Universitat de Lleida
Escola Politècnica Superior
Ingeniería Técnica en Informática de Sistemas

Trabajo final de carrera

MoniTo 4.0, una herramienta de monitorización de clusters

Autor: Sebastián Avellana Rámiz

Director: Francesc Solsona Tehàs

Septiembre de 2008

MoniTo 4.0, una herramienta de monitorización de clusters

Sebastián Avellana Ramiz

19 de septiembre de 2008

Prólogo

Las aplicaciones comerciales y científicas cada vez necesitan una mayor capacidad de cálculo. Para poder afrontar estos requerimientos se idearon una serie de soluciones que pretendían optimizar la capacidad de cómputo de las computadoras.

Las computadoras paralelas son máquinas que disponen de cientos y a veces miles de microprocesadores estrechamente coordinados. Estas máquinas tienen la capacidad de cálculo más elevada pero con un importante inconveniente, su elevado coste y que inmediatamente quedan obsoletos.

Una alternativa, que mantiene las prestaciones reduciendo los costes, es el *cluster* y están formados por computadoras de uso general y relativo bajo costo, interconectados por redes locales de baja latencia y gran ancho de banda.

Los *clusters* permiten obtener altos rendimientos de cómputo, de una forma muy económica, motivo por el cual gran cantidad de grupos de investigación y docentes se puedan permitir este sistema para poder aumentar su capacidad de cómputo.

Cuando se administra un *cluster*, es importante poder conocer en todo momento cómo se comporta el sistema para poder analizar y mejorar el rendimiento de dicho *cluster*. A este proceso de visualización y extracción de la información se le conoce como monitorización.

En este trabajo se recupera la herramienta MoniTo, implementada en trabajos anteriores para realizar dos mejoras importantes, la primera de ellas consiste en modificar su implementación utilizando las librerías LibGtop mientras que la segunda es una revisión de la unidad de visualización para mejorar su interacción con el usuario y aprovechar la potencia a la hora de diseñar interfaces web de las hojas de estilo en cascada CSS y de la implementación con lenguajes interpretados en el lado del servidor como PHP.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Conceptos previos

Cada día las exigencias de cálculo de las aplicaciones comerciales son más elevadas. Los sistemas informáticos necesitan aumentar su capacidad de cálculo de forma continuada. La tecnología avanza a pasos agigantados y los ordenadores disponen cada día de una mayor potencia de cálculo. Pero aún así, las necesidades de cálculo siempre superan al avance técnico.

Para poder asumir esta alta demanda de requerimientos se ha trabajo en la creación de supercomputadores. Para llegar a este objetivo existen cuatro tecnologías importantes:

La tecnología *"vector processor"* fue creada por Seymour Cray, considerado el padre de la supercomputación y fundador de la compañía *Cray Research*. Desde esta compañía diseñó y construyó las computadoras de propósito general con mayor rendimiento. Estas nuevas tecnologías permitían la ejecución de operaciones aritméticas en paralelo.

Los *"Massively Parallel Processor"* son sistemas informáticos cuya arquitectura está formada por varios unidades aritméticas independientes o procesadores completos, los cuales trabajan en paralelo. El término *Massively* indica sobre el orden de millares de estas unidades. En esta clase de computación, el procesamiento de todos los elementos están unidos entre sí como si fuera una gran computadora.

Otra tecnología consiste en aprovechar los ciclos de CPU ociosa en miles de ordenadores personales a lo largo del mundo, estos equipos se comunicarán a través de Internet y las tareas deben estar divididas en bloques de cálculo lo suficientemente independientes para no necesitar comunicarse y ni unirse durante horas. Un ejemplo de esta tecnología es BOINC, un proyecto de código abierto de computación voluntaria con ordenadores personales interconectados, estos proyectos pueden llevar el apoyo de universidades e instituciones, como ZIVIS surgido de la colaboración entre la Universidad de Zaragoza y el Ayuntamiento de la capital aragonesa.

Por último se introducirá la tecnología que trata este trabajo, la *Computación Distribuida* como modelo para resolver problemas de computación masiva utilizando un gran número de computadoras organizadas en *clusters* organizados en una infraestructura de telecomunicaciones distribuida, estas comunicaciones son a través de redes de área local de baja latencia y gran ancho de banda. El Doctor Thomas Sterling[1] lo define como una "arquitectura de computadora paralela basada en unir máquinas independientes, integradas por medio de redes de interconexión, para obtener un sistema coordinado capaz de procesar una carga".

La tecnología de *clusters* ha evolucionado en apoyo de actividades que van desde aplicaciones de supercómputo y software de misiones críticas, servidores Web y comercio electrónico, hasta bases de datos de alto rendimiento, entre otros usos.

Los clusters se pueden catalogar en tres tipos, esta clasificación no es estricta y el propio cluster puede presentar combinaciones de estas capacidades.

Un cluster de alto rendimiento está diseñado para dar altas prestaciones en cuanto a capacidad de cálculo, su utilización dependerá del tamaño del problema por resolver y el precio de la máquina necesaria para resolverlo.

[1] Profesor de Informática en el *Center for Computation & Technology* de la *Louisiana State University*

Un cluster de alta disponibilidad se caracteriza por compartir los discos de almacenamiento de datos y por estar constantemente monitorizándose entre sí.

Un cluster de balanceo de carga o está compuesto por varios nodos que actúan como *frontend* del cluster, y reparten las peticiones de servicio que reciba el cluster, a otros nodos que forman el *backend* de éste. Un tipo concreto de cluster cuya función es repartir la carga de proceso entre los nodos en lugar de los servicios es el cluster openMosix.

Hay que hacer mención a una propiedad muy deseable en un entorno cluster, la escalabilidad, para permitir el manejo del crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

La construcción de los ordenadores del cluster es más fácil y económica debido a su flexibilidad: pueden tener todos la misma configuración de hardware y sistema operativo *(cluster homogéneo)*, diferente rendimiento pero con arquitecturas y sistemas operativos similares *(cluster semi-homogéneo)*, o tener diferente hardware y sistema operativo *(cluster heterogéneo)*.

En general, un cluster está compuesto por:

- Nodos, estos pueden ser simples ordenadores, sistemas multiprocesador o estaciones de trabajo.
- Sistema operativo.
- Conexión de red entre nodos.
- Capa de abstracción entre el usuario y el sistema operativo llamada *Middleware* (OpenMOSIX, p.e.).
- Protocolos de comunicación y servicios
- Aplicaciones.

Dentro del campo de las aplicaciones para un cluster se puede encontrar aplicaciones de cómputo en paralelo, aplicaciones científicas, y como en el caso que trata este trabajo, aplicaciones de monitorización del cluster, entre otras muchas.

MoniTo, es una herramienta de monitorización de clusters, obtiene los datos de diversos parámetros mediante librerías LibGTop (en su última versión), utiliza una estructura cliente-servidor y muestra gráficamente la información obtenida mediante una interfaz web. Esta estructura se explicará más detalladamente en los siguientes capítulos, pero antes se van a explicar una serie de conceptos relacionados por las interfaces web y los servidores que se verán a lo largo de todo el trabajo.

Apache Servidor HTTP de código abierto multiplataforma.

HTML Lenguaje diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

XHTML Versión XML de HTML, pensado para sustituir a HTML como estándar para las páginas web.

CSS Hojas de estilo en cascada *(Cascading Style Sheets)* utilizadas para definir la presentación de un documento XHTML.

PHP Lenguaje de programación interpretado usado para la creación de páginas web dinámicas.

Applet Java Aplicación escrita en Java que se ejecuta desde un navegador web utilizando la *Java Virtual Machine*.

1.2. Motivación

La motivación de este trabajo es portar la implementación de MoniTo a las librerías LibGTop. En versiones anteriores de esta herramienta los datos se obtenían del sistema de ficheros */proc*, pero como se explicará en 2.1.2 este sufre cambios en cada versión del kernel, lo cual crea una fuerte dependencia de la aplicación MoniTo, con la versión del núcleo que ejecuta la máquina sobre la cual trabaja. Este problema se utiliza obteniendo la información de sistema con las librerías LibGTop, de las cuales se hablará en profundidad en la sección 2.1.

1.3. Estado del arte

En esta sección se va a realizar un inciso sobre algunas de las herramientas de monitorización presentes en la actualidad.

Ganglia es un sistema de monitorización distribuido y escalable. Su arquitectura es cliente/servidor, su diseño jerárquico y utiliza tecnologías como XML para la representación de datos y el sistema de ficheros *procfs* para obtenerlos. Las estructuras de datos y algoritmos están cuidadosamente seleccionados para conseguir una intrusión muy baja en cada nodo y una alta concurrencia. Ganglia es utilizado para enlazar clusters a través de varios campus universitarios alrededor del mundo y se puede escalar para manejar clusters con 2000 nodos.

Ganglia tiene la siguiente arquitectura: Esta formado por tres módulos *gmond*[2], *gmetad*[3] y el portal web de Ganglia. En cada cliente habrá un servicio *gmond* que recoge datos sobre la computadora, esta información será almacenada en un documento XML, el cual se enviará a través de una dirección de *multicast*. El servidor *gmetad* deberá recolectar la información enviada por el grupo de nodos o de otros servidores *gmetad*, de esta manera se establecerá la monitorización de varios clusters. El portal web estará situado en el mismo equipo que el demonio *gmetad* y muestra gráficas de los datos recogidos por los demonios *gmond*.

Las ventajas de Ganglia son la intuitiva interfaz del portal web, los datos mostrados como formulario XML, la escalabilidad y posibilidad de enlazar varios clusters, su fácil instalación, la posibilidad de ejecutar comandos en las máquinas monitorizadas y por último, Ganglia forma parte de las más importantes distribuciones para clusters. Su gran desventaja es que la escritura de datos en XML es exigente en recursos de la computadora para su análisis y puede causar una importante carga en la red.

CluMon es el sistema de monitorización de clusters desarrollado por el *National Center for Supercomputing Applications* de los Estados Unidos y utilizado por el *Silicon Graphics Performance Co-Pilot*. Su principal característica es su integración con el sistema de gestión de trabajos OpenPBS. CluMon está formado por demonios PCP[4], una base de datos relacional y un portal web. El demonio PCP está instalado en los clientes y su función es recolectar el estado de ciertos recursos. El demonio *clumond* está instalado en el servidor, su función es recoger los datos del demonio PCP y almacenarlos en la base de datos. Junto a al servicio PCP, CluMon también recoge datos de los procesos desde el servidor PBS[5]. El estado de los clientes, sus procesos y los trabajos PBS son monitorizados a través del portal web, este portal permite obtener información detallada sobre los trabajos PBS y gráficas de los trabajos realizados en los clientes.

Las ventajas de CluMon son su integración con el sistema de gestión de trabajos PBS, la información sobre los procesos de ciertos clientes y la posilibidad de instalar sensores arbitrarios. Por contra, no es posible enlazar varios clusters, sólo el parámetro de carga permite gráficas y la implementación no es portable.

Las herramientas explicadas anteriormente tienen en común su arquitectura cliente/servidor, en el cliente se instala un demonio capaz de recoger los datos del sistema mientras que en el servidor recoge los datos de los clientes y los muestra al usuario mediante un portal web. MoniTo ha seguido esta arquitectura en todas sus versiones hasta la actualidad.

Otras herramientas de monitorización utilizan una arquitectura basada en agentes móviles, estos son programas inteligentes y autónomos que pueden moverse a través de la red, buscando servicios que el usuario necesita e interactuando con ellos. Estos sistemas usan servidores especializados para interpretar el comportamiento y las comunicaciones del agentes con otros servidores. Un agente móvil tiene una inherente autonomía que le permite navegar por la red, para ello pregunta si debe ser enviado a otros nodos y son capaces de ejecutarse en varias máquinas sin necesidad de que su código esté instalado en cada uno de los ordenadores que visita.

1.4. Objetivos

Este trabajo tiene dos objetivos principales. Por un lado portar la implementación de los demonios de MoniTo a las librerías LibGTop, de esta manera conseguiremos que la herramienta sea independiente de la versión del núcleo que ejecutan las máquinas del cluster.

[2]Ganglia Monitoring Daemon
[3]Ganglia Meta Daemon
[4]Performance Co Pilot
[5]Portable Batch System

Por el otro lado, se detectó que la unidad de visualización de MoniTo se encontraba llena de etiquetas HTML obsoletas lo que unido a una estructura bastante enredada, motivó separar (en la medida de lo posible) la parte gráfica de los lanzadores de MoniTo, además se aprovechó para depurar el código de la unidad de visualización para hacerlo más sencillo y permitir nuevas funcionalidades gracias a la potencia de las hojas de estilo en cascada CSS y lenguajes interpretados del lado del servidor como PHP.

Los pasos seguidos para alcanzar estos objetivos han sido:

1. Eliminar las lecturas de */proc* en los demonios de MoniTo.
2. Obtener la información del sistema mediante las librerías LibGTop.
3. Eliminar, en la medida de lo posible, las llamadas relacionadas con la interfaz gráfica de los CGI's.
4. Integrar el código HTML en ficheros PHP.
5. Separar el contenido del diseño utilizando CSS.

1.5. Contenidos

El contenido de la memoria es el siguiente:

- **Capítulo 2**: Explica las diferentes tecnologías utilizadas en el desarrollo de MoniTo 4.
- **Capítulo 3**: Detalla la implementación de MoniTo 4 y explica las diversas pruebas que se han realizado.
- **Capítulo 4**: En este capítulo se incluirán las conclusiones del trabajo y posibles trabajos futuros.

Capítulo 2

Tecnologías utilizadas

2.1. Librerías LibGTop

Las *LibGTop* son unas librerías que permiten obtener información específica de un sistema como la memoria y los ciclos de procesador utilizados por los procesos en ejecución.

Aunque *LibGTop* sea una parte del entorno de escritorio GNOME, la interfaz principal de *LibGTop* es totalmente independiente de cualquier entorno de escritorio, así que pueden ser utilizadas con librerías en cualquier aplicación bajo licencia GPL.

La implementación de los demonios de MoniTo 4 utiliza estas librerías para obtener la información que posteriormente mostrará al usuario, mientras que en versiones anteriores obtenía los datos del sistema de ficheros */proc*. En los apartados 2.1.1 y 2.1.2 se intentará justificar las razones que han motivado el cambio en la implementación de MoniTo.

2.1.1. El sistema de ficheros /proc

Procfs es un sistema de ficheros virtual. En un sistema Linux se monta en */proc* y proporciona múltiple información incluyendo acceso a información del kernel desde el espacio de usuario. Su principal característica es que permite procesar la información como texto plano. Múltiples comandos de Linux (como `ps`, `top` o `pstree`) obtienen la información del sistema de ficheros *procfs*.

Todas las distribuciones de Linux montan automáticamente el sistema de archivos */proc* al iniciar el sistema. No es esencial que el montaje se realice en ese momento pero varias aplicaciones confían en encontrarlo montado al ejecutarse, por lo tanto, en la práctica */proc* se monta en el arranque.

El comando `mount` muestra los sistemas de archivos montados en el sistema. Si se selecciona la linea correspondiente a *proc* se observa como se encuentra montado:

```
proc on /proc type proc (rw,noexec,nosuid,nodev)
```

Esta línea indica que *procfs* se encuentra montado en */proc* y es un sistema de archivos de tipo *proc*. Sus opciones lectura y escritura *(Read & Write)*, *noexec* prohíbe la ejecución directa de binarios, *nosuid* impide que los bits de identificador de usuario y de grupo tengan efecto.

Por último la opción *nodev* prohíbe la creación de dispositivos en */proc*. Estas opciones quedan justificadas por la función de */proc*, que no es otra que ofrecer en espacio de usuario la lectura de información escrita por el kernel. El resto de funciones al ser innecesarias se prohíben en el montaje.

La ejecución del comando `ls` (figura 2.1) permite ver los contenidos del directorio */proc* y servirá para apoyar la explicación de los principales ficheros que contienen la información mas destacada del sistema.

Se observa que `/proc` contiene un directorio para cada proceso en ejecución (incluidos los procesos del kernel) de la forma `/proc/PID`. Algunas informaciones que aporta son:

- `/proc/PID/cmdline` contiene la linea de comando que ha originado dicho proceso.

```
:/proc$ ls -CF
1/      1966/  4138/  4670/  5138/  5320/        filesystems  scsi/
12058/  1967/  4140/  4702/  5139/  5424/        fs/          self@
12078/  1989/  4141/  4703/  5141/  5461/        ide/         slabinfo
12083/  2/     4142/  4708/  5146/  6/           interrupts   stat
12087/  2122/  4143/  4775/  5148/  7/           iomem        swaps
12090/  2123/  4413/  4826/  5150/  8133/        ioports      sys/
12101/  2178/  4507/  4899/  5151/  acpi/        irq/         sysrq-trigger
12102/  2334/  4559/  4913/  5169/  asound/      kallsyms     sysvipc/
12127/  2531/  4561/  5/     5174/  buddyinfo    kcore        tty/
12130/  3/     4582/  5063/  5195/  bus/         key-users    uptime
12131/  30/    4598/  5103/  5203/  cmdline      kmsg         version
12184/  31/    4599/  5106/  5229/  cpuinfo      loadavg      version_signature
12251/  32/    4605/  5107/  5232/  crypto       locks        vmcore
123/    3365/  4606/  5109/  5239/  devices      meminfo      vmstat
142/    3547/  4613/  5112/  5241/  diskstats    misc         zoneinfo
143/    3548/  4626/  5114/  5248/  dma          modules
144/    3862/  4629/  5121/  5251/  dri/         mounts@
145/    3880/  4643/  5122/  5267/  driver/      mtrr
1948/   4/     4656/  5129/  5312/  execdomains  net/
1949/   4137/  4669/  5135/  5317/  fb           partitions
```

Figura 2.1: Contenido del directorio */proc*

- `/proc/PID/cwd` es un enlace simbólico al directorio de trabajo actual del proceso.
- `/proc/PID/task` es un directorio que contiene enlaces duros a las tareas iniciadas por el proceso o por su padre.

En las siguientes lineas se analizarán algunos ficheros importantes de `/proc`:

- `/proc/cmdline` da información sobre los parámetros que se le pasan al kernel en el momento del arranque.
- `/proc/cpuinfo` contiene información sobre el procesador, como el identificador del fabricante, el modelo, la frecuencia a la que trabaja, tamaño de la memoria caché etc.
- `/proc/devices` lista los dispositivos de carácter y de bloque actualmente configurados.
- `/proc/diskstats` proporciona información (incluyendo el número de dispositivo) para cada disco lógico.
- `/proc/filesystems` contiene la lista de los sistemas de ficheros soportados por el kernel.
- `/proc/meminfo` resume como maneja el kernel la memoria del sistema.
- `/proc/modules` es uno de los ficheros más importantes dentro del sistema de archivos `/proc`. Su misión es mostrar los módulos que se han cargado en el kernel y da cierta información sobre sus dependencias.
- `/proc/partitions` lista el número de dispositivo, su tamaño en bloques y el nombre por el cual es identificado como partición existente.
- El directorio `/proc/net` contiene muchisima información de utilidad sobre las conexiones de red, las interfaces, estadísticas de envío y recepción de paquetes.

Como se observa, los ficheros de */proc* aportan información sobre los procesos desde el espacio de usuario en vez que desde el espacio del kernel. De esta funcionalidad se aprovechan múltiples aplicaciones y comandos, como por ejemplo `ps`, comando encargado de devolver un resumen de los principales procesos del sistema. Toda su ejecución es en modo usuario leyendo los datos de *procfs*.

2.1.2. Modificaciones en */proc* según rama del kernel

La implementación de */proc* no se mantiene idéntica a lo largo del ciclo de desarrollo del kernel de Linux, según la rama de kernel que ejecute la máquina puede haber una serie de diferencias en su implementación, que se explicarán en este apartado.

2.1.2.1. Sistema de numeración del núcleo

Como previo, conviene explicar el significado de la numeración del kernel. Hasta que empezó el desarrollo de la serie 2.6 del núcleo, existieron dos tipos de versiones del núcleo:

- Versión de producción: La version de producción, era la version estable hasta el momento. Esta version era el resultado final de las versiones de desarrollo o experimentales. Cuando el equipo de desarrollo del núcleo experimental, decidía que tenia un núcleo estable y con la suficiente calidad, se lanzaba una nueva versión de producción ó estable. Esta versión era la que se debía utilizar para un uso normal del sistema, ya que eran las versiones consideradas más estables y libres de fallos en el momento de su lanzamiento.

- Versión de desarrollo: Esta versión era experimental y era la que utilizaban los desarrolladores para programar, comprobar y verificar nuevas características, correcciones, etc. Estos núcleos solían ser inestables y no se debían usar sin saber lo que se hacia.

Los números de las versiones de las ramas por debajo de la 2.6 se interpretan de la siguiente manera. Las versiones del núcleo se numeraban con 3 números, dispuestos de la forma `AA.BB.CC`.

- `AA`: Indicaba la serie/versión principal del núcleo. Solo han existido la 1 y 2. Este número cambiaba cuando la manera de funcionamiento del kernel había sufrido un cambio muy importante.

- `BB`: Indicaba si la versión era de desarrollo ó de producción. Un número impar, significaba que era de desarrollo, uno par, que era de producción.

- `CC`: Indicaba nuevas revisiones dentro de una versión, en las que lo único que se había modificado eran fallos de programación.

Con la rama 2.6 del núcleo, el sistema de numeración así como el modelo de desarrollo han cambiado. Han desaparecido las versiones de producción y desarrollo y la numeración se realiza con 4 dígitos dispuestos de la forma: `AA.BB.CC.DD` donde:

- `AA`: Indica la rama principal del núcleo.

- `BB`: Indica la revisión principal del núcleo. Números pares e impares no tienen ningún significado hoy en día

- `CC`: Indica nuevas revisiones menores del núcleo. Cambia cuando nuevas características y drivers son soportados.

- `DD`: Este dígito cambia cuando se corrigen fallos de programación o fallos de seguridad dentro de una revisión.

En septiembre de 2006 se redefinió el sistema de numeración a propuesta de Alan Cox[9], a partir de entonces, los núcleos numerados `2.6.CC` con `CC` impar serán versiones inestables, y no se aceptarán nuevas funcionalidades para el siguiente kernel par, sino que sólo podrán corregirse bugs. Por lo tanto el kernel numerado `2.6.CC` con `CC` par será la versión estable del kernel. Esta idea ha recibido el visto bueno de Andrew Morton y Linus Torvalds.

2.1.2.2. Sistema de ficheros *sysfs*

El primer cambio importante entre las ramas 2.4 y 2.6 del kernel es la introducción del sistema de ficheros *sysfs*. Este sistema de archivos, montado por defecto en `/sys`, es una representación estructurada del árbol de dispositivos tal y como lo ve el kernel (con excepciones). Incluye unos cuantos atributos de los dispositivos detectados: nombre del dispositivo, recursos IRQ y DMA, modo de energía, y demás.

Durante el ciclo de desarrollo del kernel 2.5 , se introdujo el modelo de *drivers* para arreglar algunas deficiencias del kernel 2.4:

- Ausencia de una representación unificada de las relaciones entre los controladores y los dispositivos.
- Falta de mecanismo de *hot plug*[1] genérico.
- Existencia en */proc* de multitud de información no referente a los procesos.

Sysfs fue diseñado para exportar la información presente en el árbol de dispositivos de manera que no se abuse de *procfs*. Patrick Mochel lo implementó. Maneesh Sony escribió a posterioridad el parche *backing store* que reduce la memoria utilizada en los sistemas grandes.

Sysfs es un sistema de archivos basado inicialmente en un sistema de archivos ram. Este ejercicio de elegancia mostró lo sencillo que era escribir un sistema de archivos utilizando la nueva capa VFS. Por su simplicidad y el uso de VFS se obtuvo una buena base para futuras implementaciones de sistemas de archivos basados en memoria.

Sysfs fue llamado inicialmente *ddfs (Device Driver Filesystem)* y fue creado inicialmente para depurar el modelo de controladores que se estaba desarrollando. Anteriormente, la depuración fue realizada utilizando *procfs* para exportar el árbol de dispositivos, pero bajo orden de Linus Torvalds se convirtió en un nuevo sistema de ficheros basado en *ramfs*.

Posteriormente y durante el ciclo de desarrollo del kernel 2.5, las características estructurales del modelo de drivers y su sistema de archivos empezaron a resultar útiles para otros subsistemas de archivos. Entonces adquirió su nombre definitivo *sysfs* y se desarrolló para proporcional una mecanismo de gestión centralizado.

2.1.2.3. Más cambios en */proc*

Para encontrar más cambios en el sistema de ficheros */proc* entre versiones del kernel se han analizado varios *changelog* de algunas versiones del kernel. Ver [12]. En las siguientes lineas se encuentran una pequeña parte de los cambios que ha ido sufriendo este sistema de archivos a lo largo del ciclo de desarrollo del kernel 2.6.

El kernel 2.6 permite añadir información completa sobre la configuración en el propio fichero del kernel. Se pueden incluir aquí detalles tales como opciones de configuración, versión del compilador utilizado, y otros detalles que ayudan a regenerar un kernel similar cuando sea necesario. Esta información se puede exponer a los usuarios a través del interfaz */proc*.

Cambios en el formado de que causaban confusión en anteriores versiones del kernel.

`/proc/sys/vm/swappiness` muestra el valor de *swapping* del sistema. Esto es un número entre 0 y 100 que nos dice la tendencia que tiene el kernel a transferir memoria no usada a la partición de intercambio. A valor más alto, más *swapping*.

Ha habido una actualización de `/proc` al quedar sin soporte algunas operaciones con los hilos de ejecución.

El fichero `/proc/filesystems` contiene algunos sistemas de archivos que no se pueden montar en el espacio de usuario, pero que se utilizan internamente por el kernel para seguir la pista de algunas cosas, como por ejemplo *futexfs* y *eventpollfs*.

El fichero `/proc/tty/driver/serial` en Linux 2.4.x revelaba el número exacto de caracteres utilizados en links serie, lo que permitía que los usuarios pudieran obtener información como la longitud de las contraseñas. En la rama 2.6 sólo puede hacerlo el usuario root.

Cambios de formato en los ficheros `/proc/stat`, `/proc/irq`, `/proc/pid/maps` entre otros.

Como se puede observar algunos de estos cambios pueden afectar a la implementación de MoniTo y hacer que de una versión a otra la herramienta quede obsoleta. Esta es la motivación principal de portar la implementación de MoniTo a las librerías LibGTop.

[1] *Hot-plug*, es la capacidad que tienen algunos periféricos de poder enchufarse o desenchufarse al ordenador, sin apagar el mismo, y funcionar correctamente.

2.1.3. Descripción de las librerías LibGTop

2.1.3.1. Obtener las librerías

El código fuente de LibGTop se encuentra en el repositorio Subversion (SVN) de GNOME (`http://svn.gnome.org/viewvc/libgtop/`), también pueden conseguirse los últimos archivadores de `http://ftp.gnome.org/pub/gnome/sources/libgtop/` o desde cualquiera de sus servidores espejo.

Además, las principales distribuciones de Linux llevan en su paquetería base o bien disponibles desde sus repositorios las librerías LibGTop.

En Debian se pueden instalar utilizando el comando `aptitude install libgtop2-7 libgtop2-common libgtop2-dev` o bien el gestor de paquetes Synaptic.

La última versión es la 2.21 y fue liberada el 29 de octubre de 2007.

2.1.3.2. Plataformas soportadas

La rama estable de LibGTop soporta las siguientes plataformas:

- *Todas las versiones de Linux*
  LibGTop fue testeado bajo Linux 2.0 y 2.2 y actualmente trabaja sin problemas con las ramas 2.4 y 2.6 del kernel.
- *FreeBSD, NetBSD y OpenBSD*
- *BSD/OS*
- *Digital Unix*
- *Solaris 7*

2.1.3.3. LibGTop White Paper

Los sistemas UNIX como Solaris, BSD o Digital Unix solo permiten a procesos con privilegios leer información como el consumo de CPU y memoria o información sobre los procesos en ejecución.

