Druida: Una Herramienta Basada en Conocimiento para el Afinamiento de Sistemas Unix

Mario Droguett C. , Magister U.C.
Ignacio Casas R., Ph.D.
Francisco Aurtenechea O., M.Phil.
Depto. de Ciencia de la Computación
Escuela de Ingeniería
Pont. Universidad Católica de Chile
Casilla 6177, Santiago Chile

RESUMEN

Este documento describe el análisis, diseño e implementación de un prototipo de sistema experto para el afinamiento del sistema operativo Unix. El paradigma de afinamiento se basa tanto en reglas de cotas de rendimiento, como en modelos analíticos de redes de espera para la evaluación y predicción del rendimiento. El objetivo principal de este trabajo ha sido el estudiar la automatización del afinamiento de un sistema computacional, considerando la identificación de los problemas de rendimiento y de los parámetros a modificar. Se describe el espacio de búsqueda del sistema experto y la combinación de estrategias de búsqueda y modelación.

Palabras Claves: Evaluación del Rendimiento, Afinamiento, Sistemas Expertos.

1. Introducción

La tecnología de los sistemas computacionales modernos demanda recursos escasos y costosos, requiriéndo de procesos continuos de afinamiento, periódicas evaluaciones y rigurosas planificaciones de capacidad, para asegurar un uso eficiente y óptimo que satisfaga los crecientes requerimientos de los usuarios.

Con la creciente complejidad de los sistemas computacionales de aplicación y la proliferación de tales sistemas, el problema de la evaluación del rendimiento y el afinamiento de éstos es una tarea más difícil y llega a ser realizada por profesionales que no se encuentran, en general, lo suficientemente preparados para llevarlas a cabo.

Existe un gran número de excelentes herramientas disponibles para recolectar y manejar información sobre el rendimiento de un sistema. Mediante el examen de la información provista por éstas y comparándola con ciertos valores esperados, es factible identificar un problema de rendimiento y proponer una solución. Sin embargo, esto requiere de una gran cantidad de experiencia y conocimiento para interpretar la, generalmente, muy voluminosa información entregada por las herramientas.

Esta situación satisface, por lo tanto, los requerimientos básicos para la especificación del dominio apropiado para un Sistema Experto[1] [5] [8] [11].

Un Sistema Experto para afinamiento de sistemas computacionales debe proveer las siguientes funciones básicas [6]:

a. Recolectar y Resumir datos de monitoreo del sistema.
b. Correlacionar datos relacionados y no relacionados.
c. Aplicar reglas y juicios de valor sobre los datos.
d. Emitir conclusiones sobre los datos.
e. Recomendar cambios para mejorar el rendimiento.

Las primeras dos funciones han sido cubiertas por numerosas herramientas de monitoreo desarrolladas durante los últimos veinte años [10]. Además, actualmente existen tanto aplicaciones tradicionales, como Sistemas Expertos que abarcan las tres últimas funciones.

---

[1] Los términos *Sistema Experto* y *Sistema basado en Conocimiento* son utilizados como sinónimos en este documento.

Sin embargo, no existe una herramienta computacional que abarque las cinco funciones para diferentes ambientes computacionales [9]. En la práctica es necesario un experto.

En este trabajo se estudia la utilización de Modelos de Redes de Espera en un Sistema Experto de Afinamiento, identificando las ventajas y desventajas de su utilización.

Deseamos determinar la factibilidad de utilizar Modelos Analíticos de Redes de Espera en un Sistema Experto de Afinamiento, obteniendo así una cuantificación del efecto de su utilización para la Evaluación del Rendimiento de sistemas computacionales Unix.

Para comprobar nuestro objetivo hemos desarrollado un prototipo de Sistema Experto, llamado Druida, el cual nos ha permitido analizar el efecto de la utilización de Modelos de Redes de Espera en un Sistema Experto de Afinamiento.

A continuación se describen los conceptos básicos de afinamiento y transmisión del conocimiento. La siguiente sección describe a Druida, los objetivos de su desarrollo, su diseño y cómo caracteriza al sistema. Las secciones siguientes describen el conocimiento manejado por Druida y cómo éste es representado. Se entregan los resultados obtenidos al aplicar el prototipo a un problema real y, finalmente, las conclusiones son acompañadas con una visión del trabajo futuro en el área.

2. Concepto de Afinamiento.

Por afinamiento de sistemas computacionales entendemos, dada la carga de trabajo que el sistema enfrenta, el ajustar ciertos parámetros para obtener un mejor rendimiento.

El trabajo de afinamiento es generalmente realizado a través del uso de un método heurístico, que, en realidad, es una síntesis de reglas formalizadas y experiencia.

