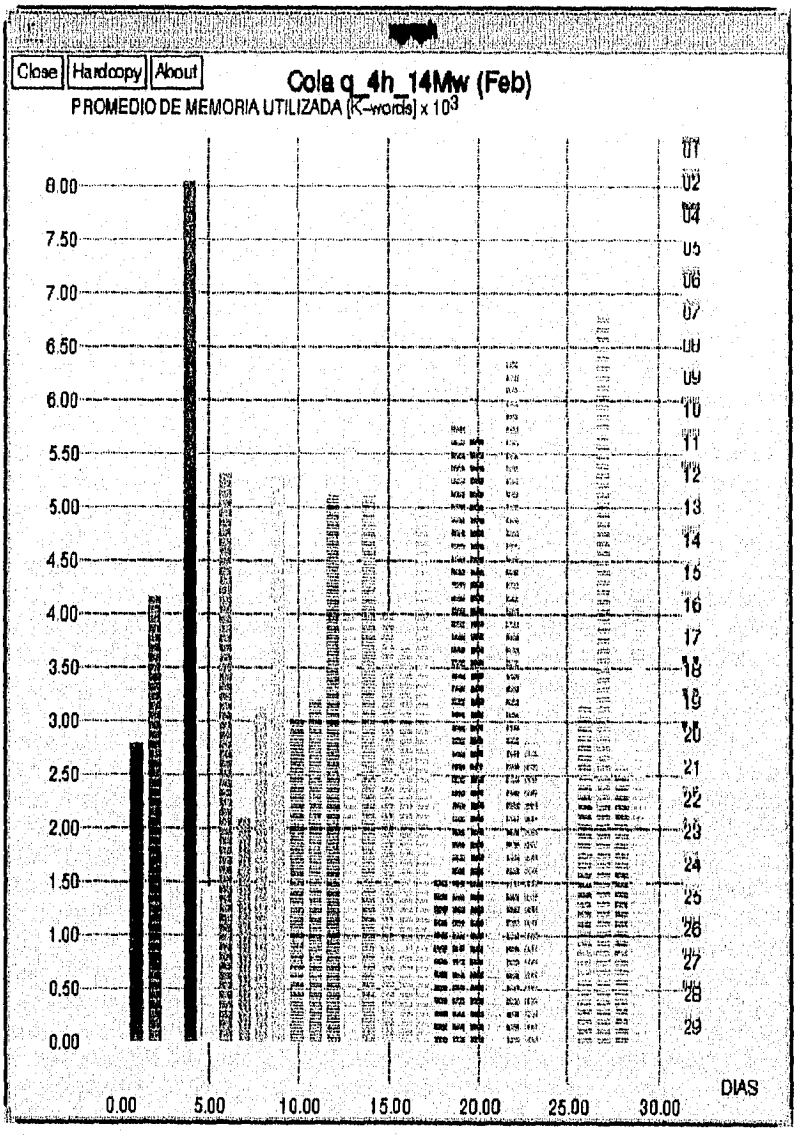
GRÁFICA 2a.


Close
Hardcopy
About
Cola q_4h_14Mw (01/Feb/96-29/Feb/96)
NO. DE PROCESOS
60.00
55.00
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
0.00
5.00
10.00
15.00
20.00
25.00
USUARIOS

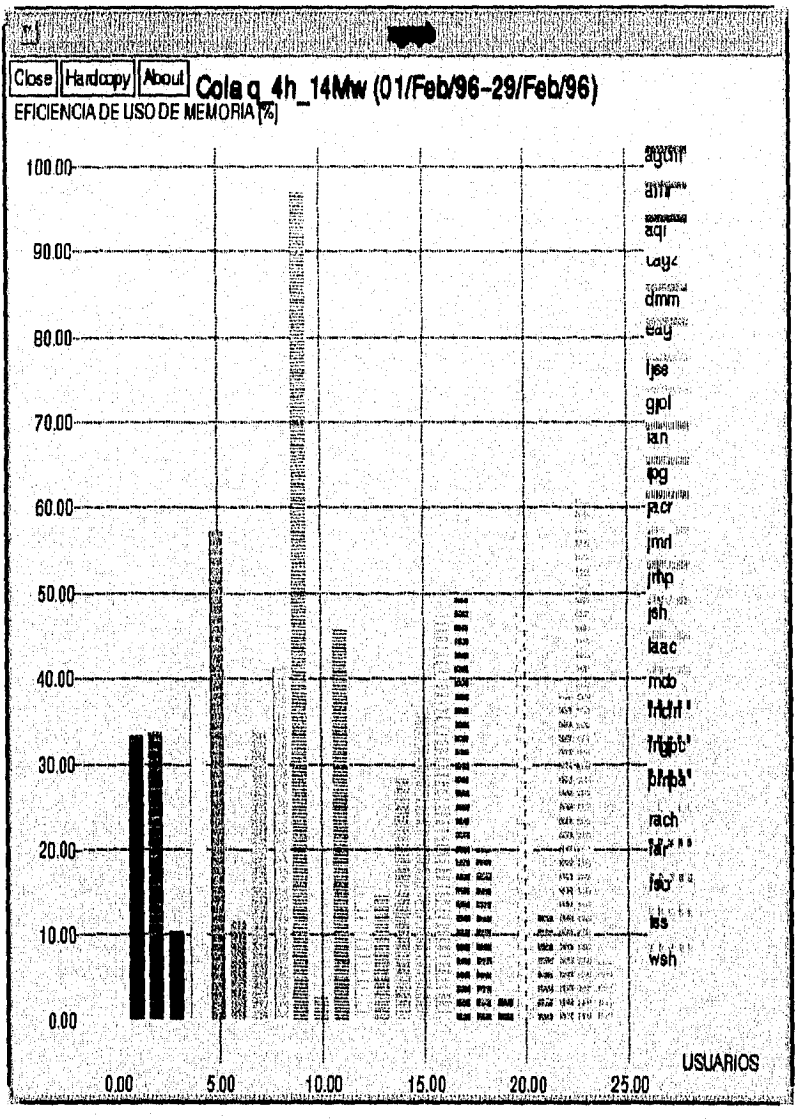
GRÁFICA 3.