- BSD, por ejemplo, no tiene otra manera de obtener estos datos que leyendo directamente de */dev/kmem*, además el usuario deberá pertenecer al grupo *kmem* para estar habilitado para acceder a dicha información.
- Otros sistemas, como Digital Unix, permite a todos los usuarios obtener estadísticas de uso de procesador y memoria, pero no pueden obtener información sobre sus propios procesos sino son *root*.
- Linux tiene un bonito sistema de archivos */proc*, pero leerlo y analizarlo es lento e ineficiente.
- Solaris funciona un poco mejor, pero es necesario estar en el grupo *sys* o ser *root* para obtener algún dato.

Por esto, utilidades de sistema como *ps*, *uptime* o *top* acostumbran a tener como *setgid* a *kmem* o *setuid* al *root*. Normalmente, son muy específicas del sistema en el que han sido escritos y no son portables fácilmente hacia otros sistemas sin tener que realizar un gran esfuerzo.

Esto se convierte en un problema en herramientas gráficas como *gtop*. Para el proyecto GNOME es necesario un tipo de librería que proporcione toda la información requerida de una manera portable, ya que hay más de un programa que quiere utilizarla (por ejemplo, *gtop* y las aplicaciones del panel *multiload*, *cpumemusage* y *netload*).

Sobre el desarrollo de la interfaz de LibGTop considerar que fue necesario reunir todos los datos que la librería debía proporcionar y colocarlos en estructuras de C. Sin embargo esto no fue fácil ya que LibGTop debía ser portable a cualquier sistema UNIX actual con una librería común en parte a todos aquellos sistemas, pero considerando que los datos se devolvían de manera diferente según el sistema. Por ejemplo, algunos sistemas suportan la memoria compartida, pero otros no.

Los ficheros de cabecera que definen estas estructuras en C (independientes del sistema) están compartidas entre cliente y servidor. De esta manera, se puede llamar al código dependiente del sistema directamente donde no es necesario disponer de privilegios especiales para hacerlo.

Todas las estructuras contienen un atributo con *flags*, se interpretan como una máscara de bit y comunican a la llamada a la función de librería que campos de los devueltos por la estructura son válidos y cuales no.

El servidor LibGTop es un binario con *setgid* y *setuid* que contiene todo el código dependiente del sistema que necesita privilegios especiales. Sólo se utiliza si el sistema lo requiere (por ejemplo, el kernel de Linux proporciona todos los datos a través del sistema de ficheros */proc* por lo que no es imprescindible, aunque si recomendable, utilizar el servidor) y sólo contiene las características para las que se necesitan privilegios.

En el caso de no ser necesario ningún privilegio para coger todos los datos de una alguna de las estructuras requeridas, la librería llama directamente a las funciones de sistema en lugar de utilizar el servidor.

2.1.3.4. Funciones de sistema utilizadas

En este apartado se detallarán las funciones de sistema utilizadas en la nueva implementación de MoniTo y la lectura de fichero en el sistema de archivos */proc* al que sustituyen. De todos los atributos disponibles en las estructuras sólo se dará la explicación de aquellos que han sido utilizados en la implementación de MoniTo 4.

Uso de CPU

Función de librería `glibtop_get_cpu`:

```
void glibtop_get_cpu (glibtop_cpu *buf);
```

Declaración de `glibtop_cpu` en `<glibtop/cpu.h>`:

```
typedef struct _glibtop_cpu     glibtop_cpu;

    struct _glibtop_cpu
    {
        guint64   flags,
            total,
            user,
            nice,
            sys,
            idle,
            iowait,
            irq,
            softirq,
            frequency,
            xcpu_total [GLIBTOP_NCPU],
            xcpu_user [GLIBTOP_NCPU],
            xcpu_nice [GLIBTOP_NCPU],
            xcpu_sys  [GLIBTOP_NCPU],
            xcpu_idle [GLIBTOP_NCPU],
            xcpu_iowait [GLIBTOP_NCPU],
            xcpu_irq [GLIBTOP_NCPU],
            xcpu_softirq [GLIBTOP_NCPU],
            xcpu_flags;
    };
```

- **user** ⇒ Número de ciclos de reloj que gasta el sistema en modo usuario.
- **nice** ⇒ Número de ciclos de reloj que gasta el sistema en modo usuario y prioridad nice.

- **sys** ⇒ Número de ciclos de reloj que gasta el sistema en modo sistema.
- **idle** ⇒ Número de ciclos de reloj que gasta el sistema en modo ocioso.
- **iowait** ⇒ Número de ciclos de reloj que gasta el sistema en modo esperando a completar entrada / salida.

Esta llamada evita la lectura del fichero `/proc/stat`.

Uso de Memoria

Función de librería `glibtop_get_mem`:

```
void glibtop_get_mem (glibtop_mem *buf);
```

Declaración de `glibtop_mem` en `<glibtop/mem.h>`:

```
typedef struct _glibtop_mem     glibtop_mem;

struct _glibtop_mem
{
    guint64   flags,
        total,
        used,
        free,
        shared,
        buffer,
        cached,
        user,
        locked;
};
```

Todas las unidades están expresadas en bytes.

- **total** ⇒ Tamaño total de memoria física.
- **used** ⇒ Tamaño de memoria utilizada.
- **free** ⇒ Tamaño de memoria libre.
- **shared** ⇒ Tamaño de memoria compartida.

Esta llamada evita la lectura del fichero `/proc/meminfo`.

Uso de Swap

Función de librería `glibtop_get_swap`:

```
void glibtop_get_swap (glibtop_swap *buf);
```

Declaración de `glibtop_swap` en `<glibtop/swap.h>`:

```
typedef struct _glibtop_swap    glibtop_swap;

struct _glibtop_swap
{
    guint64   flags,
        total,
```

```
        used,
        free,
        pagein,
        pageout;
};
```

Todas las unidades están expresadas en bytes.

- **total** ⇒ Total de espacio swap utilizado en el sistema.
- **used** ⇒ Espacio swap utilizado.
- **free** ⇒ Espacio swap libre.

Esta llamada evita la lectura del fichero `/proc/swaps`.

Uptime

Función de librería `glibtop_get_uptime`:

```
void glibtop_get_uptime (glibtop_uptime *buf);
```

Declaración de `glibtop_uptime` en `<glibtop/uptime.h>`:

```
typedef struct _glibtop_uptime  glibtop_uptime;

struct _glibtop_uptime
{
    guint64 flags;
    double uptime,
        idletime;
    guint64 boot_time;
};
```

- **uptime** ⇒ Tiempo en segundos desde el arranque del sistema.
- **idletime** ⇒ Tiempo en segundos que gasta el sistema en estado ocioso desde el arranque del sistema.

Estos datos también se pueden obtener operando con atributos de la estructura `glibtop_cpu`.
Esta llamada evita la lectura del fichero `/proc/uptime`.

Memoria de un proceso

Función de librería `glibtop_get_proc_mem`:

```
void glibtop_get_proc_mem (glibtop_proc_mem *buf, pid_t pid);
```

Declaración de `glibtop_proc_mem` en `<glibtop/procmem.h>`:

```
typedef struct _glibtop_proc_mem        glibtop_proc_mem;

struct _glibtop_proc_mem
{
    guint64   flags,
        size,
        vsize,
```

```
        resident,
        share,
        rss,
        rss_rlim;
};
```

- **size** ⇒ Número de páginas de memoria.
- **vsize** ⇒ Número de páginas de memoria virtual.
- **rss** ⇒ Número de páginas que el proceso tiene en memoria real.
- **rss_rlim** ⇒ Límite actual en bytes del conjunto residente (rss) del proceso.

Esta llamada evita la lectura del fichero `/proc/<pid>/statm`.

Tiempo de un proceso

Función de librería `glibtop_get_proc_time`:

```
void glibtop_get_proc_time (glibtop_proc_time *buf, pid_t pid);
```

Declaración de `glibtop_proc_time` en `<glibtop/proctime.h>`:

```
typedef struct _glibtop_proc_time       glibtop_proc_time;

struct _glibtop_proc_time
{
    guint64   flags,
        start_time,
        rtime,
        utime,
        stime,
        cutime,
        cstime,
        timeout,
        it_real_value,
        frequency,
        xcpu_utime [GLIBTOP_NCPU],
        xcpu_stime [GLIBTOP_NCPU],
        xcpu_flags;
};
```

- **rtime** ⇒ Tiempo real acumulado por el proceso.
- **utime** ⇒ Tiempo de CPU en modo usuario acumulado por el proceso.
- **stime** ⇒ Tiempo de CPU en modo kernel acumulado por el proceso.

Esta llamada evita la lectura del fichero `/proc/<pid>/stat`.

Procesos en el kernel

Función de librería `glibtop_get_proc_kernel`:

```
void glibtop_get_proc_kernel (glibtop_proc_kernel *buf, pid_t pid);
```

Declaración de `glibtop_proc_kernel` en `<glibtop/prockernel.h>`:

```
typedef struct _glibtop_proc_kernel     glibtop_proc_kernel;

struct _glibtop_proc_kernel
{
    guint64 flags;
    guint64 k_flags,
        min_flt,
        maj_flt,
        cmin_flt,
        cmaj_flt,
        kstk_esp,
        kstk_eip,
        nwchan;
    char wchan [40];
};
```

- **min_flt** ⇒ Número de faltas menores que ha realizado un proceso y que no han requerido cargar una página desde el disco.
- **maj_flt** ⇒ Número de faltas mayores que ha realizado un proceso y que han requerido cargar una página desde el disco.

Esta llamada evita la lectura del fichero `/proc/<pid>/stat`.

Estadísticas de red

Función de librería `glibtop_get_netload`:

```
void glibtop_get_netload (glibtop_netload *buf, const char *interface);
```

Declaración de`glibtop_netload` en `<glibtop/netload.h>`:

```
typedef struct _glibtop_netload glibtop_netload;

struct _glibtop_netload
{
    guint64   flags,
        if_flags,
        mtu,
        subnet,
        address,
        packets_in,
        packets_out,
        packets_total,
        bytes_in,
        bytes_out,
        bytes_total,
        errors_in,
        errors_out,
        errors_total,
        collisions;
};
```

- **bytes_in** ⇒ Número total de bytes de entrada.
- **bytes_out** ⇒ Número total de bytes de salida.
- **bytes_total** ⇒ Número total de bytes.
- **errors_in** ⇒ Número total de errores en dirección de entrada.
- **errors_out** ⇒ Número total de errores en dirección de salida.
- **errors_total** ⇒ Número total de errores.
- **collisions** ⇒ Número total de colisiones.

Esta llamada evita la lectura del fichero `/proc/net/dev`.

2.2. Portal Web de MoniTo 4

En todas las versiones de MoniTo existe un portal web que hace la función de *frontend*, es decir interactua con el usuario para obtener los parámetros necesarios para efectuar la monitorización y posteriormente mostrar los datos.

En el capítulo 3 se concretará este tema incidiendo en el diseño particular de la herramienta, sirva por tanto, este apartado para presentar las herramientas utilizadas para el diseño y gestión del portal web.

2.2.1. XHTML 1.0

Este apartado define XHTML 1.0, una reformulación de HTML 4 como aplicación de XML 1.0, y los tres DTD correspondientes a los definidos en HTML 4. La semántica de los elementos y sus atributos se definen en la Recomendación del W3C para HTML 4. Estos semántica proporciona las bases para la futura extensión de XHTML. La compatibilidad con agentes de usuario HTML existentes es posible siguiendo un pequeño conjunto de directrices.

2.2.1.1. Introducción a XHTML

XHTML[2] es un tipo de documento que reproduce, recoge y amplia HTML 4. Los documentos XHTML están basados en XML y está siendo diseñado para poder trabajar con herramientas basadas en XML.

XHTML es el siguiente paso en la evolución de Internet. Migrar hoy a XHTML permite a los desarrolladores de contenido entrar en el mundo de XML y disfrutar de todos sus beneficios, al tiempo de tener garantizada la compatibilidad hacia atrás.

HTML 4

HTML 4 es un SGML[3] y es ampliamente considerado como el lenguaje estándar de la edición de la World Wide Web.

SGML es una lenguaje para describir lenguajes de marcado, particularmente los utilizados en el intercambio electrónico, gestión y publicación del documento. HTML es un ejemplo de un lenguaje definido en SGML.

HTML, tal como se concibió originalmente, iba a ser un lenguaje para el intercambio de información científica y otros documentos técnicos, adecuado para ser utilizado por no especialistas. HTML abordó el problema de la complejidad de SGML especificando un pequeño conjunto de etiquetas estructurales y semánticas adecuadas para la realización de documentos. Aparte de simplificar la estructura del documento, añadió soporte para hipertexto y capacidades multimedia.

En muy poco tiempo, HTML se hizo tremendamente popular y rápidamente sobrepasó su propósito original. Desde la creación de HTML se han ido inventando nuevos elementos dispuestos a ser utilizados dentro del estándar

[2]The Extensible HyperText Markup Language
[3]Standard Generalized Markup Language

HTML, por lo que hubo que adaptar HTML a los nuevos tiempos, sin embargo ha acabado llevando a problemas de interoperabilidad de documentos a través de diferentes plataformas.

XML

XML es el acrónimo de Extensible Markup Language. Fue concebido como un medio para recuperar el potencial y la flexibilidad de SGML pero sin la mayor parte de su complejidad. Sin abandonar estas características beneficiosas, XML elimina muchas de las características más complejas de SGML que hacen del diseño de programas informáticos adecuados difícil y costoso.

Necesidad de XHTML

Algunos de los beneficios de la migración a XHTML, en general, son los siguientes:

- Desarrolladores y diseñadores de agentes de usuario descubren constantemente nuevas maneras de expresar sus ideas a través del nuevo marcado. En XML, es relativamente fácil introducir nuevos elementos o atributos elemento adicional. La familia XHTML está diseñado para acomodar estas extensiones a través de módulos XHTML y técnicas para desarrollar nuevos módulos. Estos módulos permitirán la combinación de los ya existentes y de las nuevas características en el desarrollo de contenidos y en el diseño de nuevos *user-agents*.
- Vías alternativas de acceso a Internet se están introduciendo constantemente. XHTML está diseñado para permitir la interoperabilidad general entre agentes de usuario.

2.2.1.2. Definiciones

Agente de usuario Un agente de usuario es un sistema que procesa documentos XHTML, conforme a la especificación del W3C.

Análisis sintáctico Análisis sintáctico es el acto por el cual un documento es escaneado, y la información contenida en el documento se filtra en el contexto de los elementos en los que la información está estructurada.

Atributo Un atributo es un parámetro de un elemento declarado en la DTD. Un atributo del tipo y rango de valor, incluido un posible valor por defecto, se definen en la DTD.

DTD Un DTD o definición de tipo de documento, es una colección de declaraciones de marcado XML que, como colección, define la estructura, los elementos y atributos que están disponibles para su uso en un documento que se ajusta a la DTD.

Validación La validación es un proceso por el cual los documentos se verifican contra la DTD asociada, asegurándose de que la estructura, uso de elementos, y el uso de atributos son consistentes con las definiciones de la DTD.

2.2.1.3. Normas de XHTML 1.0

En este apartado se explicarán algunas de las normas de XHTML 1.0 que han debido ser tenidas en cuenta a lo largo de la implementación del portal web de MoniTo 4.

- Debe ajustarse a las limitaciones expresadas en el DocType.
- El elemento raíz siempre será `html`.
- La raíz del documento debe contener la declaración *xmlns* del espacio de nombres de XHTML, como por ejemplo.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

- Debe haber una declaración DOCTYPE en el documento antes del elemento raíz. El identificador público incluido en la declaración DOCTYPE debe hacer referencia a uno de los tres DTDs siguientes.

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

- Los elementos anidados deben tener un correcto orden de apertura/cierre (el que se abre último, debe cerrarse primero).
- Los elementos y nombres de atributos deben estar en minúsculas
- Los elementos no vacíos también deben cerrarse siempre.
- Los valores de los atributos deben siempre ir encerrados entre comillas (simples o dobles).
- Los elementos vacíos deben constar de un fin de etiqueta, como por ejemplo `<br />` o `<img />`.
- En XML, los identificadores de fragmentos son de tipo `ID`, y solamente puede haber un único atributo de tipo `ID` por elemento.
- Los atributos desaprobados en HTML 4 no forman parte de XHTML, como por ejemplo `font` o `center`.

2.2.2. Hojas de estilo en cascada CSS

Hojas de Estilo en Cascada *(Cascading Style Sheets)*, es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML (y por extensión en XHTML), separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

```
h1 {color: red;}
```

donde `h1` es el selector y `color: red;` es la declaración.

El selector funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto. En el ejemplo anterior, el selector `h1`[4] indica que todos los elementos `h1` se verán afectados por la declaración donde se establece que la propiedad color va a tener el valor `red` *(rojo)* para todos los elementos `h1` del documento o documentos que estén vinculados a esa hoja de estilos.

Las tres formas más conocidas de dar estilo a un documento son las siguientes:

[4] `h1` representa un encabezado de primer orden

- Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento `<link>`, el cual debe ir situado en la sección `<head>`.

```
<link rel="stylesheet" type="text/css" href="*.css" />
```

