Evaluación del Desempeño de sistemas Unix*en ambiente distribuido

Eduardo Pavez F.
Depto. de Cs. de la Computación
Universidad de Chile
epavez@uchdcc.uchile.cl

Gonzalo Urrea R.
Depto. de Cs. de la Computación
Universidad de Chile
gurrea@uchdcc.uchile.cl

Resumen

El sistema operativo Unix es uno de los más utilizados hoy en día, ya sea en centros de investigación o en las distintas actividades de las empresas. Sin embargo, no existe una buena difusión de técnicas de modelamiento de Unix que permitan evaluar su desempeño y aprovechar mejor sus características de administración y asignación de recursos.

Este trabajo consiste en el desarrollo de un modelo para evaluación del desempeño de sistemas que ejecutan en el sistema operativo Unix en un ambiente distribuido, basándose en los conceptos bajos los cuales se rige el modelo construido por Ramamurthy [13].

Los resultados obtenidos en la validación, del modelo de Ramamurthy, en un ambiente local y distribuido de multiprogramación difieren con respecto a los resultados reales en un 10%.

Por otra parte, es necesario destacar que la principal desventaja del modelo es el costo computacional de los algoritmos para evaluación de medidas de rendimiento, los que dependen del número de tipos de trabajos distintos, del número de trabajos por tipo, número de dispositivos modelados y otros parámetros que aumentan tanto los requerimientos de memoria como los de tiempo de ejecución del algoritmo.

keywords: modelamiento, evaluación del rendimiento, sistemas computacionales, sistemas distribuidos, sistema operativo Unix.

1 Introducción

El sistema operativo Unix es uno de los más utilizados en la actualidad, debido a la versatilidad y amplitud de su aplicación. Una de sus características principales es que es un sistema de

*Unix es marca registrada de Bell Laboratories

tiempo compartido que permite a varios procesos estar ejecutando simultáneamente, los que pueden tener distintos objetivos, por ejemplo: planificación de procesos, manejo de comandos del sistema operativo, edición de texto, consultas a bases de datos y en la actualidad acceso a redes de computadores.

Es posible que configuraciones basadas en Unix puedan sufrir problemas de desempeño y requerir modificaciones, esto genera interrogantes acerca de la utilización de los distintos dispositivos y las cargas de trabajo que provocan tales situaciones.

Para poder predecir los problemas y analizar las características de sistemas que ejecutan en Unix en un ambiente local o distribuido, deben existir herramientas que permitan realizar el registro de información del sistema, estas herramientas son conocidas como monitores para evaluación del desempeño [4][8].

La información entregada por los monitores junto a la información de hardware (ucp, discos, terminales, controladores de discos) y conocimiento del tipo de trabajos que ejecutan en el sistema, ha permitido y permitirá construir modelos que representen adecuadamente las características de Unix. Existen algunos estudios de Unix, los que han sido desarrollados por [10][11][9] y, uno de los más sofisticados, el modelo de Ramamurthy[13] sobre el cual está basada esta investigación.

2 Descripción de Unix para el diseño de un modelo distribuido

El sistema operativo Unix fue desarrollado por Ken Thompson, y luego reescrito en el lenguaje C por Dennis Ritchie [6], para laboratorios Bell. Con el aumento en desarrollo y poder de los microprocesadores, otras compañías llevaron el sistema Unix a nuevas máquinas, y por su simplicidad y claridad llevó a muchos ingenieros de sistemas a adaptarlo a sus propias necesidades, resultando varias versiones del sistema original.

Como principal característica es que permite a varios procesos ejecutar en forma simultánea, permitiendo a varios usuarios estar conectados a la vez. Es Unix el encargado de administrar la ejecución de los distintos procesos, mediante un esquema de tiempo compartido y asignación de prioridades [1].

En la Figura 1, se muestra la arquitectura lógica del sistema operativo Unix. El sistema operativo interactúa directamente con el hardware, entregando servicios a los procesos que ejecutan en él. Entonces, podemos ver al sistema operativo como un conjunto de componentes [1], en que la principal es conocida como el *kernel* del sistema, y tiene como función establecer la comunicación con el hardware.