Esto no garantiza el encontrar una buena solución al problema, ya que, generalmente, la experiencia es escasa y las reglas no son suficientes. Por último, dada la decisión de solucionar un problema, no es sencillo predecir el efecto real de ésta.

Así, para la construcción de nuestro Sistema Experto, que llamaremos Druida, las reglas son complementadas con modelos cuantitativos. El efecto de esta estrategia, es la capacidad de Druida de predecir el efecto de las recomendaciones que realiza el sistema.

3. Druida.

A continuación se presentan los principios de diseño y consideraciones de evaluación del rendimiento contempladas en la construcción del Sistema Experto Druida.

Druida es una herramienta orientada a apoyar el afinamiento de sistemas Unix System V. Druida, contiene conocimiento sobre Unix, análisis de datos y modelación de sistemas computacionales para:

a. Diagnosticar un problema de rendimiento.
b. Ubicar áreas que puedan ser modificadas para mejorar el rendimiento.
c. Hacer recomendaciones para el afinamiento del sistema.

3.1 Objetivos en el desarrollo de Druida.

La finalidad principal de Druida es actuar como una herramienta para el afinamiento de sistemas computacionales Unix System V.

Para realizar lo anterior obtiene de forma automática las medidas de rendimiento actuales del sistema y, utilizando el conocimiento experto almacenado en su base de conocimientos, determina las modificaciones necesarias para mejorar su rendimiento.

Para llevar el rendimiento de un sistema computacional a un nivel satisfactorio, dos estrategias principales pueden llevarse a cabo: afinar el sistema actual y/o comprar hardware adicional. [16] [2] [3]

El afinamiento es normalmente la opción más barata y debe ser realizada primero. Si un sistema no ha sido recientemente afinado, un considerable aumento en su rendimiento puede ser obtenido a un bajo costo. [3]

3.2 Diseño Conceptual

Para el desarrollo de Druida se elige la orientación a Objetos como paradigma de diseño y programación. Druida se ve como una colección de objetos que realizan tareas específicas y se comunican para obtener información del sistema, requerir información del usuario, realizar el

diagnóstico, y entregar sugerencias para mejorar el rendimiento del sistema. En la figura 1 se muestran las relaciones entre los distintos objetos que conforman Druida.

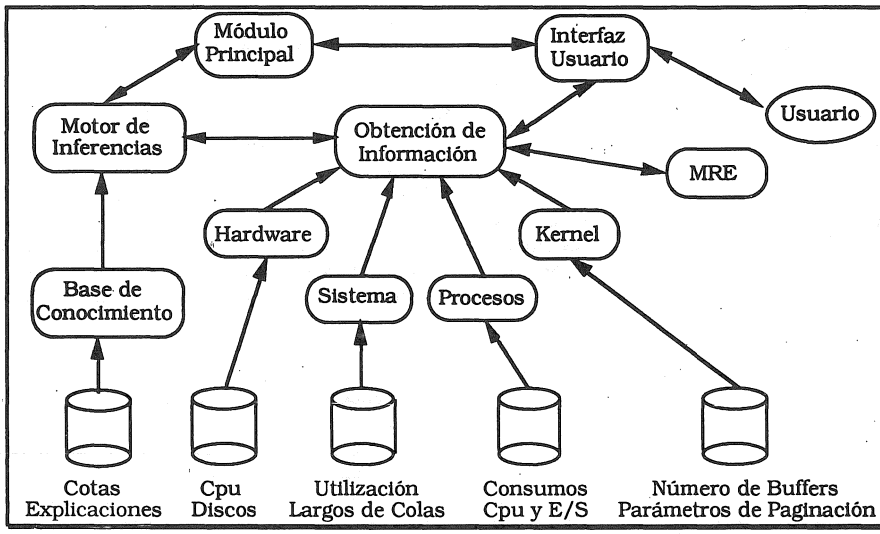

Figura 1. Relaciones entre los Objetos de Druida.

Dada la selección de este paradigma, se utiliza para la implementación de Druida el lenguaje de programación C++[2]. El uso de C++ radica principalmente en la posibilidad de reutilizar otras herramientas ya desarrolladas, la eficiencia del código generado y en la productividad de las tareas de diseño y desarrollo de prototipos.

Los distintos objetos que conforman Druida proveen entonces una serie de métodos para comunicarse con otros objetos y para realizar sus tareas.

4. Adquisición del Conocimiento

El conocimiento necesario para realizar el afinamiento de un sistema computacional es obtenido de múltiples fuentes, siendo las principales los manuales del sistema, publicaciones sobre el tema, opiniones de expertos y experimentación.