- Utilizando el elemento `<style>`, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección `<head>`. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<style type="text/css"> (...)  </style>
```

- Utilizando estilos directamente sobre aquellos elementos que lo permiten a través del atributo `<style>` dentro de `<body>`. Pero este tipo de definición del estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

2.2.2.1. Modelo de cajas CSS

Dentro de toda la especificación de CSS es interesante abordar el tema del modelo de cajas, ya que ha sido una parte importante del rediseño de la interfaz web de MoniTo 4.

Cada caja tiene un área de contenido (ej., texto, una imagen, etc.) y las áreas circundantes opcionales de `padding, border` y `margin`; el tamaño de cada área es especificado por las propiedades que se definen abajo. La figura 2.2 muestra cómo se relacionan estas áreas y la terminología usada para referirse a las partes de `margin, border` y `padding`:

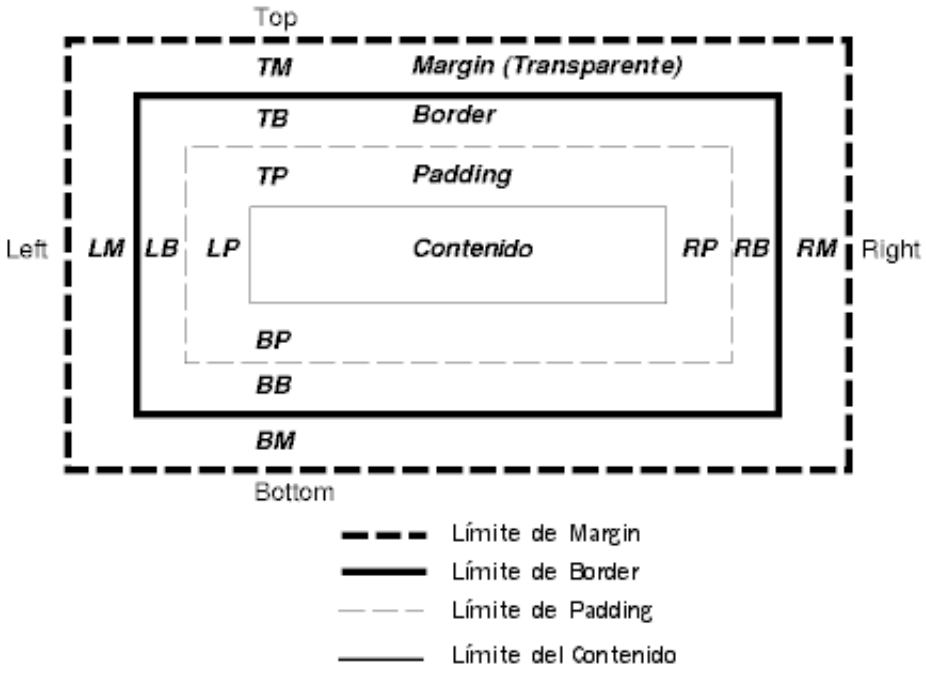

Figura 2.2: Modelo de cajas CSS

`Margin, border` y `padding` pueden ser divididos en los segmentos `left, right, top y bottom` (ej., en la figura 2.2, ‘‘`LM`’’ para `left margin`, ‘‘`RP`’’ para `right padding`, ‘‘`TB`’’ para `top border`, etc.).

El perímetro de cada una de las cuatro áreas *(contenido, padding, border y margin)* es llamado límite, de manera que cada caja tiene cuatro límites:

límite del contenido El límite interno del contenido rodea al contenido procesado del elemento.

límite de padding El límite de `padding` *(relleno)* rodea a la caja de relleno. Si `padding` tiene un ancho de 0, el límite del relleno es el mismo que el límite de contenido. El límite del relleno de una caja define el límite de la caja de contención establecida por la caja.

límite de border El límite de `border` *(borde)* rodea el borde de la caja. Si `border` tiene un ancho de 0, el límite del borde es el mismo que el límite de `padding`.

límite de margin El límite de `margin` externo rodea el margen de la caja. Si `margin` tiene un ancho de 0, el límite del margen es el mismo que el límite de `border`.

Cada límite puede dividirse en `left` *(izquierdo)*, `right` *(derecho)*, `top` *(superior)* y `bottom` *(inferior)*.

Las dimensiones del área del contenido de una caja (el ancho del contenido y la altura del contenido) dependen de varios factores: si el elemento que genera la caja tiene asignadas las propiedades `width` o `height`, si la caja contiene texto u otras cajas, si la caja es una tabla, etc.

El ancho de la caja está dado por la suma de los márgenes, bordes y rellenos izquierdos y derechos, y el ancho del contenido. La altura está dada por la suma de los márgenes, bordes y rellenos superiores e inferiores, y la altura del contenido.

El estilo del fondo de las distintas áreas de una caja es determinado como sigue:

Área del contenido La propiedad `background` del elemento generador.

Area del padding La propiedad `background` del elemento generador.

Area del border Las propiedades del borde del elemento generador.

Area del margin Los márgenes son siempre transparentes.

La etiqueta que permite crear un bloque dentro de un documento XHTML es `<div>`.

Es importante es concepto ya que permite eliminar la maquetación del portal web utilizando tablas. Sus ventajas son la facilidad de administración del código generado, no tienen ningún estilo asociado por defecto lo que permite una mayor versatilidad a la hora de diseñar, mayor respeto por los estándares y por último obliga a utilizar las tablas para su función, mostrar datos tabulados.

En el capítulo 3 se tratará más en profundidad este tema aplicado al caso particular del portal web de MoniTo.

2.2.2.2. Propiedades y atributos importantes

La tablas de este apartado muestran una pequeña lista de los atributos más importantes que se han utilizado en el desarrollo del portal web de MoniTo. La lista de atributos ha sido obtenida del *World Wide Web Consortium*, un consorcio internacional que produce estándares para la World Wide Web. Está dirigida por Tim Berners-Lee. La referencia completa se encuentra en [15].

Modelo de Cajas

Márgenes		
Propiedad	Descripción	Valores

`margin-top`
`margin-right`
`margin-bottom`
`margin-left`
	Tamaño del margen superior, derecho, inferior e izquierdo	*longitud, porcentaje, auto*

`margin`	Tamaño del margen superior, derecho, inferior e izquierdo	*longitud, porcentaje, auto, 1-4*

Cuadro 2.1: Atributos para el margen

Relleno		
Propiedad	Descripción	Valores

`padding-top`
`padding-right`
`padding-bottom`
`padding-left`
	Tamaño del relleno superior, derecho, inferior e izquierdo	*longitud, porcentaje, auto*

`padding`	Tamaño del relleno superior, derecho, inferior e izquierdo	*longitud, porcentaje, auto,* {*1-4*}

Cuadro 2.2: Atributos para el relleno

Bordes		
Propiedad	Descripción	Valores

`border-top-width`
`border-right-width`
`border-bottom-width`
`border-left-width`
	Anchura del borde superior, derecho, inferior o izquierdo	*thin, medium, thick, longitud*

`border-width`	Anchos de varios bordes individuales	*thin, medium, thick, longitud,* {*1-4*}

`border-top-color`
`border-right-color`
`border-bottom-color`
`border-left-color`
	Color del borde superior, derecho, inferior o izquierdo	*color, transparent*

`border-color`	Colores de varios bordes individuales	*color, transparent*

`border-top-style`
`border-right-style`
`border-bottom-style`
`border-left-style`
	Estilo del borde superior, derecho, inferior o izquierdo	*none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset*

`border-style`	Estilos de varios bordes individuales	*none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset* {*1-4*}

`border-top`
`border-right`
`border-bottom`
`border-left`
	Ancho, estilo y el color para el borde superior, derecho, inferior o izquierdo	*border-top-width, border-top-style, border-top-color*

`border`	Ancho, el estilo y el color para los 4 bordes	*border-top-width, border-top-style, border-top-color*

Cuadro 2.3: Atributos de los bordes

Modelo de Formato Visual

Propiedad	Descripción	Valores
`display`	Comportamiento del contenedor	*inline, block, list-item, run-in, inline-block, table, inline-table, table-row-group, table-header-group, table-footer-group, table-row, table-column-group, table-column, table-cell, table-caption, none*
`position`	Esquema de posicionamiento	*static, relative, absolute, fixed*

`top`
`right`
`bottom`
`left`
	Desplazamiento de la caja (respecto al límite superior, derecho, inferior o izquierdo del contenedor)	*longitud, porcentaje, auto*

`float`	Posicionamiento flotante	*left, right, none*
`clear`	Control de cajas adyacentes a los `float`	*none, left, right, both*
`z-index`	Solapamiento de niveles de capas	*Auto, entero con signo*
`width` `height`	Ancho Alto	*Longitud, porcentaje, none*
`overflow`	Comportamiento del contenido si se desborda en la caja	*visible, hidden, scroll, auto*

Cuadro 2.4: Atributos de formato visual

Contenido generado, numeración automática y listas

Propiedad	Descripción	Valores
`list-style-type`	Estilo aplicable a los marcadores visuales de las listas	*disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-greek, lower-latin, upper-latin, armenian, georgian, lower-alpha, upper-alpha, none*
`list-style-position`	Posición dentro de la lista de los elementos marcadores de las listas	*inside, outside*
`list-style`	Permite establecer el estilo de la lista, la imagen y/o la posición	*list-style-type, list-style-position, list-style-image*

Cuadro 2.5: Atributos de listas

Colores y fondo

Propiedad	Descripción	Valores
`color`	Color del primer plano	*color*
`background-color`	Color de fondo	*color, transparent*

Cuadro 2.6: Atributos de colores y fondo

Fuentes y texto

Propiedad	Descripción	Valores
`font-family`	Familias de fuentes	*nombre-familia, familia-genérica*
`font-style`	Estilo de la fuente	*normal, italic, oblique*
`font-weight`	Intensidad de la fuente	*normal, bold, bolder, lighter, 100-900*
`font-size`	Tamaño de la fuente	*small, medium, large, longitud, porcentaje*
`text-indent`	Desplazamiento de la primera línea del texto	*longitud, porcentaje*
`text-align`	Alineamiento del texto	*left, right, center, justify*
`text-decoration`	Efectos de subrayado, tachado, parpadeo	*none, underline, overline, line-through, blink*

Cuadro 2.7: Atributos de fuentes y texto

Tablas

Propiedad	Descripción	Valores
`border-collapse`	Selección del modelo de los bordes	*collapse, separate*

Cuadro 2.8: Atributos de tablas

En resumen, la principal ventaja de utilizar hojas de estilo es que permite separar en dos partes la implementación del portal web, por un lado la estructura del texto en el fichero XHTML, mientras que por otro definir su apariencia en una hoja de estilo CSS. La lista de atributos que se ha mostrado en este apartado muestra la gran potencia a la hora de diseñar interfaces web que otorga CSS, y que han motivado ser elegidos como una parte importante de la reimplementación del portal web de MoniTo 4.

2.2.3. PHP, procesador de hipertexto

PHP, acrónimo de *"PHP: Hypertext Preprocessor"*, es un lenguaje de código abierto interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.

Su interpretación y ejecución se da en el servidor web, en el cual se encuentra almacenado el script, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, y regresa el resultado al servidor, el cual se encarga de devolverselo al cliente.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite; lo cual permite la creación de Aplicaciones web muy robustas.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

Actualmente la rama estable de PHP es la 5, lanzada el 13 de julio de 2004. La versión más reciente de PHP es la 5.2.6 (1 de mayo de 2008), que incluye todas las ventajas que provee el nuevo motor Zend Engine 2 como:

- Soporte sólido y REAL para Programación Orientada a Objetos (o OOP) con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.

- Mejor soporte a XML (XPath, DOM...).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Excepciones de errores.

2.2.3.1. Características

Como todo lenguaje de programación tiene sus ventajas y desventajas, las cuales se enumerarán a continuación:

Ventajas

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones).
- Amplia documentación, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.

Desventajas

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas para cada motor.
- No posee adecuado manejo de internacionalización, unicode, etc.

Queda claro que la potencia de PHP es amplia, y aunque como se explicará en el capítulo 3 no ha sido necesario explotar en exceso esta capacidad, se pretende sentar las bases de mejora de las capacidades y posibilidades de MoniTo utilizando páginas dinámicas.

2.2.4. Servidor HTTP Apache

El Servidor Apache HTTP, comúnmente conocido como Apache, es un servidor web con notable rol en el crecimiento inicial del *World Wide Web*. Desde abril de 1996 Apache ha sido el servidor HTTP más popular pero desde marzo de 2006 ha experimentado una disminución constante de su cuota de mercado. Su nombre se debe a la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de Internet Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA.

Apache es desarrollado y mantenido por una comunidad de desarrolladores con el soporte de la Apache Software Foundation. La aplicación está disponible para una amplia variedad de sistemas operativos, como Unix, FreeBSD,

Linux, Solaris, Novell NetWare, Mac OS X y Windows. El código fuente de Apache es abierto, por lo que es un servidor de libre distribución.

La arquitectura del servidor Apache es muy modular. El servidor consta de un núcleo y gran parte de la funcionalidad que podría considerarse básica para un servidor web es provista por módulos. Estas funciones pueden ir desde el soporte para lenguajes de programación del lado del servidor como PHP (`mod_php`), módulos de autenticación (`mod_auth`), reescritura de URL's servidas (`mod_rewrite`), logs personalizados (`mod_log_config`) o compresión de sitios web (`mod_gzip`) entre otros muchos. Otra característica importante es el soporte para sitios virtuales, es decir la capacidad de una instalación de Apache de servir varios sitios web distintos.

Apache dispone de dos ramas de desarrollo, los módulos de cada versión son exclusivos de ella por lo que un módulo para la rama 1.3 no funcionará en la rama 2.2. Actualmente, la versión recomendada por los desarrolladores es la 2.2.8 (liberada el 21 de enero de 2008) por encima todas las anteriores. A continuación se enumerarán algunos cambios importantes entre versiones, que pueden considerarse a la hora de servir páginas para MoniTo.

Apache 1.3

- Los módulos de Apache ahora pueden cargarse en tiempo de ejecución, por lo que implica que sólo se cargarán en el servidor en el momento que sean necesarios.
- Reorganización de los ficheros, especialmente en la configuración de los módulos.
- En la versión 1.3.4 se han reunificado los archivos de configuración en `httpd.conf`.
- Los scripts CGI transfieren la salida tal como la reciben, en vez de esperar a tener el buffer completo o a finalizar la ejecución del CGI.

Apache 2.2

- Diferente estructura de los ficheros de configuración, especialmente para definir sitios virtuales y módulos activados.
- Soporte para hilos POSIX.
- `mod_ssl` integrado en Apache 2.
- Los módulos de Apache pueden ser escritos como filtros que actúan como flujo de contenido, ya que se entregan hacia o desde el servidor. Esto permite, por ejemplo, que la salida de los CGI's sea analizada por las directivas *Server Side Include*.
- Simplificación de directivas confusas.

Se ha comprobado el funcionamiento de MoniTo 4 tanto con Apache 1.3 como con Apache 2. En próximos apartados se explicará la configuración detallada para cada una de las versiones del servidor.

Capítulo 3

MoniTo 4

MoniTo es una herramienta de monitorización de clusters implementada por el grupo de investigación de Computación Distribuida del Departamento de Informática e Ingeniería Industrial de la Universidad de Lleida.

MoniTo necesita ser instalado en cada uno de los nodos del cluster que se quiere monitorizar. Además también deberá estar presente en un servidor que recopila la información que obtienen cada uno de los nodos. La herramienta dispone de una interfaz amigable y fácil de utilizar debido a que el acceso al monitor se realiza a través de una interfaz web.

La monitorización se implementa mediante una serie de módulos que recogen información relevante sobre el rendimiento del cluster. Estos módulos se conocen como monitores. La información recogida por los monitores se puede clasificar en varias categorías:

1. Información sobre al carga del sistema, relacionada con el uso de CPU.
2. Información sobre la memoria, relacionada con el uso de memoria física y swap.
3. Información sobre la red, relacionada con el uso de la interfaz de red por la que se comunican los nodos del cluster.

En este capítulo se explica más detalladamente la estructura y el diseño de MoniTo y como hacen sus monitores para extraer la información relacionada con el funcionamiento del sistema. También se explicará cada uno de los demonios y componentes que forman el sistema MoniTo.

3.1. Diseño de MoniTo

MoniTo recopila valores de los diferentes parámetros monitorizados en cada uno de los nodos que forman parte del cluster. Esta información se recopila en un sistema central y se transforma en un formato fácil de comprender y de analizar por el usuario. Esta estructura puede verse en la figura 3.1.

3.1.1. ScD. Slave collector Daemon

El demonio ScD se ejecuta en cada nodo del cluster, su función es recoger información sobre el funcionamiento del nodo y de sus procesos. Los datos se consiguen a partir de los monitores que obtienen información del sistema operativo utilizando las librerías LibGTop, explicadas en el apartado 2.1.

La función principal se encarga de controlar todos los monitores, inicializar el servidor LibGTop y de comunicarse con el McD. El ScD está implementado utilizando hilos POSIX que permiten a los diferentes monitores atender peticiones de forma simultanea, compartir memoria y de paso, se facilita la comunicación entre los monitores y el módulo principal.

Los monitores que se utilizan para obtener la información son:

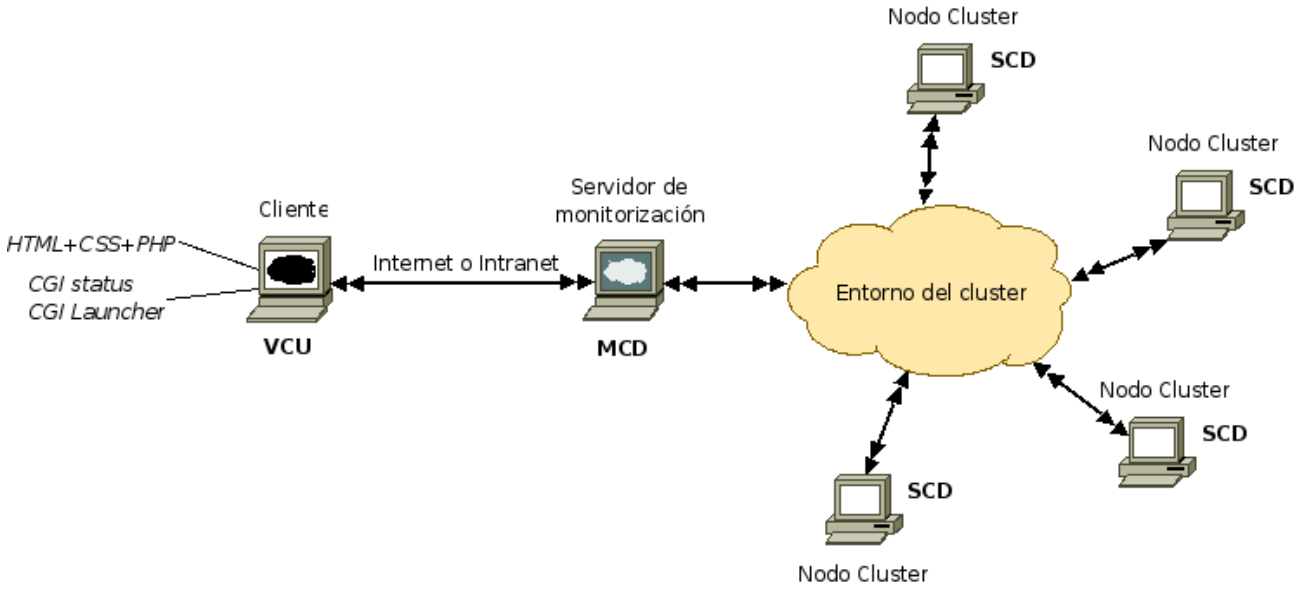

Figura 3.1: Estructura Monito

LoadMon: Monitoriza los parámetros de carga del procesador.

MemMon: Monitoriza diversos parámetros sobre el estado de la memoria física y la partición swap.

NetMon: Monitoriza los parámetros de utilización de la red.

ClassMon: Obtiene información estadística sobre el uso del nodo durante un intervalo establecido por el usuario.

EstaMon: Obtiene estadísticas de uso de cada nodo del cluster y se las comunica al demonio estadístico.

Más adelanta se explicará más detalladamente la información de cada nodo y la manera de la que obtiene cada uno la información mediante las librerías LibGTop.

3.1.2. McD. Master collector Daemon

El demonio MCD se ejecuta en el servidor de monitorización. Su función es centralizar la información proporcionada por los nodos (mediante el ScD) y enviarla a la unidad de visualización. Generalmente se instala en la misma máquina que el servidor web, es decir junto al portal web de la unidad de visualización.

Este demonio recibe peticiones de la unidad de visualización, las traduce en peticiones individuales que remite al ScD de cada nodo implicado. Para cada petición que realiza crea un proceso hijo, que se encarga de recibir las muestras y enviarlas a la unidad de visualización, para ser mostradas gráficamente.

Este demonio no ha sufrido ninguna modificación en la migración de MoniTo a las librerías LibGTop, por lo que poco más se comentará sobre el.

3.1.3. VcU. Visualization Control Unit

La unidad de visualización se encarga de interactuar con el usuario. Gracias a ella se pueden visualizar los resultados de las monitorizaciones de un modo gráfico.

Los datos se muestran al usuario mediante un portal web. Para ello se necesita un servidor web Apache, encargado de servir las páginas con la información que solicita el usuario, ejecutar los scripts de PHP para generar la página dinámica y permitir en caso de ser necesaria, la autenticación de usuarios.

En la unidad de visualización se pueden distinguir cinco partes:

PHP Lenguaje interpretado en el lado del servidor utilizado para generar páginas dinámicas fácilmente

XHTML La parte visible de la web se encuentra embebida en las llamadas a ficheros PHP. Para imprimir el código HTML desde PHP se utiliza la función `echo`.

CSS Las hojas de estilo en cascada permiten separar la presentación de la estructura de la web. De esta manera no es necesario editar (e incluso recompilar) múltiples ficheros para cambiar la presentación de la interfaz web.

CGI Son ejecutables escritos en C que generan un contenido en las páginas web. Se ejecutan a través de PHP con la función `passthru`. Se dispone de dos tipos de CGI's:

- El CGI *status* se conecta al McD y obtiene una lista de nodos disponibles junto al estado de sus monitores.
- Los CGI *launcher* generan una (o varias) ventanas emergentes en las que se muestran los resultados de la monitorización.

Applet Java Muestra de forma gráfica la información monitorizada en los monitores de información de nodos, procesos y en el monitor de comportamiento del proceso.

3.2. Implementación de los monitores y demonios

Una de las grandes modificaciones de esta versión de MoniTo es la migración a LibGTop. Gracias a estas librerías se obtiene la información de sistema sin necesidad de recurrir al sistema de ficheros */proc*.

En el apartado 2.1 se han presentado las estructuras y llamadas a funciones que se han utilizado en la implementación. Ahora se va a concretar para cada monitor como se ha realizado la implementación en cada monitor y en el ScD.

3.2.1. ScD. Slave Collector Daemon

Este demonio debe ser instalado en todos los nodos que forman parte del cluster. Se encargará principalmente de tomar las muestras de los parámetros a monitorizar. La tabla 3.1 muestra la estructura interna actual del demonio ScD.

LibGTop Server	NetMon	`netstat` & LibGTop
	ClassMon	LibGTop
	LoadMon	
	MemMon	
	EstaMon	`/tmp/avg_data_file.txt`

Cuadro 3.1: Estructura interna del demonio ScD

La tabla 3.1 tiene el siguiente significado, en el función `main` del ScD se realiza la inicialización del servidor LibGTop mediante la llamada `glibtop_init();`. Aunque en un sistema Linux no es imprescindible hacerla, ya que la información se puede obtener mayoritariamente sin privilegios de superusuario, es buena práctica realizar dicha llamada para evitar futuros problemas de compatibilidad.

El monitor de red obtiene parte de sus datos utilizando la librería LibGTop y otra parte mediante la ejecución del comando `netstat`. La ventaja de utilizar comandos de sistema es un mayor nivel de portabilidad entre versiones del núcleo pero tiene la desventaja es que este método no es tan rápido debido a la carga de CPU que produce la ejecución del comando.

Los monitores *LoadMon*, *MemMon* y *ClassMon* obtienen toda la información de las librerías LibGTop, en estos monitores se han sustituido completamente las lecturas de ficheros del sistema de archivos */proc*.

Por último, el monitor estadístico toma muestras de los datos cada minuto y los almacena en el archivo `/tmp/avg_data_file.txt` para poder obtener los datos estadísticos que LibGTop no proporciona por si mismo.

Aparte de la inicialización del servidor LibGTop antes de atender ninguna petición de los monitores o del McD no ha habido ninguna modificación más en el código del ScD. Todo el código modificado se puede consular en el apendice B.1.1.

3.2.2. Demonio estadístico

El demonio estadístico se encarga de recoger datos del sistema cada un cierto tiempo (60 segundos) ya que no se puede obtener directamente estadísticas de uso de */proc* y tampoco con las librerías LibGTop.

Este demonio ha sufrido fuertes modificaciones en la manera de obtener los datos en la función `pick_data`. Esta función copia los datos obtenidos en una estructura pasada por parámetro.

Anteriormente obtenía los datos de los ficheros de */proc*, sin embargo en la versión actual las obtiene directamente de las estructuras de datos de LibGTop explicadas en el apartado 2.1.3.4. La migración, consigue obtener la información será algo más rápida con LibGTop.

El código fuente modificado se puede consultar en el apéndice B.1.2

3.2.3. Monitor de carga - LoadMon

El monitor de carga se encuentra en el fichero `load_func.c`. Su función es monitorizar diferentes parámetros de la carga del procesador. Este monitor obtiene toda la información de la librería LibGTop.

Para el cálculo de los parámetros del nodo utiliza la estructura `glibtop_cpu` (ver 2.1.3.4). De esta estructura se obtiene la información detallada en la tabla 3.2.

Atributo	Información obtenida	Nombre variable
`user`	Usuario	`cuser`
`nice`	Usuario baja prioridad	`cnice`
`sys`	Sistema	`csys`
`idle`	Ociosa	`cidle`
`iowait`	Espera de E/S	`cio`

Cuadro 3.2: Información obtenida de `glibtop_cpu`

Para la obtención de los valores se ha utilizado el mismo procedimiento que en versiones anteriores, ya que lo único que se ha sustituido es la manera de obtener la información del sistema. Una vez obtenidos los datos se trabaja de idéntica manera con ellos.

```
diff = ( cuser + cnice + csys + cidle + cio) - anterior_total
anterior_total = cuser + cnice + csys + cidle + cio
tiempo_usado = (cuser + cnice + csys) - (cuser_anterior + cnice_anterior + csys_anterior )
Total CPU Load = ( tiempo_usado * 100) / diff
User CPU Load = ( cuser * 100 ) / diff
System CPU load = ( csys * 100 ) / diff
Nice CPU Load = ( cnice * 100 ) / diff
IO CPU wait = ( cio * 100 ) / diff
cuser_anterior = cuser
cnice_anterior = cnice
csys_anterior = csys
cidle_anterior = cidle
```

Para obtener los tiempos de un proceso se utiliza la llamada a `glibtop_get_proc_time` (ver 2.1.3.4). De esta estructura se obtiene la información detallada en la tabla 3.3.

Atributo	Información obtenida	Nombre variable
`utime`	Jiffies en modo usuario	`utime`
`rtime`	Tiempo entre inicio de sistema y momento de ejecución del proceso	`start`
`stime`	Jiffies en modo sistema	`stime`

Cuadro 3.3: Información obtenida de `glibtop_get_proc_time`

El cálculo de los tiempos del proceso se mantiene como en anteriores versiones:

```
Process CPU load = ( ( utime + stime ) - ( utime_anterior + stime_anterior ) ) * 100 / diff
utime_anterior = utime
stime_anterior = stime
```

El tiempo total en segundos que el sistema lleva encendido se obtiene a partir de dos parámetros obtenidos de `glibtop_cpu` de la siguiente manera:

```
uptime = (double) cpu.total / (double) cpu.frequency;
```

Los parámetros obtenidos se enviarán al McD de idéntica manera a la que realizaba en versiones anteriores.

El último cambio que hay que reseñar es la modificación en la función `load_init`, anteriormente comprobaba si los ficheros de */proc* que se utilizaban para hallar la carga del sistema estaban disponibles. Obviamente todas esas comprobaciones han sido eliminadas y sólo se realiza la conexión del *socket* por el que se comunicarán el McD y el ScD.

3.2.3.1. Monitor de memoria -MemMon

El monitor de memoria se encuentra en el fichero `mem_func.c`. Su función es monitorizar diferentes parámetros sobre el estado de la memoria. Este monitor obtiene toda la información de la librería LibGTop.

Los datos de memoria física se obtienen de la estructura `glibtop_mem` (ver 2.1.3.4). De esta estructura se obtiene la información detallada en la tabla 3.4.

Atributo	Información obtenida	Nombre variable
`total`	Memoria total	`mt`
`free`	Memoria libre	`mf`
`shared`	Memoria compartida	`ms`

Cuadro 3.4: Información obtenida de `glibtop_mem`

La memoria utilizada se obtiene a partir de la total y la libre con la siguiente operación:

```
memoria total - memoria libre = memoria utilizada
```

Los datos de memoria swap se obtienen de la estructura `glibtop_swap` (ver 2.1.3.4). De esta estructura se obtiene la información detallada en la tabla 3.5.

Atributo	Información obtenida	Nombre variable
`total`	Swap total	`st`
`free`	Swap libre	`sf`

Cuadro 3.5: Información obtenida de `glibtop_swap`

La cantidad de swap utilizada se obtiene de manera análoga a la memoria física utilizada:

```
swap total - swap libre = swap utilizada
```

La información de fallos de página se obtienen de la estructura `glibtop_get_proc_kernel` (ver 2.1.3.4). De esta estructura se obtiene la información detallada en la tabla 3.6.

Atributo	Información obtenida	Nombre variable
`min_flt`	Fallos de pagina menores (copy-on-write)	`wcf`
`maj_flt`	Fallos de pagina mayores. Acceso a disco	`pif`

Cuadro 3.6: Información obtenida de `glibtop_get_proc_kernel`

Otros datos relevantes sobre la memoria de un proceso se obtienen de la estructura `glibtop_get_proc_mem` (ver 2.1.3.4). De esta estructura se obtiene la información detallada en la tabla 3.7.

Atributo	Información obtenida	Nombre variable
`vsize`	Tamaño de memoria virtual	`vsp`
`rss`	Tamaño del conjunto residente	`rsp`
`resident`	Páginas residentes en M.P.	`resident`
`size`	Total páginas en M.P.	`size`

Cuadro 3.7: Información obtenida de `glibtop_get_proc_mem`

Algúnos parámetros requieren cálculos y estos se seguirán realizando de idéntica forma a versiones anteriores:

```
Pages in swap memory = totalpag - RSS
Virtual memory size = virt
Resident pages = RSS
Pages touched = minfl + maxfl
Pages fetch (min_flt) = minfl
Pages fetch(maj_flt) = maxfl
```

De manera análoga al monitor de carga se ha modificado la función `load_init`. De esta manera ya no comprueba si los ficheros de */proc* están disponibles, simplemente conecta el socket e inicializa el monitor.

3.2.3.2. Monitor de red - NetMon

Monitoriza diferentes parámetros sobre el uso de la red. Este monitor obtiene parte de la información de las librerías LibGTop y la información específica de los procesos mediante el comando `netstat`.

La parte de información que se consigue con LibGTop se obtiene de la estructura `glibtop_get_netload` (ver 2.1.3.4). De esta estructura se obtiene la información detallada en la tabla 3.8.

Atributo	Información obtenida	Nombre variable
`bytes_out`	Número de bytes enviados	`bytesend`
`bytes_in`	Número de bytes recibidos	`byterecv`
`errors_out`	Número de errores de envío	`errorsend`
`errors_in`	Número de errores de recepción	`errorrecv`
`collisions`	Número de colisiones	`collisions`

Cuadro 3.8: Información obtenida de `glibtop_get_netload`

En este monitor también se ha eliminado la comprobración de ficheros en */proc* al ser totalmente innecesaria. La interfaz de red a través de la cual se conecta el demonio al cluster se encuentra en el fichero de configuración `/etc/monito/netdev.conf`, este fichero está compuesto de una única linea con el identificador de la interfaz de red del cluster, este fichero puede ser editado por el usuario administrador del cluster y su valor por defecto es `eth0`.

3.2.3.3. Monitor de comunicación - ClassMon

Este monitor, introducido en MoniTo 3 permite obtener una comparativa entre los recursos utilizados por un proceso comparados con los del sistema. Mas en concreto:

- Bytes de un proceso y bytes del nodo.
- Memoria residente de un proceso y memoria utilizada por el nodo.
- Carga total de CPU del sistema y del proceso.

Dentro del propio fichero `class_func.c` sólo se utiliza la llamada a `glibtop_get_netload`, cuya referencia se encuentra en la tabla 3.8. También realiza un escaneo de puertos con el ejecutable `ports`, del cual se comentarán sus modificaciones en 3.2.3.4.

Como ha sucedido con el resto de monitores no comprueba en su inicialización la existencia de */proc*, en este sentido actúa exáctamente igual que el monitor de red.

3.2.3.4. Escaner de puertos - ports

El proceso `ports` es utilizado por el monitor de comunicación. Su función es escuchar el tráfico de red que provoca un proceso. Estos datos se almacenan en un fichero temporal, el cual utiliza después el monitor para obtener los parámetros deseados.

Al plantearse la actualización de MoniTo se detectaron dos deficiencias en la implementación de ports.

La primera de ellas fue que utilizaba un comando ya obsoleto `tethereal`, el antiguamente conocido como *Ethereal* ha pasado a llamarse *Wireshark* debido a que *Ethereal* estaba registrado en la empresa en la que el creador, Gerald Combs, trabajaba (NIS). Al cambiar de puesto de trabajo para unirse a CACE, no pudo llevarse los derechos. Sin embargo, sigue manteniendo todo el equipo de desarrollo. Por este motivo se ha decidido incluir el *tshark* (nombre de la versión en linea de comandos) en la implementación de `ports` y en las dependencias de los paquetes de instalación (ver 3.5).

La segunda fue la detección de que `ports` fallaba en procesos con varios puertos activos. Esto era debido a que montaba mal los filtros de puertos del `tethereal`, por lo que `ports` cerraba su ejecución y de paso mataba la del `ScD`, por lo que los resultados obtenidos en el monitor de comunicación eran nulos.

Para solucionar este problema se implementó una llamada al comando `wc -l`, el comando cuenta el número de lineas de un fichero, concretamente la salida de la ejecución de `netstat --numeric-ports -ptuw` para el proceso seleccionado. La opción `--numeric-ports` muestra el número de puerto sin afectar por la resolución de nombres de equipo, por su parte `-p` muestra el nombre de los procesos propietarios de los sockets, `-tu` indica que debe mostrar tanto sockets TCP como UDP y por último `-w` muestra la entrada *raw*. El nombre del proceso para el que está ejecutandonse el comando *ports* se filtra mediante el comando de sistema `grep` dejando únicamente las lineas que afectan a dicho proceso.

Una vez obtenido el número de puertos utilizados por el proceso y sus datos, se construye la ejecución de `tshark` adecuada al número de puertos del proceso.

Un ejemplo de como se montaría el comando `tshark` es el siguiente. Consideremos un proceso `pidgin` que se encuentra la conexión establecida en dos puertos:

```
# netstat --numeric-ports -ptuw | grep pidgin
Proto Recv-Q Send-Q Local Address           Foreign Address         State      PID/Program name
tcp        0      0 192.168.2.101:33048     207.46.111.59:1863      ESTABLISHED6839/pidgin
tcp        0      0 192.168.2.101:39606     209.85.163.125:5222     ESTABLISHED6839/pidgin
```

Por tanto, con `wc -l` obtendremos que hay dos puertos en el fichero `/tmp/ports` donde se direcciona la salida de este comando y así construir correctamente el comando `tshark` para capturar paquetes:

```
# /usr/bin/ports pidgin eth1
tshark -l -p -i eth1 -f  "  src port 33048  or  src port 39606  " -o
column.format:"temps",%t,"len",%L -T text >> /tmp/class_fifo
Capturing on eth1
```

En el caso de ser un proceso con una conexión activa, como sería Amarok reproduciendo un *streaming* por internet tendríamos:

```
# /usr/bin/ports amarokapp eth1
tshark -l -p -i eth1 -f "src port 42932" -o
column.format:"temps",%t,"len",%L -T text >> /tmp/class_fifo
```

Mientras que si el proceso no tiene conexiones activas y para evitar cuelgues en el `ScD` por no recibir ningún tipo de información se ha decidido mantener la implementación anterior y que capture paquetes en el puerto 0.