La labor principal de Unix es el control de los recursos via la ejecución de llamadas al sistema. Los programas mostrados en la Figura 1, como GREP o VI, interactúan con el *kernel* a través de la ejecución de un conjunto de llamadas al sistema, conocidas como *system calls*. Estas llamadas al sistema hacen variadas operaciones a través de llamadas a rutinas proporcionadas por el sistema operativo, y por medio de estas se intercambia

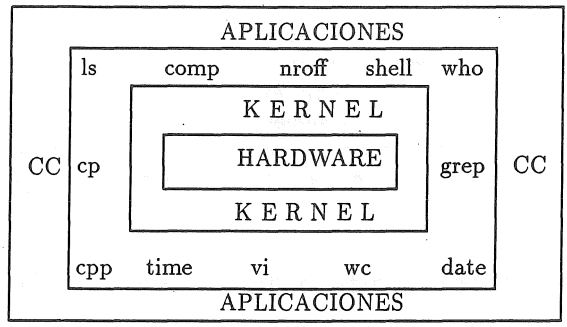


Figura 1. Arquitectura lógica del sistema operativo Unix

información entre el *kernel* y el programa del usuario. En definitiva, un programa en ejecución se transforma en un conjunto de llamadas al sistema junto al código efectivo del programa del usuario.

Por otra parte, el *kernel* se encarga de otras labores como asignar la UCP a los procesos, controlar la creación, suspensión comunicación y finalización de procesos, asignar memoria principal para procesos en ejecución, junto con la protección de áreas restringidas, administración de memoria secundaria, lo que incluye además el control y administración del sistema de archivos.

Este mismo esquema de Unix es posible ampliarlo a un sistema distribuido, como por ejemplo, si un comando antes ejecutaba una lectura en un disco local, ahora, en un esquema distribuido es probable que esta lectura se realice en forma remota, mediante alguna llamada al sistema dedicada a esa operación.

El modelo para Unix en ambiente distribuido, desarrollado en esta investigación, está basado en los recursos que requiere un proceso o comando para ser ejecutado en el sistema distribuido. Las definiciones utilizadas para sistemas distribuidos están basadas principalmente en las implementaciones para Unix descritas en [1].

Se pudo establecer en la práctica que la principal diferencia con un sistema local de multiprogramación, basado en Unix, radica en el lugar donde se atienden ciertos requerimientos, que no necesariamente los entrega la máquina donde se ejecuta el proceso. De acuerdo a esta característica, un proceso que no pueda satisfacer localmente una necesidad, realizando llamadas al sistema local, las efectuará en forma remota, y estas llamadas remotas al sistema son provistas por Unix.

El hecho que exista un servidor de archivos implica que el sistema tiene que ejecutar

procesos que le permitan enviar el requerimiento a través de la red, siendo atendido en el servidor y luego devuelto al cliente con la respuesta. El recorrido a través de la red produce un aumento en el tiempo de respuesta, comparado con un ambiente local. También, existe un aumento del tiempo de respuesta debido a que existirán una gran cantidad de requerimientos remotos sobre el servidor, producto de un aumento significativo de la carga. Si en el servidor la carga es baja, el cliente notará un tiempo de respuesta equivalente a como si el sistema atendiera el pedido localmente, y, por el contrario si la carga es alta el tiempo de respuesta será aún mayor.

3 Descripción del Modelo de Unix en ambiente distribuido

El modelo de Unix de Ramamurthy está construido en base a un Modelo de Red de Colas con Prioridades [5], que permite representar a un sistema computacional ejecutando en el sistema operativo Unix.

Cada sistema computacional en el sistema distribuido puede tener discos locales o hacer uso de un sistema de discos existente en otro computador, ver Figura 2 y 3.

