



# Supervisión y cálculo de estadísticas de uso de servidores

Iván Cabrera Altamirano, Tanibet Pérez de los Santos Mondragón, *Seguridad en Sistemas de Información.*

*Dr. Arturo Díaz Pérez, CINVESTAV-IPN. Departamento de Computación*

**Resumen—** El presente documento describe el uso de la herramienta *Ganglia* para la supervisión y cálculo de estadísticas de uso de servidores. Inicialmente se provee una breve descripción de la herramienta y de sus componentes básicos. Después, se aborda la instalación, configuración, y puesta en marcha del sistema. Además se propone una arquitectura para la realización de pruebas funcionales y de rendimiento con el fin de evaluar la herramienta en un ambiente controlado. Finalmente se presentan los resultados de las pruebas aplicadas al sistema, así como las conclusiones derivadas de la realización de este trabajo.

## I. INTRODUCCIÓN

El término *clúster* se aplica a un conjunto de computadoras que utilizan componentes de hardware en común y se comportan como si fuesen una computadora única. En la actualidad se utilizan para solucionar problemas de la ciencia e ingeniería.

En otras palabras, *clúster* es un grupo de varias computadoras unidas mediante una red de alta velocidad. Son empleados para mejorar el rendimiento y/o la disponibilidad que provee una sola computadora; entonces de un *cluster* se esperan los siguientes servicios:

- Alto rendimiento.
- Alta disponibilidad.
- Alta eficiencia.

Para que un *clúster* funcione correctamente, no es suficiente con conectar los equipos entre sí, ya que es necesario contar con un sistema de manejo de *clúster*, el cual se encargará de interactuar con el usuario y administrar los procesos que corren para optimizar el funcionamiento.

Para optimizar el funcionamiento de un *clúster* es necesario su monitoreo para cubrir algunas problemáticas tales como: la escalabilidad, confiabilidad, heterogeneidad, gestión, y la evolución del sistema.

El monitoreo de un *clúster* es necesario para procesar y mostrar información referente al rendimiento o disponibilidad de los recursos.

A continuación se enlistan las principales problemáticas para un sistema de monitoreo distribuido:

- Escalabilidad: Al añadir nodos en el sistema es necesario que el software de monitoreo sea capaz de identificar las fallas para que puedan ser reparadas en un tiempo adecuado.
- Robustez: El sistema debe seguir funcionando a pesar de fallas en la red.
- Extensibilidad: El sistema debe ser extensible para monitorear diferentes tipos de datos.
- Manejabilidad: El sistema debe gestionar las sobrecargas que afecten el escalamiento.
- Portabilidad: Debe ser portable en diversos sistemas operativos.
- Sobrecarga: El sistema debe incurrir a bajas sobrecargas de recursos (*CPU*, ancho de banda) por cada nodo.

Existen métricas que ayudan a estimar si se cumple una característica de software. Por ejemplo para medir la disponibilidad se aplica la métrica de *uptime* (tiempo de subida), la cual es la fracción de tiempo que un sitio tarda en manipular los datos.

Al aplicar algunas métricas se pueden obtener datos que pueden ser procesados para generar estadísticas detalladas de los recursos de cómputo. También, con la información recopilada, se pueden generar gráficas las cuales simplifican el análisis de los datos.

Al analizar esta información se puede optimizar el funcionamiento del *clúster*. Ya que con un monitoreo profundo de los parámetros de operación se puede saber el comportamiento detallado de los recursos, esto facilita la detección de errores en distintos los niveles.

*Ganglia* es un sistema distribuido escalable de monitoreo para sistemas de cómputo de alto desempeño, tales como *clusters* y *grids*.

Este sistema está basado en un diseño jerárquico orientado a conjuntos de clusters. Este aprovecha tecnologías ampliamente utilizados como *XML*, *XDR* y *RRDTool*. Utiliza estructuras de datos y algoritmos cuidadosamente diseñados para alcanzar sobrecargas muy bajas por nodo y lograr una alta concurrencia.

## II. DESCRIPCIÓN DE LA HERRAMIENTA

En esta sección se describe la funcionalidad de la herramienta *Ganglia*.

### Componentes del Sistema

Los principales componentes de un sistema de monitoreo basado en *Ganglia* son:

#### a) *Ganglia Monitoring Daemon (gmond)*

*Gmond* es un demonio multihilos que se ejecuta en cada nodo del *cluster* que se desee monitorear. Su instalación es fácil, no se necesita tener un sistema de archivos común como *NFS* o una base de datos, cuentas especiales de instalación, mantener archivos de configuración, ni otros molestos problemas.