Close
Hardcopy
About
Periodo (01/Feb/96)
PROMEDIO DE MEMORIA UTILIZADA [K-words] x 10^3
18.00
17.00
16.00
15.00
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
0.00
5.00
10.00
15.00
COLAS
q_1min_4Mw
q_1min_28Mw
q_15min_4Mw
q_15min_14Mw
q_15min_20Mw
q_15min_28Mw
q_15min_56Mw
q_4h_4Mw
q_4h_14Mw
q_4h_20Mw
q_4h_28Mw
q_24h_4Mw
q_24h_14Mw
q_24h_20Mw
q_24h_28Mw
q_48h_28Mw
q_48h_56Mw
q_96h_28Mw

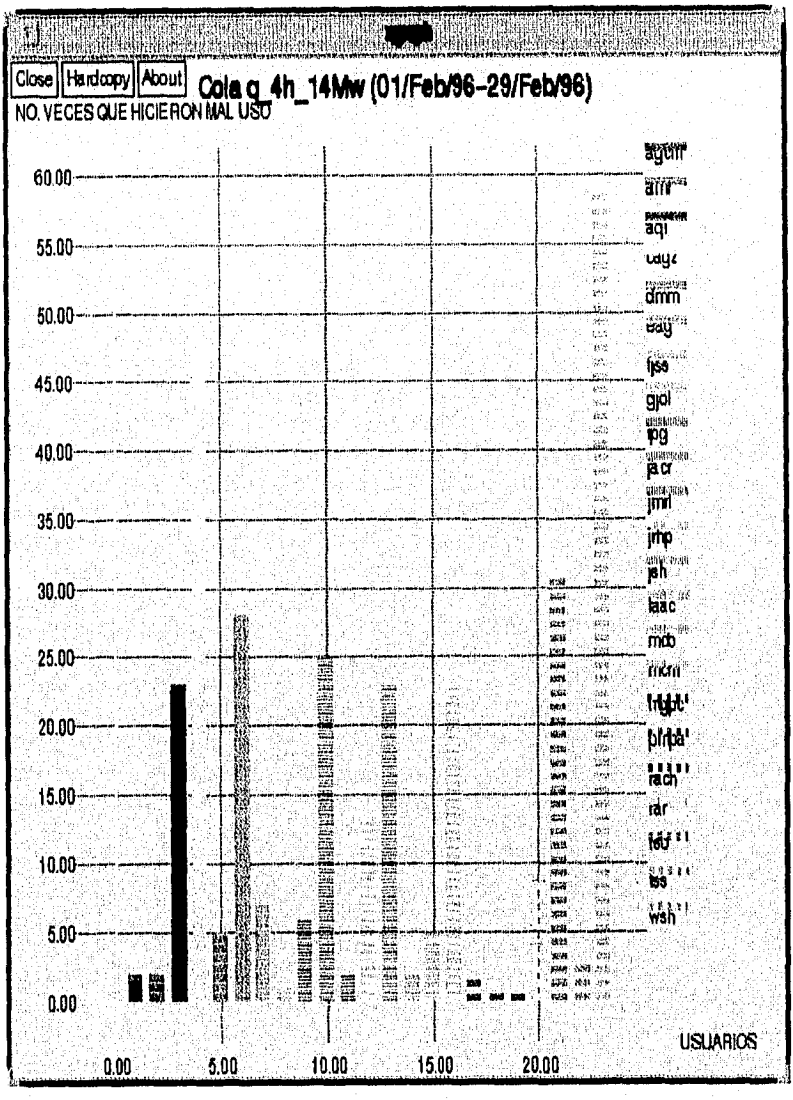
GRÁFICA 4.


Close
Hardcopy
About
Cola q_4h_14Mw (Feb)
PROMEDIO DE MEMORIA UTILIZADA [K-words] x 10^3
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
5.00
10.00
15.00
20.00
25.00
30.00
DIAS

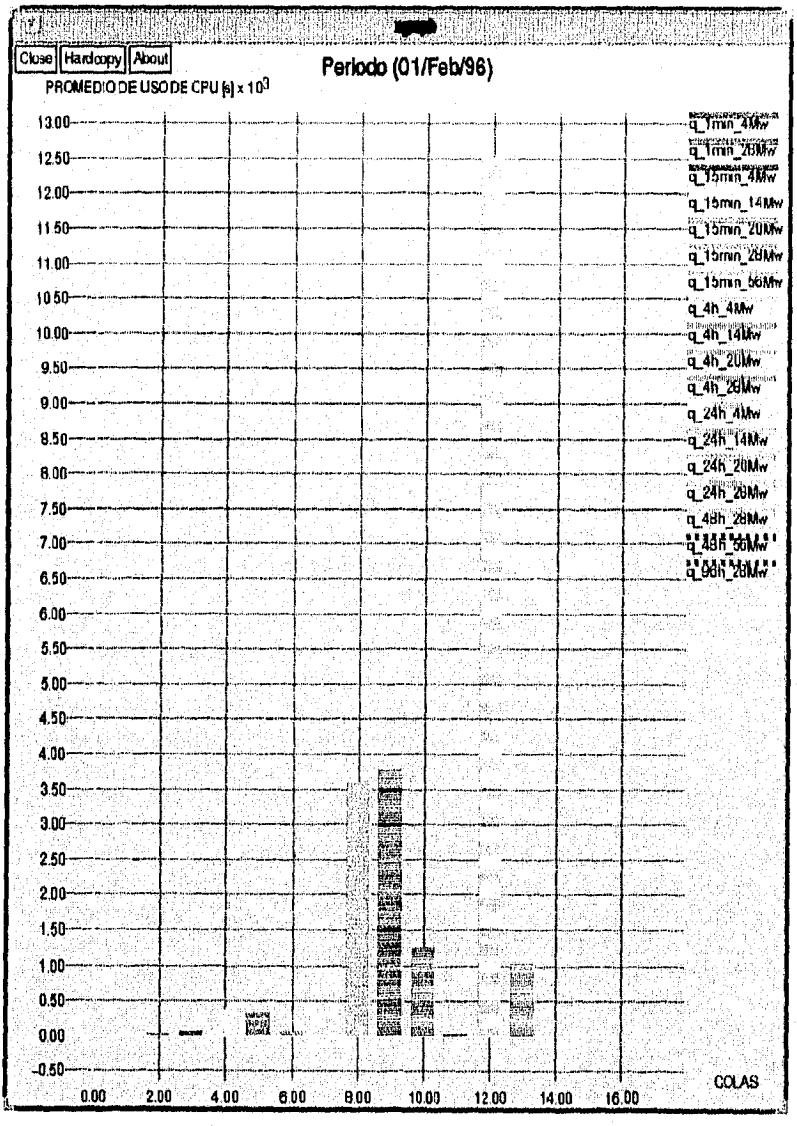
GRÁFICA 4a.


Close
Hardcopy
About
Cola q_4h_14Mw (01/Feb/96-29/Feb/96)
EFICIENCIA DE USO DE MEMORIA [%]
100.00
90.00
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
0.00
5.00
10.00
15.00
20.00
25.00
USUARIOS