```
# /usr/bin/ports evince eth1
tshark -l -p -i eth1 -f "src port 0" -o
column.format:"temps",%t,"len",%L -T text >> /tmp/class_fifo
```

El código fuente del proceso `ports` se encuentra en el apendice **??**.

3.3. Implementación de la VcU

La VcU (Unidad de Visualización y Control) es la parte gráfica de Monito. En esta versión esta parte ha sufrido fuertes cambios, algunos visibles por el usuario y otros en limpieza de su código fuente, utilización de hojas de estilos en cascada y funciones de PHP para facilitar su administración.

3.3.1. Árbol de directorios

En la figura 3.2 se muestra la estructura de los directorios que se han creado en MoniTo para estructurar mejor los diferentes ficheros creados en la nueva estructura.

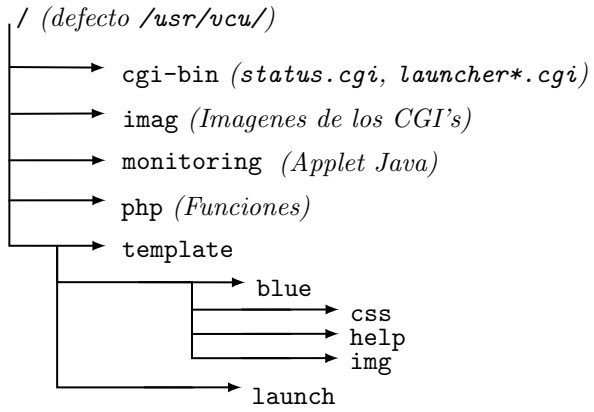

Figura 3.2: Árbol de directorios de MoniTo 4

En el directorio raíz se sitúan todos los ficheros *php* que muestran la interfaz web de MoniTo. Ese directorio raíz por defecto es `/usr/vcu/` y la página de inicio es *index.php*. En la figura 3.3 se muestra la página principal de la VcU de MoniTo.

El directorio `cgi-bin` contiene unos ejecutables que generan parte del contenido del portal web. En 3.3.2 se explicarán más detalladamente los cambios realizados, pero como adelanto, se ha sacado del CGI prácticamente todo el código HTML posible para conseguir la mayor independencia posible entre el portal web y la aplicación.

El directorio `imag` contiene las imágenes que utilizan las (pocas) porciones HTML incrustadas en los CGI. Se han separado del template ya que al encontrarse en un ejecutable precompilado no puede determinarse antes de su ejecución la ruta de la piel del portal web. Este directorio en otras versiones se llamaba `images`, pero ha sido cambiado el nombre para evitar configuraciones extra en Apache. Esto era debido a que Apache buscaba esas imágenes en `/usr/share/images/`, con el nuevo nombre no se da pie a que Apache busque las imágenes en un directorio de similar nombre, sino en el sitio virtual que ya se ha definido.

El directorio `monitoring` contiene los ficheros del Applet Java. Su función es realizar una representación gráfica de la información enviada por los ScD. No se realizarán más menciones ya que su implementación se mantiene respecto a versiones anteriores.

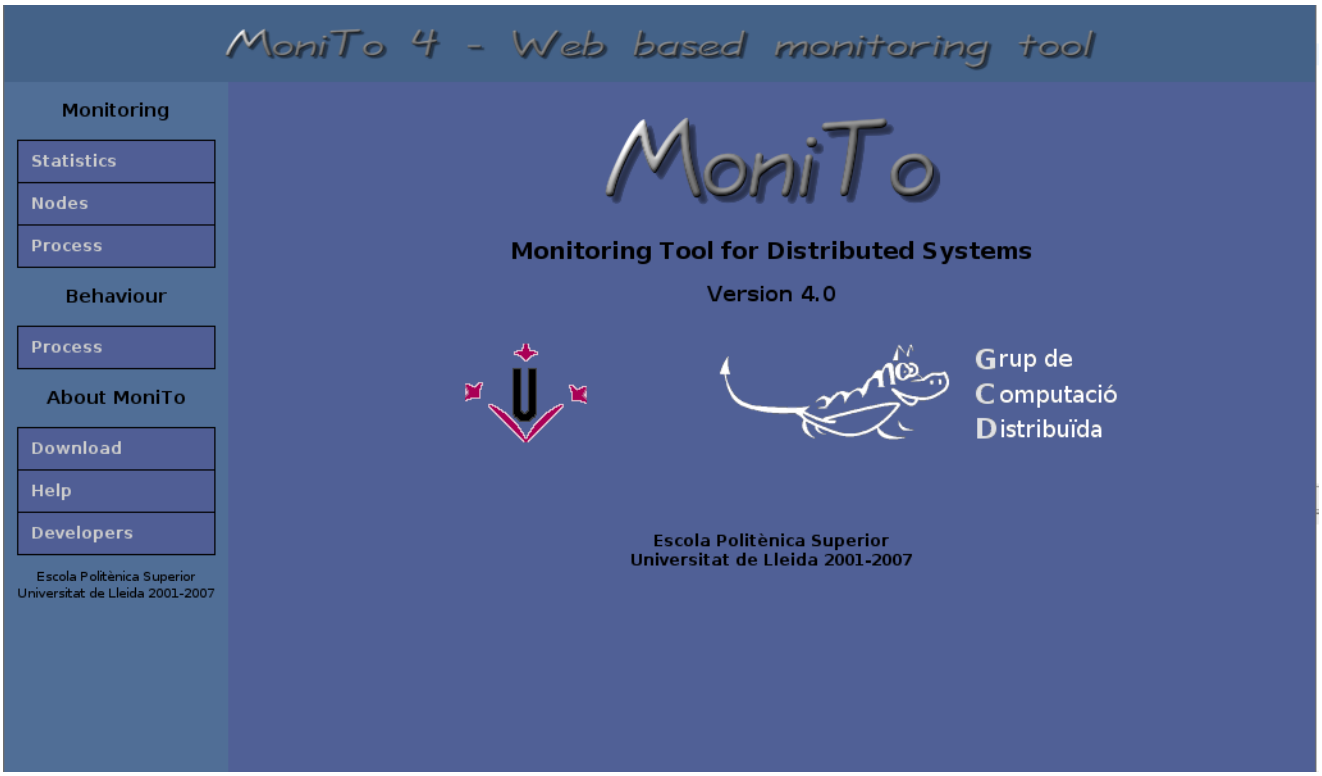

Figura 3.3: Página principal de MoniTo 4

En el directorio `php` se encuentran los ficheros con funciones comunes a varios (o incluso a todos) los ficheros del portal web. Ahí se encuentra el fichero de configuración, las funciones javascript y las propias de cada grupo de ficheros.

El directorio `template` es el directorio que contiene todos los ficheros CSS y demás archivos necesarios para generar el diseño del portal web. En un primer nivel encontramos el directorio `launch` con los estilos de las ventanas emergentes generadas tras la monitorización. Por otra parte el directorio `blue` es el *template* por defecto, este se puede cambiar editando el fichero `php/config.php` indicando la ruta del nuevo *template*. Dentro del directorio encontramos:

- `css`: Incluye todas las hojas de estilo en cascada del *template*.
- `img`: Imágenes propias del *template*.
- `help`: Imágenes propias de la ayuda incluída en la interfaz web. Se ha creado esta carpeta por si se desea incluir imágenes de la nueva piel en la ayuda, en vez de utilizar las generadas por el *template* por defecto.

3.3.2. Modificaciones en los CGI

3.3.2.1. status.cgi

Los CGI de `status` se encargan de conectarse al McD y obtener una lista de host junto al estado de sus monitores. Como se ha dicho anteriormente, el objetivo de conseguir independencia entre el portal web y el monitor implicaba cambios en este CGI.

En la figura 3.4 se marca con aerógrafo negro la parte generada por `status.cgi`. Son los datos del host a monitorizar y su estado. Esas celdas se obtienen durante la ejecución del CGI, además se ha implementado de manera que pueda variar su diseño según los parámetros dados en la hoja de estilo.

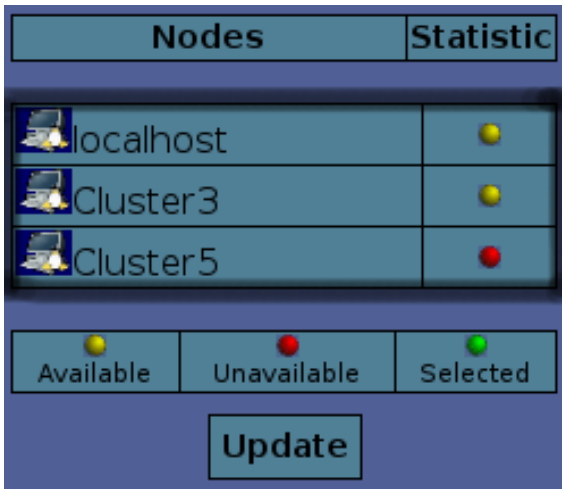

Figura 3.4: Ejecución del CGI de estado

En la versión anterior, `status.cgi` generaba la página completa, lo que implicaba que para cualquier cambio que se quisiese hacer fuera obligatorio recompilar el CGI. Ahora la leyenda, las funciones javascript y el enlace de actualización se incluyen desde un fichero `status*.php`.

La ejecución del CGI se realiza gracias a la función de PHP `passthru` de la siguiente manera:

```
passthru('cgi-bin/status.cgi 5');
```

No se ha podido parametrizar en un único fichero de `status` en PHP debido a que la llamada a `passthru` no acepta el paso de variables. Sin embargo, si que se ha podido mantener un único CGI `status` recogiendo el número de monitor desde parámetro.

En los anexos se incluye el código fuente de ambos ficheros involucrados.

3.3.2.2. launcher.cgi

Los CGI de lanzamiento (`launcher*.cgi`) generan una (o varias) ventana emergente en el navegador web del cliente inicia un proceso de monitorización en el cluster.

Dentro de la nueva política seguida en MoniTo 4 estos ficheros de lanzador han sufrido bastantes modificaciones. La primera de ellas es que únicamente se encargarán de generar la ventana emergente con los parámetros de la monitorización. Los parámetros se recogerán desde una página generada en PHP y después de la monitorización se volverá a esa misma página, en vez de cargar una generada por el CGI como hacía anteriormente. La principal ventaja de este sistema es la separación de roles, por un lado la interfaz gráfica, por otro las páginas HTML (generadas por PHP) y los CGI para generar la información imprescindible para la ejecución de MoniTo. La figura 3.5 muestra el diagrama de flujo de la ejecución del `launcher2.cgi`, correspondiente al monitor de estadísticas en un momento determinado del cluster.

Primero de todo, en la página `sta_main.php` se encuentra el formulario donde el usuario selecciona el intervalo de tiempo, la agrupación y los parámetros que desea monitorizar. Al apretar el botón `Submit` el formulario se envía y se inicia la ejecución de `launcher2.cgi`.

La ejecución del `launcher` incluye una etiqueta `meta` con una acción `refresh`, de manera que vuelve a cargar el fichero `sta_main.php`, que como se ha dicho anteriormente es dónde se encuentra el formulario de selección de parámetros. Anteriormente se tenía una versión duplicada de la misma página, por un lado en HTML y por otro en la función `html_params_page` del *launcher*, lo cual implicaba propagar cualquier cambio en los dos ficheros (incluyendo una recompilación) para que no se notase la diferencia, y además no permitía el uso en la página precompilada de hojas de estilo en cascada.

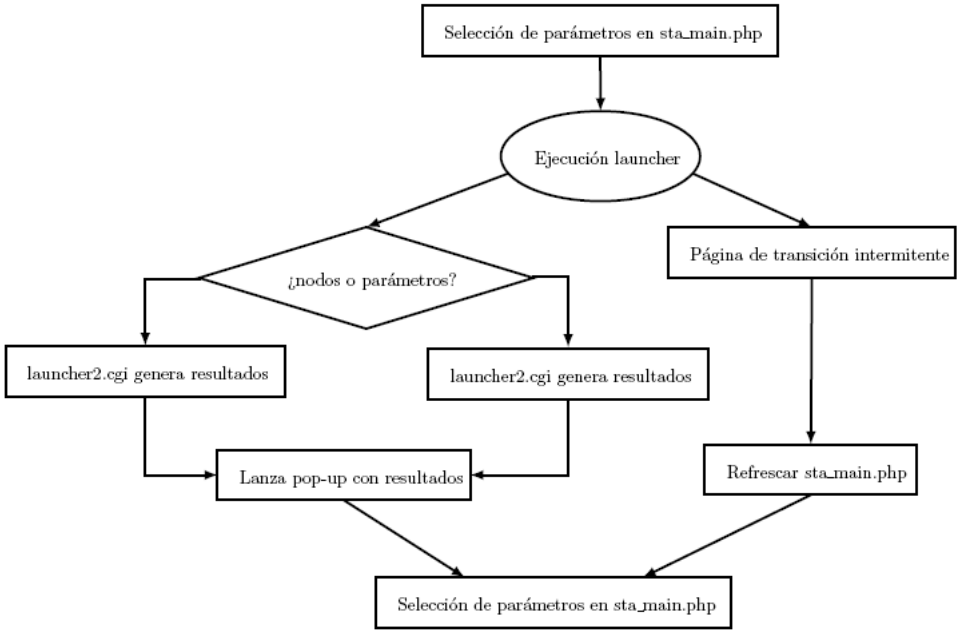

Figura 3.5: Diagrama de flujo de la ejecución de `launcher2.cgi`

Por el otro lado del diagrama, el *launcher* según la agrupación realizada (nodos o parámetros) genera una ventana emergente con los resultados de la monitorización. Esta ventana puede contener un *applet* en Java o una página HTML con una tabla.

Una vez finalizado todo vuelve al estado inicial, la página `sta_main.php` esperando la introducción de parámetros a monitorizar.

3.3.2.3. Configuración de la VcU con PHP

Desde este trabajo se ha pretendido sentar las bases del uso de un lenguaje interpretado del lado del servidor como PHP. La inclusión de instrucciones PHP atiende a los siguientes motivos:

- Controlar el *template* de la VcU desde un fichero de configuración.
- Imprimir código HTML común a varias páginas de la VcU mediante funciones.
- Poder utilizar instrucciones de programación para la interfaz web.

Seguidamente se dará un ejemplo de cada item en la implementación del portal web de MoniTo 4.

Una hoja de estilos en cascada se incluye en HTML con la siguiente instrucción:

```
<link rel="stylesheet" type="text/css" href="template/blue/css/mainstyle.css" />
```

La ruta donde se encuentra dicha hoja de estilos viene determinada por una variable `TemplatePath` cuyo valor se puede consultar o modificar en el fichero de configuración `config.php`. De esta manera con la edición de un fichero se puede cambiar completamente la apariencia de la unidad de visualización de MoniTo.

Del segundo item se puede considerar el caso de incluir en HTML una leyenda con la interpretación del estado del host que se va a monitorizar (ver en figura 3.4 la tabla con texto *Available*, *Unavailable* y *Selected*). Esa tabla se va a incluir en todos los ficheros de `status` y para evitar duplicidad de código y facilitar el mantenimiento o posteriores modificaciones en esa tabla se ha creado una función llamada `legend_table` para implementar esta capacidad.

Para el tercer y último item se considera el caso de querer generar un código HTML diferente según el monitor que esté siendo utilizado. Pues para generarlo automáticamente a través de un algoritmo es bastante simple utilizando PHP.

3.3.2.4. Hojas de estilo en cascada CSS

Tal como se ha explicado en varias ocasiones, en MoniTo 4 se han utilizado hojas de estilos en cascada para definir la presentación de la unidad de visualización y de paso, separarla de la estructura de dicha VcU.

En este apartado se explicarán las decisiones de diseño más relevantes que se han tomado en MoniTo 4. En el apartado 3.3.1 se observa que todas las hojas de estilo se encuentran dentro del directorio `template` seguido del identificador de tema (`blue` por defecto).

Dentro de cada hoja se encuentran los estilos definidos para cada una de las partes de la unidad de visualización de MoniTo.

Una decisión de diseño para simplificar el código de la aplicación es evitar el uso de tablas para maquetar el contenido en la medida de lo posible. Las ventajas principales de utilizar hojas de estilo en cascada para maquetar el contenido en vez de tablas serían:

- **Accesibilidad:** la separación de forma y contenido permite acceder a las personas con discapacidades a los contenidos de un sitio.
- **Menor cantidad de código que redunda en menores tiempos de carga:** Jugando con el posicionamiento es posible presentar unas partes del contenido antes que otras dando aún mayor sensación de velocidad.
- **Mantenimiento:** el cambio de aspecto del sitio resulta mucho más sencillo al estar todas las normas de presentación ubicadas en un punto único: las hojas CSS.
- **Futuro:** La maquetación con tablas es cosa del pasado. Si todos los fabricantes se han dado cuenta de la importancia de los estándares y los adoptan, estamos garantizando una viabilidad a largo plazo de nuestros trabajos. Hay que ponerse sobre el camino adecuado.
- **Gestión:** el contenido se presenta agrupado basándose en criterios lógicos gracias a la utilización de etiquetas div, pudiendo presentarse un módulo de contenidos con diferentes aspectos según la página desde la que es llamado o incluso no presentarlo.

De esta manera las tablas se utilizan para lo que inicialmente fueron concebidas, mostrar datos tabulados. Este caso sería la tabla de hosts y su disponibilidad o la tabla que se genera para mostrar las gráficas del monitor estadístico.

Un ejemplo de maquetación con listas y hojas de estilo en cascada se encuentra en el menú lateral `type.php`

```
<div class="menu4">
    <ul>
        <li><a>Statistics</a></li>
        <li><a>Nodes</a></li>
        <li><a>Process</a></li>
    </ul>
</div>
```

Dentro del div *menu4* se encuentran los tres items de una lista desordenada. Para simplificar se ha eliminado el destino de los enlaces. La definición del estilo de la lista lo proporciona el código CSS siguiente:

```
ul {
        list-style: none;
}
```

`list-style: none;` hace que la lista no tenga ningún tipo de item como indicador, como podría ser un circulo negro al principio del item.

```
.menu4 li a:link, .menu4 li a:visited
{
        background-color: #505e95;
        color: #CCC;
        display: block;
        padding: 8px 0 0 10px;
}

.menu4 li a:hover
{
        color: #000;
        background:  #506e96 0 -32px;
        padding: 8px 0 0 10px;
}
```

En estas dos instrucciones de CSS obtenemos el distinto comportamiento del item. En el momento en el que el puntero del ratón (`.menu4 li a:hover`) pasa por encima del item este cambia su color de fondo y de fuente mientras que si tiene un link en su interior (`a:link`) o ese propio link ha sido visitado (`a:visited`) mantiene el color de fondo `#505e95;` y cambiará el color de la fuente respecto del estado *hover*.

En la figura 3.6 se puede observar claramente el diseño comentado anteriormente:

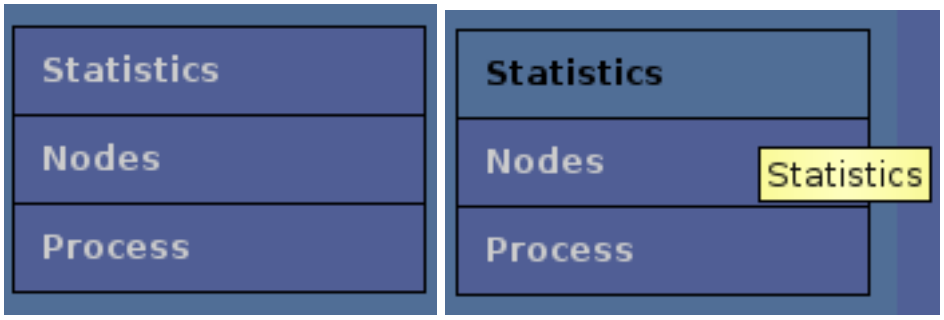

Figura 3.6: Menú diseñado con hojas de estilo en cascada

Este sólo es un pequeño ejemplo de la potencia de las hojas de estilo en cascada a la hora de diseñar y maquetar un sitio web.

Claramente representan una sustancial mejora tanto en claridad de código como en velocidad de transmisión ya que el diseño de la página sólo se transfiere la primera vez que se descarga la hoja de estilos, además de cara a un futuro mantenimiento todo este diseño se encuentra centralizado en un mismo sitio.

Además en versiones anteriores de MoniTo este mantenimiento implicaba tener que recompilar los lanzadores y los cgi de status ya que en ellos se generaba absolutamente todo el sitio web, con estilos embebidos en el propio fichero html.

3.4. Manual de usuario

3.4.1. Configuración del servidor Apache

3.4.1.1. Apache 1.3

En la versión 1.3 del servidor Apache la configuración importante se encuentra en un único fichero `httpd.conf`. De la configuración por defecto se deberían modificar los siguientes parámetros:

```
User www-data
```

```
Group www-data
```

`www-data` es el usuario por defecto que ejecuta el servidor Apache y es al que pertenecerán igualmente por defecto todos los ficheros de la unidad de visualización. Para evitar problemas de acceso a ficheros relacionados con los permisos de usuario estos deberán estar fijados a `755` (lectura, escritura y ejecución para el propietario, lectura y ejecución para grupo y resto de usuarios) para todos los ficheros del sitio.

```
DocumentRoot /usr/vcu/
```

La raíz del sitio Apache deberá direccionarse al directorio `/usr/vcu/`, es decir la ubicación por defecto de la unidad de visualización de MoniTo. No existe ninguna limitación para situar la VcU en cualquier directorio del sistema, simplemente deberá indicarse en la siguiente linea del archivo de configuración de Apache 1.3 para que el servidor vaya a buscar las páginas de MoniTo al lugar indicado.

```
    Alias /images/ /usr/share/images/

    <Directory /usr/share/images>
         Options MultiViews
         AllowOverride None
         Order allow,deny
         Allow from all
    </Directory>
```

Es conveniente realizar un comentario sobre el siguiente bloque. En versiones anteriores existía un directorio `images` con ciertos archivos de imagen que se utilizaban en la VcU. Pero al utilizar un alias, el servidor Apache los buscaba en `/usr/share/images`, para evitar configurar este aspecto se evita la confusión renombrando el directorio `images` a un simple `imag`, aunque actualmente la mayor parte de las imágenes se encuentran en el `template` del portal web.

```
<IfModule mod_alias.c>
    ScriptAlias /cgi-bin/ /usr/vcu/cgi-bin/
    <Directory /usr/vcu/cgi-bin/>
        AllowOverride None
        Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>
</IfModule>
```

Es imprescindible direccionar el directorio de ejecución de los CGI de MoniTo ya que en caso contrario no se podrá realizar ningún tipo de monitorización. Como se ha explicado en el apartado 3.3.2 los CGI se encargan de obtener información sobre el estado de los clusters a monitorizar y generar las ventanas emergentes con los resultados de la monitorización. Estos ejecutables escritos en C se encuentran por defecto en el directorio `/usr/vcu/cgi-bin`. Para evitar tener que mantener copias de los CGI's en directorios ajenos a la VcU hay que direccionar la ejecución de ellos hacía la ruta de la unidad de visualización y mas en concreto hacia el directorio `cgi-bin`.

```
# Default charset to iso-8859-1 (http://www.apache.org/info/css-security/).

AddDefaultCharset off
```

Cambiar la codificación por defecto de Apache a UTF-8 es otra de las opciones de configuración a realizar. UTF-8 es una codificación de caracteres de longitud variable. Usa grupos de bytes para representar el estándar Unicode para la mayoría de los lenguajes del mundo.

Toda la unidad de visualización emplea la codificación UTF-8 al incluir la siguiente etiqueta en la cabecera:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

Además el servidor Apache debe codificar los ficheros según UTF-8 para que aparezcan los carácteres correctamente y no tener que recurrir a entidades de HTML como `&egrave;` para representar el carácter è. El motivo del uso de UTF-8 es una mejor compatibilidad futura con AJAX y con posibles traducciones de la interfaz web a otros idiomas, que si tienen tíldes (como ocurre con el español y el catalán) simplifican la escritura del código ya se puede escribir todos los carácteres con tíldes y eñes sin problemas de codificación posteriores.

Una vez realizadas las siguientes modificaciones será obligatorio reiniciar el servidor Apache.

3.4.2. Apache 2.2

En la rama 2.2 se efectuaron cambios en la disposición y contenidos de los ficheros de configuración.

La definición del sitio virtual donde se encuentra MoniTo se encuentra en el fichero `sites-enabled` y se realiza de la siguiente manera:

```
DocumentRoot /usr/vcu
<Directory /usr/vcu/>
Options Indexes FollowSymLinks MultiViews
AllowOverride None
Order allow,deny
allow from all
</Directory>
```

También se puede configurar el servidor Apache para que busque la unidad de visualización en el directorio `home` de un usuario. Esta solución es interesante comentarla desde el punto de vista del administrador del cluster ya que el usuario tendrá permisos totales sobre los directorios de la unidad de visualización, al contrario de lo que ocurre en el directorio por defecto (`/usr/vcu/`), donde los permisos para un usuario son únicamente de lectura. Es también una solución interesante de cara a mantener ordenado el sistema de archivos del cluster, ya que de esta manera el servidor de MoniTo no toca directorios importantes en el sistema, simplemente se mantiene en la cuenta de usuario con todos los beneficios que ello conlleva.

En el fichero de configuración de Apache deberá incluirse la siguiente directiva:

```
    <Directory /home/*/public_html>
        AllowOverride FileInfo AuthConfig Limit
        Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
        <Limit GET POST OPTIONS PROPFIND>
            Order allow,deny
            Allow from all
        </Limit>
        <Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
            Order deny,allow
            Deny from all
        </Limit>
    </Directory>
</IfModule>
```

Los modulos activos se encuentran dentro del fichero `mods-enable`, dentro de ese fichero será imprescindible que se encuentren activos los módulos de interpretación de PHP y también la ejecución de cgi's.

Al igual que con Apache 1.3 es imprescindible que toda la unidad de visualización tenga como usuario y grupo a `www-data` salvo que se haya definido otro grupo para ellos. El cambio de propietario de la VcU evita problemas de `Premature end of script headers` relacionados con los permisos de usuario.

3.5. Paquetería en Linux

La paquetería es el método más común para distribuir software precompilado bajo Linux. Un paquete es un conjunto de programas y ficheros empaquetados. Los programas que contienen dichos paquetes son el resultado de la compilación del código fuente de una determinada aplicación.

Cada paquete tiene en su interior un archivo con información que contiene el nombre del paquete, la descripción, dependencias con otros paquetes, nombre de los mantenedores etc. Dentro de cada paquete uno de los campos más importantes es de las dependencias con otros paquetes. Cada aplicación suele depender de otras aplicaciones y librerías. con lo que en el sistema operativo se acaba generando un árbol de dependencias entre los diferentes paquetes que contienen estas aplicaciones y librerías. Es frecuente que al instalar un paquete tendremos que instalar otros paquetes del los que depende este primero.

Monito 4 se encuentra empaquetado en los dos principales sistemas de paquetes de Linux, los cuales son los paquetes `RPM` utilizados por Fedora y Suse y los paquetes `DEB` utilizados por Debian y derivadas como Ubuntu. Otro sistema de empaquetamiento es el `autopackage`, este sistema comprueba que se cumplen las dependencias necesarias para instalar el paquete. Si alguna no se cumple, las instala de manera automática. Los paquetes `.package` son realmente *scripts* de *bash* ejecutables, por lo que pueden ser instalados simplemente ejecutándolos. Sin embargo este sistema ha sido descartado al ser una aplicación relacionada con el nucleo del sistema operativo, motivo por el cual no es recomendable utilizar *autopackage* por razones de velocidad y compatibilidad. En estos momentos *autopackage* se encuentra en fase de desarrollo activo.

3.5.1. Estructura Interna

Los paquetes `deb` son un archivador estándar. Estos paquetes contienen tres archivos:

- `debian-binary`: Número de versión del formato deb
- `control.tar.gz`: Contiene toda la metainformación del paquete.
- `data.tar.gz`: Archivos y directorios que se instalan.

Los paquetes `rpm` también son un archivador estandar. En su caso se encuentran compuestos por:

- `spec`: Fichero de especificación con toda la información necesaria para poder construir el paquete `rpm`.
- La firma del paquete para asegurar su integridad y/o autenticidad.
- Un archivador comprimido en *gzip* con los archivos de la aplicación.

Como se ha dicho anteriormente, todos los formatos de paquetes binarios son muy parecidos. Todos cumplen la misma función, y solo se diferencian en la estructura y forma interna del paquete. Eso posibilita que cualquier tipo de paquete binario se pueda convertir a cualquier otro tipo. Con este objetivo se creó una herramienta llamada *Alien*.

Alien conoce varios tipos de paquetes binarios ente ellos los dos formatos existentes para Linux, los paquetes tipo `rpm` y los paquetes tipo `deb`. Alien puede transformar un paquete `rpm` a `deb` y viceversa, conservando las dependencias y el resto de información.

3.5.2. Estructuración en paquetes de la herramienta

En esta versión de Monito se distribuye en un paquete el código fuente, la unidad de visualización y el Master Collector Daemon (McD) y en otro el cliente Slave Collector Daemon (ScD). Estos paquetes son:

- `monito-vcu-mcd_4.0-1_i386.*` con el McD y el portal web.
- `monito-scd_4.0-1_i386.*` con el ScD

No se ha considerado incluir el servidor web Apache con el módulo del lenguaje PHP por si llegado el momento de eliminar o bien la aplicación o bien el servidor Apache para sustituirlo por otro podría ocasionar problemas de dependencias que llevaran al usuario a tener que desinstalar paquetes no deseados.

Por tanto, se deja en manos del usuario que instale por su cuenta un servidor capaz de interpretar PHP. Como se ha dicho anteriormente este servidor requerirá una pequeña configuración previa antes de empezar a utilizarse la unidad de visualización.

Mientras que el ScD no dará por supuesto que el sistema donde va a ser instalado dispone de las librerías LibGTop razón por la cual han sido incluídas como dependencia en este paquete.

Para facilitar la creación de los paquetes rpm y deb con los que se distrubuiría la herramienta precompilada se creó en la versión 2 de la herramienta dos scripts llamados *crea_deb* y *crea_rpm*. Estos dos scripts han sido actualizados para seguir siendo compatibles con la nueva versión de la herramienta Monito.

3.6. Experimentación

Para probar el monitor de un cluster MoniTo se realizaron una serie de pruebas especialmente encaminadas a evaluar su intrusión en los sistemas sobre los cuales se ejecuta la monitorización.

Un monitor debe estar cuidadosamente implementado, con el fin de utilizar el menor número de recursos posibles.

Para realizar una prueba de intrusión se han utilizado dos equipos, uno en el rol de servidor y otro en el de cliente con las siguientes características:

SERVIDOR: Intel Celeron 2.40 GHz, 128 MB de memoria caché, 512 MB de memoria RAM y sistema operativo Ubuntu 7.04

CLIENTE: Intel Celeron 2.40 GHz, 256 MB de memoria caché, 512 MB de memoria RAM y sistema operativo Ubuntu 7.10

Ambos equipos se encuentran interconectados mediante sus tarjetas de red Ethernet, el servidor tiene dirección IP 192.168.0.2 mientras que el cliente 192.168.0.1, por lo que respecta a la salida a Internet ambos equipos utilizan la interfaz inalámbrica `wlan0` con direcciones 172.30.1.201 y 172.30.1.202.

MoniTo utiliza el protocolo UDP para el paso de mensajes entre cliente y servidor a través de la red. En este protocolo de la capa de transporte los paquetes basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación, ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o de recepción.

Como se puede comprobar en la captura de tráfico de red generado por una monitorización de cuatro parámetros en el monitor estadístico y agrupados en un nodo la intrusión en la red es pequeña comunicando entre dos equipos, un mensaje en cada sentido de la transmisión. La captura ha sido realizada con el *sniffer* de libre distribución *Wireshark* y se muestra en la figura 3.7.

	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.2	192.168.0.1	UDP	Source port: 4004 Destination port: 5004
2	0.000908	192.168.0.1	192.168.0.2	UDP	Source port: 5004 Destination port: 4004

Figura 3.7: Tráfico de red generado por MoniTo

La ejecución tanto en el cliente y en el servidor realiza una carga de procesador despreciable en los equipos del cluster, al igual que tampoco supone un consumo de memoria considerable y si se extrapolan los datos de intrusión de red a un cluster con un cierto número de equipos, un mensaje en cada sentido por monitorizació no supone una intrusión considerable por lo que en este aspecto la herramienta se podría decir que no tiene una importante afección en el sistema.

Sin embargo hay un apartado de la herramienta global que genera una amplia carga en el sistema y no es otro que el uso de Java para la representación de las gráficas con los resultados de la monitorización.

Para confirmar este hecho se ha realizado una prueba monitorizando el uso de procesador del sistema comparado con un proceso que a priori requiere muy poco uso de CPU, este proceso es `wget`. Con `wget` se pueden descargar archivos desde la web de forma no interactiva.

El archivo seleccionado para la descarga ha sido seleccionado teniendo en cuenta que tenga un tamaño importante para no limitar la duración de la prueba. Durante la descarga del archivo la salida a internet mediante la interfaz *wlan0* estaba utilizando el 100 % del ancho de banda disponible (55.6 KB/s) pero al utilizar el monitor mediante la interfaz cableada *eth0* no afecta a las transferencias internas de MoniTo.

El monitor seleccionado para la prueba es el de comportamiento, este monitor añadido en MoniTo 3 permite comparar un parámetro de un proceso en comparación con ese mismo parámetro para todo el sistema. El tiempo de duración de la monitorización será de 10 segundos y muestra los resultados obtenidos en la figura 3.8.

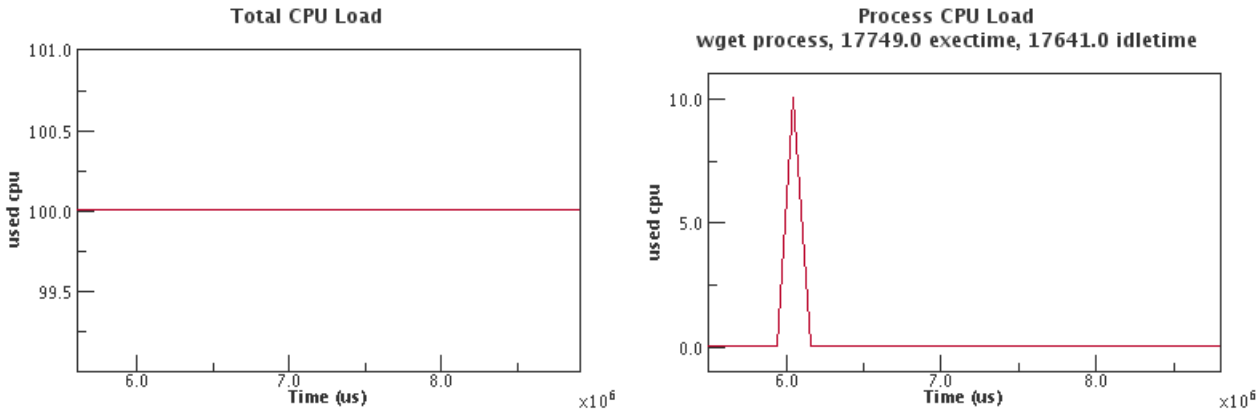

Figura 3.8: Gráficas de uso de CPU del sistema que ejecuta la VcU de MoniTo comparado con wget

Como se puede observar en la figura 3.8 el procesador se encuentra al 100 % de carga pero prácticamente ninguno de esos ciclos de procesador han sido consumidos por el proceso `wget`, a excepción del pequeño pico del 10 % de carga.

Para afianzar la idea global también se muestra una captura del comando `top` durante la ejecución de la monitorización y generación de la gráfica en java. La figura 3.9 confirma que casi todo el procesador lo consume el proceso `java_vm`, quedandose un porcentaje reseñable el navegador `swiftfox-bin` y viendose como el proceso monitorizado `wget` utiliza un ínfima parte de los ciclos de procesador del sistema. Como dato curioso observese que consume más memoria el navegador que la máquina virtual de java.

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 8685 telecent  15   0  229m  39m  16m S 84.0  8.0   1:13.60 java_vm
 8259 telecent  16   0  170m  51m  22m S  7.9 10.4   2:13.74 swiftfox-bin
 4935 root      15   0  227m  29m 9588 S  4.2  5.9   4:41.33 Xorg
 5359 telecent  15   0 48492  22m  15m R  1.2  4.5   0:28.40 gnome-panel
 6886 telecent  15   0 59968  18m  10m S  0.9  3.8   0:06.27 gnome-terminal
 8652 telecent  15   0  4160 1552 1256 S  0.7  0.3   0:01.78 wget
```