En la interacción con expertos se trabajó principalmente con expertos de UNISYS CHILE y administradores experimentados del Departamento de Ciencia de la Computación. La interacción con los primeros se basó principalmente en la discusión de los resultados obtenidos por el prototipo inicial (basado en la información de manuales), y con los segundos se trabajó en la prueba de la Base de Conocimientos y la ampliación de las fuentes de información. Para mayores detalles en este caso específico refiérase a [12].

Los manuales del sistema entregan información que es específica al sistema en estudio. En este sentido proveen procedimientos para detectar y solucionar problemas, así como recomendaciones para aumentar el rendimiento del sistema. Estas recomendaciones son generales, por lo cual deben ser adecuadas al sistema en estudio para ser útiles.

Las publicaciones sobre el tema y las opiniones de expertos no siempre se refieren a sistemas Unix[3]. Es por ésto que la información obtenida de ellos debe ser verificada y ajustada para ser utilizada en un sistema Unix. Para validar esta información se utiliza experimentación.

Los experimentos consisten en medir el sistema, aplicar las medidas obtenidas a un criterio propuesto[4], por ejemplo, para un sistema VMS, y obtener una recomendación sobre la

---

[2] Para información sobre C++ refiérase a [22].

[3] Para publicaciones sobre Unix System V, refiérase a [21].

[4] En [24] una serie de criterios son presentados para diversos tipos de sistemas.

base de este criterio. Luego, se procede a efectuar la modificación recomendada en el sistema Unix, y éste es medido nuevamente. Los resultados obtenidos son analizados para determinar la validez del criterio propuesto, de modo de ajustarlo a un sistema Unix, o determinar que este criterio no es aplicable.

4.1. Parámetros de Afinamiento.

Sobre la base de los objetivos que dan partida al desarrollo de Druida, inicialmente es necesario identificar aquellos parámetros relativos al desempeño de un sistema Unix que son medibles y modificables de manera automática.

La primera tarea a realizar para efectuar un afinamiento es detectar y diagnosticar el problema de rendimiento que éste presenta. Identificado el dispositivo que produce la baja en el rendimiento, una reducción en su carga de trabajo puede aumentar sensiblemente el rendimiento. [6]

4.2. Medidas de Rendimiento.

Una vez identificados, tanto los parámetros modificables automáticamente como los que deben ser modificados de forma manual, y conocida la forma de reconstruir automáticamente un sistema Unix, se deben obtener medidas de rendimiento del sistema en estudio, los valores actuales de los parámetros del kernel y la configuración de hardware del sistema.

Del cúmulo de información de rendimiento que se puede obtener mediante los utilitarios descritos en [23], es necesario definir aquella que es relevante para el estudio del sistema que realiza Druida.

Por experiencias previas en estudios de modelación y opiniones vertidas en publicaciones del área, se eligen como medidas relevantes del rendimiento del sistema Unix las siguientes:

Medida	Componente de Aplicación
Utilización	Cpu, Discos.
Largos de Cola	Cpu, Discos.
Demanda de servicio	Clases de procesos sobre Cpu y Discos.
Actividad de Paging & Swap	Memoria y Partición de swap en disco.
Tiempo de Respuesta	Clases de procesos sobre el Sistema completo.

Además de las medidas señaladas anteriormente, es necesaria información adicional sobre el comportamiento del sistema para realizar el diagnóstico y determinar una solución al problema. Esta información corresponde, por ejemplo, al número de páginas libres en memoria, necesaria para determinar si un aumento en el número de "buffers" del sistema producirá como efecto lateral un aumento en la actividad de swap.

4.3. Utilización de Modelos de Redes de Espera.

En la evaluación del rendimiento de sistemas computacionales, los Modelos de Redes de Espera son la principal herramienta utilizada para analizar y predecir el rendimiento de un sistema computacional [3] [8] [19].

Un Modelo de Redes de Espera (MRE) es una representación abstracta de un sistema computacional, que se apoya en la definición de las Clases de Procesos (Clientes), unidades (Centros de Servicio), y Parámetros de Carga de Trabajo (Tasas de Llegada, Tiempo de Servicio, etc) [3].

El modelo es evaluado por medio de técnicas analíticas (resolución de sistemas de ecuaciones basados en teoría de colas), para obtener una evaluación del desempeño o rendimiento del sistema (Utilización de la CPU, Tiempo de Respuesta, Caudal y Largos de Cola) [16].

Los MRE permiten evaluar el rendimiento del sistema, identificando las causas que limitan este rendimiento. Utilizando MRE podemos, entonces, realizar las tareas de detección y diagnóstico que son deseables en un Sistema Experto.

El uso de MRE permite, además, determinar las formas de mejorar este rendimiento, focalizando el esfuerzo de afinamiento [4]. Apoyamos, así, la determinación de recomendaciones para solucionar los problemas detectados, al focalizar las áreas de análisis.