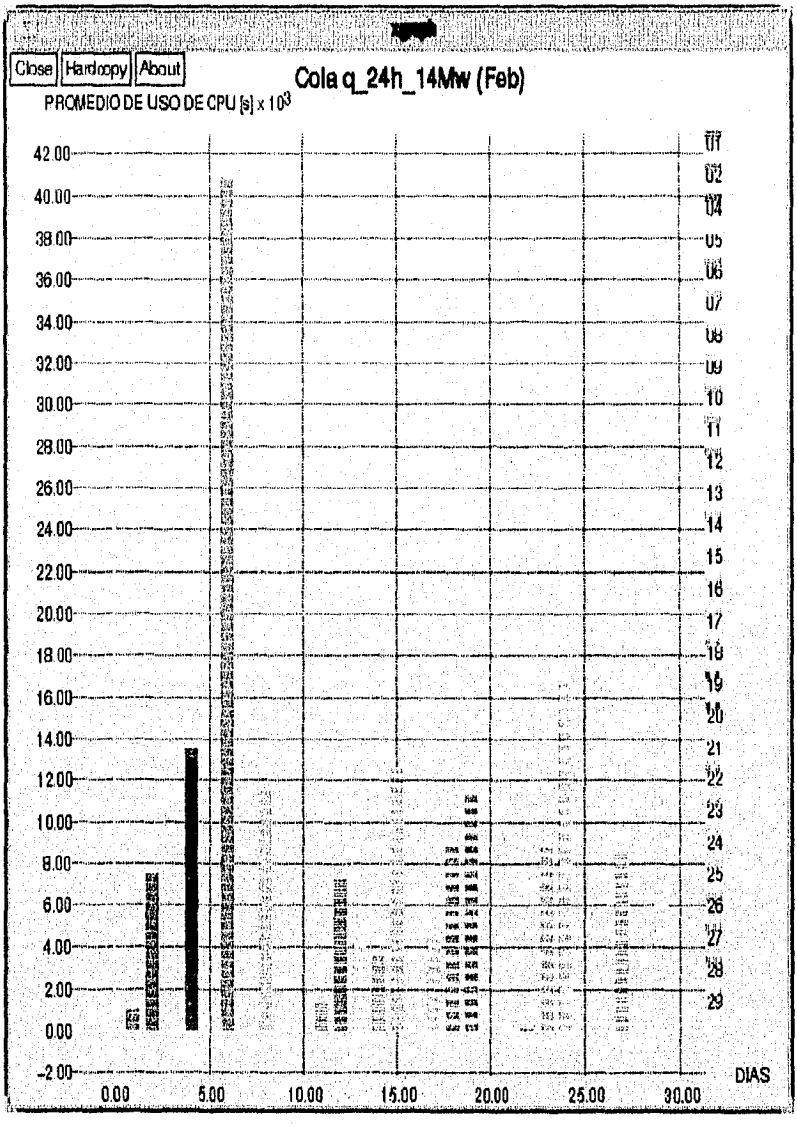
GRÁFICA 5.


Close
Hardcopy
About
Cola q_4h_14Mw (01/Feb/96-29/Feb/96)
NO. VECES QUE HICIERON MAL USO
60.00
55.00
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
0.00
5.00
10.00
15.00
20.00
USUARIOS

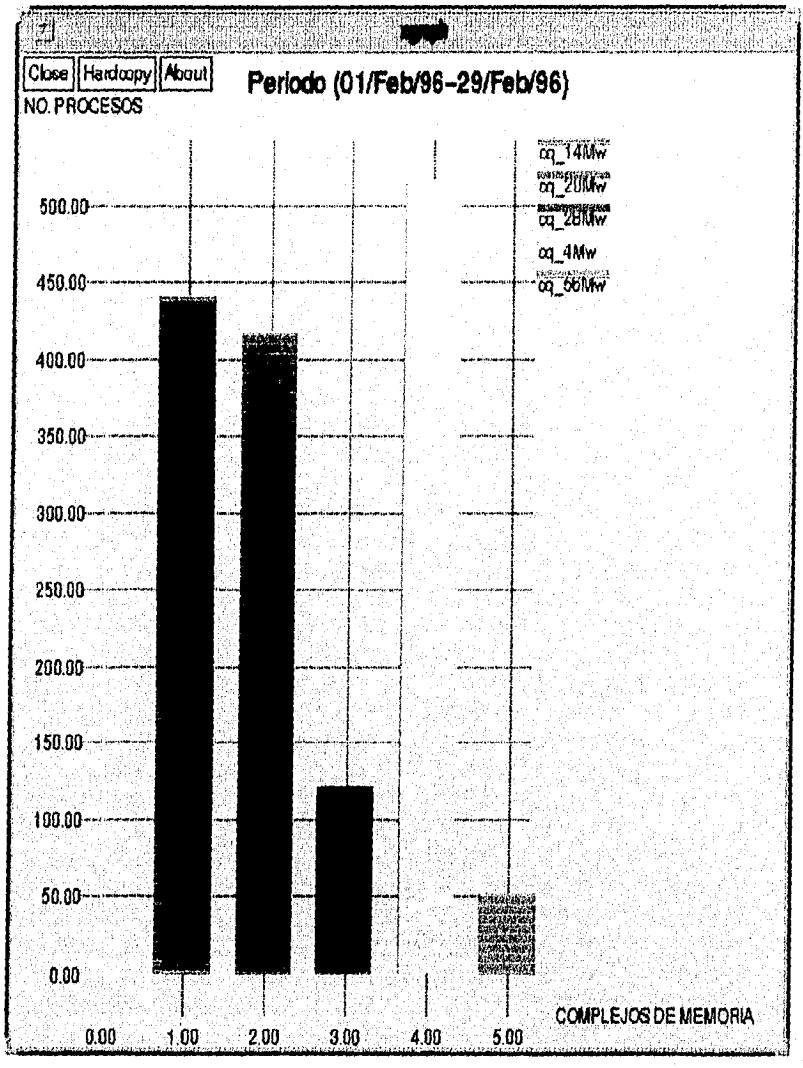
GRÁFICA 5a.


Close
Hardcopy
About
Periodo (01/Feb/96)
PROMEDIO DE USO DE CPU [s] x 10^0
13.00
12.50
12.00
11.50
11.00
10.50
10.00
9.50
9.00
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
-0.50
0.00
2.00
4.00
6.00
8.00
10.00
12.00
14.00
16.00
q_1min_4Mw
q_1min_28Mw
q_15min_4Mw
q_15min_14Mw
q_15min_20Mw
q_15min_28Mw
q_15min_56Mw
q_4h_4Mw
q_4h_14Mw
q_4h_20Mw
q_4h_28Mw
q_24h_4Mw
q_24h_14Mw
q_24h_20Mw
q_24h_28Mw
q_48h_28Mw
q_48h_56Mw
q_96h_28Mw
COLAS