*Gmond* tiene cuatro responsabilidades principales: monitorear cambios en el estado del cliente, anunciar cambios relevantes, escuchar el estado de otros nodos *Ganglia* vía canales *unicast* o *multicast* y responder peticiones de una descripción *XML* del estado de un *cluster*.

Cada *gmond* transmite información en dos diferentes formas: estado del host en *unicast/multicast* mediante un formato de representación externa de datos (*XDR*) usando mensajes *UDP* o enviando archivos *XML* sobre una conexión *TCP*.

*Gmond* se configura a través del archivo `/etc/gmond.conf`.

#### b) *Ganglia Metadata Daemon (gmetad)*

*Gmetad* es un demonio contenido en un solo archivo, el cual se encarga de consolidar los datos traídos de nodo en una base de datos *RRDtools*. Se necesita una instancia de este demonio por cada *cluster*. En *clusters* masivos, es posible tener más de un demonio *gmetad* organizados de forma jerárquica.

Mientras que *gmond* utiliza canales *multicast* en una forma par a par, *gmetad* envía el archivo de descripción *XML* desde las fuentes de datos de *Ganglia*, sobre rutas *unicast*.

*Gmetad* es el componente del sistema que es accedido indirectamente por la interfaz de usuario a través de la página *web* de *Ganglia*. *Gmetad* almacena información histórica a bases de datos con turnos circulares y exporta resúmenes en formato *XML* que representan poderosas instantáneas y tendencias de todos los nodos monitoreados por *Ganglia*.

*Gmetad* se configura a través del archivo `/etc/gmetad.conf`

#### c) *Ganglia Web Frontend*

El componente de interacción con el usuario a través del *web* proporciona una vista de la información recogida en tiempo real a través de páginas *web* dinámicas. Lo más importante

que muestra *Ganglia* es una presentación de datos de una manera significativa para los administradores de sistema y los usuarios de equipos de cómputo. A pesar de que la interfaz *web* de *Ganglia* comenzó como una simple vista *HTML* del árbol *XML*, que se ha convertido en un sistema que mantiene una atractiva historia de todos los datos recogidos.

#### d) Base de datos y servidor *web*.

*Ganglia* utiliza la bien conocida y respetada herramienta para bases de datos de código abierto llamada *RRDTool*.

Cualquier servidor con soporte para *PHP*, *SSL* y *XML* puede ser utilizado para *Ganglia*, comúnmente es utilizado *Apache2*.

En la figura 1, se puede apreciar el flujo de datos en un sistema *Ganglia*.

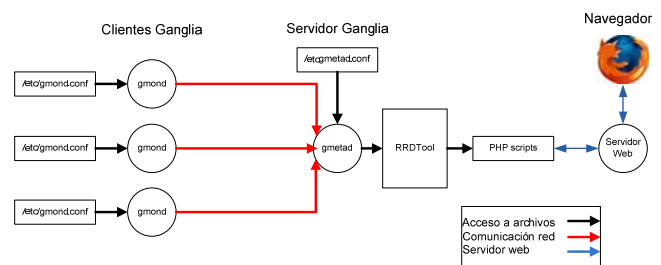


Figura 1. Estructura del sistema *Ganglia*.

### Entradas del Sistema

Las principales métricas sobre las cuales se pueden generar información son las siguientes:

boottime	cpu_user	mem_buffers	part_max_used
bytes_in	cpu_wio	mem_cached	pkts_in
bytes_out	disk_free	mem_free	pkts_out
cpu_idle	disk_total	mem_report	proc_run
cpu_nice	gexec	mem_shared	proc_total
cpu_num	load_fifteen	mem_total	swap_free
cpu_report	load_five	network_report	swap_total
cpu_speed	load_one	os_name	
cpu_system	load_report	os_release	
	machine_type	packet_report	

Adicionalmente se pueden generar estadísticas tomando en cuenta parámetros como:

- Nodo
- Día
- Mes
- Hora
- Semana
- Año

### Salidas del Sistema

El sistema de monitoreo presenta su información de salida a manera de graficas, generadas de forma dinámica. En la figura 2, se aprecia la salida del sistema, para una consulta de la métrica *network\_report*.

Reporte generado por el sistema:

Grafica 1: Representa la carga del nodo en la última hora.

Grafica 2: Uso del CPU en la ultima hora.

Grafica 3: Uso de la memoria en la ultima hora.

Grafica 4: Uso de la red en la ultima hora.

Abajo tenemos la información del uso de la red, para los demás nodos.



Figura 2. Gráficas generadas para la métrica *network\_report*.