En los MRE las modificaciones sugeridas para el afinamiento pueden ser reflejadas en el modelo del sistema. Evaluando el modelo se puede predecir el rendimiento futuro del sistema, luego de las modificaciones.[3].

Druida incorpora MRE en su diseño de manera de acceder a las ventajas señaladas anteriormente. Así, la información obtenida del sistema es manejada por Druida desde un nivel de abstracción mayor que el que proveen los utilitarios del sistema.

5. Representación del Conocimiento

Para la representación del conocimiento se elige un esquema basado en reglas. El conjunto de las reglas definidas es utilizado por el Motor de Inferencias para realizar el afinamiento del sistema. A continuación se describen el Motor de Inferencias de Druida, mostrando ejemplos de la representación del conocimiento utilizada, y su relación con los mecanismos de explicación provistos.

5.1 El Motor de Inferencias.

Desde el punto de vista de los objetos que conforman Druida, el Motor de Inferencias de éste no es más que otro objeto con el cual pueden comunicarse, recibir y entregar datos. Mas, en este caso, se incorpora, como Motor de Inferencias de Druida, el motor de inferencias de una herramienta disponible en el mercado, la herramienta incorporada corresponde a KES[5]. La selección de KES se basa principalmente en claridad de uso y disponibilidad para experimentación.

Para incorporar el motor de inferencias en un programa escrito en C++, KES provee una librería de funciones que permiten manejar la información que entrega y/o recibe el motor de inferencia, y controlar el proceso de razonamiento. Provee mecanismos de explicación simples que pueden ser manejados y ampliados por el programa que controla el Motor de Inferencias.

5.2 La Base de Conocimientos.

La Base de Conocimientos de Druida se encuentra escrita en el lenguaje provisto por KES para su Motor de Inferencias. Se representa en ella sobre la base de reglas y comprende dos áreas principales de conocimiento relativas al afinamiento de sistemas Unix. Estas áreas corresponden a Diagnóstico y Sugerencias. Las áreas de Diagnóstico y Sugerencias proveen reglas para:

- **a.** Realizar la identificación de el o los posibles problemas existentes en el sistema en estudio.
- **b.** Obtener los datos necesarios desde el sistema o el usuario y entregar el diagnóstico realizado.
- **c.** Sugerir, dado el problema determinado, la solución más adecuada al problema detectado.

5.3 Reglas y Modelos de Redes de Espera.

Para analizar como se relacionan los Modelos de Redes de Espera y la Base de Conocimientos descrita en base a reglas utilizaremos el siguiente ejemplo:

Supongamos que el motor de inferencias ha determinado que el cuello de botella del sistema en estudio corresponde al sub-sistema de disco. Luego el motor de inferencias intentará determinar una solución a este problema. Para esto, por ejemplo, tratará de satisfacer la siguiente regla:

```
if Balance Disk Tuning = yes          (1)
then
Disk Tuning = Balance Disk.
endif.
```

Esta regla puede interpretarse de la siguiente manera: Para ver si el valor de esta regla es verdadero Druida intentará obtener el valor de la variable **Balance Disk Tuning**, para ésto comparará el valor obtenido con la constante **yes** y, si esto es efectivo, concluirá que se puede obtener una mejora en el rendimiento del sistema realizando un balance de carga en el sub-sistema de disco.

---

[5] KES Knowledge Engineering System. Copyright 1990 Software Architecture & Engineering, Inc. Distribuido por UNISYS Corp.

Para obtener el valor de la variable **Balance Disk Tuning** el motor de inferencias utilizará entonces la siguiente regla:

```
if MRE Balance Disk Test le 0.9          (2)
then
Balance Disk Tuning = yes.
      endif.
```

Para obtener a su vez el valor de **MRE Balance Disk Test** Druida utilizará el siguiente cálculo:

```
MRE Balance Disk Test: real
  (3)
[default: (MRE Balance Disk Response Time) / (MRE Response Time)].
```

Luego, para realizar el cálculo el motor de inferencias debe obtener los valores de las variables involucradas, que en este caso corresponden a **MRE Balance Disk Response Time** y **MRE Response Time**. Para obtener estos valores el motor de inferencias realiza las siguientes consultas:

```
MRE Response Time: real
    (4)
{question: "Ingrese MRE Response Time:",
   "MRE","RTIME"}.
```

```
MRE Balance Disk Response Time: real          (5)
{question:
   "Ingrese MRE Balance Disk Response Time:",
   "MRE","BDISK"}.
```

Estas consultas son generadas por el objeto Motor de Inferencias y enviadas como mensajes al objeto Módulo de Obtención de Información. Este identifica las consultas y las envía al objeto MRE, como se muestra en la figura 2.