Figura 3.9: Salida del comando `top` durante monitorización con Java

Esta fuerte intrusión en el sistema se producirá en el sistema desde el cual se acceda a la interfaz web, bien podría ser el propio equipo (*localhost*) que la aloja o bien accedida desde un lugar externo. Sea cual sea el caso y conocida la tendencia a desaparecer los *applets* de Java de un sitio web daría lugar a replantearse la conveniencia de seguir utilizando Java para representar los resultados de la monitorización, esta tarea queda enunciada para posibles trabajos futuros.

Capítulo 4

Conclusiones y trabajo futuro

4.1. Conclusiones

En este proyecto de final de carrera se ha realizado un análisis detallado de las librerías LibGTop que se han utilizado en la nueva implementación de la herramienta de monitorización MoniTo, sistema avanzado de monitorización de clusters. También se ha aprovechado la revisión de la herramienta para mejorar el cliente web que muestra los resultados de la monitorización.

Respecto a la migración de la obtención de datos utilizando los librerías LibGTop en vez de leer los datos del sistema de archivos */proc* se puede observar una mejora de la aplicación ya que el uso de las librerías evita la excesiva dependencia de cambios en el núcleo de Linux que pudieran afectar al funcionamiento de la herramienta y un nivel de abstracción mayor en el código fuente. El rendimiento de MoniTo sigue siendo bueno y es posible que haya mejorado con la sustitución de lecturas y parseo de los datos obtenidos en */proc* por la obtención directa de las estructuras de LibGTop.

La nueva implementación facilita la adición de nuevos parámetros a monitorizar dentro de la herramienta, ya que no obliga a tener conocimiento de la estructura de /proc sino únicamente a añadir el parámetro que se quiere obtener de las estructuras de LibGTop y enviar los datos al ScD.

Se ha mejorado sustancialmente la interfaz web de la unidad de visualización, esta presentaba problemas como el uso de etiquetas html obsoletas, mucho código redundante y una nula separación entre la estructura y el diseño de la interfaz. Para arreglar estas deficiencias se ha eliminado todo el código html incluído en el código precompilado de los lanzadores de la monitorización que no influyese en el funcionamiento de la herramienta, separar el diseño de la estructura utilizando hojas de estilo en cascada, poner las bases para el uso de templates y muchas más posibilidades utilizando el lenguaje de programación de páginas dinámicas PHP y por último reducir la sobrecarga del código evitando el uso de tablas para maquetar en la medida de lo posible.

Todas estas modificaciones en la unidad de visualización han conseguido realizar una interfaz más sencilla, amigable y fácil de modificar para el usuario.

En global la herramienta ha salido ganando con el cambio en la forma de obtener los datos y con la mejora de la interfaz web. De todas formas los aspectos que se detallan en este apartado y el siguiente son claramente mejorables.

4.2. Trabajo futuro

En este apartado se trata de explicar detalles o mejoras que quedan pendientes y que pueden ser realizadas en trabajos futuros.

Un aspecto que se podría mejorar en versiones futuras es buscar alternativas al uso de Java para realizar las gráficas generadas durante la monitorización. La principal ventaja de Java es su enorme potencialidad a la hora de poder generar gráficas pero a cambio tiene una gran desventaja en la carga de procesador que genera en el equipo cliente donde se ejecuta la máquina virtual de Java.

Otros aspectos que se pueden introducir en la interfaz web es nuevas tecnologías como AJAX[1] de manera que la aplicación se ejecutaría en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. También se podrían añadir usuarios, una interfaz de administración del cluster vía web y diseñar nuevos *templates* que complementen al azul actual que lleva la herramienta por defecto.

Otro trabajo importante sería mantener la herramienta compatible con versiones recientes de todas las herramientas que utiliza, desde las librerías LibGTop pasando por el servidor Apache o el lenguaje PHP y acabando en cualquier nueva tecnología que se incluya en la herramienta. Si una aplicación no se adapta a estos cambios, su código se acaba volviendo obsoleto y llega un momento que deja de funcionar. Por lo tanto, MoniTo requiere un esfuerzo para mantener el código siempre actualizado a los últimos cambios del sistema operativo evitando así que el código se vuelva obsoleto.

[1] Acrónimo de Asynchronous JavaScript And XML

Apéndice A

Código fuente de la interfaz Web de MoniTo

A.1. Ficheros PHP de interfaz web

A.1.1. index.php

```
<?php

require_once('php/config.php');

frameset_header();

echo"\n<head>\n\t<title>{$title}</title>\n</head>\n";

echo<<<FRAMESET
<frameset rows="60,*" border="0">
    <frame src="+title.php" name="title" scrolling="no" />
        <frameset cols="175,*" border="0">
            <frame src="+type.php" name="type" noresize="noresize" />
            <frame src="+main.php" name="main" noresize="noresize" />
        </frameset>
</frameset>
</html>

FRAMESET;

?>
```

A.1.2. +title.php

```
<?php

require_once('php/config.php');
require_once('php/javascriptfunctions.php');
print_header();

echo"\n<link rel=\"stylesheet\" type=\"text/css\"
```

```
href=\"{$TemplatePath}css/titlestyle.css\" />\n";

javascript_function();

echo<<<HTML
</head>
<body>
    <a href="+main.php" target="main">
    <img src="{$TemplatePath}img/title.png" alt="{$title}" />
    </a>
</body>
</html>
HTML;

?>
```

A.1.3. +type.php

```
<?php

require_once('php/config.php');
print_header();

echo"\n<link rel=\"stylesheet\" type=\"text/css\"
href=\"{$TemplatePath}css/typestyle.css\" />\n</head>\n";

echo<<<BODY
<body>
<p class="options">Monitoring</p>
<div class="menu4">
    <ul>
        <li><a href="_stadistic.php" title="Statistics" target="main">Statistics</a></li>
        <li><a href="_node.php" title="Nodes" target="main">Nodes</a></li>
        <li><a href="_process.php" title="Process" target="main">Process</a></li>
    </ul>
</div>
    <p class="options">Behaviour</p>
<div class="menu4">
    <ul>
        <li><a href="_class.php" title="Process" target="main">Process</a></li>
    </ul>
</div>
    <p class="options">About MoniTo</p>
<div class="menu4">
    <ul>
        <li><a href="download.php" title="Download" target="main">Download</a></li>
        <li><a href="help.php" title="Help" target="main">Help</a></li>
        <li><a href="help.php#developers" title="Developers" target="main">Developers</a></li>
    </ul>
</div>
    <p class="date"> Escola Polit&egrave;nica Superior<br/>Universitat de Lleida 2001-2008</p>
</body>
```

```
</html>
BODY;

?>
```

A.1.4. +main.php

```
<?php

require_once('php/config.php');
require_once('php/launchfunctions.php');

print_header();

echo"\n<link rel=\"stylesheet\" type=\"text/css\"
href=\"{$TemplatePath}css/mainstyle.css\" />\n</head>\n";

echo<<<BODY
<body>
    <div id="logo">
        <img src="{$TemplatePath}img/logo.png" alt="{$title}"/>
        <h1>Monitoring Tool for Distributed Systems</h1>
        <h2>Version 4.0</h2></div>
    <div id="trescol">
        <ul>
            <li class="col1"><a href="http://www.eps.udl.es" target="_parent">
            <img src="{$TemplatePath}img/logo_udl.gif" alt="EPS" /></a></li>
            <li class="col2"><a href="http://gcd.udl.es" target="_parent">
            <img src="{$TemplatePath}img/logo_gcd.png" alt="GCD" /></a></li>
        </ul>
    </div>
    <p>Escola  Politènica Superior<br/>Universitat de Lleida 2001-2007</p>
BODY;

echo"\t\n</body>\n</html>";
?>
```

A.1.5. _statistic.php

```
 <?php

require_once('php/config.php');

frameset_header();

echo"\n<head>\n\t<title>{$title}</title>\n</head>\n";

echo<<<FRAMESET
<frameset cols="250,*" framespacing="0" border="0" frameborder="no">
    <frame src="status2.php" name="cluster" noresize>
    <frame src="sta_main.php" name="main" noresize>
</frameset>
```

```
</html>
FRAMESET;

?>
```

A.1.6. _node.php

```
<?php

require_once('php/config.php');

frameset_header();

echo"\n<head>\n\t<title>{$title}</title>\n</head>\n";

echo<<<FRAMESET
<frameset cols="250,*" framespacing="0" border="0" frameborder="no">
    <frame src="status1.php" name="cluster" noresize>
    <frame src="node_main.php" name="main" noresize>
</frameset>
</html>
FRAMESET;

?>
```

A.1.7. _process.php

```
<?php

require_once('php/config.php');

frameset_header();

echo"\n<head>\n\t<title>{$title}</title>\n</head>\n";

echo<<<FRAMESET
<frameset cols="250,*" framespacing="0" border="0" frameborder="no">
    <frame src="status1.php" name="cluster" noresize>
    <frame src="process_main.php" name="main" noresize>
</frameset>
</html>
FRAMESET;

?>
```

A.1.8. _class.php

```
<?php

require_once('php/config.php');

frameset_header();
```

```
echo"\n<head>\n\t<title>{$title}</title>\n</head>\n";

echo<<<FRAMESET
<frameset cols="250,*" framespacing="0" border="0" frameborder="no">
    <frame src="status3.php" name="cluster" noresize>
    <frame src="class_main.php" name="main" noresize>
</frameset>
</html>
FRAMESET;

?>
```

A.1.9. status.php

Existen tres scripts de *status*, uno por cada conjunto distinto de parámetros que se puede monitorizar. El código es idéntico para los tres salvo el parámetro que se pasa utilizando la función `init_hosts_table` con el que se formará la tabla de estado de los monitores en el equipo que ejecuta una instancia del ScD.

```
<?php

require_once('php/config.php');
require_once('php/statusfunctions.php');

print_header();

echo"\n<link rel=\"stylesheet\" type=\"text/css\"
href=\"{$TemplatePath}css/statusstyle.css\" />\n";
javascript_functions();

echo"\n</head>\n<body>\n";

// Definir los monitores que se quieren activar, el código será
// idéntico para la función init_hosts_table y para el CGI status.cgi
$monact = '123';

init_hosts_table($monact);

echo"<table>\n";

passthru('cgi-bin/status.cgi 123');

echo"</table>\n";

echo"<br/>\n";
legend_table();
update_hosts();
echo"\n\t</body>\n";
echo"</html>";
?>
```

A.1.10. node_main

```
<?php

require_once('php/config.php');
require_once('php/launchfunctions.php');

print_header();

echo"<head>\n<link rel=\"stylesheet\" type=\"text/css\"
href=\"{$TemplatePath}css/nodemainstyle.css\" />";

javascript_functions();
echo"</head>";

echo<<<BODY
<body>
<h1>NODE SPECIFIC INFORMATION</h1>
    <form name="form1" action="cgi-bin/launcher3.cgi">
    <h2>General Monitoring Parametres</h2>
        <div class="form_center">
            <ul class="grouped_list">
                <li><span>Rate</span><br/>
                    <input type="text" name="rate" size="10"> (>=100ms)</li>
                <li><span>Time interval</span><br/>
                    <input type="text" name="interval" size="10"> (s)</li>
                <li><span>Monitoring type</span><br/>
                    <input type="radio" name="mtype" value="0" checked> On-line
                    <input type="radio" name="mtype" value="2">Off-Line</li>
                <li><span>FileName (off-line)</span><br/>
                    <input type="text" name="filename" size="31"></li>
            </ul>
        </div>
    <h2>Specific Monitor Parameter</h2>
        <div class="form_center">
            <ul class="time_list">
                <li><span>CPU</span></li>
                <li><span>MEMORY</span></li>
                <li><span>NET</span></li>
                <li>
                <select name="lparameter" size="3" multiple>
                    <option value="0-Total CPU Load-used cpu" selected>Total CPU load
                    <option value="1-User CPU Load-used cpu">User CPU load
                    <option value="2-Nice CPU Load-used cpu">Nice CPU load
                    <option value="3-System CPU Load-used cpu">System CPU load
                    <option value="4-Idle CPU-idle cpu">Idle CPU
                    <option value="5-IO CPU wait-IO cpu wait">IO CPU wait
                </select>
                </li>
                <li>
                <select name="mparameter" size="3" multiple>
                    <option value="8- Total Main Mem-Mbytes" selected>Total Main Mem
```

```
                    <option value="9- Used Main Mem-Mbytes">Used Main Mem
                    <option value="10- Free Main Mem-Mbytes">Free Main Mem
                    <option value="11- Shared Main M-Mbytes">Shared Main Mem
                    <option value="12- Total Swap Mem-Mbytes">Total Swap Mem
                    <option value="13- Used Swap Mem-Mbytes">Used Swap Mem
                    <option value="14- Free Swap Mem-Mbytes">Free Swap Mem
                </select>
                </li>
                <li>
                <select name="cparameter" size="3" multiple>
                    <option value="4-Num. kbytes send-kbytes" selected>Kbytes Send
                    <option value="5-Num. kbytes received-kbytes">Kbytes Received
                    <option value="6-Num. errors sending-packets">Errors Send
                    <option value="7-Num. errors receiving-packets">Errors Received
                    <option value="8-Collisions-packets">Collisions
                </select>
                </li>
            </ul>
        </div>
    <div class="submit">
        <input type="submit" name="blaunch" value="Submit">
        <input type="hidden" name="hparams" value=" ">
    </div>
BODY;

view_offline_file();

echo"\n\t\t</form>\n\t</body>\n</html>";

?>
```

A.1.11. process_main

```
<?php

require_once('php/config.php');
require_once('php/launchfunctions.php');

print_header();

echo"\n<link rel=\"stylesheet\" type=\"text/css\"
href=\"{$TemplatePath}css/processmainstyle.css\" />";

javascript_functions();
echo"</head>";

echo<<<BODY
<body>
<h1>PROCESS SPECIFIC INFORMATION</h1>
    <form name="form1" action="cgi-bin/launcher.cgi">
    <h2>General Monitoring Parametres</h2>
        <div class="form_center">
```

```
            <ul class="grouped_list">
                <li><span>Process Name</span><br/>
                    <input type="text" name="process" size="24"></li>
                <li><span>Rate</span>
                    <input type="text" name="rate" size="10"> (>=100ms)<br/>
                <span>Time Interval</span>
                    <input type="text" name="interval" size="10"> (s)</li>
                <li><span>Monitoring type</span><br/>
                    <input type="radio" name="mtype" value="0" checked> On-line
                    <input type="radio" name="mtype" value="2"> Off-Line</li>
                <li><span>FileName (off-line)</span><br/>
                    <input type="text" name="filename" size="31"></li>
            </ul>
        </div>
    <h2>Specific Monitor Parameter</h2>
        <div class="form_center">
            <ul class="time_list">
                <li><span>CPU</span></li>
                <li><span>MEMORY</span></li>
                <li><span>NET</span></li>
                <li>
                <select name="lparameter" size="3" multiple>
                 <option value="6-Process CPU Load-used cpu" selected>Process CPU load</select>
                </li>
                <li>
                <select name="mparameter" size="3" multiple>
                 <option value="0-Pages in swap file-Pages" selected>Pages in swap file
                 <option value="1-Virtual memory size-Pages">Virtual Memory Size
                 <option value="2-Resident pages-Pages">Resident pages
                 <option value="3-Pages touched-Pages">Pages touched
                 <option value="4-Pages fetch. min_flt-Pages">Pages fetch (min_flt)
                 <option value="5-Pages fetch. maj_flt-Pages">Pages fetch (maj_flt)
                </select>
                </li>
                <li>
                <select name="cparameter" size="3" multiple>
                  <option value="0-Bytes in Receive Queue-Bytes" selected>Bytes in Receive Queue
                  <option value="1-Bytes in Send Queue-Bytes">Bytes in Send Queue
                </select>
                </li>
            </ul>
        </div>
    <div class="submit">
        <input type="submit" name="blaunch" value="Submit">
        <input type="hidden" name="hparams" value=" ">
    </div>
BODY;

view_offline_file();

echo"\n\t\t</form>\n\t</body>\n</html>";
?>
```

A.1.12. sta_main.php

```
<?php

require_once('php/config.php');

print_header();

echo"<head>\n<link rel=\"stylesheet\" type=\"text/css\"
href=\"{$TemplatePath}css/stamainstyle.css\" />\n</head>";

echo<<<MAIN
<body>
<h1>NODE UTILIZATION STATISTICS INFORMATION</h1>
        <form name="form1" action="cgi-bin/launcher2.cgi">
    <h2>Analising Period of Time</h2>
        <div class="form_center">
            <ul class="time_list">
                <li>
                <input type="radio" name="time" value="1" tabindex="0" checked>Last minut</li>
                <li>
                <input type="radio" name="time" value="2" tabindex="1">Last 5 minuts</li>
                <li>
                <input type="radio" name="time" value="5" tabindex="4">Last 12 hours</li>
                <li>
                <input type="radio" name="time" value="3" tabindex="2">Last 15 minuts</li>
                <li>
                <input type="radio" name="time" value="4" tabindex="3">Last hour</li>
                <li>
                <input type="radio" name="time" value="7" tabindex="6">Last week</li>
            </ul>
        </div>
    <h2>Graphic Agrupations</h2>
        <div class="form_center">
            <ul class="grouped_list">
                <li>
                <input type="radio" name="draw" value="1" tabindex="1" checked>
                Grouped by parameter</li>
                <li>
                <input type="radio" name="draw" value="2" tabindex="2">
                Grouped by node</li>
            </ul>
        </div>
    <h2>Analising Parameters</h2>
        <div class="form_center">
            <ul class="time_list">
                <li><span>CPU</span></li>
                <li><span>MEMORY</span></li>
                <li><span>NET</span></li>
                <li>
                    <select name="cpu" size="5" multiple>
                        <option value="0" selected>Total CPU load
```

```
                        <option value="1">User CPU load
                        <option value="2">Nice CPU load
                        <option value="3">System CPU load
                        <option value="4">Idle CPU
                        </select>
                </li>
                <li>
                    <select name="memory" size="5" multiple>
                        <option value="0" selected>Total Main Mem
                        <option value="1">Used Main Mem
                        <option value="2">Free Main Mem
                        <option value="3">Shared Main Mem
                        <option value="4">Total Swap Mem
                        <option value="5">Used Swap Mem
                        <option value="6">Free Swap Mem
                    </select>
                </li>
                <li>
                    <select name="net" size="5" multiple>
                        <option value="0" selected>Kbytes Send
                        <option value="1">Kbytes Received
                        <option value="2">Errors Send
                        <option value="3">Errors Received
                        <option value="4">Collisions
                    </select>
                </li>
            </ul>
        </div>
        <div class="submit">
                <input type="submit" name="blaunch" value="Submit">
                <input type="reset" name="bclear" value="Clear">
                <input type="hidden" name="hparams" value=" ">
        </div>
        </form>
    </body>
</html>
MAIN;

?>
```

A.2. Configuración y funciones

A.2.1. config.php

```
<?php

// Variables comunes a todo MoniTo
$TemplatePath = 'template/blue/';
$title = 'Monito 4.0 - Web based monitoring tool';

// Header comun a todo MoniTo
```

```
function print_header()
{
echo<<<HEADER
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>{$title}</title>
HEADER;
}

function frameset_header()
{
echo<<<HEADER
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
HEADER;
}

?>
```

A.2.2. javascriptfunctions.php

```
<?php

function javascript_function()
{
echo<<<JAVASCRIPT
    <script language="JavaScript" type="text/javascript"><!--
        function winopen(h, n, p) {
            myWin = open(h, n, p + ", toolbar=no, resizeable=no,
            location=no, menubar=no, scrolling=yes, scrollbars=yes");
        }
// -->
    </script>
JAVASCRIPT;
}

?>
```

A.2.3. launchfunctions.php

```
<?php

function javascript_functions()
{
echo<<<JAVASCRIPT
    <script language="JavaScript" type="text/javascript">
    <!--
```

```
        function MM_openBrWindow(theURL,winName,features) { //v2.0
            window.open(theURL,winName,features);
        }
//-->
    </script>
JAVASCRIPT;
}

function view_offline_file()
{
echo<<<OFFLINE
    <div class="submit">
        <h2>View Off-line file</h2>
                <p>Previous Off-line file</p>
                    <input type="text" name="offfilename" size="31">
                    <input type="submit" name="bview" value=" View ">
    </div>
OFFLINE;
}

?>
```

A.2.4. statusfunctions.php

```
<?php

function legend_table()
{
echo<<<LEGENDTABLE
<table>
<tr>
    <td class="legend">
        <img src="../imag/esyellow.gif" alt="Yellow" /><br/>Available
    </td>
    <td class="legend">
        <img src="../imag/esred.gif" alt="Red" /><br/>Unavailable
    </td>
    <td class="legend">
        <img src="../imag/esgreen.gif" alt="Green" /><br/>Selected
    </td>
</tr>
</table>
LEGENDTABLE;
}

function update_hosts()
{
echo<<<UPDATE
<div id="update">
    <a href="javascript:parent.document.location.reload();" >Update</a>
</div>
UPDATE;
```

```
}

function init_hosts_table($monact)
{
    $i = 0;
    $namemon = array("Load", "Mem", "Com", "Statistic", "Behaviour");

// Imprimir el esqueleto genérico de cada tabla
    echo"<table>\n";
    echo"\t<tr id=\"titles\">\n";
    echo"\t\t<td id=\"row1\" >Nodes</td>\n";
// Selecciona las celdas dependiendo del monitor activo
    switch ( $monact )
    {
        case '123':
            echo"\t\t<td id=\"row12\" >Load</td>\n";
            echo"\t\t<td id=\"row13\" >Mem</td>\n";
            echo"\t\t<td id=\"row14\" >Com</td>\n";
        break;
        case '4':
            echo"\t\t<td id=\"row2\" >Statistic</td>\n";
        break;
        case '5':
            echo"\t\t<td id=\"row2\" >Behaviour</td>\n";
        break;
    }
    echo"\t</tr>\n";
    echo"</table>\n";
    echo"<br/>\n";
}

function javascript_functions()
{
echo<<<JAVASCRIPT
<script language="JavaScript" type="text/javascript">
    var selected = new Array();
    var selcount = 0;
    function CompSwap(host_num, host_name, monitor_num, obj)
    { //Oscar
        var x;
        var isselect = 0;
        var field = window.parent.parent.main.main.document.form1.hparams;
        if ((x=MM_findObj(obj))!=null)
        {
            for(i=0;i<selected.length&&isselect==0;i++)
            {
                if (obj==selected[i]) isselect=i+1;
            }
            if (isselect)
            {
                selected[i-1]='';
                MM_swapImage(obj,'','../imag/esyellow.gif',1)
```

```
            }
            else
            {
                selected[selcount++] = obj;
                MM_swapImage(obj,'','../imag/esgreen.gif',1)
            }
            if (monitor_num != 5)
            {
                field.value = field.value + host_num + '*' + host_name +
                '*' + monitor_num + '-';
            }
            else
            {
                field.value = field.value + host_num + '*' + host_name +
                '*' + '1' + '-'+ host_num + '*' + host_name + '*' + '2' +
                '-'+ host_num + '*' + host_name + '*' + monitor_num + '-';
            }
        }
    }
</script>

<script language="JavaScript" type="text/javascript">
    <!-
    function MM_swapImgRestore() { //v3.0
        var i,x,a=document.MM_sr;
        for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
    }
    function MM_preloadImages() { //v3.0
        var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
        var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
        if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
    }
    function MM_findObj(n, d) { //v4.0
        var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
        d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
        if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
        for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
        if(!x && document.getElementById) x=document.getElementById(n); return x;
    }
    function MM_swapImage() { //v3.0
        var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array;
            for(i=0;i<(a.length-2);i+=3)
        if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x;
            if(!x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
    }
    //-->
</script>
JAVASCRIPT;
}
?>
```

A.3. Hojas de estilo en cascada

Se encuentran dentro del directorio *template/(nombre del template)/css* y representan el diseño de prácticamente todos los aspectos de la unidad de visualización.