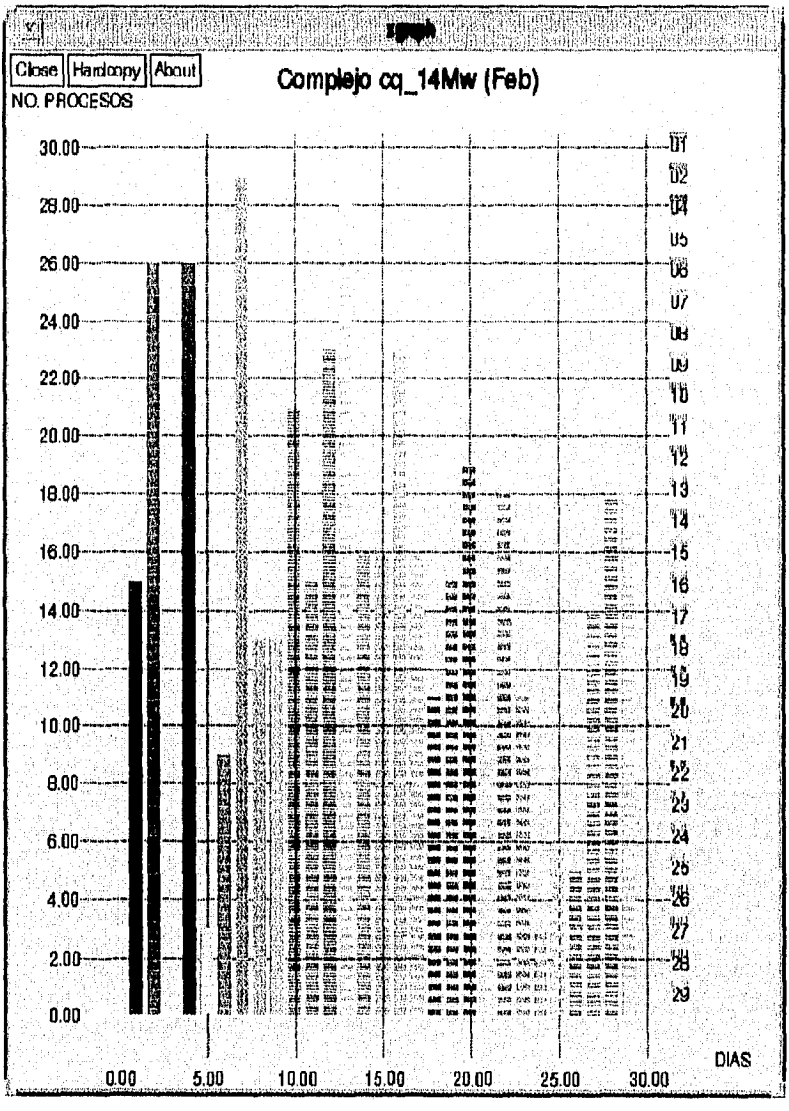
GRÁFICA 6.


Close
Hardcopy
About
Cola q_24h_14Mw (Feb)
PROMEDIO DE USO DE CPU [s] x 10^3
42.00
40.00
38.00
36.00
34.00
32.00
30.00
28.00
26.00
24.00
22.00
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
-2.00
0.00
5.00
10.00
15.00
20.00
25.00
30.00
DIAS

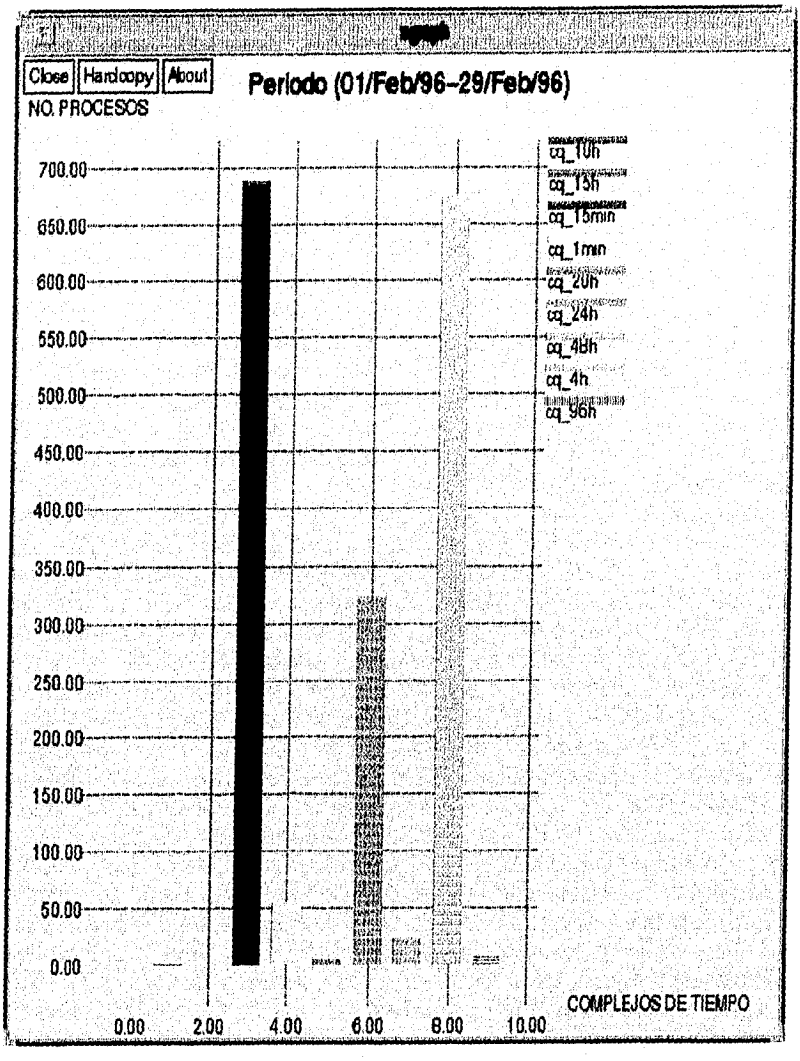
GRÁFICA 6a.


Close
Hardcopy
About
Periodo (01/Feb/96-29/Feb/96)
NO. PROCESOS
cq_14Mw
cq_20Mw
cq_28Mw
cq_4Mw
cq_56Mw
500.00
450.00
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
0.00
1.00
2.00
3.00
4.00
5.00
COMPLEJOS DE MEMORIA

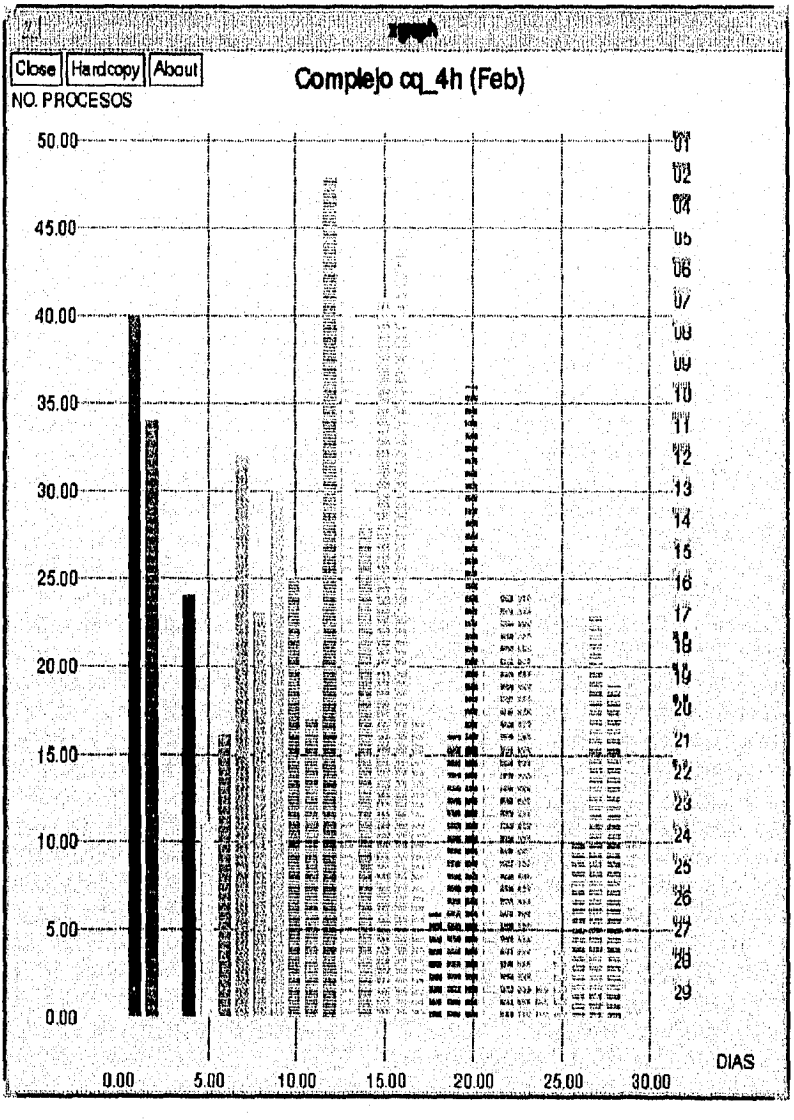
GRÁFICA 7.