Para la Primera consulta (4) el objeto MRE construye un Modelo de Redes de Espera del sistema en estudio[6], enviando para ésto mensajes al módulo de obtención de información para determinar las componentes del modelo y los valores de los parámetros que lo definen. El módulo de obtención de información genera a su vez mensajes a los objetos Hardware, Sistema, Procesos y Kernel para obtener los datos solicitados (ver figura 1).

Luego el modelo construido es resuelto y validado por el objeto MRE[7]. Luego de validar el modelo el valor del tiempo de respuesta promedio persivido por las clases del modelo es retornado al motor de inferencias como valor de la variable **MRE Response Time**.

Para resolver la siguiente consulta (5) el objeto MRE utiliza el modelo original del sistema definido en la consulta anterior, pero en éste, la carga de trabajo al sub-sistema de discos es distribuida de manera uniforme entre los discos que conforman el sub-sistema.

Luego el modelo es resuelto, y el valor del tiempo de respuesta promedio persivido por las clases del modelo modificado (balanceado en disco) es retornado al motor de inferencias como valor de la variable **MRE Balance Disk Response Time**.

Una vez obtenidos estos valores el Motor de inferencias realiza el cálculo indicado en (3), el cual define el valor de la variable **MRE Balance Disk Test**, si este valor es menor que 0.9 indica que la diferencia entre el tiempo de respuesta persibido por un usuario promedio del sistema, y el que persiviría luego de realizado una balance de carga en el sub-sistema de disco tiene una reducción mayor al 10%.

---

[6] Una vez construido el modelo éste se mantiene dentro de Druida. Posteriormente este modelo puede ser modificado y re-evaluado.

[7] Se utilizan para ésto los algoritmos de Análisis de Valor Medio descritos en [16].