A.3.1. downloadstyle.css

```
body
{
        background: #506096;
        padding-top: 10px;
        padding-left: 1%;
        padding-right: 1%;
        padding-bottom: 20px;
        margin: 0;
}

h1
{
        background-color: #5090A6;
        border: 1px solid #000;
        font-family: sans-serif;
        font-size: 18px;
        margin-left: 5px;
        padding-top: 5px;
        padding-bottom: 5px;
        text-align: center;
}

h2
{
        border-bottom: 1px solid black;
        font-family: sans-serif;
        font-size: 16px;
        margin-left: 2%;
        margin-right: 25%;
        text-align: left;
}

h3
{
        font-family: sans-serif;
        font-size: 14px;
        margin-left: 2%;
        margin-right: 25%;
        text-align: left;
}

#block1
{
        width: 94%;
        margin-left: 3%;
```

```
        margin-right: 3%;
        background-color: #5090A6;
        border: 1px solid #000;
}

#block1 p
{
        font-family: sans-serif;
        font-size: 12px;
        margin-left: 4%;
        margin-right: 4%;
        text-align: left;
}

#block1 ol
{
        font-family: sans-serif;
        font-size: 12px;
        margin-left: 5%;
        margin-right: 5%;
        text-align: left;
}

#block1 li
{
        padding-bottom: 2px;
}

a
{
        color: black;
        font-family: sans-serif;
        font-size: 12px;
        padding: 0 10px 0 10px;
}

a:hover
{
        color: black;
        font-family: sans-serif;
        font-size: 12px;
        font-weight: bold;
        padding: 0 10px 0 10px;
}

#block1 .term
{
        font-family: serif;
        font-size: 12px;
        font-weight: bold;
}
```

A.3.2. helpstyle.css

```
body
{
        background-color: #506096;
        padding-top: 10px;
        padding-left: 1%;
        padding-right: 1%;
        padding-bottom: 20px;
        margin: 0;
}

strong
{
        font-weight: bold;
}

em
{
        font-weight: italic;
}

h1
{
        background-color: #5090A6;
        border: 1px solid #000;
        font-family: sans-serif;
        font-size: 18px;
        margin-left: 5px;
        padding-top: 5px;
        padding-bottom: 5px;
        text-align: center;
}

h2
{
        border-bottom: 1px solid black;
        font-family: sans-serif;
        font-size: 16px;
        margin-left: 2%;
        margin-right: 25%;
        text-align: left;
}

h3
{
        font-family: sans-serif;
        font-size: 14px;
        margin-left: 2%;
        margin-right: 25%;
        text-align: left;
}
```

```
#block1
{
        background-color: #5090A6;
        border: 1px solid #000;
        margin-left: 3%;
        margin-right: 3%;
        padding-top: 20px;
        padding-bottom: 20px;
        width: 94%;
        clear: both;
        overflow:hidden;
}

#block1 a
{
        color: black;
        font-family: sans-serif;
        font-size: 16px;
        font-weight: bold;
        text-decoration: none;
}

#block1 span a
{
        font-family: sans-serif;
        font-size: 12px;
        text-decoration: none;
}


#block1 img
{
        border: 0;
        margin-right: 20px;
}

#block1 table
{
        color: black;
        font-family: sans-serif;
        font-size: 12px;
}

#block1 td
{
        border-bottom: 1px solid black;
        padding-top: 3px;
        padding-bottom: 3px;
}

.items li
```

```
{
        background-color: #5090A6;
        border-color: #5090A6;
        border-width: 1px;
        border-style: solid;
        color: black;
        font-family: sans-serif;
        list-style: none;
        margin-left: 25%;
        margin-right: 25%;
        padding-top: 10px;
        padding-bottom: 10px;
        text-align: center;
}

.items li:hover
{
        background-color: #6090A6;
        border-color: #000000;
        border-width: 1px;
        border-style: solid;
        color: black;
        font-family: sans-serif;
        list-style: none;
        padding-top: 10px;
        padding-bottom: 10px;
        margin-left: 25%;
        margin-right: 25%;
        text-align: center;
}

#block2
{
        background-color: #5090A6;
        border: 1px solid #000;
        margin-left: 3%;
        margin-right: 3%;
        margin-bottom: 15px;
        padding-top: 5px;
        padding-bottom: 20px;
        width: 94%;
        clear: both;
}

#block2 span
{
        margin: 0 10px;
        padding: 0;
        float: left;
}

#block2 span a
```

```
{
        display: block;
        width: 100%;
        padding: 0 4px;
        text-decoration: none;
        text-align: center;
        font-size: 12px;
        color: #FFFFFF;
        background-color: #5090A6;
        border-width: 1px;
        border-style: none;
        color: black;
        font-family: sans-serif;
        font-weight: bold;
}

#block2 span a:hover
{
        color: #CCCCCC;
        background-color: #506096;
}


.column1
{
        width: 34%;
        margin-right: 1%;
        margin-left: 1%;
        float: left;
        clear: both;
}

.column1 p
{
        color: black;
        font-family: sans-serif;
        font-size: 12px;
        font-weight: bold;
        text-align: center;
}

.column2
{
        width: 60%;
        margin-right: 2%;
        float: left;
        clear: none;
}

.column2 p
{
        color: black;
```

```
        font-family: sans-serif;
        font-size: 12px;
}

.imgcontainer
{
        margin-left: 15%;
        margin-right: 15%;
        text-align: center;
}

.imgcontainer img
{
        width: 80%;
        padding-bottom: 5px;
}

.imgcontainer span
{
        color: black;
        font-family: sans-serif;
        font-size: 12px;
        font-weight: bold;
        text-align: center;
}

.block12
{
        background-color: #5090A6;
        margin-left: 3%;
        margin-right: 3%;
        width: 94%;
         clear: both;
        overflow:hidden;
}

.block12 p
{
        color: black;
        font-family: sans-serif;
        font-size: 12px;
        text-align: left;
}

.block12 ul
{
        text-align: left;
}

.block12 li
{
        color: black;
```

```
        font-family: sans-serif;
        font-size: 12px;
        list-style: none;
        padding-top: 10px;
        padding-bottom: 10px;
}

.order li
{
        color: black;
        font-family: sans-serif;
        font-size: 12px;
        list-style-type: upper-roman;
        margin-left: 10%;
        padding-top: 10px;
        padding-bottom: 10px;
}

.preformat
{
        font-family: Monospace;
}
```

A.3.3. indexstyle.css

```
body
{
        background: #506096;
}
```

A.3.4. mainstyle.css

```
html
{
        border: none;
}

body
{
        background: #506096;
        border: none;
}

#logo
{
        margin: 25px 10px 25px 10px;
        text-align: center;
}

img
{
        border: 0;
```

```
}

h1
{
        font-family: sans-serif;
        font-size: 18px;
        padding-top: 10px;
}

h2
{
        font-family: sans-serif;
        font-size: 16px;
}

#trescol
{
        margin: 0 15% 150px 15%;
        padding: 0;
}

#trescol ul
{
        list-style:none;
}

#trescol li
{
        margin:2px;
        padding:2px;
        float:left;
}

#trescol li.col1
{
        padding-right: 50px;
        padding-left: 10px;
}

p
{
        padding-top: 25px;
        font-size: 12px;
        font-family: sans-serif;
        font-weight: bold;
        text-align: center;
}

#trescol li.col2
{
        padding-right: 10px;
        padding-left: 50px;
```

```
}
```

A.3.5. nodemainstyle.css

```
body
{
        background: #506096;
        padding: 0 0 10px 0;
        margin: 0;
}

h1
{
        font-family: sans-serif;
        font-size: 18px;
        text-align: center;
        border: 1px solid #000;
        margin-left: 5px;
        margin-right: 5px;
        background-color: #508096;
}

h2
{
        font-family: sans-serif;
        font-size: 16px;
        text-align: center;
        border-bottom: 1px solid black;
        margin-left: 10px;
        margin-right: 10px;
}

div
{
        clear: both;
}

li span
{
        font-weight: bold;
}

.form_center
{
        background-color: #508096;
        overflow:hidden;
        border: 1px solid black;
        margin-left: 3%;
        margin-right: 3%;
}
```

```
ul.time_list
{
        padding-right: 5%;
        padding-left: 5%;
        float: left;
        padding-bottom: 0px;
        margin: 15px 0px;
        width: 100%;
        padding-top: 0px;
        list-style-type: none;
}

ul.time_list li
{
        font-size: 12px;
        font-family: sans-serif;
        color: #000;
        display: inline;
        padding-left: 12px;
        float: left;
        padding-bottom: 2px;
        width: 30%;
        padding-top: 2px;
}

ul.grouped_list
{
        padding-right: 5%;
        padding-left: 5%;
        float: left;
        padding-bottom: 0px;
        margin: 15px 0px;
        width: 100%;
        padding-top: 0px;
        list-style-type: none;
}

ul.grouped_list li
{
        font-size: 12px;
        font-family: sans-serif;
        color: #000;
        display: inline;
        padding-left: 12px;
        float: left;
        padding-bottom: 2px;
        width: 45%;
        padding-top: 2px;
}
```

```
.submit
{
        width: 100%;
        text-align: center;
        padding-top: 10px;
}

.submit p
{
        font-family: sans-serif;
        font-size: 12px;
        font-weight: bold;
        text-align: center;
}
```

A.3.6. processmainstyle.css

```
body
{
        background: #506096;
        padding: 0 0 10px 0;
        margin: 0;
}

h1
{
        font-family: sans-serif;
        font-size: 18px;
        text-align: center;
        border: 1px solid #000;
        margin-left: 5px;
        margin-right: 5px;
        background-color: #508096;
}

h2
{
        font-family: sans-serif;
        font-size: 16px;
        text-align: center;
        border-bottom: 1px solid black;
        margin-left: 10px;
        margin-right: 10px;
}

div
{
        clear: both;
}
```

```
li span
{
        font-weight: bold;
        padding-right: 10px;
}


.form_center
{
        background-color: #508096;
        overflow:hidden;
        border: 1px solid black;
        margin-left: 3%;
        margin-right: 3%;
}

ul.time_list
{
        padding-right: 5%;
        padding-left: 5%;
        float: left;
        padding-bottom: 0px;
        margin: 15px 0px;
        width: 100%;
        padding-top: 0px;
        list-style-type: none;
}

ul.time_list li
{
        font-size: 12px;
        font-family: sans-serif;
        color: #000;
        display: inline;
        padding-left: 12px;
        float: left;
        padding-bottom: 2px;
        width: 30%;
        padding-top: 2px;
}

ul.grouped_list
{
        padding-right: 5%;
        padding-left: 5%;
        float: left;
        padding-bottom: 0px;
        margin: 15px 0px;
        width: 100%;
        padding-top: 0px;
        list-style-type: none;
}
```

```
ul.grouped_list li
{
        font-size: 12px;
        font-family: sans-serif;
        color: #000;
        display: inline;
        padding-left: 12px;
        float: left;
        padding-bottom: 2px;
        width: 45%;
        padding-top: 2px;
}

.submit
{
        width: 100%;
        text-align: center;
        padding-top: 10px;
}

.submit p
{
        font-family: sans-serif;
        font-size: 12px;
        font-weight: bold;
        text-align: center;
}
```

A.3.7. stamainstyle.css

```
body
{
        background: #506096;
        padding: 0;
        margin: 0;
}

h1
{
        font-family: sans-serif;
        font-size: 18px;
        text-align: center;
        border: 1px solid #000;
        margin-left: 5px;
        margin-right: 5px;
        background-color: #508096;
}

h2
{
        font-family: sans-serif;
        font-size: 16px;
```

```
        text-align: center;
        border-bottom: 1px solid black;
        margin-left: 10px;
        margin-right: 10px;
}

.submit
{
        width: 100%;
        text-align: center;
        padding-top: 10px;
}
```

A.3.8. statusstyle.css

```
body
{
        background: #506096;
        padding: 20px 5px 5px 5px;
        margin: 0;
}

img
{
        border: 0;
}

table
{
        border-collapse: collapse;
        width: 100%;
        padding-bottom: 5px;
}

td
{
         border: 1px solid #000;
        background-color: #508096;
        font-family: sans-serif;
}

tr#titles
{
        font-size: 14px;
        font-weight: bold;
        text-align: center;
}

td#row1
{
        width: 75%;
}
```

```
td#row2
{
        width: 25%;
        text-align: center;
}

td.legend
{
        font-size: 11px;
        text-align: center;
}

#update
{
        padding-top: 15px;
        text-align: center;

}

#update a
{
        background-color: #508096;
        border: 1px solid black;
        color: black;
        padding: 5px;
        font-size: 14px;
        font-family: sans-serif;
        font-weight: bold;
        text-decoration: none;
}

#update a:hover
{
        background-color: #505e95;
        border: 1px solid black;
        color: #cccccc;
        padding: 5px;
        font-size: 14px;
        font-family: sans-serif;
        font-weight: bold;
        text-decoration: none;
}

td#row11
{
        width: 52%;
}

td#row12
{
        width: 16%;
```

```
        text-align: center;
}

td#row13
{
        width: 16%;
        text-align: center;
}

td#row14
{
        width: 16%;
        text-align: center;
}
```

A.3.9. titlestyle.css

```
body
{
        background-color: #446288;
        text-align: center;
}

img
{
        border: 0;
}
```

A.3.10. typestyle.css

```
body
{
        background: #506e96;
        font-family: sans-serif;
        padding: 0;
        margin: 0;
}

.options
{
        border: none;
        font-family: sans-serif;
        font-weight: bold;
        font-size: 14px;
        text-align: center;
}

ul {
        list-style: none;
        margin: 0;
        padding: 0;
}
```

```
.menu4
{
        border-color: #000;
        border-style: solid solid none solid;
        border-width: 1px;
        margin: 10px;
}

.menu4 li
{
        border-color: #000;
        border-style: none none solid none;
        border-width: 1px;
        text-decoration: none;
}

.menu4 li a
{
        font-family: sans-serif;
        font-size: 12px;
        font-weight: bold;
        height: 24px;
        text-decoration: none;
        voice-family: "\"}\"";
        voice-family: inherit;
}

.menu4 li a:link, .menu4 li a:visited
{
        background-color: #505e95;
        color: #CCC;
        display: block;
        padding: 8px 0 0 10px;
}

.menu4 li a:hover
{
        color: #000;
        background:  #506e96 0 -32px;
        padding: 8px 0 0 10px;
}

.date
{
        font-size: 10px;
        text-align: center;
}
```

Apéndice B

Código fuente modificado de MoniTo

En esta sección se incluye el código fuente de MoniTo que ha sufrido modificaciones durante la migración a LibGTop. Dentro de cada fichero solamente se considerarán las funciones que han sufrido algun tipo de cambio.

B.1. Demonios ScD y Estadistico

B.1.1. ScD.c

En el *Slave Collector Daemon* se ha incluído la inicialización del servidor LibGTop.

```
#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <netdb.h>/* -> gethostbyname */
#include <sys/time.h>
#include <sys/socket.h>
#include <glibtop.h>

#include "mon_common.h"

int main(int argc,char *argv[])
{
    struct monitor_desc **monitor;
    struct timeval timeout;
    fd_set fdset;
    int *monitor_type;
    struct servent *sent;
    int nmonitors=0, i;
    pthread_t thread;
    FILE *f;
    char ntpserver[20],command[20];

//      Leer infomacion del servidor ntp y configurar hora

    f=fopen("/etc/monito/ntp.scd", "rb");
    if (f!=NULL)
    {
        fscanf(f, "%s", ntpserver);
```

```
        printf("NTP server %s\n",ntpserver);
        sprintf(command,"ntpdate %s", ntpserver);
        /* Actualiza la hora */
        system(command);
        fclose(f);
    }
    else
    {
        printf("WARNING: Cannot open /etc/monito/ntp.conf, to sync de clock.\n");
    }

    if ((sent = getservbyname("ScD","udp")) == NULL)
    {
        fprintf(stderr,"DMonitor: Not service registered in services file.\n");
        exit(-1);
    }

    DMONPORT = htons(sent->s_port);

    for_each_monitor(monitor)
    {
        (*monitor)->init();
        nmonitors++;
    }
    printf("================================================================\n\n");
    monitor_type = (int *) malloc(sizeof(int)*nmonitors);
    if (monitor_type == NULL)
    {
        fprintf(stderr,"DMonitor: No enougth memory.\n");
        exit(-1);
    }

    FD_ZERO(&fdset);

    glibtop_init();
    printf("*********** LibGTop server initialized ***********\n");

    for(;;)
    {
        i=0;
        FD_ZERO(&fdset);
        for_each_monitor(monitor)
        {
            if (!((*monitor)->petition) && (*monitor)->isinit)
                FD_SET( (*monitor)->sockfd, &fdset );
        }
        timeout.tv_sec=2;timeout.tv_usec=0;
        switch( select(FD_SETSIZE, &fdset, NULL, NULL, &timeout))
        {
            case 1 ... FD_SETSIZE: // sockets
                for_each_monitor(monitor)
                {
```

```
                    if (FD_ISSET((*monitor)->sockfd, &fdset))
                    {
                        (*monitor)->getmsg();
                        if ((*monitor)->petition)
                        {
                            pthread_create(&thread, NULL,
                            thread_service, (void*) (*monitor));
                            printf("thread creado en el ScD...\n");
                        }
                    }
                }
                break;
            case 0: //printf("\n  TIMEOUT !!\n");
                break;
        }
    }
}
```

B.1.2. estadistic.c

```
void pick_data(param* data)
{
    int cuser, cnice, csys, cidle, tused;
    int i;
    float cdiff;
    unsigned long int no_sirve;
    FILE *file;
    long int pos1, pos2, pos3, pos4, pos5;
    char buff[256];
    long int mt,mf,ms,st,sf;
    char INTERFAZ[7];
    int contador;
    long int eth=0;
    long int bytesend,byterecv,errorsend,errorrecv,collisions;

    glibtop_cpu cpu;
    glibtop_mem memory;
    glibtop_swap swap;
    glibtop_netload net;

    glibtop_get_cpu(&cpu);
    cuser = cpu.user;
    cnice = cpu.nice;
    csys = cpu.sys;
    cidle = cpu.idle;

    cdiff = (cuser + cnice + csys + cidle) - cpu_values_load[4];
    cpu_values_load[4] = cuser + cnice + csys + cidle;
    tused = (cuser + cnice + csys) -
    ( cpu_values_load[0] + cpu_values_load[1] +  cpu_values_load[2]);

    (*data).load[0] = (int) (100 * tused / cdiff);
```

```
    (*data).load[1] = (int) (100 * (cuser - cpu_values_load[0]) / cdiff);
    (*data).load[2] = (int) (100 * (cnice - cpu_values_load[1]) / cdiff);
    (*data).load[3] = (int) (100 * (csys - cpu_values_load[2]) / cdiff);
    (*data).load[4] = (int) (100 * (cidle - cpu_values_load[3]) / cdiff);

    cpu_values_load[0] = cuser;
    cpu_values_load[1] = cnice;
    cpu_values_load[2] = csys;
    cpu_values_load[3] = cidle;
    glibtop_get_mem ( &memory );
    mt = memory.total / 1024;
    mf = memory.free / 1024;
    ms = memory.shared / 1024;

    glibtop_get_swap ( &swap );
    st = swap.total  / 1024;
    sf = swap.free  / 1024;

    (*data).mem[0]=mt;
    (*data).mem[1]=mt-mf;
    (*data).mem[2]=mf;
    (*data).mem[3]=ms;
    (*data).mem[4]=st;
    (*data).mem[5]=st-sf;
    (*data).mem[6]=sf;

    file=fopen("/etc/monito/netdev.conf","r");
    if ( file==NULL )
    {
        printf("ERROR. Cannot open /etc/monito/netdev.conf!
        Read the manual and create it!\n");
        exit(-1);
    }
    else
    {
        fscanf(file, "%s", INTERFAZ);
    }

    glibtop_get_netload(&net,INTERFAZ);

    bytesend = net.bytes_out;
    byterecv = net.bytes_in;
    errorsend = net.errors_out;
    errorrecv = net.errors_in;
    collisions = net.collisions;

    (*data).net[0]=bytesend;
    (*data).net[1]=byterecv;
    (*data).net[2]=errorsend;
    (*data).net[3]=errorrecv;
    (*data).net[4]=collisions;
}
```

B.2. Monitores

B.2.1. load_func.c

```
#include "mon_common.h"
#include <glibtop/cpu.h>
#include <glibtop/uptime.h>
#include <glibtop/proctime.h>

int load_scan(char* buf)
{
    int err=0, cuser, cnice, csys, cidle, cio, tused;
    float cdiff,result;
    unsigned long utime,stime,start,uptime;
    glibtop_cpu cpu;
    glibtop_proc_time ProcTime;
    FILE *fpid;
    char buf2[100];
    char *str;

    glibtop_get_cpu(&cpu);

    cuser = cpu.user;
    cnice = cpu.nice;
    csys = cpu.sys;
    cidle = cpu.idle;
    cio = cpu.iowait;
    cdiff = (cuser + cnice + csys + cidle + cio) - cpu_values_load[5];
    cpu_values_load[5] = cuser + cnice + csys + cidle + cio;
    tused = (cuser + cnice + csys) -
    ( cpu_values_load[0] + cpu_values_load[1] +  cpu_values_load[2]);
    sprintf(buf, "%d %d %d %d %d %d",(int) (100 * tused / cdiff),
        (int) (100 * (cuser - cpu_values_load[0]) / cdiff),
        (int) (100 * (cnice - cpu_values_load[1]) / cdiff),
        (int) (100 * (csys - cpu_values_load[2]) / cdiff),
        (int) (100 * (cidle - cpu_values_load[3]) / cdiff),
        (int) (100 * (cio - cpu_values_load[4]) / cdiff));
    cpu_values_load[0] = cuser;
    cpu_values_load[1] = cnice;
    cpu_values_load[2] = csys;
    cpu_values_load[3] = cidle;
    cpu_values_load[4] = cio;

    if (pid_process == -1)
    {
        sprintf(buf2,"ps -A |grep %s >/tmp/pidload", load_msg_req->process);
        system(buf2);
        fpid = fopen("/tmp/pidload","r");
        fread(&str, sizeof(char), 1, fpid);
        if(feof(fpid))
        {
            fclose(fpid);
```

```
            pid_process=-1;
        }
        else
        {
            fseek(fpid, 0, SEEK_SET);
            fscanf(fpid,"%li",&pid_process);
            fclose(fpid);
        }
    }

    glibtop_get_proc_time (&ProcTime, pid_process);

    utime = ProcTime.utime;
    start = ProcTime.rtime;
    stime = ProcTime.stime;

    result=(((utime+stime)-(utimeant+stimeant))*100)/cdiff;
    utimeant=utime;
    stimeant=stime;

    uptime = (double) cpu.total / (double) cpu.frequency;
    // Guarda ultimo timer
    timer=(uptime*100)-start;

    sprintf(buf,"%s %d %d %d nomsg",buf,(int)result, (int)timer, (int)(timer-utime-stime));

    return err;
}

void load_init ()
{
    char cont=TRUE;

    printf("\n\nLOADMON INIT...\n");

    if ( (loadmon.sockfd = connect_sock(DMONPORT+loadmon.type, SOCK_DGRAM))==-1 )
    {
        printf("-Cannot create LOADMON socket! This monitor will be disabled.\n");
        cont = FALSE;
    }
    else {printf("-Socket succesfully created!\n");}

    scan_precision = 50;
    loadmon.isinit = cont;

    if (loadmon.isinit==TRUE) printf("** LOADMON ENABLED **\n");
    else printf("** LOADMON DISABLED **\n");
}
```

B.2.2. mem_func.c

```
int mem_scan2(char* buf, char* process)
{
    int diff = -1;
    long int vsp = -1, rsp = -1, wcf = -1, pif = -1, size = -1, resident = -1;
    unsigned long int mem[7];
    long int mt,mf,ms,st,sf;
    glibtop_proc_kernel procKernel;
    glibtop_proc_mem procMem;
    glibtop_mem memory;
    glibtop_swap swap;

    if(pid==-1)
    {
        sprintf(buf, "-1 -1 -1 -1 -1 -1 -1 -1");
        diff=-1;
    }
    else
    {
        diff=0;
        glibtop_get_proc_kernel(&procKernel,pid);
        wcf = procKernel.min_flt;
        pif = procKernel.maj_flt;

        glibtop_get_proc_mem(&procMem, pid);
        vsp = procMem.vsize;
        rsp = procMem.rss + 3;
        resident = procMem.resident;
        size = procMem.size;
        sprintf(buf, "%li %li %li %li %li %li -1 -1",
        size-resident, vsp, rsp, wcf+pif , wcf, pif);
    }

    glibtop_get_mem ( &memory );
    mt = memory.total / 1024;
    mf = memory.free / 1024;
    ms = memory.shared / 1024;

    glibtop_get_swap ( &swap );
    st = swap.total  / 1024;
    sf = swap.free  / 1024;

    mem[0]=mt;
    mem[1]=mt-mf;
    mem[2]=mf;
    if(pos3==0) mem[3]=0;
    else mem[3]=ms;
    mem[4]=st;
    mem[5]=st-sf;
    mem[6]=sf;
```

```
    sprintf(buf, "%s %li %li %li %li %li %li %li %s",
    buf,mem[0],mem[1],mem[2],mem[3],mem[4],mem[5],mem[6],process);

    return diff;
}

void mem_init ()
{
    char cont=TRUE;

    printf("\n\nMEMMON INIT...\n");

    if ( (memmon.sockfd = connect_sock(DMONPORT+memmon.type, SOCK_DGRAM)) == -1 )
    {
        printf("-Cannot create MEMMON socket! This monitor will be disabled.\n");
        cont = FALSE;
    }
    else {printf("-Socket succesfully created!\n");}

    scan_precision = 50;
    memmon.isinit = cont;

    if (memmon.isinit==TRUE) printf("** MEMMON ENABLED **\n");
    else printf("** MEMMON DISABLED **\n");
}
```