Close
Hardcopy
About
Complejo cq_14Mw (Feb)
NO. PROCESOS
30.00
28.00
26.00
24.00
22.00
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
0.00
5.00
10.00
15.00
20.00
25.00
30.00
DIAS

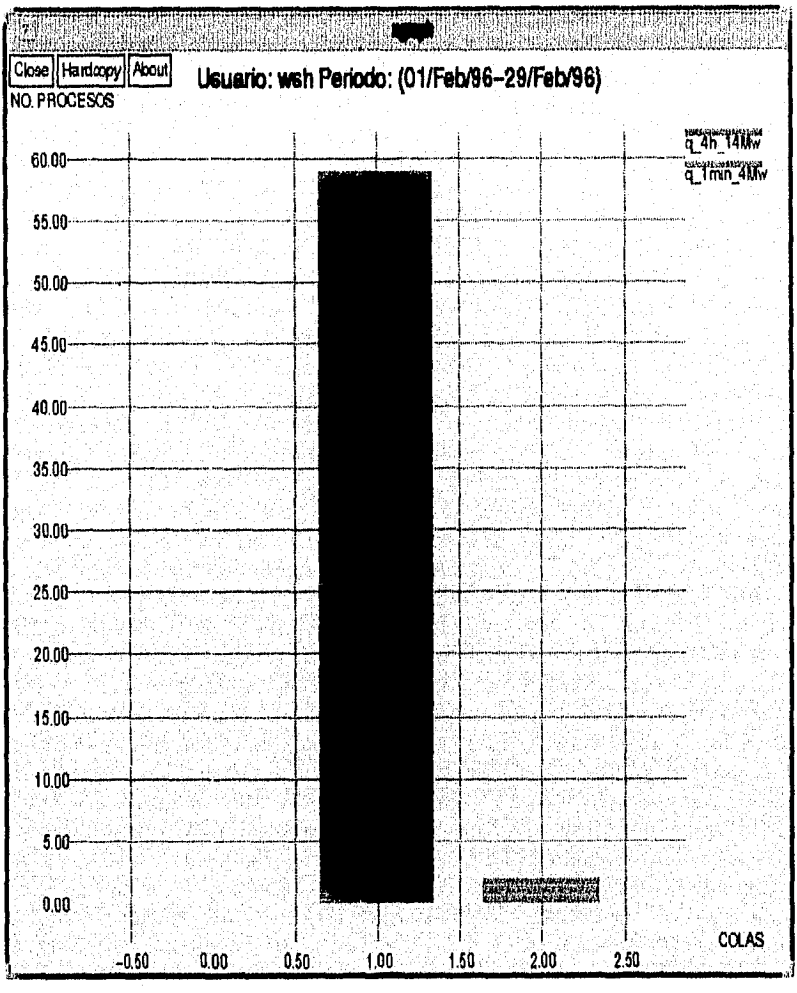
GRÁFICA 7a.


Close
Hardcopy
About
Periodo (01/Feb/96-29/Feb/96)
NO. PROCESOS
700.00
650.00
600.00
550.00
500.00
450.00
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
cq_10h
cq_15h
cq_15min
cq_1min
cq_20h
cq_24h
cq_48h
cq_4h
cq_96h
0.00
2.00
4.00
6.00
8.00
10.00
COMPLEJOS DE TIEMPO

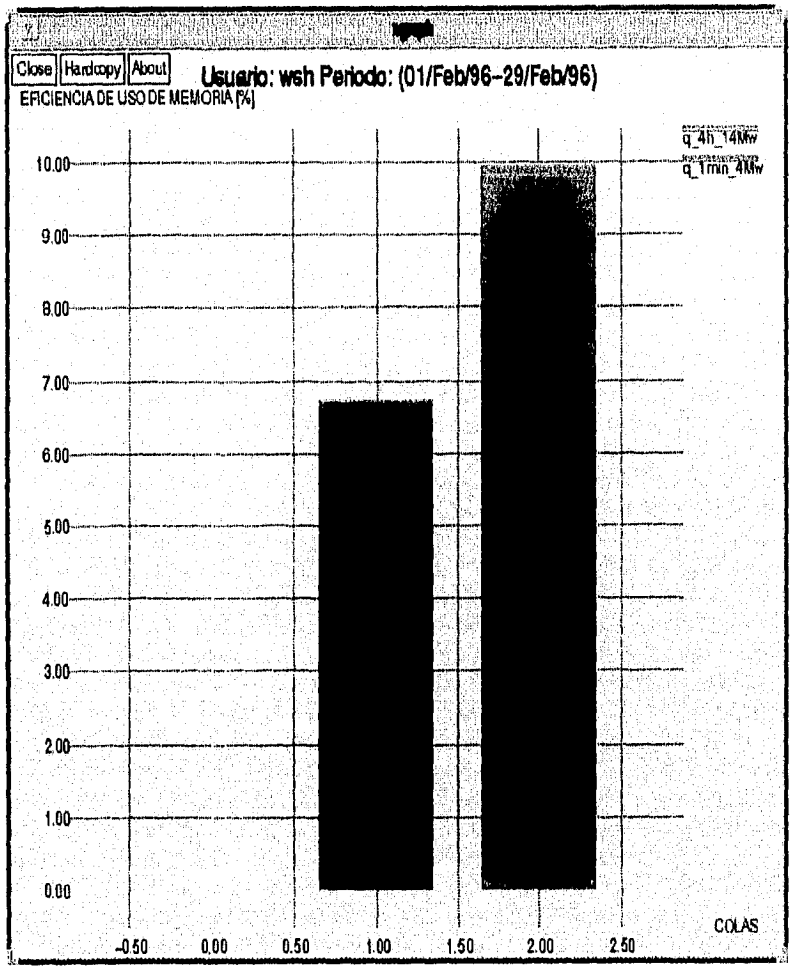
GRÁFICA 8.