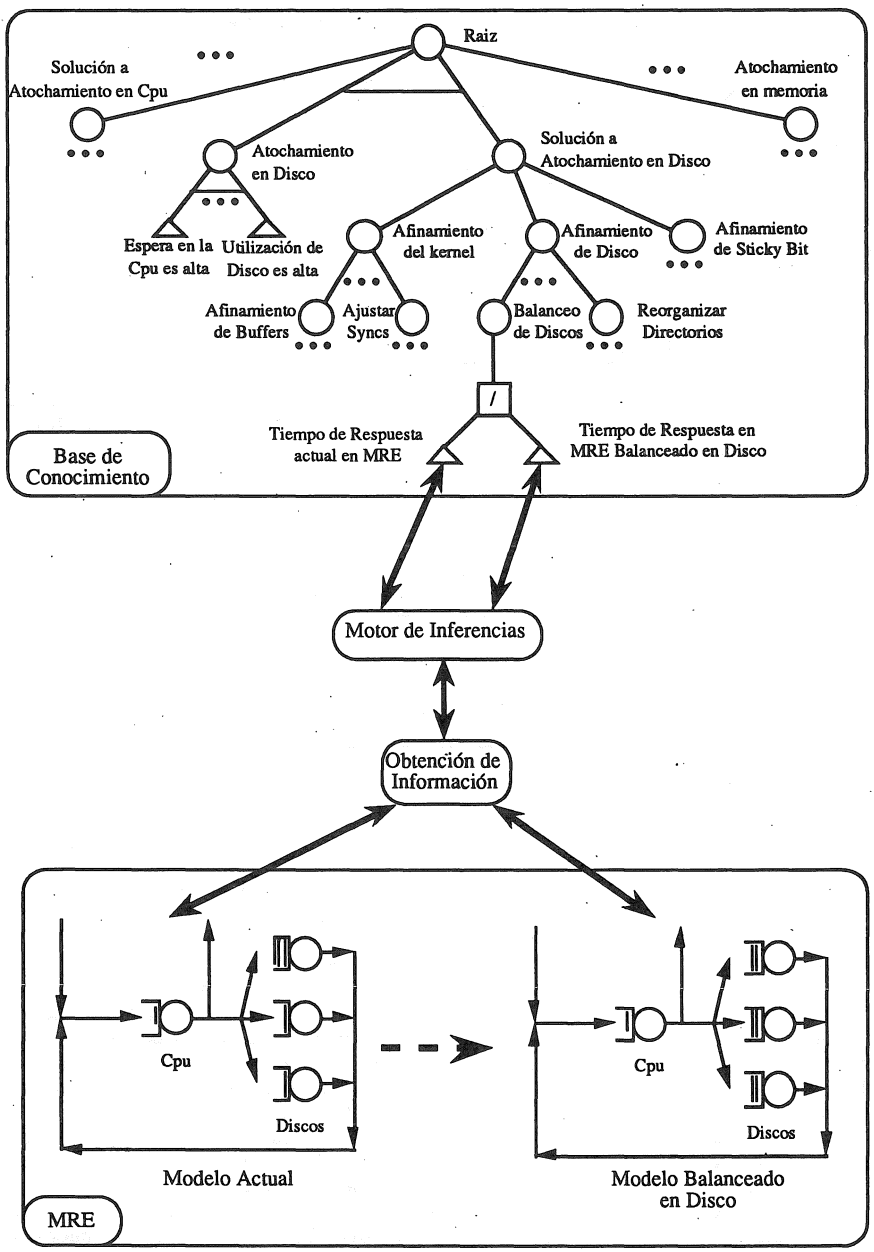

Figura 2. Realación entre la Base de Conocimientos y los Modelos de Redes de Espera.

La condición anterior es verificada por la regla (2), lo que permite hacer verdadera la regla (1), la cual hace al motor de inferencias concluir que una mejora al rendimiento puede ser obtenida mediante un balance de carga en el sub-sistema de disco.

6. Caracterización del Sistema y su Carga de Trabajo.

En Unix System V existe una serie de utilitarios que recolectan y generan informes sobre las medidas de rendimiento del sistema en estudio [20] [17]. Ejemplos de estos utilitarios son el Generador de Reportes de la Actividad del Sistema sar(1)[8], y acctcom(1), que entrega información sobre los procesos ejecutados en el sistema.

La información provista por estas rutinas puede ser utilizada para caracterizar la carga de trabajo del sistema y analizar su rendimiento, pero ésta es muy voluminosa y requiere invertir gran cantidad de trabajo para que llegue a ser útil.

Por esto Druida se apoya, para procesar esta información, en las facilidades que provee una herramienta desarrollada por el Departamento de Ciencia de la Computación de la Universidad Católica: Asterix[9].

Asterix es utilizado para realizar la caracterización de la carga de trabajo del sistema y obtener las medidas de rendimiento del mismo. Druida utiliza la información que entrega Asterix tanto para diagnosticar el sistema en estudio, como para entregar recomendaciones para mejorar su rendimiento.

Cuando el Motor de Inferencia de Druida requiere, por lo tanto, algún dato específico en su cadena de razonamiento, y éste no puede ser inferido a partir de otros, genera un mensaje al módulo de obtención de información. Este, a su vez, obtiene el dato requerido desde el sistema.

7. Resultados obtenidos

Se describe a continuación la utilización de Druida en el afinamiento de un sistema Unix. El sistema es descrito en cuanto a sus componentes de hardware y la carga inicial que enfrenta.

El sistema en estudio es 'Puyehue', un equipo UNISYS U6000/55 instalado en el Departamento de Ciencia de la Computación de la Universidad Católica de Chile. Puyehue cuenta con un sistema operativo Unix System V versión 3.2, y se encuentra dedicado a investigación. Las características técnicas de Puyehue son las siguientes:

- **a.** Cpu INTEL 80386 a 33 Mhz.
- **b.** Procesador matemático esclavo Weitek.
- **c.** 16 Mb de Memoria Principal.
- **d.** 1 unidad de Cinta.
- **e.** 3 discos SCSI que totalizan 1,2 Gb.
- **f.** 8 terminales uvt1224.
- **g.** 10 80386sx conectados en red Ethernet.
- **h.** Conexión Ethernet a la Red Interdepartamental de la Escuela de Ingeniería.

Los días 8, 10 y 15 de Octubre de 1991 fueron presentados a Druida, para la obtención de recomendaciones para mejorar el rendimiento. Cada día es monitoreado desde las 9:00 horas hasta las 19:00 horas.

Al analizar la información de rendimiento de estos días, y resolver los modelos asociados, Druida señala que no existen atochamientos importantes. Luego, indica que el rendimiento puede ser mejorado efectuando un balanceo de la carga que enfrentan los discos.

Druida no recomienda en este caso modificaciones a los parámetros del kernel, ya que su modificación no tiene un impacto significativo en el rendimiento.

Para cuantificar objetivamente el efecto del afinamiento sugerido por Druida se utiliza un experimento[10] que permite determinar el tiempo de respuesta percibido por el usuario para un comando estandar del sistema.

---

[8] Para información sobre los nombres de comandos en Unix System V refiérase a [23]

[9] Asterix es una herramienta de apoyo al Análisis, Supervisión, y Evaluación de Rendimiento de Sistemas Unix. Permite monitorear y obtener información del sistema en estudio, enfocada ésta hacia un estudio de Modelación. Para mayor información refiérase a [4].

[10] Este experimento se encuentra propuesto en [18] y es utilizado para analizar el cambio en el rendimiento de un sistema VAX11-780 con sistema operativo Unix.