B.2.3. net_func.c

```
int net_scan(char* buf)
{
#ifdef PVM_GETPVMDSTATS
    struct pvmdstats spvmdstats;
    unsigned long pvmd_saddr, saddr;
    unsigned int pvmd_sport, sport;
    FILE *fdstadsoc;
    char buff2[256];
#endif

    glibtop_netload net;
    FILE *fdev;
    int tx,rx;
    char nor[7];
    unsigned int bytesend, byterecv, errorsend;
    unsigned int errorrecv, collisions;

    if(pid==-1)
    {
        sprintf(buf,"-1 -1 -1 -1");
    }
    else
    {
        sprintf(buf,"netstat -tpna | grep %s > /tmp/netstat",
```

```
        net_msg_req->process);
        system(buf);

        fdev = fopen("/tmp/netstat","r");
        fread(&str, sizeof(char), 1, fdev);
        if(feof(fdev))
        {
            fclose(fdev);
            sprintf(buf,"-1 -1 -1 -1");
        }
        else
        {
            fseek(fdev, 0, SEEK_SET);
            fscanf(fdev, "%s %d %d\n",nor,&rx,&tx);
            sprintf(buf,"%d %d -3 -4",rx,tx);
            fclose(fdev);
        }
    }

    glibtop_get_netload(&net,INTERFAZ);
    bytesend = net.bytes_out;
    byterecv = net.bytes_in;
    errorsend = net.errors_out;
    errorrecv = net.errors_in;
    collisions = net.collisions;

    sprintf(buf, "%s %u %u %u %u %u nomsg",
    buf, bytesend / 1024, byterecv / 1024, errorsend, errorrecv, collisions);

    return 0;
}

void net_init ()
{
    char cont=TRUE;
    FILE *f;

    printf("\n\nNETMON INIT...\n");

    f=fopen("/etc/monito/netdev.conf","r");
    if ( f==NULL )
    {
        printf("-Cannot open /etc/monito/netdev.conf!
        Read the manual and create it!\n");
        cont = FALSE;
    }
    else
    {
        fscanf(f, "%s", INTERFAZ);
        printf("-Cluster net interface: %s\n",INTERFAZ);
    }
```

```
    if ( (netmon.sockfd = connect_sock(DMONPORT+netmon.type, SOCK_DGRAM)) == -1 )
    {
        printf("-Cannot create NETMON socket! This monitor will be disabled.\n");
        cont = FALSE;
    }
    else {printf("-Socket succesfully created!\n");}

    scan_precision = 50;
    netmon.isinit = cont;

    if (netmon.isinit==TRUE) printf("** NETMON ENABLED **\n");
    else printf("** NETMON DISABLED **\n");
}
```

B.2.4. class_func.c

```
int class_scan(char* buf)
{
    unsigned int bytesend, byterecv, errorsend, errorrecv, collisions, tot_bytes = 0;
    struct timeval tr;
    float ftr;
    int val = 0;
    glibtop_netload net;

    glibtop_get_netload(&net,INTERFAZ);
    bytesend = net.bytes_out;
    byterecv = net.bytes_in;
    errorsend = net.errors_out;
    errorrecv = net.errors_in;
    collisions = net.collisions;

    diff_time(next_scan_class, time_inici, &tr);
    ftr = tr.tv_sec + (tr.tv_usec * 0.000001);
    printf("================ ftr: %f\n", ftr);
    while(temps <= ftr && val != EOF)
    {
        tot_bytes = tot_bytes + bytes;
        printf("================ tot_bytes: %d - temps: %f - bytes: %d \n",
        tot_bytes, temps, bytes);
        val = fscanf(ftub, "      %f %d\n", &temps, &bytes);
    }

    if (val == EOF) {bytes=0; temps=0;}

    if(classmon.last_nscan == classmon.nscan) fclose(ftub);

    sprintf(buf, "-1 -1 -1 -1 %u %d nomsg", (bytesend + byterecv- bytesendant), tot_bytes);

    bytesendant = bytesend + byterecv;
    return(0);
}
```

```
void class_init ()
{
    char cont=TRUE;
    FILE *f;

    printf("\n\nCLASSMON INIT...\n");

    f=fopen("/etc/monito/netdev.conf","r");
    if ( f==NULL )
    {
        printf("-Cannot open /etc/monito/netdev.conf!
        Read the manual and create it!\n");
        cont = FALSE;
    }
    else
    {
        fscanf(f, "%s", INTERFAZ);
        printf("-Cluster net interface: %s\n",INTERFAZ);
    }

    if ( (classmon.sockfd = connect_sock(DMONPORT+classmon.type, SOCK_DGRAM))==-1 )
    {
        printf("-Cannot create CLASSMON socket! This monitor will be disabled.\n");
        cont = FALSE;
    }
    else {printf("-Socket succesfully created!\n");}

    scan_precision = 50;
    classmon.isinit = cont;

    if (classmon.isinit==TRUE) printf("** CLASSMON ENABLED **\n");
    else printf("** CLASSMON DISABLED **\n");
}

void class_getmsg()
{
    int nbytes, ssock_cli;
    char buf[100];
    unsigned int byterecv, errorsend, errorrecv, collisions;
    glibtop_netload net;

    ssock_cli = sizeof (struct sockaddr);
    nbytes = recvfrom( classmon.sockfd, &dmon_msg_req_class,
    sizeof(dmon_msg_req_class), 0, (struct sockaddr*) &class_addr_cli, &ssock_cli);

    if (nbytes < 0)
    {
        printf("classMon: GetMsg: Error al recv.\n");
        perror("classMon");
        clear_monitor(&classmon);
        return;
    }
```

```
class_msg_req = (struct mem_msg_request *) &dmon_msg_req_class;

printf("classMON->cli sec: %li \n", class_msg_req->client_time.tv_sec);
printf("classMON->cli usec: %li \n", class_msg_req->client_time.tv_usec);
printf("classMON->cli int: %li \n", class_msg_req->interval);
printf("classMON->cli left: %li \n", class_msg_req->left_time);
printf("classMON->cli proc: %s \n", class_msg_req->process);
printf("classMON->cli proc: %s \n", class_msg_req->offline_filename);
printf("classMON->cli port: %i \n", ntohs(class_addr_cli.sin_port));

inc_time(class_msg_req->client_time, ++(classmon.nscan)
* class_msg_req->interval, &next_scan_class);

classmon.calc_next_timeout();
printf("classMON->cli sec: %li \n", classmon.next_timeout.tv_sec);
printf("classMON->cli usec: %li \n", classmon.next_timeout.tv_usec);

classmon.petition = TRUE;
classmon.last_nscan = (class_msg_req->left_time * 1000) / class_msg_req->interval;

if (class_msg_req->method == BUFFERING)
{
    classmon.nbuf = class_msg_req->interval < RATE_BUFFERING ?
    (int) RATE_BUFFERING / class_msg_req->interval : 0;
    classmon.nbuf = classmon.nbuf == 1 ? 2 : classmon.nbuf;
    classmon.buffmsgreplys = (struct msg_reply *) malloc(sizeof(struct msg_reply)
    * (classmon.nbuf + 1));
    if (classmon.buffmsgreplys == NULL)
    {
        printf("ClassMon: Not enought memory. Buffering deactivated.\n");
        class_msg_req->method = DIRECT;
        classmon.nbuf = 0;
    }
    else
    {
        classmon.buffmsgreplys[classmon.nbuf].nscan = 0;
    }
}

glibtop_get_netload(&net,INTERFAZ);
byterecv = net.bytes_in;
errorsend = net.errors_out;
errorrecv = net.errors_in;
collisions = net.collisions;

// Inicialización del scanner de puertos
mkfifo(pathname, mode);
sprintf(buf, "ports %s %s &", class_msg_req->process, INTERFAZ);
system(buf);
ftub = fdopen(open(pathname, O_RDONLY|O_NONBLOCK), "r");
temps = 0; bytes = 0;
```

```
    gettimeofday(&time_inici, NULL);
}
```

B.2.5. ports.c

```
main(int argc, char *argv[])
{
    long int pid_process, pid_prc, pid_actual;
    FILE *fport, *fpid, *fsnif, *fnumport;
    char buf[500], buf2[100], buf3[100];
    char *str0, *str1, *str2, *str3, *str4, *str5, *str6, *str7, *str8;
    int ip1, ip2, ip3, ip4;
    int port, totalports, bytes, tot_bytes;
    int i=0, numport[10];
    float temps;

    // control de los parametros
    if (argc != 3)
    {
    printf ("Parameters ERROR\n\t You should call: \n");
    printf (" ./ports <process> <interface> \n");
    printf (" <process> Process name to monitorize \n");
    printf (" <interficie> Interface to monitorize \n");
    exit(1);
    }

    // Creación de la lista de puertos que utiliza el proceso
    sprintf(buf2,"netstat --numeric-ports -ptuw | grep %s > /tmp/ports", argv[1]);
    system(buf2);

    // se crea el comando tshark para escanear todos los puertos del proceso
    sprintf(buf,"tshark -l -p -i %s", argv[2]); // argv[2] es la interface del cluster

    // Obtener el numero de puertos distintos que utiliza un proceso
    sprintf(buf3,"wc -l /tmp/ports > /tmp/totalports");
    system(buf3);
    fnumport = fopen("/tmp/totalports","r");
    fscanf(fnumport,"%i \n",&totalports);
    fclose(fnumport);

    fport = fopen("/tmp/ports","r");
    switch ( totalports )
    {
    case 0: // Engaña al tshark en procesos que no tienen conexiones activas...
        fscanf(fport, "%s %s %s %d.%d.%d.%d:%d %s %s \n",
        &str1, &str2, &str3, &ip1, &ip2, &ip3, &ip4, &port, &str4, &str5);
        sprintf(buf, "%s -f \"src port %d\"", buf, port);
        break;
    case 1:
        fscanf(fport, "%s %s %s %d.%d.%d.%d:%d %s %s \n",
        &str1, &str2, &str3, &ip1, &ip2, &ip3, &ip4, &port, &str4, &str5);
```

```
    sprintf(buf, "%s -f \"src port %d\"", buf, port);
    break;
default:
    while (!feof(fport))
    {
    fscanf(fport, "%s %s %s %d.%d.%d.%d:%d %s %s \n",
    &str1, &str2, &str3, &ip1, &ip2, &ip3, &ip4, &port, &str4, &str5);
    numport[i] = port;
    i++;
    }
    i = 0;
    sprintf(buf,"%s -f  \" ",buf);
    while ( i < totalports )
    {
    sprintf(buf,"%s src port %d ",buf,numport[i]);
    i++;
    if ( i == totalports ) sprintf(buf,"%s \"",buf);
    else sprintf(buf,"%s or ",buf);
    }
break;
}

fclose(fport);
sprintf(buf,"%s -o column.format:\"temps\",%%t,\"len\",%%L -T text >> /tmp/class_fifo", buf);

// pid del proceso a monitorizar
pid_process = ident_proces(argv[1]);
if (pid_process == -1)
{
exit(0);
}
fpid = fopen("/tmp/ident_prc","r");
fseek(fpid, 0, SEEK_SET);
fscanf(fpid,"%li", &pid_process);

// si existe se acaba la ejecucion del anterior dnotify
sprintf(buf2, "killall -9 dnotify");
system(buf2);

// se crea un comando dnotify para que avise
// cuando se modifique algun puerto del proceso
while (!feof(fpid))
{
sprintf(buf2,"\n\ndnotify -a --times 1 /proc/%d/fd/ -e ports %s %s &",
pid_process, argv[1], argv[2]);
printf("%s\n", buf2);
system(buf2);
fscanf(fpid,"%li", &pid_process);
}
fclose(fpid);

// Si existe, se acaba la ejecución del antiguo tshark
```

```
    pid_prc = ident_proces("tshark");
    while (pid_prc != -1)
    {
    sprintf(buf2, "kill -2 %d", pid_prc);
    system(buf2);
    pid_prc = ident_proces("tshark");
    }

    // se ejecuta un nuevo tshark
    printf("\n\n%s\n", buf);
    system(buf);
    exit(0);
}
```

B.3. CGI's

B.3.1. html_common.c

De la anterior versión se han eliminado múltiples funciones que ahora se generan dinámicamente mediante html embebido en sentencias php.

```
void host_show(char *host_name, int host_num, char active, char *monitors_status, char *monact)
{
    char imghost_status[]= "cmplin1.gif";
    int monitor_num = 0, numchars, i, monshow=1, j=0;
    char buff[6]="", buf[30]="", buf2[2]="";
    char comment[] = "ScD not actived in this node.";

    printf("  <tr>\n");
    printf("    <td id=\"row11\">%s</td>\n", host_name);

    sprintf(buf2,"%c\0",monact[0]);
    for (i=0; i < atoi(buf2); i++)
    {
    sprintf(buf,"%s%d#",buf,i);
    }
    monitors_status += strspn(monitors_status, buf);


    while ( monitors_status[0] != '\0' && active &&
    (numchars = strspn(monitors_status, monact)) && j<strlen(monact))
    {   // numero de monitors que quieres aceptar los tres primeros
    sprintf(buf2,"%c\0",monact[j++]);
    monshow = atoi(buf2);
    strncpy(buff, monitors_status, numchars);
    monitors_status += numchars;
    monitor_num = atoi(buff);
    for (i = monshow; i < monitor_num; i++)
    { // Monitores no activos
        printf("    <td id=\"row12\">\n");
        printf("      <img src=\"../imag/esred.gif\" / alt=\"Red\" />\n");
        printf("    </td>\n");
```

```
        monshow++; j++;
    }

    printf("    <td id=\"row12\">\n");
    printf("<a href=\"javascript:;\" onClick=\"CompSwap('%i','%s', '%i', 'imgc%im%i')\">
    <img src=\"../imag/esyellow.gif\" name=\"imgc%im%i\" alt=\"Yellow\" /></a>\n",
     host_num, host_name, monitor_num, host_num, monitor_num, host_num, monitor_num);
    printf("    </td>\n");
    monshow++;

    monitors_status += strspn (monitors_status, "#");
    }
    sprintf(buf2,"%c\0",monact[strlen(monact)-1]);

    for (i = monitor_num + 1; i <= atoi(buf2); i++)
    {
    printf("    <td id=\"row12\">\n");
    printf("      <img src=\"../imag/esred.gif\" alt=\"Red\" />\n");
    printf("    </td>\n");
    }
    printf("  </tr>");
}

void xhtml_doctype()
{
    printf("<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
    Transitional//EN\"\n\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n
    <html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n");
}

html_msg(char *m)
{
    printf("<script language=\"JavaScript\">\n");
    printf( "alert('%s');\n", m);
    printf("</script>\n");
}
```

B.3.2. launcher_proc.c

```
void launch_applet(/*char *q*/)
{
    register int m;
    register int h;
    char hs[BUFFSIZE*3]="";

    for (m=1; m <= maxmonid; m++)
    {
    for (h=0; h <= maxhostid; h++)
    {
        if (hostsmonitors[h][m])
        sprintf(hs,"%s%i*%s-", hs, h, hostsnames[h]);
    }
```

```
    if (strlen(hs) != 0)
    {
        hs[strlen(hs)] = '\0';
        html_launcher_file_write(m, hs);
        printf("<script language=\"JavaScript\">\n");
        printf( "window.parent.parent.title.winopen
        ('../launch/launch%i.htm', '_blank', 'width=%i,height=%i');\n",
        applets_launched - 1, appw + 20, apph + 20);
        printf("</script>\n");
    }
    hs[0]='\0';
    }
}

void html_params_page ()
{
    printf("<body bgcolor=\"#506096\"></body>");
}
```

B.3.3. launcher_esta.c

```
int main(void)
{
char query[4096]="";
FILE *file;

int sockfd, nbytes, ssock_mcd;
struct hostent *ent;
char *host_mcd, mcd_host[MAXHOSTNAMEELEN];
struct sockaddr_in addr_mcd;
struct mcd_msg_request msg_req = {0, 0, 0, ""}; //type,interval,lef_time,msg
char smsg_req[sizeof(msg_req)+30];
struct hoststatus msgreply_status[MAXHOSTS+1] = { { -1, "", "" } };

unsigned char smsgreply[sizeof(struct msg_reply)+20];
int i, j, petitions=0, error=0;
printf("Content-type:  text/html\n\n");
xhtml_doctype();
printf("<head>\n");
printf("<meta content=\"0;url='../sta_main.php'\" http-equiv=\"refresh\">");

strcpy(query, getenv("QUERY_STRING"));
parse_parameters(query);

if ( button==BLAUNCH && hp && hayparams)
{
    for(i=0;i<MAXHOSTS;i++)
    {
        if (hostsmonitors[i][4]==1)
        {
            sprintf(msg_req.msg, "%s%i-",msg_req.msg,i);
```

```
            petitions++;
        }
        param_val[i][0]=-1;
    }
sprintf(msg_req.msg, "%s#END MSG#",msg_req.msg);
sockfd = socket(AF_INET, SOCK_STREAM, 0);
strcpy(mcd_host,MCDHOST);
ent = gethostbyname(mcd_host);

if ( !ent )   fprintf(stderr,"Error! Not found: %s \n", host_mcd);

memset(&addr_mcd,0,sizeof (struct sockaddr_in));
addr_mcd.sin_family=AF_INET;
addr_mcd.sin_addr.s_addr=(((struct in_addr *)ent->h_addr)->s_addr);
addr_mcd.sin_port=htons(MCDPORT);
connect( sockfd, (struct sockaddr *) &addr_mcd, sizeof(struct sockaddr_in) );

sprintf(smsg_req, "4 0 %li 0 %s",aparam.time,msg_req.msg);
nbytes = send(sockfd, smsg_req, strlen(smsg_req), 0);

if ( nbytes < 0 )   fprintf(stderr,"Error on send to laucher_esta\n");

memset(smsgreply, '\0', strlen(smsgreply));

//recibir el resultado de la peticion
for(i=0;i<petitions;i++)
{
    nbytes = recvfrom(sockfd, &smsgreply, sizeof(smsgreply),
    0,(struct sockaddr*) &addr_mcd, &ssock_mcd);

    if (nbytes<0)    fprintf(stderr,"Error on recv\n");
    else
    {
    parse_sms(smsgreply);
    for(j=0;j<AVG_maxload;j++)
    {
        param_val[host][j] = result[host].load[j];
        if( result[host].load[j] < 0 || result[host].load[j] > 100 ) error = 1;
    }
    for(j=0;j<AVG_maxmem;j++)
    {
        switch(j)
        {
        case 0: //total main mem
        case 4: //total swap mem
            param_val[host][AVG_maxload + j] =
             (float)result[host].mem[j] / 1024; //mem in kbytes
            break;
        default:
            if(j<4)  param_val[host][AVG_maxload + j] =
            ( (float)result[host].mem[j] / (float)result[host].mem[0]) * 10000;
            else   param_val[host][AVG_maxload + j] =
```

```
            ( (float)result[host].mem[j] / (float)result[host].mem[4]) * 10000;
            break;
        }
        if( param_val[host][AVG_maxload + j] < 0) error = 1;
    }
    for(j=0;j<AVG_maxnet;j++)
    {
        if( param_val[host][AVG_maxload + AVG_maxmem + j] < 0 ) error = 1;
        param_val[host][AVG_maxload + AVG_maxmem + j] =  result[host].net[j];
    }
    }
    memset(smsgreply, '\0', strlen(smsgreply));
}

sleep(1);  // per donarli temps...

if(!error)
{
    switch ( aparam.draw )
    {
    case 1:  //Grouped by parameter
    for(i=0;i<MAXESTAPARAMS;i++)
    {
        if(aparam.parameterid[i] > 0)
        {
        html_graphics_initpage("",parameters_names[i],i);
        html_graphics_params(&param_val[0][0],i);
        html_graphics_closepage();
        printf("<script language=\"JavaScript\" text=\"text/javascript\" >\n");
        printf( "window.parent.parent.title.winopen('../launch/html_graphic%i.htm',
        '_blank', 'width=%i,height=%i');  \n",i,size_x,size_y);
        printf("</script>");
        }
    }
    break;
    case 2:  //Grouped by node
    for(i=0;i<MAXHOSTS;i++)
    {
        if (param_val[i][0] != -1)
        {
        html_graphics_initpage("Statistic information of",hostsnames[i],i);
        html_graphics_nodes(MAXESTAPARAMS,param_val[i]);
        html_graphics_closepage();
        printf("<script language=\"JavaScript\" text=\"text/javascript\" >\n");
        printf( "window.parent.parent.title.winopen('../launch/html_graphic%i.htm',
        '_blank', 'width=%i,height=%i');  \n",i,size_x,size_y);
        printf("</script>");
        }
    }
    break;
    }
}
```

```
else
    html_msg("Transmision ERROR. It is possible that the statistic modul
    is not being executed so much time or there is a possible saturation
    of net or daemons, please wait a moment and retry!");
    close(sockfd);
}
else
{
    if(!hp)
    html_msg("Bad parameters. Please SELECT a host");
    else
    if(hayparams==0)
        html_msg("Bad parameters. Please SELECT a parameter");
    else
        html_msg("Bad parameters");
}

html_params_page();

return 0;
}

void html_graphics_initpage(char* s,char* name, int petitions)
{
    char str[100];

    sprintf(str,"../launch/html_graphic%i.htm",petitions);
    html=fopen(str,"w");

    fprintf(html,"<!DOCTYPE html PUBLIC \"-\"//W3C//DTD XHTML 1.0 Transitional//EN\"\n
    \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n
    <html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n
    <meta http-equiv=\"content-type\" content=\"text/html; charset=utf-8\" />\n
    <head>\n
    <link rel=\"stylesheet\" type=\"text/css\"
    href=\"../template/launch/css/graphicstyle.css\" /></head>\n
    <body>\n
    <div id=\"title\">%s %s</div>\n
    <div id=\"graphic\">",s,name);
}

// Página de graficos en agrupación por nodos
void html_graphics_nodes(int num, int* values)
{
    int i, j, count = 0, count_old = 0, linedrawed[3] = {0,0,0};
    char printed[num]; //Guarda si se ha imprimido el tipo de tabla

    for (i=0;i<num;i++)
    {
    printed[i] = FALSE;
    if(aparam.parameterid[i] > 0)
    {
```

```
if ( i < AVG_maxload && linedrawed[0]==0 ) // parametros del monitor de carga de cpu
{
    linedrawed[0]=1;
    strcpy(unit1,"% load");
}
if ( (i < AVG_maxmem + AVG_maxload && i >= AVG_maxload) && linedrawed[1]==0 )
{
    count_old = count;
    count = 0;
    linedrawed[1]=1;
    strcpy(unit1,"% Mbytes");
    if(linedrawed[0]==1)
    {
    strcpy(unit1,"% load");
    strcpy(unit2,"% Mbytes");
    }
}
if ( (i >= AVG_maxmem + AVG_maxload ) && linedrawed[2]==0)
{
    if(count_old < count)    count_old = count;
    count = 0;
    linedrawed[2]=1;
    strcpy(unit1,"Quantity");
    if(linedrawed[0]==1)
    {
    strcpy(unit1,"% load");
    strcpy(unit2,"Quantity");
    }
    if(linedrawed[1]==1)
    {
    strcpy(unit1,"% Mbytes");
    strcpy(unit2,"Quantity");
    }
}
fprintf(html,"<table class=\"fondo\">\n");
switch (i)
{
case 0+AVG_maxload:
case 4+AVG_maxload:
    fprintf(html,"\n<tr>\n<td class=\"porcentaje\">%iMb</td></tr>
    <tr><td height=\"100\" class=\"barra\">TOTAL</td></tr>
    \n</table>\n",values[i]);
    printed[i] = TRUE;
    break;
default:
    if(i<AVG_maxload)
    {
    fprintf(html,"<tr><td class=\"porcentaje\">%i%</td></tr>
    <tr><td height=\"%i\" class=\"barra\"></td>\n</table>"
    ,values[i],values[i]);
    printed[i] = TRUE;
    }
```

```
        if(i>=AVG_maxload && i<AVG_maxload+AVG_maxmem)
        {
        if(i<AVG_maxload + 4)
        {
            fprintf(html,"<tr><td class=\"porcentaje\">%.0fMb</td></tr>
            <tr><td height=\"%i\" class=\"barra\">%.2f%\n\n</td>"
            ,/*(float)values[i]/100*/ (float)
             ((values[i]/100)*(values[AVG_maxload]))/100,
            (int)values[i]/100,(float)values[i]/100);
            printed[i] = TRUE;
        }
        else
        {
            fprintf(html,"<tr><td class=\"porcentaje\">%.0fMb</td></tr>
            <tr><td height=\"%i\" class=\"barra\">%.2f%</td></tr>"
            ,/*(float)values[i]/100*/ (float)
             ((values[i]/100)*(values[AVG_maxload+4]))/100,
            (int)values[i]/100,(float)values[i]/100);
            printed[i] = TRUE;
        }
        }
        if(i>=AVG_maxload+AVG_maxmem)
        {
        fprintf(html,"<tr><td class=\"porcentaje\">%i</td></tr><tr>
        <td height=\"%i\" class=\"barra\"></td></tr>\n",
        values[i],values[i]==0?0:100);
        printed[i] = TRUE;
        }
    break;
    }
    count ++;
    }
    if ( printed[i] == TRUE )
    {
    fprintf(html,"</table>\n");
    fprintf(html,"<p>%s</p>",parameters_names[i]);
    }
}
fprintf(html,"</div>");
if(linedrawed[0]==1 && linedrawed[1]==1 && linedrawed[2]==1)
{
    strcpy(unit1,"% load");
    strcpy(unit2,"% Mbytes");
    strcpy(unit3,"Quantity");
}

size_x = (count_old > count) ? ((count_old*65)+110):((count*65)+110);
//numero de columnas * tamaño(65) + 20 pixels linea negra + 100 para la ventana
size_y = (linedrawed[0] * 210)+(linedrawed[1] * 210)+(linedrawed[2] * 210) + 65;
}
```

```
// Dibuja las columnas para una agrupación por columnas
void html_graphics_params(int* val, int param)
{
int i, j, count = 0;

fprintf(html,"<table class=\"fondo\">\n");

/******* Columnas **************/
for (i=0;i<MAXHOSTS;i++)
{
    if ( val[i*MAXESTAPARAMS] != -1)
    {
    /********************** LOAD ******************/
    if ( param < AVG_maxload )
    {
        fprintf(html,"<tr><td class=\"porcentaje\">%i%</td></tr>
        <tr><td height=\"%i\" class=\"barra\" ></td></tr>\n",
        val[(i*MAXESTAPARAMS)+param],val[(i*MAXESTAPARAMS)+param]);
        strcpy(unit1,"% load");
    }
    /************************ MEMORY ***************************************************/
    if ( param >= AVG_maxload && param < AVG_maxload+AVG_maxmem)
    {
        switch (param)
        {
        case 0+AVG_maxload:  // caso especial para el total de memoria
        case 4+AVG_maxload:
            fprintf(html,"<tr>\n<td class=\"porcentaje\">%iMb</td></tr>
            <tr><td height=\"100\" class=\"barra\">TOTAL</td></tr>\n",
            val[(i*MAXESTAPARAMS)+param]/*/1024*/);
            break;
        default:
            if(param < AVG_maxload + 4)
            fprintf(html,"<tr>\n<td class=\"porcentaje\">\n%.0fMb</td></tr>
            <tr><td height=\"%i\" class=\"barra\">%.2f%</td></tr>\n",
            (float) ( (val[(i*MAXESTAPARAMS)+param]/100)*
            (val[(i*MAXESTAPARAMS)+AVG_maxload]/*/1024*/) )/100,
            (int)val[(i*MAXESTAPARAMS)+param]/100,
            (float)val[(i*MAXESTAPARAMS)+param]/100);
            else
            fprintf(html,"<tr>\n<td class=\"porcentaje\">%.0fMb</td></tr>
            <tr><td height=\"%i\" class=\"barra\">%.2f%</td></tr>\n",
            (float) ( (val[(i*MAXESTAPARAMS)+param]/100)*
            (val[(i*MAXESTAPARAMS)+AVG_maxload+4]))/100,
            (int)val[(i*MAXESTAPARAMS)+param]/100,
            (float)val[(i*MAXESTAPARAMS)+param]/100);
            break;
        }
    strcpy(unit1,"% Mbytes");
    }
```

```
/*********************** NET ****************************/
    if ( param >= AVG_maxload+AVG_maxmem)
    {
        fprintf(html,"<tr><td class=\"porcentaje\">%i</td></tr>
        <tr><td height=\"%i\" class=\"barra\"></td></tr>\n",
        val[(i*MAXESTAPARAMS)+param]/1024,
        val[(i*MAXESTAPARAMS)+param]==0?0:100);
        // imprime en kbytes y cada kbyte tiene 20 de altura
        strcpy(unit1,"Quantity");
    }
    count++;
    }
}
//     Final de la tabla de la gráfica
fprintf(html,"</table>\n");

/****************** NOMBRE DE LOS HOSTS *********************************/
for (i=0;i<MAXHOSTS;i++)
    if ( val[i*MAXESTAPARAMS] != -1)
    fprintf(html,"<p>\n%s\n</p>",hostsnames[i]);
    fprintf(html,"</div>"); // Cierre del div graphic

size_x = (count*65)+110;
    size_y = 265;
}

void html_graphics_closepage()
{
// Lista desordenada con las unidades empleadas
    fprintf(html,"<div>\n<ul><li>%s</li>\n
    <li>%s</li><li>%s</li>\n
    </div>\n
    </body>\n
    </html>",unit1,unit2,unit3);
    fclose(html);
}

void html_params_page()
{
    printf("<body bgcolor=\"#506096\"></body>");
}
```