Los procesos, ejecutando en alguno de estos sistemas, son representados mediante un esquema de múltiples clases y prioridades, caracterizadas por las llamadas al sistema y la propia ejecución.

Los usuarios son modelados como unidades que envían un proceso a ejecución, y tienen asociado un tiempo de reflexión, que corresponde al tiempo que se demora un usuario en enviar el siguiente comando.

Los discos corresponden a unidades que atienden procesos, en orden de llegada, y que son caracterizados por un tiempo de servicio por operación realizada en el disco y el número de veces que un proceso hace uso de ese disco.

La UCP está caracterizada como una unidad que atiende procesos de acuerdo a un esquema de prioridades, cuyos parámetros son el tiempo de servicio o *time slice* en Unix (fijo para todos los procesos) y el número de veces que un proceso visita la UCP. El esquema de prioridades es el siguiente:

- prioridad 1 y 2 son procesos del *kernel* ejecutando en modo *kernel*, y no son desalojables por otros procesos.
- prioridad 3 hasta K-1, son procesos ejecutando en modo *kernel* con prioridad a nivel de usuario y son desalojables por los procesos de prioridad 1 y 2.
- prioridad K, son procesos de usuario ejecutando a nivel usuario y son desalojables por procesos de mayor prioridad.

En este esquema de prioridades un menor valor de prioridad representa una prioridad mayor de atención, frente a otros procesos.

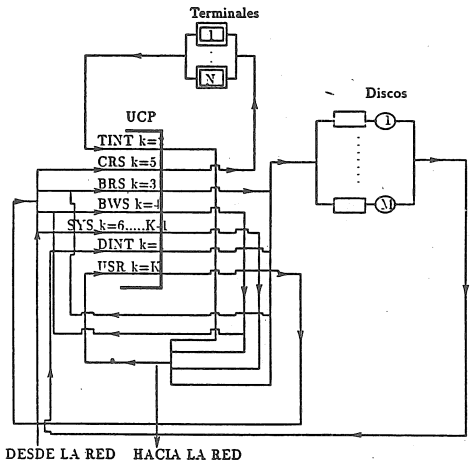


Figura 3. Modelo de Unix para un servidor de la red

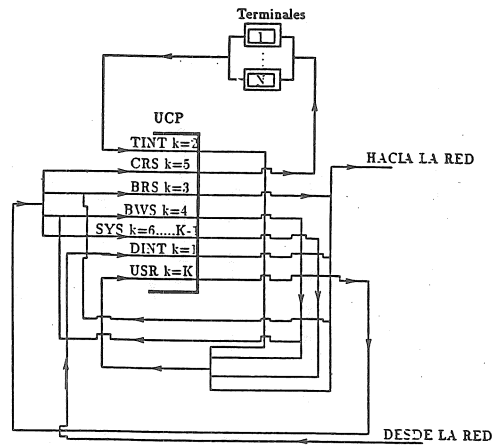


Figura 2. Modelo de Unix para un sistema sin discos

Inicialmente, un proceso que empieza su ejecución activa un interrupción asociada al terminal, con prioridad 2, llamada TINT. Luego, comenzará su ejecución realizando llamadas al sistema, las que corresponden a:

- DINT, interrupción desde el sistema de discos, indicando que el bloque requerido ha sido transferido al *buffer cache* de disco, con pioridad 1.
- CRS, lectura desde un terminal, con prioridad 5.
- BRS, llamada al sistema para escribir de disco, con prioridad 3 (*Block Read*), local y remota.
- BWS, escrituras a disco, con prioridad 4 (*Block Write*), local y remota.
- SYS, otras llamadas a rutinas de I/O con prioridad 6 a K-1. Estas pueden ser: *access*, *fork*, *lseek*, *open* o *wait*, entre otras.
- USR, ejecución a nivel usuario con prioridad K.

La principal característica, de acuerdo a la descripción anterior, es que el modelo de Ramamurthy representa a un tipo o clase de trabajo por sus diferentes llamadas al sistema.