Close
Hardcopy
About
Complejo cq_4h (Feb)
NO. PROCESOS
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
0.00
5.00
10.00
15.00
20.00
25.00
30.00
DIAS

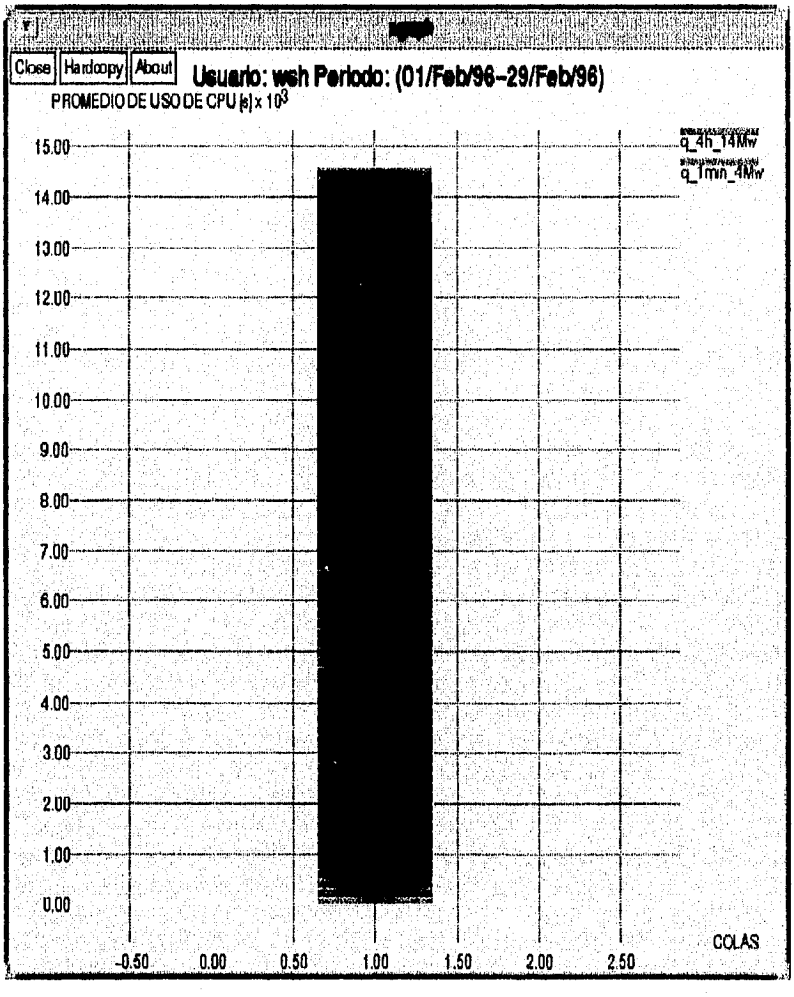
GRÁFICA 8a.


Close
Hardcopy
About
Usuario: wsh Periodo: (01/Feb/96-29/Feb/96)
NO. PROCESOS
q_4h_14Mw
q_1min_4Mw
60.00
55.00
50.00
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
-0.50
0.00
0.50
1.00
1.50
2.00
2.50
COLAS

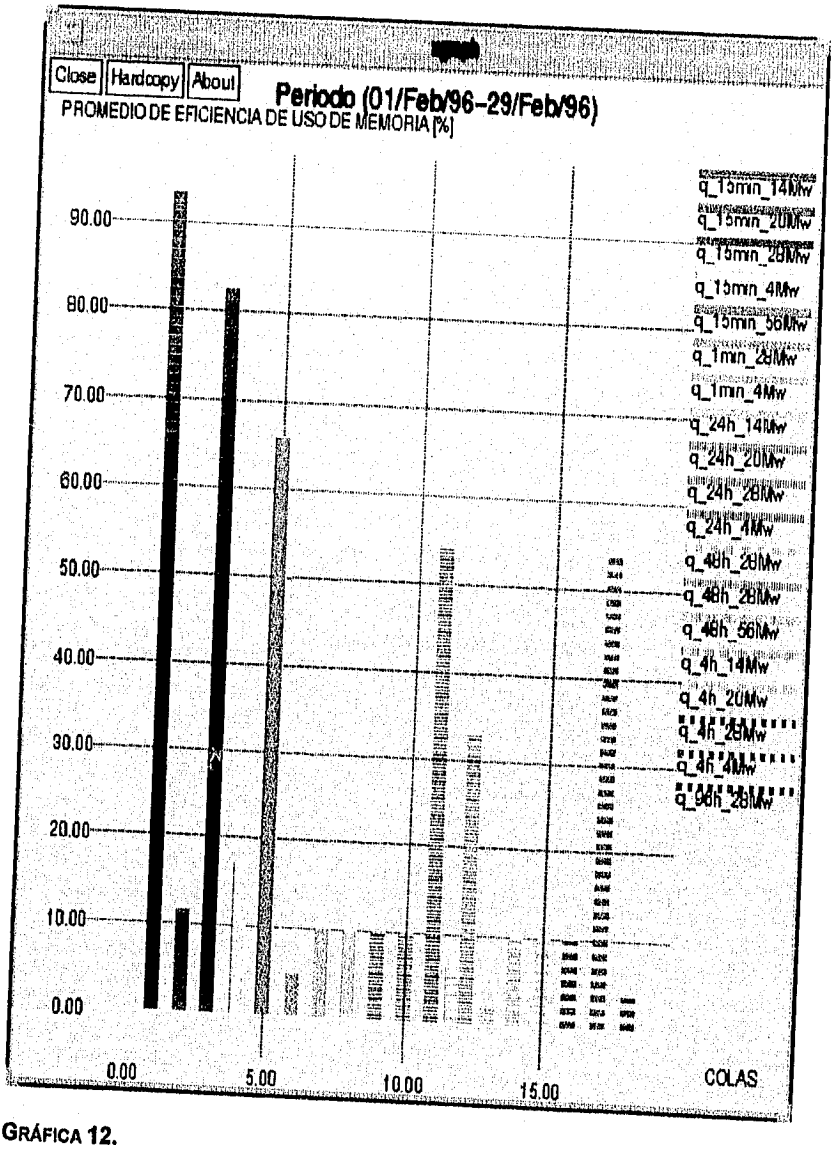
GRÁFICA 9.


Close
Hardcopy
About
Usuario: wsh Periodo: (01/Feb/96-29/Feb/96)
EFICIENCIA DE USO DE MEMORIA [%]
q_4h_14Mw
q_1min_4Mw
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
-0.50
0.00
0.50
1.00
1.50
2.00
2.50
COLAS