B.3.4. launcher_node

```
void html_launcher_file_write(int montype, char *hs)
{
    FILE *file;
    char filename[MAXFILENAMELEN] = "";
    char paramsid[5*MAXPARAMS] = "", ytitle[BUFFSIZE*MAXPARAMS] = "";
    register int i;
    struct timeval time;
    gettimeofday(&time, NULL);
```

```
appw = ((nparams[montype] > APPLETCOLS) ? APPLETCOLS
: nparams[montype]) * APPLETWIDTH;
apph = (((nparams[montype] / APPLETCOLS) +
((nparams[montype] % APPLETCOLS)>0 ? 1: 0)) * APPLETHEIGHT);

aparam.monitortype = montype;
strcpy(aparam.hoststring, hs);

sprintf(filename, "../launch/launch%i.htm", applets_launched++);

sprintf(paramsid, "%i", aparam.parameterid[aparam.monitortype][0]);
sprintf(aparam.title, "%s", paramnames[aparam.monitortype][0]);
sprintf(ytitle, "%s", ytitles[aparam.monitortype][0]);

for (i=1; i<nparams[aparam.monitortype]; i++)
{
sprintf(paramsid, "%s-%i", paramsid, aparam.parameterid[aparam.monitortype][i]);
sprintf(aparam.title, "%s-%s", aparam.title, paramnames[aparam.monitortype][i]);
sprintf(ytitle, "%s-%s", ytitle, ytitles[aparam.monitortype][i]);
}

file=fopen(filename,"w");
fprintf(file, "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
Transitional//EN\"\n\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n
<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n");
fprintf(file, "\t<head>\n<meta http-equiv=\"content-type\" content=\"text/html;
charset=utf-8\" />\n");
fprintf(file, "<title>%s</title>\n", aparam.title);
fprintf(file, "</head>\n");
fprintf(file, "<body text=\"white\" bgcolor=\"black\">\n\n");
fprintf(file, "<div align=\"center\"> \n");
fprintf(file, "<!--\"CONVERTED_APPLET\"-->\n");
fprintf(file, "<!-- HTML CONVERTER -->\n");
fprintf(file, "<OBJECT classid=\"clsid:CAFEEFAC-0013-0001-0000-ABCDEFFEDCBA\"\n");
fprintf(file, "WIDTH = %i HEIGHT = %i
codebase=\"http://java.sun.com/products/plugin/1.3.1/jinstall-131-win32.cab#Version=1,3,1,0\">
\n", appw, apph);
fprintf(file, "<PARAM NAME = CODE VALUE = \"jdmonchart/jMultiNowMoniTo.class\" >\n");
fprintf(file, "<PARAM NAME = CODEBASE VALUE = \"../monitoring\" >\n");
fprintf(file, "\n");
fprintf(file, "<PARAM NAME=\"type\" VALUE=\"application/x-java-applet;version=1.3\">\n");
fprintf(file, "<PARAM NAME=\"scriptable\" VALUE=\"false\">\n");
fprintf(file, "<PARAM NAME = \"title\" VALUE =\"%s\">\n", aparam.title);
fprintf(file, "<PARAM NAME = \"monitortype\" VALUE =\"%i\"> \n", aparam.monitortype);
fprintf(file, "<PARAM NAME = \"rate\" VALUE =\"%i\"> \n", aparam.rate);
fprintf(file, "<PARAM NAME = \"lefttime\" VALUE =\"%i\"> \n", aparam.lefttime   );
fprintf(file, "<PARAM NAME = \"hoststring\" VALUE =\"%s\"> \n", aparam.hoststring );
fprintf(file, "<PARAM NAME = \"processname\" VALUE =\"%s\"> \n", aparam.processname);
fprintf(file, "<PARAM NAME = \"parameterid\" VALUE =\"%s\">\n", paramsid);
fprintf(file, "<PARAM NAME = \"mtype\" VALUE =\"%i\">\n", aparam.mtype);
fprintf(file, "<PARAM NAME = \"ytitle\" VALUE =\"%s\">\n", ytitle);
```

```
    fprintf(file, "<PARAM NAME = \"numgraphs\" VALUE =\"%i\">\n", nparams[montype]);
    fprintf(file, "<PARAM NAME = \"sec\" VALUE =\"%li\">\n", time.tv_sec);
    fprintf(file, "<PARAM NAME = \"usec\" VALUE =\"%li\">\n", time.tv_usec);
    fprintf(file, "<COMMENT>\n");
    fprintf(file, "<EMBED type=\"application/x-java-applet;version=1.3\"
    CODE = \"jdmonchart/jMultiNowMoniTo.class\" CODEBASE = \"../monitoring\"\n");
    fprintf(file, "WIDTH = %i HEIGHT = %i \n", appw, apph);
    fprintf(file, "title = \"%s\" \n", aparam.title);
    fprintf(file, "monitortype = \"%i\"\n", aparam.monitortype);
    fprintf(file, "rate = \"%i\" \n", aparam.rate       );
    fprintf(file, "lefttime = \"%i\"\n", aparam.lefttime   );
    fprintf(file, "hoststring = \"%s\"\n", aparam.hoststring );
    fprintf(file, "processname = \"%s\" \n", aparam.processname);
    fprintf(file, "parameterid = \"%s\" \n", paramsid);
    fprintf(file, "mtype = \"%i\" \n", aparam.mtype);
    fprintf(file, "ytitle = \"%s\" \n", ytitle );
    fprintf(file, "numgraphs = \"%i\" \n", nparams[montype]);
    fprintf(file, "sec = \"%li\" \n", time.tv_sec);
    fprintf(file, "usec = \"%li\" \n", time.tv_usec);
    fprintf(file, "scriptable=false
    pluginspage=\"http://java.sun.com/products/plugin/1.3.1/plugin-install.html\"> \n");
    fprintf(file, "<NOEMBED>\n \n </NOEMBED> \n");
    fprintf(file, "</EMBED>");
    fprintf(file, "</COMMENT>\n");
    fprintf(file, "</OBJECT>\n");
    fprintf(file, "\n");
    fprintf(file, "<!--\n");
    fprintf(file, "<APPLET CODE = \"jdmonchart/DMonChart.class\"
    CODEBASE = \"../monitoring\" WIDTH = %i HEIGHT = %i>\n", appw, apph);
    fprintf(file, "<PARAM NAME = \"title\" VALUE =\"%s\"> \n", aparam.title);
    fprintf(file, "<PARAM NAME = \"monitortype\" VALUE =\"%i\"> \n", aparam.monitortype);
    fprintf(file, "<PARAM NAME = \"rate\" VALUE =\"%i\"> \n", aparam.rate       );
    fprintf(file, "<PARAM NAME = \"lefttime\" VALUE =\"%i\"> \n", aparam.lefttime   );
    fprintf(file, "<PARAM NAME = \"hoststring\" VALUE =\"%s\"> \n", aparam.hoststring );
    fprintf(file, "<PARAM NAME = \"processname\" VALUE =\"%s\"> \n", aparam.processname);
    fprintf(file, "<PARAM NAME = \"parameterid\" VALUE =\"%s\"> \n", paramsid);
    fprintf(file, "<PARAM NAME = \"mtype\" VALUE =\"%i\"> \n", aparam.mtype);
    fprintf(file, "<PARAM NAME = \"ytitle\" VALUE =\"%s\"> \n", ytitle);
    fprintf(file, "<PARAM NAME = \"numgraphs\" VALUE =\"%i\"> \n", nparams[montype]);
    fprintf(file, "\n");
    fprintf(file, "\n");
    fprintf(file, "</APPLET>\n");
    fprintf(file, "-->\n");
    fprintf(file, "<!--\"END_CONVERTED_APPLET\"-->\n");
    fprintf(file, "   </div> \n");
    fprintf(file, "      </body>\n");
    fprintf(file, "</html>\n");
    fclose(file);
}

void launch_applet()
{
```

```
    register int m;
    register int h;
    char hs[BUFFSIZE*3]="";

    for (m=1; m <= maxmonid; m++)
    {
    for (h=0; h <= maxhostid; h++)
    {
        if (hostsmonitors[h][m])
        sprintf(hs,"%s%i*%s-", hs, h, hostsnames[h]);
        perror(hs);
        }
    if (strlen(hs) != 0)
    {
        hs[strlen(hs)] = '\0';
        html_launcher_file_write(m, hs);
        printf("<script language=\"JavaScript\" type=\"text/javascript\">\n");
        printf( "    window.parent.parent.title.winopen('../launch/launch%i.htm', '_blank',
        'width=%i,height=%i');  \n", applets_launched - 1, appw + 20, apph + 20);
        printf("</script>\n");
    }
    hs[0]='\0';
    }
}

void html_params_page()
{
    printf("<body bgcolor=\"#506096\"></body>");
}
```

B.3.5. launcher_class

```
void html_launcher_file_write(int montype, char *hs)
{
    FILE *file;
    char filename[MAXFILENAMELEN] = "";
    char paramsid[5*MAXPARAMS] = "", ytitle[BUFFSIZE*MAXPARAMS] = "";
    register int i;
    struct timeval time;
    gettimeofday(&time, NULL);

    appw = ((nparams[montype] > APPLETCOLS) ?
    APPLETCOLS : nparams[montype]) * APPLETWIDTH;
    apph = (((nparams[montype] / APPLETCOLS) + ((nparams[montype]
    % APPLETCOLS)>0 ? 1: 0)) * APPLETHEIGHT);

    aparam.monitortype = montype;
    strcpy(aparam.hoststring, hs);
    make_title(aparam.title, montype);

    sprintf(filename, "../launch/launch%i.htm", applets_launched++);
```

```
sprintf(paramsid, "%i", aparam.parameterid[aparam.monitortype][0]);
sprintf(aparam.title, "%s", paramnames[aparam.monitortype][0]);
sprintf(ytitle, "%s", ytitles[aparam.monitortype][0]);
for (i=1; i<nparams[aparam.monitortype]; i++)
{
sprintf(paramsid, "%s-%i", paramsid, aparam.parameterid[aparam.monitortype][i]);
sprintf(aparam.title, "%s-%s", aparam.title, paramnames[aparam.monitortype][i]);
sprintf(ytitle, "%s-%s", ytitle, ytitles[aparam.monitortype][i]);
}

file=fopen(filename,"w");
fprintf(file, "<HTML>\n");
fprintf(file, "   <HEAD>\n");
fprintf(file, "      <TITLE>%s</TITLE>\n", aparam.title);
fprintf(file, "   </HEAD>\n");
fprintf(file, "   <BODY TEXT=\"white\" BGCOLOR=\"black\">\n\n");
fprintf(file, "   <div align=\"center\"> \n");
fprintf(file, "<!--\"CONVERTED_APPLET\"-->\n");
fprintf(file, "<!-- HTML CONVERTER --> \n");
fprintf(file, "<OBJECT classid=\"clsid:CAFEEFAC-0013-0001-0000-ABCDEFFEDCBA\" \n");
fprintf(file, "WIDTH = %i HEIGHT = %i
codebase=\"http://java.sun.com/products/plugin/1.3.1/jinstall-131-win32.cab#Version=1,3,1,0\">
\n", appw, apph);
fprintf(file, "<PARAM NAME = CODE VALUE = \"jdmonchart/jMultiNowMoniTo.class\" > \n");
fprintf(file, "<PARAM NAME = CODEBASE VALUE = \"../monitoring\" > \n");
fprintf(file, "\n");
fprintf(file, "<PARAM NAME=\"type\" VALUE=\"application/x-java-applet;version=1.3\">\n");
fprintf(file, "<PARAM NAME=\"scriptable\" VALUE=\"false\">      \n");
fprintf(file, "<PARAM NAME = \"title\" VALUE =\"%s\">  \n", aparam.title);
fprintf(file, "<PARAM NAME = \"monitortype\" VALUE =\"%i\">  \n", aparam.monitortype);
fprintf(file, "<PARAM NAME = \"rate\" VALUE =\"%i\">     \n", aparam.rate       );
fprintf(file, "<PARAM NAME = \"lefttime\" VALUE =\"%i\">     \n", aparam.lefttime   );
fprintf(file, "<PARAM NAME = \"hoststring\" VALUE =\"%s\">      \n", aparam.hoststring );
fprintf(file, "<PARAM NAME = \"processname\" VALUE =\"%s\">  \n", aparam.processname);
fprintf(file, "<PARAM NAME = \"parameterid\" VALUE =\"%s\">  \n", paramsid);
fprintf(file, "<PARAM NAME = \"mtype\" VALUE =\"%i\">  \n", aparam.mtype);
fprintf(file, "<PARAM NAME = \"ytitle\" VALUE =\"%s\">  \n", ytitle);
fprintf(file, "<PARAM NAME = \"numgraphs\" VALUE =\"%i\">  \n", nparams[montype]);
fprintf(file, "<PARAM NAME = \"sec\" VALUE =\"%li\">  \n", time.tv_sec);
fprintf(file, "<PARAM NAME = \"usec\" VALUE =\"%li\">  \n", time.tv_usec);
fprintf(file, "<COMMENT>     \n");
fprintf(file, "<EMBED type=\"application/x-java-applet;version=1.3\"
CODE = \"jdmonchart/jMultiNowMoniTo.class\" CODEBASE = \"../monitoring\"\n");
fprintf(file, "WIDTH = %i HEIGHT = %i  \n", appw, apph);
fprintf(file, "title = \"%s\" \n", aparam.title);
fprintf(file, "monitortype = \"%i\"    \n", aparam.monitortype);
fprintf(file, "rate = \"%i\"    \n", aparam.rate       );
fprintf(file, "lefttime = \"%i\"    \n", aparam.lefttime   );
fprintf(file, "hoststring = \"%s\"    \n", aparam.hoststring );
fprintf(file, "processname = \"%s\"    \n", aparam.processname);
fprintf(file, "parameterid = \"%s\"    \n", paramsid);
fprintf(file, "mtype = \"%i\"    \n", aparam.mtype);
```

```
    fprintf(file, "ytitle = \"%s\"    \n", ytitle );
    fprintf(file, "numgraphs = \"%i\"    \n", nparams[montype]);
    fprintf(file, "sec = \"%li\"    \n", time.tv_sec);
    fprintf(file, "usec = \"%li\"    \n", time.tv_usec);
    fprintf(file, "scriptable=false
    pluginspage=\"http://java.sun.com/products/plugin/1.3.1/plugin-install.html\">\n");
    fprintf(file, "<NOEMBED> \n");
    fprintf(file, " \n");
    fprintf(file, "</NOEMBED>\n");
    fprintf(file, "</EMBED> \n");
    fprintf(file, "</COMMENT>\n");
    fprintf(file, "</OBJECT> \n");
    fprintf(file, " \n");
    fprintf(file, "<!--     \n");
    fprintf(file, "<APPLET CODE = \"jdmonchart/DMonChart.class\" CODEBASE = \"../monitoring\"
    WIDTH = %i HEIGHT = %i>  \n", appw, apph);
    fprintf(file, "<PARAM NAME = \"title\" VALUE =\"%s\">\n", aparam.title);
    fprintf(file, "<PARAM NAME = \"monitortype\" VALUE =\"%i\">\n", aparam.monitortype);
    fprintf(file, "<PARAM NAME = \"rate\" VALUE =\"%i\">\n", aparam.rate       );
    fprintf(file, "<PARAM NAME = \"lefttime\" VALUE =\"%i\">\n", aparam.lefttime   );
    fprintf(file, "<PARAM NAME = \"hoststring\" VALUE =\"%s\">\n", aparam.hoststring );
    fprintf(file, "<PARAM NAME = \"processname\" VALUE =\"%s\">\n", aparam.processname);
    fprintf(file, "<PARAM NAME = \"parameterid\" VALUE =\"%s\">\n", paramsid);
    fprintf(file, "<PARAM NAME = \"mtype\" VALUE =\"%i\">\n", aparam.mtype);
    fprintf(file, "<PARAM NAME = \"ytitle\" VALUE =\"%s\">  \n", ytitle);
    fprintf(file, "<PARAM NAME = \"numgraphs\" VALUE =\"%i\">  \n", nparams[montype]);
    fprintf(file, " \n");
    fprintf(file, " \n");
    fprintf(file, "</APPLET> \n");
    fprintf(file, "-->      \n");
    fprintf(file, "<!--\"END_CONVERTED_APPLET\"-->       \n");
    fprintf(file, "   </div> \n");
    fprintf(file, "      </BODY>      \n");
    fprintf(file, "</HTML>  \n");
    fclose(file);
}

void launch_applet(/*char *q*/)
{
    register int m;
    register int h;
    char hs[BUFFSIZE*3]="";

    for (m=1; m <= maxmonid; m++)
    {
    for (h=0; h <= maxhostid; h++)
    {
       if (hostsmonitors[h][m])
        sprintf(hs,"%s%i*%s-", hs, h, hostsnames[h]);
    }
    if (strlen(hs) != 0)
    {
```

```
        hs[strlen(hs)] = '\0';
        html_launcher_file_write(m, hs);
        printf("<script language=\"JavaScript\">\n");
        printf( "    window.parent.parent.title.winopen
        ('../launch/launch%i.htm', '_blank', 'width=%i,height=%i');
        \n", applets_launched - 1, appw + 20, apph + 20);
        printf("</script>\n");
    }
    hs[0]='\0';
    }
}

void html_params_page()
{
    printf("<body bgcolor=\"#506096\"></body>");
}
```

B.3.6. status

```
int main(int argc, char *argv[])
{
    int sockfd, nbytes, ssock_mcd, nchars, aux;
    struct hostent *ent;
    char *host_mcd, mcd_host[MAXHOSTNAMEELEN];
    struct mcd_msg_request msg_req = {0, 0, 0, "STATUS"};
    struct sockaddr_in addr_mcd;
    char smsg_req[sizeof(msg_req)+30];
    struct hoststatus msgreply_status[MAXHOSTS+1] = { { -1, "", "" } };
    char query[300]="";
    char *monact;

    nchars = strcspn(query, "=");
    monact = argv[1];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    strcpy(mcd_host,MCDHOST);

    ent = gethostbyname(mcd_host);
    if ( !ent )
    {
       fprintf(stderr,"Error! Not found: %s \n", host_mcd);
    }

    memset(&addr_mcd,0,sizeof (struct sockaddr_in));
    addr_mcd.sin_family=AF_INET;
    addr_mcd.sin_addr.s_addr=(((struct in_addr *)ent->h_addr)->s_addr);
    addr_mcd.sin_port=htons(MCDPORT);

    connect( sockfd, (struct sockaddr *) &addr_mcd, sizeof(struct sockaddr_in) );

    sprintf(smsg_req, "%i %li %li 0 %s",
```

```
            msg_req.type,
            msg_req.interval,
            msg_req.left_time,
            msg_req.msg);

    nbytes = send(sockfd, smsg_req, strlen(smsg_req), 0);
    if ( nbytes < 0 )
    {
        fprintf(stderr,"Error on sendto \n");
    }

    nbytes = recvfrom(sockfd, &msgreply_status, sizeof(msgreply_status),0,
            (struct sockaddr*) &addr_mcd, &ssock_mcd);
    if (nbytes<0)
    {
        fprintf(stderr,"Error on recv\n");
    }
    close(sockfd);

    nbytes = 0;

    while (msgreply_status[nbytes].hostnum != -1 && nbytes <= MAXHOSTS )
    {
        aux = host_show(msgreply_status[nbytes].host_name,
        msgreply_status[nbytes].hostnum,
        (msgreply_status[nbytes].host_monitors[0]) != '#',
        msgreply_status[nbytes].host_monitors, monact);
        nbytes++;
    }
    return 0;
}
```

B.4. Compilación de Monito y paquetería

B.4.1. Makefile

Fichero Makefile para compilar el ScD, McD y los monitores adaptado para las librerías LibGTop

```
#--------------- Makefile configuration---------
#    Modified to work with LibGTop Libraries

#
# Enable Monitors
#
LOADMON = y
MEMMON = y
NETMON = y
ESTAMON = y
CLASSMON = y

#
# libgtop2 path
```

```
#
LIBGTOP2_PATH= -L/usr/lib/libgtop-2.0.so

#
# NetMon options
#
WITH_PVMDGETSTATS = n
PVMINCLUDE = -I/usr/local/pvm3/include
PVMLIBSPATH = -L/usr/local/pvm3/lib/LINUX

#--------------------------------------------
CC = env LC_ALL=C gcc-3.4
CFLAGS = -g -O0 -Wall -D_GNU_SOURCE -DDEBUG
LIBGTOPFLAGS= -D_IN_LIBGTOP -D_GNU_SOURCE -DGLIBTOP_NAMES -DHAVE_STRERROR -DHAVE_CONFIG_H
LIBGTOPINCLUDES= -I/usr/include/libgtop-2.0/glibtop -I/usr/include/libgtop-2.0
-I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include
LIBS = -lpthread -lgtop-2.0 -lglib-2.0
OBJS = mon_common.o load_func.o mem_func.o net_func.o esta_func.o class_func.o
COMMONOBJ = mon_common.o
DEFINES =
INCLUDES =
#mem_func.o
DAEMONS = ScD McD

ifeq ($(LOADMON),y)
    DEFINES += -DENABLE_LOADMON
endif

ifeq ($(MEMMON),y)
    DEFINES += -DENABLE_MEMMON
endif

ifeq ($(NETMON),y)
    DEFINES += -DENABLE_NETMON
    ifeq ($(WITH_PVMDGETSTATS),y)
       DEFINES += -DPVM_GETPVMDSTATS
       INCLUDES += $(PVMINCLUDE)
       LIBS += -lpvm3 -lgpvm3
    endif
endif

ifeq ($(ESTAMON),y)
    DEFINES += -DENABLE_ESTAMON
endif

ifeq ($(CLASSMON),y)
    DEFINES += -DENABLE_CLASSMON
endif

all:
    $(CC) -o ports ports.c
    $(CC) $(CFLAGS) $(LIBGTOPFLAGS) $(LIBGTOPINCLUDES) -c mon_common.c -o mon_common.o
```

```
    $(CC) $(CFLAGS) $(LIBGTOPFLAGS) $(LIBGTOPINCLUDES) -c load_func.c -o load_func.o
    $(CC) $(CFLAGS) $(LIBGTOPFLAGS) $(LIBGTOPINCLUDES) -c mem_func.c -o mem_func.o
    $(CC) $(CFLAGS) $(DEFINES) $(INCLUDES) $(LIBGTOPFLAGS) $(LIBGTOPINCLUDES)
    -c net_func.c -o net_func.o
    $(CC) $(CFLAGS) -c esta_func.c -o esta_func.o
    $(CC) $(CFLAGS) $(LIBGTOPFLAGS) $(LIBGTOPINCLUDES) -c class_func.c -o class_func.o
    $(CC) $(CFLAGS) $(LIBGTOPFLAGS) $(LIBGTOPINCLUDES) -o scd ScD.c
    $(DEFINES) $(OBJS) $(LIBS) $(PVMLIBSPATH)
    $(CC) $(CFLAGS) -o mcd McD.c mon_common.o

clean:
    rm *.o scd mcd
```

Makefile del Monitor Estadistico

```
CFLAGS+= -I$/../dmon/ -I/usr/include/libgtop-2.0/glibtop
-I/usr/include/libgtop-2.0 -I/usr/include/glib-2.0
-I/usr/lib/glib-2.0/include
LIBGTOPFLAGS= -D_IN_LIBGTOP -D_GNU_SOURCE -DGLIBTOP_NAMES -DHAVE_STRERROR -DHAVE_CONFIG_H
LIBGTOPINCLUDES=
LIBS = -lpthread -lgtop-2.0 -lglib-2.0

all:
    gcc-3.4  $(CFLAGS) $(LIBGTOPFLAGS) $(LIBGTOPINCLUDES) $(LIBS) -o esta estadistic.c
    gcc-3.4  $(CFLAGS) -o compro comprovar.c
```

B.4.2. Script Compila

Script en bash para compilar los cgi's lanzadores y de estado

```
#!/bin/bash

cd ../..
RUTA=$(pwd)

gcc-3.4 -c -o $RUTA/vcu/src/common/html_common.o $RUTA/vcu/src/common/html_common.c

gcc-3.4 -I $RUTA/vcu/dmon/ -I$RUTA/vcu/src/common/ -o $RUTA/vcu/src/launcher/launcher.cgi
$RUTA/vcu/src/launcher/launcher.c $RUTA/vcu/dmon/mon_common.o
$RUTA/vcu/src/common/html_common.o
cp $RUTA/vcu/src/launcher/launcher.cgi $RUTA/vcu/cgi-bin/

gcc-3.4  -I $RUTA/vcu/dmon/ -I $RUTA/vcu/src/common/ -o $RUTA/vcu/src/launcher_esta/launcher2.cgi
$RUTA/vcu/src/launcher_esta/launcher_esta.c $RUTA/vcu/dmon/mon_common.o
$RUTA/vcu/src/common/html_common.o
cp $RUTA/vcu/src/launcher_esta/launcher2.cgi $RUTA/vcu/cgi-bin/

gcc-3.4  -I $RUTA/vcu/dmon/ -I $RUTA/vcu/src/common/ -o $RUTA/vcu/src/launcher_node/launcher3.cgi
$RUTA/vcu/src/launcher_node/launcher_node.c $RUTA/vcu/dmon/mon_common.o
$RUTA/vcu/src/common/html_common.o
cp $RUTA/vcu/src/launcher_node/launcher3.cgi $RUTA/vcu/cgi-bin/

gcc-3.4  -I $RUTA/vcu/dmon/ -I $RUTA/vcu/src/common/ -o $RUTA/vcu/src/launcher_class/launcher4.cgi
```

```
$RUTA/vcu/src/launcher_class/launcher_class.c $RUTA/vcu/dmon/mon_common.o
$RUTA/vcu/src/common/html_common.o
cp $RUTA/vcu/src/launcher_class/launcher4.cgi $RUTA/vcu/cgi-bin/

gcc-3.4 -g  -I $RUTA/vcu/dmon/ -I $RUTA/vcu/src/common/ -o $RUTA/vcu/src/status/status.cgi
$RUTA/vcu/src/status/status.c $RUTA/vcu/dmon/mon_common.o
$RUTA/vcu/src/common/html_common.o
cp $RUTA/vcu/src/status/status.cgi $RUTA/vcu/cgi-bin/
```

B.4.3. Scripts de creación de paquetes crea_deb y crea_rpm

Bibliografía

[1] Wikipedia http://www.wikipedia.org

[2] Seymour Cray http://www.cray.com/about_cray/seymourcray.html

[3] BOINC http://boinc.berkeley.edu/

[4] Proyecto ZIVIS http://zivis.zaragoza.es/

[5] Thomas Sterling http://www.cacr.caltech.edu/~tron/

[6] Ganglia http://www.ganglia.info

[7] CluMon http://clumon.ncsa.uiuc.edu/

[8] linux-es http://www.linux-es.org/kernel

[9] kerneltrap.org http://kerneltrap.org/node/7153

[10] http://linuxgazette.net/115/nirendra.html

[11] http://www.kniggit.net/wwol26.html

[12] http://www.kernel.org/pub/linux/kernel/v2.6/

[13] XHTML http://www.w3.org/TR/xhtml1/

[14] CSS http://www.sidar.org/recur/desdi/traduc/es/css/cover.html

[15] Guía de Referencia Rápida CSS http://www.w3c.es/Divulgacion/GuiasReferencia/CSS21/

[16] PHP: Hypertext Preprocessor http://www.php.net

[17] Apache HTTPd server http://httpd.apache.org/

[18] Alejandro Reche. "Recuperación y mejora de un sistema de monitorización de clusters". TFC, EUP, Universitat de Lleida. Febrero 2005.

[19] Josep Bringué. "Disseny i Implementació d'una Eina de Monitorització per Clusters". TFC, EPS, Universitat de Lleida. Enero 2006