En este caso el comando estandar corresponde a la compilación en C++ de un trozo de código de aproximadamente 700 líneas. El experimento consiste en realizar una compilación de este módulo cada 5 minutos, utilizando para ello el utilitario crontab de Unix. Mientras se realiza la prueba el sistema mantiene u operación normal, atendiendo otros procesos y usuarios.

Para medir el tiempo de respuesta se utiliza el comando timex de Unix, el cual permite obtener tanto el tiempo de respuesta que percibe el usuario, como los consumos del proceso en el sistema.

Luego de realizado el afinamiento, el sistema es medido nuevamente los días 29 y 30 de Octubre, y el día 20 de Noviembre. Cada día es monitoredo desde las 9:00 horas hasta las 19:00 horas.

Analizaremos los resultados obtenidos del afinamiento mediante la técnica de gráficos de "Caja y Bigote" o "Centro y Periferia" ("Boxplot" o "Box and whiskers plot") [13] [15].

Esta técnica permite representar gráficamente los datos obtenidos de un modo coherente, cuidando que los valores extremos obtenidos en los datos no afecten la percepción del fenómeno en estudio.

En la figura 3 se compara, mediante gráficos de caja y bigote, los tiempos de respuesta del sistema a las compilaciones antes y después del afinamiento propuesto por Druida. Para cada día de observación se presentan los valores mínimo, medio y máximo de respuesta del sistema a las compilaciones, medidos en segundos. Se representa además en el gráfico las utilizaciones de Cpu promedio de cada día.

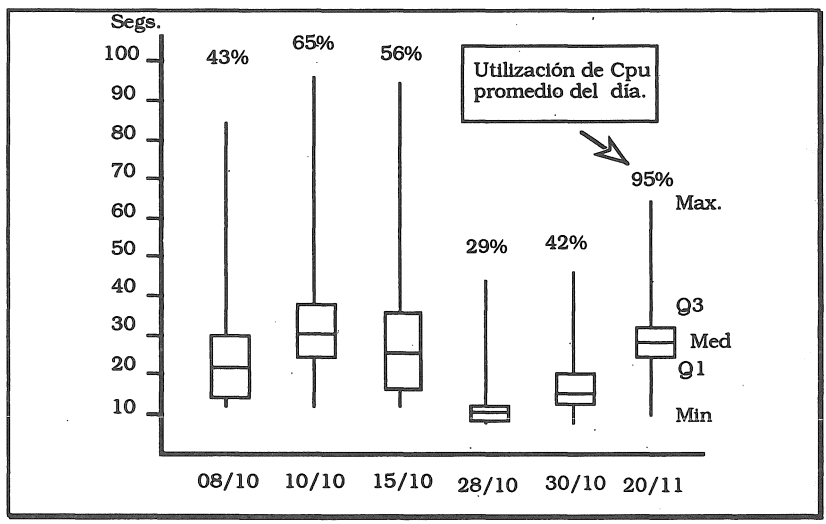

Figura 3. Respuesta del Sistema antes y después del Afinamiento.

Vemos así que el afinamiento propuesto por Druida ha producido una reducción visible del tiempo de respuesta del sistema. Si se comparan los días 8 y 30 de Octubre podemos ver que para utilizaciones promedio de Cpu similares el tiempo de respuesta del sistema es menor, y la dispersión del mismo es también menor.

Al analizar el día 20 de Noviembre vemos que al enfrentar el sistema una utilización de Cpu muy superior la respuesta del sistema es en promedio similar a la obtenida para utilizaciones entre 50 y 60%, pero la dispersión de estas respuestas es muy inferior.

Esto nos permite señalar entonces que el efecto del afinamiento realizado ha llevado a una reducción notable en el tiempo de respuesta del equipo a las compilaciones. El análisis anterior es descrito en mayor detalle en [12].

8. Conclusiones

Se ha descrito Druida, una herramienta basada en conocimiento para el afinamiento de sistemas computacionales Unix System V.

Se ha observado, durante el uso de la herramienta como prototipo, que ésta no sólo ayuda a operadores y administradores con poca experiencia, sino que también ayuda a operadores de sistema expertos.

Al obtener información útil desde grandes volumenes de datos, y entregar medidas de rendimiento comprensibles sobre los sistemas y al liberarlos de experimentaciones innecesarias, los administradores son más productivos en su trabajo.

Otro aspecto importante observado en el desarrollo de este trabajo es que, para tener éxito en el afinamiento, las reglas basadas en cotas de rendimiento no son suficientes, y es necesario complementar este conocimiento con técnicas cuantitativas y modelos.