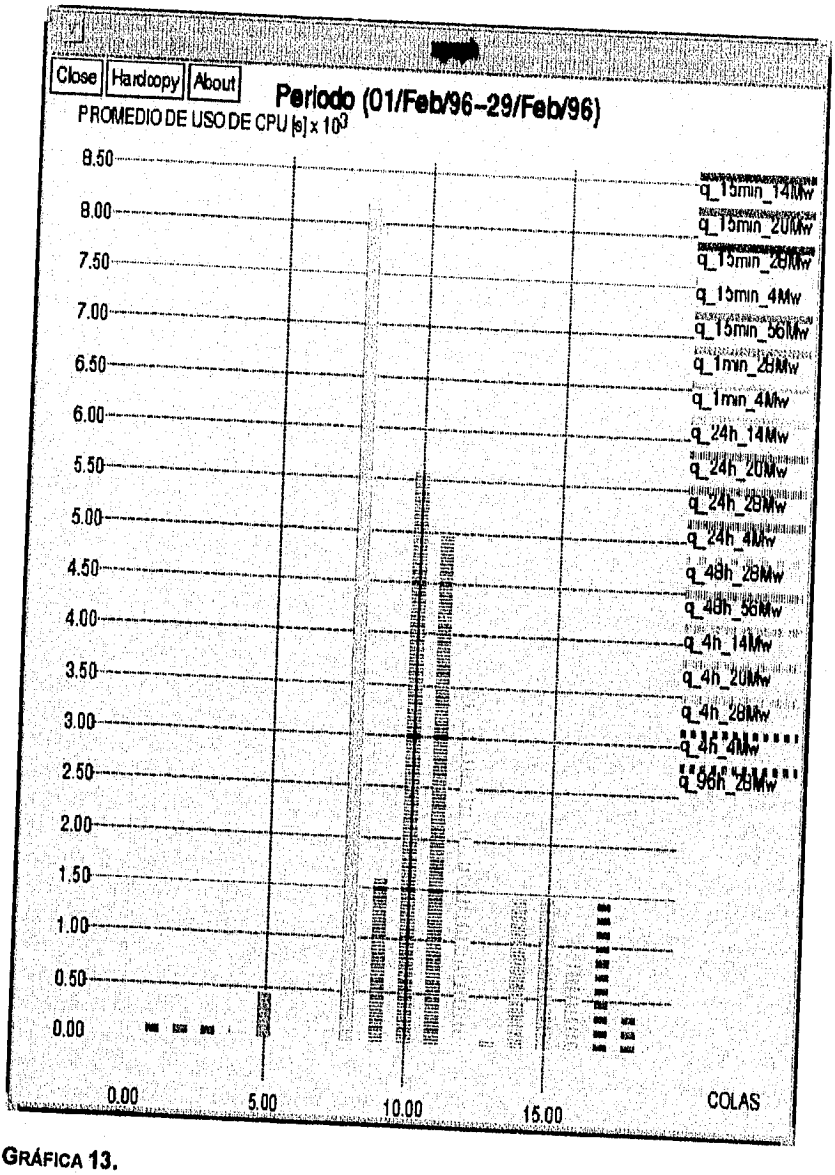
GRÁFICA 10.


Close
Hardcopy
About
Usuario: weh Periodo: (01/Feb/96-29/Feb/96)
PROMEDIO DE USO DE CPU [s] x 10^3
q_4h_14Mw
q_1mn_4Mw
15.00
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
-0.50
0.00
0.50
1.00
1.50
2.00
2.50
COLAS

GRÁFICA 11.


Close
Hardcopy
About
Periodo (01/Feb/96-29/Feb/96)
PROMEDIO DE EFICIENCIA DE USO DE MEMORIA [%]
90.00
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
0.00
5.00
10.00
15.00
COLAS
q_15min_14Mw
q_15min_20Mw
q_15min_28Mw
q_15min_4Mw
q_15min_56Mw
q_1min_28Mw
q_1min_4Mw
q_24h_14Mw
q_24h_20Mw
q_24h_28Mw
q_24h_4Mw
q_48h_28Mw
q_48h_28Mw
q_48h_56Mw
q_4h_14Mw
q_4h_20Mw
q_4h_28Mw
q_4h_4Mw
q_96h_28Mw

GRÁFICA 12.


Close
Hardcopy
About
Periodo (01/Feb/96-29/Feb/96)
PROMEDIO DE USO DE CPU [s] x 10^3
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
5.00
10.00
15.00
COLAS
q_15min_14Mw
q_15min_20Mw
q_15min_28Mw
q_15min_4Mw
q_15min_56Mw
q_1min_28Mw
q_1min_4Mw
q_24h_14Mw
q_24h_20Mw
q_24h_28Mw
q_24h_4Mw
q_48h_28Mw
q_48h_56Mw
q_4h_14Mw
q_4h_20Mw
q_4h_28Mw
q_4h_4Mw
q_96h_28Mw

GRÁFICA 13.

GLOSARIO DE TÉRMINOS

archivos de inicialización (.cshrc, .login, .profile)

Archivos que son leídos al momento de iniciar una sesión con el sistema. Estos archivos establecen los valores de las variables de ambiente y ejecutan algunos comandos de UNIX.

área de swap

Área en disco destinada para almacenar aquellos procesos o porciones de procesos, que fueron seleccionados por medio de la paginación o el swap para liberar memoria principal.

benchmark

Programa o conjunto de programas diseñados especialmente para incrementar la carga de trabajo de un sistema determinado, o bien algún recurso en especifico cuyo comportamiento se quiera evaluar dadas las circunstancias provocadas por dichos programas.

bootstrap

Proceso de inicialización de un sistema UNIX. Su función involucra desde la cargar del kernel en la memoria, hasta la operación del equipo en modo single-user o modo multiusuario.

BSD

Versión de UNIX desarrollada por la Universidad de California, Berkeley. BSD introdujo el concepto de memoria virtual y de NFS, al mundo de los sistemas UNIX, además de ser la base del sistema operativo SunOS.

buffer

Espacio temporal de almacenamiento que puede ser un archivo o un área de memoria. La mayoría de los editores almacenan el archivo que se esta editando en un buffer y cuando se salva dicho buffer es copiado sobre el archivo original.

buffer cache

Área de memoria intermedia entre los programas de aplicación y los discos. Cuando un programa escribe datos, primero son enviados al buffer cache y tiempo después, son enviados a disco, por lo cuál si es necesario leer información que se acaba de enviar al disco, puede ser que aún se encuentre en el buffer cache, lo que minimiza el tiempo de búsqueda en disco.

calendarizador

Parte del núcleo del sistema operativo, que se encarga de asignar el o los procesadores disponibles a los procesos del sistema, para que se ejecuten.

checkpoint

Acción de salvar el estado de las peticiones sometidas a NQS, para posteriormente poder reanudar su ejecución a partir del punto en que se suspendió. Lo anterior se realiza automáticamente si el sistema de colas NQS o el sistema completo son dados de baja y reinicializados nuevamente.

cola en lote

Mecanismo mediante el cual se ejecutan trabajos secuencialmente. Una cola en batch, recibe los trabajos de los usuarios, para después ejecutarlos en un momento dado. Las colas en batch son muy efectivas para administrar la carga de trabajo.

daemon

Proceso que es invisible para los usuarios pero que proporciona servicios importantes dentro del sistema, como por ejemplo verificar los e-mails que llegan al sistema o bien detectar aquellas peticiones de conexión en la red. Los deamons normalmente pasan la mayor parte del tiempo esperando a que algo pase, por lo cual no cargan de manera considerable la actividad del sistema.

default

En un programa o variable que presenta mas de una opción, la opción de default, o valor por default, es aquel que se establece sin necesidad de elegirlo. Cabe mencionar que el default es normalmente la opción más elegida.

device driver

Porción de software que interactua con el controlador del dispositivo, para realizar operaciones de entrada y salida. Los device drivers son parte del kernel del sistema operativo.

dirección IP

Dirección única de 32 bits, que identifica a un nodo dentro de una red internet.

e-mail

Correo electrónico entre usuarios de un mismo sistema o de diferentes sistemas. Existen diferentes programas en sistemas UNIX para enviar y leer correo electrónico.

firmware

Programas de microcódigo almacenados y ejecutados desde un almacenamiento de control de muy alta velocidad, normalmente grabados en memorias de solo lectura como ROM´s o PROM´s.

host

Computadora con una sola conexión a la red internet.

idle time

Tiempo que consume el procesador sin hacer nada, esperando que algún evento externo ocurra.

inodo

Estructura de datos que describe a cada archivo dentro del sistema UNIX. Para cualquier sistema de archivos, el número de inodos y por lo tanto el número máximo de archivos para el sistema de archivos en cuestión, se establece cuando el sistema de archivos es creado.

internet

Red a nivel mundial que permite establecer una comunicación universal entre todos sus nodos utilizando los protocolos de comunicación de TCP/IP. Esta red es la sucesora de la ARPANET construida por la DARPA.