Esta representación permitió a Ramamurthy desarrollar un conjunto de fórmulas analíticas basadas en prioridades [5][7] que permiten evaluar las medidas de desempeño, ya sea tiempo de respuesta por clase, throughput o utilización de distintos dispositivos. Algunas de las definiciones y fórmulas del modelo de Unix en ambiente distribuido se presentan a continuación:

r = caracteriza un tipo de trabajo (editor, consulta, transacción).

e = número de nodo en la red, de un total de E nodos o máquinas Unix.

c = dispositivo en un nodo (ucp, disco, cinta).

p = prioridad de ejecución en Unix.

N = número trabajos en el sistema = $\sum_{e=1}^{E} \sum_{r=1}^{R} N_{e,r}$

N_e = número de trabajos en el nodo e.

$S_{e,c,r,p}$ = tiempo de servicio para un trabajo de tipo r ejecutando con prioridad p, en el dispositivo c del nodo e.

$V_{e,c,r,p}$ = visitas que realiza un trabajo de tipo r ejecutando con prioridad p, en el dispositivo c del nodo e.

$\rho^r_{e,c,r,p}(N_e)$ = probabilidad que un trabajo de tipo r encuentre a un trabajo de tipo m ejecutando con prioridad p, en el dispositivo c del nodo e, cunado hay N_e trabajos en el nodo e.

$L_{e,c,r,p}(N_e)$ = largo promedio de la cola de trabajos de tipo r que ejecutan con prioridad p, en el dispositivo c del nodo e, cuando hay N_e trabajos en el nodo.

$L_{disco,r}(N_e)$ = largo promedio de la cola de requerimientos a disco, por tipo de trabajo, en el nodo e, cuando hay N_e trabajos en el nodo.

$W_{e,ucp,r,1}(N_e)$ = $S_{e,c,r,1} + \sum_{m=1}^{R_e} (L^r_{e,ucp,m,1}(N_e) - \rho^r_{e,ucp,m,1}(N_e)) * S_{e,ucp,m,1}$

$+ \sum_{m=1}^{R_e} \sum_{j=1}^{2} \rho^r_{e,ucp,m,1}(N_e) * S_{e,ucp,m,j}$

tiempo de respuesta para un trabajo de tipo r que está ejecutando en la UCP, en la máquina e, con prioridad 1, cuando realiza una visita a la UCP.

$W_{e,ucp,r,2}(N_e)$ = $S_{e,ucp,r,2} + \frac{\sum_{m=1}^{R_e} \sum_{j=1}^{2} L^r_{e,ucp,m,j}(N_e) * S_{e,ucp,m,j}}{1 - \sum_{m=1}^{R_e} X^r_{e,ucp,m,1}(N_e) * S_{e,ucp,m,1}}$

tiempo de respuesta para un trabajo de tipo r que está ejecutando en la UCP, en la máquina e, con prioridad 2.

$W_{e,disco,r}(N)$ = $W_{transmision} + S_{disco} * (1 + \sum_{e=1}^{E} \sum_{r=1}^{R_e} \sum_{r=1}^{R_e} L_{disco,r}(N_e - e_r))$

tiempo que se demora una operación a disco en un nodo.

$W_{transmision}$ = tiempo que se demora en transmitir la información a través de la red. Para el caso de una red con carga baja se puede asumir como 0. situación que ocurría en la validación, ver sección 4.

El esquema de modelamiento planteado por Ramamurthy tiene la ventaja de representar adecuadamente el esquema de prioridades de Unix. Pero las desventajas están relacionadas con la labor experimental que se debe realizar para encontrar los parámetros y además, el costo computacional de los cálculos involucrados para evaluar el modelo, aspecto que puede ser verificado en las fórmulas anteriores.

Como una primera labor de investigación, se desarrollo un conjunto de programas para evaluar y validar el modelo de Ramamurthy en ambiente local, lo que permitió ganar experiencia [14], para luego aplicar el estudio a un sistema distribuido con las características

que se especifican en la siguiente sección.