Finalmente, como trabajo futuro, se planea la incorporación de capacidad de aprendizaje a Druida, de modo que éste se auto-afine con respecto a sus recomendaciones y los valores de la cotas de rendimiento para el afinamiento del sistema específico en que se encuentra instalado.

Asimismo, la definición automática de las clases para un MRE es fundamental para una mayor efectividad en el diagnóstico del sistema y la realización de predicciones.

AGRADECIMIENTOS

Este trabajo ha sido realizado en el contexto del proyecto Asterix del Depto. de Ciencia de la Computación de la P.U.C., con financiamiento parcial de los proyectos Fondecyt 91-0786, DIUC 90/8, y convenio DICTUC-Unisys (Chile) Corp. Se agradece especialmente la colaboración prestada por los expertos de Unisys (Chile) Corp.

REFERENCIAS

[1] Berry Robert, Hellerstein Joseph: "Experts Systems for Capacity management." 1990. CMG Transactions Summer 1990.

[2] Cady J., Howarth B. "Computer Systems Performance Management and Capacity Planning", Prentice Hall, Sydney, 1990.

[3] Casas I.,"Modelación de Sistemas Computacionales.". Apuntes de Clases. 1989.Universidad Católica de Chile, Escuela de Ingeniería, Departamento de Ciencia de la Computación.

[4] Casas Ignacio, Aurtenechea Francisco, Bustos Eduardo, Droguett Mario: "Diseño de una Herramienta para el Análisis, Supervisión y Evaluación del Rendimiento de Sistemas Unix." 1991. Universidad Católica de Chile, Escuela de Ingeniería, Departamento de Ciencia de la Computación.

[5] Dawson Jeffrey, Shubin Hal : "Interactive Expert Assistant for Computer Performance Evaluation." 1988. Conference Proceedings of CMG '88.

[6] Deese Donald : "Designing an Expert System for Computer Performance Evaluation." 1988. Conference Proceedings of CMG '88.

[7] Domansky Bernard, Soberman Sidney : "Expert Systems for Computer Performance Evaluation: On MetaRules and Knowledge Engineering." 1988. Conference Proceedings of CMG '88.

[8] Domansky Bernard: "An Expert System's Tutorial for Computer Performance evaluation." 1988. Conference Proceedings of CMG '88.

[9] Domansky Bernard, Soberman Sidney : "Expert Systems for CPE: On Evolution and Data Analysis." 1989. Conference Proceedings of CMG '89.

[10] Domansky Bernard: "A PROLOG-based Expert System for Tuning MVS/XA." 1989. Performance Evaluation Review, Febrero 1989.

[11] Domansky Bernard: "An Overview of Expert Systems." 1990. CMG Transactions Summer 1990.

[12] Droguett Mario: "Una herramienta basada en conocimiento para el afinamiento de sistemas UNIX". Tesis de Magister, Escuela de Ingeniería, Pontificia Universidad Católica de Chile, 1992.

[13] du Toit, S.H.C., Steyn A.G.W., "Graphical Exploratory Data Analysis". Springer-Verlag, New York, 1986.

[14] Roberto Holtheuer, Carlos Vizcaya "Desarrollo de una Metodología para la Evaluación y Selección de Sistemas Computacionales" Memoria de título Ingeniería, Escuela de Ingeniería, Pontificia Universidad Católica de Chile, 1989.

[15] Ishikawa, Kaoru, "Guide to Quality Control". Asian Productivity Organization, 1987.

[16] Lazowska E., et al: "Quantitative System Performance, Computer System Analysis Using Queueing Network Models" Prentice-Hall New Jersey 1984.

[17] Lee Paul, Farrel Brian, Dronamraju Krishna: "Performance Measurement and Evaluation in Unix System V." 1989. Conference Proceedings of CMG '89.

[18] Madhar S. Phadke: "Quality Engineering using robust design", Prentice-Hall, New Jersey, 1989.

[19] Ovalle F. "Modelación y Evaluación del desempeño de un Sistema Computacional Centralizado". Memoria de Título, Departamento de Computación, Escuela de Ingeniería, Pontificia Universidad Católica de Chile, Abril de 1988.

[20] Ramamurthy G. : "A Capacity Planning and Configuration Modeling Methodology for Unix Systems" 1989. Conference Proceedings of CMG '89.

[21] Samadi Behrokh: "TUNEX: A Knowledge-Based System for Performance Tuning of the Unix Operating System." 1989. IEEE Transactions on Software Engineering, Julio 1989.

[22] Stroustrup Bjarne: "The C++ Programming Language" Addison-Wesley 1985.

[23] Unisys Corp. "Unix System V : Administrator Guide" Versión de sistema 3.0.

[24] Zimmer Harry : "Rules of Thumb '90." 1990. CMG Transactions Spring 1990.