I/O, Input/Output

Entrada/Salida proveniente de algún software o hardware.

kernel

Parte del sistema operativo UNIX, que se encarga de funciones como manejo de memoria, operaciones de I/O y muchas otras operaciones de bajo nivel. Se puede considerar al kernel como el corazón del sistema operativo.

modo multiusuario

Modo de operación del sistema en el cual es posible atender a más de un usuario, por lo que se permiten accesos al sistema de las cuentas existentes dentro del mismo.

modo single-user

Modo de operación del sistema en el cual solo la partición de root se encuentra montada y solo se permite el acceso al sistema desde la consola y a través de la cuenta de superusuario. Este modo de operación también es conocido como "modo de mantenimiento del sistema"

NFS, Network File System

Sistema de Archivos de Red. Utilieria que permite a los sistemas UNIX y a muchos otros sistemas que no son UNIX, compartir sistemas de archivos vía una red TCP/IP. Sujeto a ciertas restricciones de seguridad, es posible el acceso absoluto a archivos compartidos bajo este concepto.

paginación

Técnica que utiliza el kernel de UNIX, para liberar espacio en memoria física. El kernel detecta aquellas paginas que no han sido accesadas recientemente y las copia en el área de paginación del disco, para después reasignar la memoria física liberada a otras funciones. Un sistema empieza a paginar, cuando le queda poca memoria disponible.

partición

Porción del disco duro. Los discos bajo sistemas UNIX, tipicamente cuentan con ocho particiones definidas, aunque no todas ellas sean utilizadas.

password

Contraseña requerida por el sistema, para que cada cuenta existente, pueda accesar al mismo.

PATH

Variable de ambiente que contiene la lista de directorios que el shell consultará para encontrar aquellos programas que el usuario desea ejecutar.

petición en lote

Proceso que se coloca en una cola de NQS, o en cualquier otro sistema de colas, para que posteriormente sea ejecutado aprovechando así al máximo los recursos del sistema.

prioridad

Número que determina que tan seguido el kernel correrá un proceso determinado. Un proceso con una prioridad alta, correrá más seguido y por lo tanto terminará más rápido que un proceso con baja prioridad.

prompt del shell

Símbolo del shell (cuando se tiene una sesión interactiva), que indica que se encuentra listo para interpretar una línea de comandos. Por default, el símbolo de porcentaje (%) es el prompt del C Shell, mientras que el símbolo de pesos ($) es el prompt del Bourne Shell y del Korn Shell.

script

Archivo de texto que contiene una serie de acciones o comandos, que se pueden aplicar para modificar algún otro archivo, o bien para realizar otras funciones a través de estructuras y comandos en un lenguaje de programación determinado.

shell

Programa que lee e interpreta las líneas de comandos por lo que también se le conoce como "interprete de comandos". Además cuenta con un lenguaje de programación mediante el cual se pueden realizar scripts en shell y posteriormente ejecutarse.

swap

Técnica que utiliza el kernel de UNIX, para liberar espacio en memoria física. El kernel mueve procesos completos de la memoria principal hacia el área de swap en disco, para después reasignará la memoria física liberada a otras funciones. Los procesos candidatos a ser enviados al área de swap, son aquellos que han permanecido en estado de idle, durante un cierto tiempo

System V

Versión de UNIX desarrollada por AT&T y que actualmente sigue desarrollándose.

TCP/IP, Transmition Control Protocol/Internet Protocol

Protocolo de Control de Transmisión/Protocolo Internet. Conjunto de protocolos que permiten interconectar redes de diferentes tecnologías de manera transparente para el usuario. Algunos de los protocolos que incluye son IP, ICMP, UDP, TCP, SNMP, TELNET, FTP, etc.

tamaño del bloque

Máxima cantidad de datos, que los sistemas de archivos de UNIX almacenaran contiguamente. Por ejemplo, si el tamaño del bloque de un sistema de archivos es de 8KB, aquellos archivos menores de 8KB, serán almacenados contiguamente, mientras que aquellos mayores al tamaño del bloque del sistema de archivos, serán almacenados en diferentes bloques 8KB ubicados en diferentes sectores dentro del disco. Es importante señalar que el tamaño del bloque de un sistema de archivos es diferente al tamaño del bloque del disco físico, que casi siempre es de 512 bytes.

valor nice

Número que se utiliza para calcular la prioridad de ejecución de un proceso. Un valor nice bajo, significa alta prioridad, mientras que un valor nice alto significa baja prioridad.

XENIX

Primera versión de UNIX para computadoras personales y una de las pocas que corre en un sistema 80286. XENIX proviene de la Versión 7 de UNIX, versión desarrollada por AT&T en la década de los 70's.

zombie

Procesos terminados, que por alguna razón, no pudieron ser borrados de las tablas de procesos. Es imposible borrar estos procesos de las tablas de procesos, por lo cual permanecen ahí hasta que el sistema es reinicializado. Independientemente de ocupar un lugar en la tabla de procesos, los procesos zombies no consumen ningún recurso del sistema,

BIBLIOGRAFIA

Ferrari, D., Serazzi, G. and Zeigner, A. 1983. *Measurment and Tuning of Computer Systems,* Prentice Hall Software Series. ISBN 0-13-568519-2

Frish, A.. 1991. *Essential System Administratrion,* O'Reilly & Associates, Inc. ISBN 0-937175-80-3

Jain, R. 1991. *The Art of Computer Systems Performance Analysis,* John Wiley & Sons, Inc. ISBN 0-471-50336-3

Loukides, M. 1992. *System Performance Tunning,* O'Reilly & Associates, Inc. ISBN 0-937175-60-9

Nemeeth, E., Snyder, G., and Seebass, S. 1989. *UNIX System Administration Handbook,* Prentice Hall Software Series. ISBN 0-13-933441-6

Peek, J., O'Reilly, T., and Loukides, M. *UNIX Power Tools,* O'Reilly & Associates, Inc. ISBN 0-679-79073-X

Pressman, R.S. 1990. *Ingenieria del Software un Enfoque Practico,* Pressman McGraw-Hill. ISBN 84-7615-222-1

Manuales Técnicos de CRAY

Network Queuing System User's Guide

UNICOS Administrator's Commands Reference Manual

UNICOS Functional Overview

UNICOS Performance Evaluation and Tuning

UNICOS System Administration

UNICOS Tuning Guide

UNICOS User Commands Reference Manual