4 Validación del Modelo de Unix en ambiente distribuido

El sistema estudiado, para realizar la validación del modelo de Unix distribuido, estaba compuesto por:

- 5 estaciones de trabajo, SUN 3/50, sin disco local, con sistema operativo Sun 3.
- 1 LAN Ethernet, de 10 Mbits/s.
- 1 servidor con un disco de 200 Mbytes, con sistema operativo Sun3.

En este sistema se ejecutaron un conjunto de programas bien definidos, de manera tal que se pudo cuantificar cada llamada al sistema en forma exacta, utilizando el monitor *trace*. Este conjunto de programas escritos en el lenguaje de programación C se llamaron **prueba1**, **prueba2** y **prueba3**. Prueba1 realizaba la lectura de una matriz de 300x300, ejecutaba un grupo de operaciones y luego, escribía la matriz, en un archivo. Prueba2 era equivalente a prueba1 pero no escribía la matriz. Prueba3 correspondía a la compilación de un programa escrito en C, de 320 líneas.

Se efectuaron pruebas con distintas mezclas de estos programas. Estas pruebas entregaron un conjunto de resultados de tiempo de respuesta, los que se muestran en los gráficos 1, 2 y 3. Para poder comparar los tiempos de respuesta calculados con aquellos medidos, se utilizó el monitor de registro por procesos, conocido como *acctcom*, que es hoy en día una herramienta estándar en Unix [8].

También, se obtuvieron resultados del tiempo de respuesta por tipo de trabajo en el servidor, lo que sirvió para determinar que no hubo un retraso mayor debido a la transmisión. Esta situación se debió a que el grupo de procesos (prueba1, prueba2 y prueba3) no hacia un uso apreciable de la red.

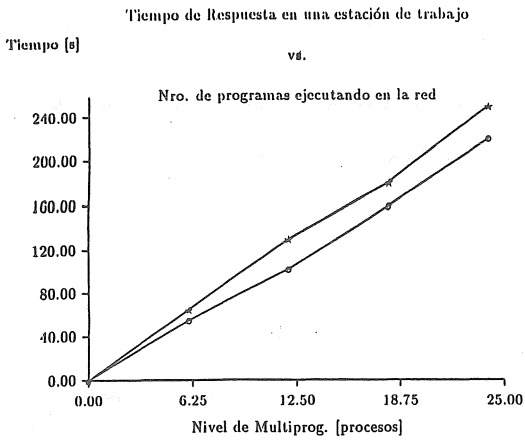
5 Conclusiones

Las conclusiones principales del desarrollo del modelo, aprovechando el modelo desarrollado por Ramamurthy, es que gracias a que la descripción de Unix es bastante cercana a la realidad es posible, sin un costo significativo, modificarlo y adecuarlo a un ambiente distribuido.

La medida de desempeño evaluada, que correspondió al tiempo de respuesta, tiene una precisión adecuada, menor o igual al 10 %, y que permite predecir adecuadamente el comportamiento futuro de un sistema basado en Unix. Es cierto que se requiere de una mayor cantidad de validaciones, pero el hecho de disponer de un modelo como el descrito ya

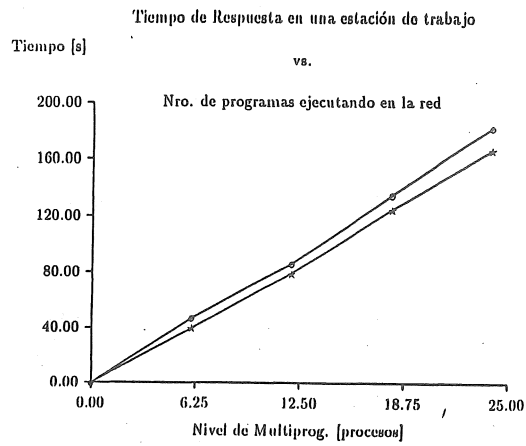
permite avanzar en esta dirección. Un punto de gran interés es poder caracterizar el tipo de proceso que realmente se realiza en una estación de trabajo, en las que en la actualidad ejecutan en ambientes de ventanas y realizan una utilización de recursos excesiva.

Finalmente, el modelo de Unix en ambiente distribuido desarrollado es posible utilizarlo para aumentar su potencialidad, agregando el modelamiento explícito de una red LAN (como Ethernet) [3], u otra, y evaluar una configuración adecuada para un cierto sistema que se desee desarrollar.



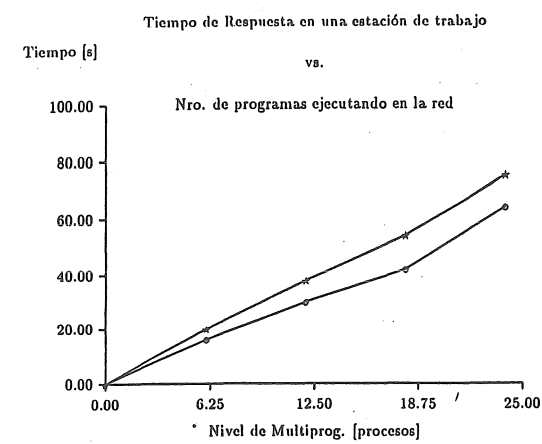
o = tiempo de respuesta calculado para trabajo prueba1
* = tiempo de respuesta real para trabajo prueba1

Gráfico 1. Tiempo de Respuesta en una estación de la red



o = tiempo de respuesta calculado para trabajo prueba2
* = tiempo de respuesta real para trabajo prueba2

Gráfico 2. Tiempo de Respuesta en una estación de la red



o = tiempo de respuesta calculado para trabajo prueba3
* = tiempo de respuesta real para trabajo prueba3

Gráfico 3. Tiempo de Respuesta en una estación de la red

Referencias

[1] M. J. Bach, The Design of the Unix Operating System, Prentice -Hall, Englewood Cliffs, New Jersey, 1986.

[2] K. M. Chandy & D. N. Neuse, Linearizer: A Heuristic Algorithm for Queueing Network Models of Computer Systems, Communications of the ACM, Vol. 25, No. 2, pp. 126-134, February 1982.

[3] J. L. Hammond & P. J. P. ÓReilly, Performance Analysis of L ocal Computer Networks, Addison-Wesley, 1986.

[4] R. Jain, The Art of Computer Systems Performance Analysis, Jo hn Wiley & Sons, 1991.

[5] N. K. Jaiswal, Priority Queues, Academic Press, New York, 196 8.

[6] K. Christian, The Unix Operating System, Wiley-Interscience, 1983.

[7] E. Lazowska et al., Computer Performance Modeling Handbook, Academic Press, New York, 1983.

[8] M. Loukides, Performance Evaluation in Unix, ÓReilly & Asso icattes, 1991.

[9] A. Park and J. C. Becker, Measurements of the Paging Behaviour of Unix, Proceedings of the 1991 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, San Diego, California, pp. 216-217, May 1991.

[10] A. Perez and L. Dowdy, Parameter Interdependencies of File Placement Models in a UNIX System, Proceedings of the 1984 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Cambridge Massachusetts, pp. 15-26, August 1984.

[11] D. R. Peachey et al., An Experimental Investigation of Scheduling Strategies for Unix, Proceedings of the 1984 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Cambridge, Massachusetts, pp. 158-166, August 1984.

[12] M. Reiser & S. S. Lavenberg, Mean value analysis of closed multichain queueing networks, Journal of the ACM, Vol. 27, No. 2, pp. 313-322,April 1980.

[13] G. Ramamurthy, An Analytical Model for Unix Systems, AT&T Technical Journal, Vol. 7, No. 5, pp. 86-99, September/October, 1988.

[14] G. Urrea, Evaluación del desempeño de Unix en ambiente local y distribuido, Memoria de Ingeniería Civil en Computación, U. de Chile, 1991.