### III. ARQUITECTURA DEL SISTEMA DE PRUEBAS

La arquitectura que se utilizará en el desarrollo de este proyecto, tal y como se muestra en la figura 3, está compuesta por 4 equipos principales:

- **Servidor Ganglia:** Ejecutando el demonio *gmetad*. Este servidor además tiene habilitado un servicio *HTTP* para consultar las estadísticas.
- **Cientes Ganglia:** Ejecutando el demonio *gmond*. Estos clientes además ejecutan servicios como: *HTTP*, *SSH* y Correo Electrónico.

La plataforma se complementa con equipos adicionales, los cuales consisten en:

- **Cientes de consulta:** Estos clientes hacen peticiones al servidor *HTTP* para ver las estadísticas que han generado los clientes *Ganglia*.
- **Cientes para generación de tráfico:** Estos clientes hacen peticiones a los servidores *HTTP*, *SSH* y de correo electrónico para generar tráfico dentro de la red, y que este pueda ser registrado por los demonios correspondientes.

Todos estos equipos se encuentran conectados mediante un *Switch Fast Ethernet 10/100 Mbps*. Las maquinas que ejecutan clientes o servidores *Ganglia*, tienen el sistema operativo *GNU/Linux* instalado, mientras que los clientes de consulta y de generación de tráfico pueden estar operando bajo *Windows* o *GNU/Linux*.

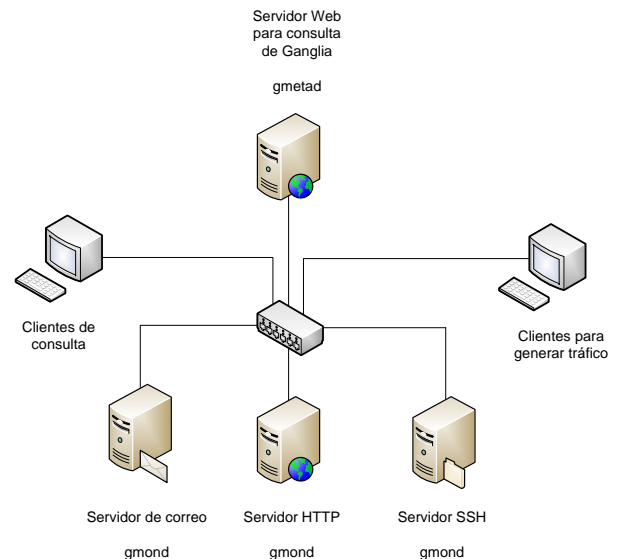


Figura 3. Arquitectura del sistema de evaluación de Ganglia

### IV. INSTALACIÓN

Para la versión 3.1.2:

- Descargamos el archivo *ganglia-3.1.2.tar.gz* desde el sitio del proyecto *Ganglia*.
- Lo descomprimos con el siguiente comando:  
# tar zxvf ganglia-3.1.2.tar.gz

A continuación presentamos las instrucciones específicas para el cliente y servidor. Es importante recordar que para esta etapa, las librerías y aplicaciones marcadas como requerimientos ya deben estar correctamente instalados.

#### Requerimientos

Para instalar esta herramienta requerimos tener los siguientes paquetes instalados:

- **Apache Portable Runtime:** Librería de soporte para el servidor web apache.  
<http://apr.apache.org/>
- **libConfuse:** Parser para archivos de configuración.  
<http://www.nongnu.org/confuse/>
- **Expat:** Parser para archivos XML  
<http://expat.sourceforge.net/>

- Python: Compilador y herramientas para desarrollo en Python.  
<http://www.python.org/>
- RRDtool: Base de datos para mediciones temporales.  
<http://oss.oetiker.ch/rrdtool/>
- GNU Make: Controla la generación de archivos ejecutables.  
<http://www.gnu.org/software/make/>
- GNU C Compiler: Compilador de C para GNU/Linux.  
<http://gcc.gnu.org/>

En los sitios de Internet listados para cada herramienta, se presentan las instrucciones específicas para la instalación de cada una de ellas. Además es posible instalarlas a través de repositorios con herramientas como: `synaptics`, `apt-get`, `aptitude`, `wajig`.

### Instalación de cliente (gmond).

Como root, ejecutamos los siguientes comandos:

- Cambiamos al directorio principal y ejecutamos para crear el archivo `makefile`.  
`# ./configure`
- Compilamos la aplicación con el comando `make`:  
`# make`
- Instalamos la aplicación:  
`# make install`

Opcionalmente podemos instalar `gmond` a través de un repositorio con:

```
# sudo apt-get install ganglia-monitor
```

Otra opción para instalar `gmond` es a través de un paquete `.deb`, el cual podemos descargar desde:

<http://packages.debian.org/sid/i386/ganglia-monitor/download>

Esta última opción es la más recomendada.

### Instalación de servidor (gmetad).

Como root, ejecutamos los siguientes comandos:

- Cambiamos al directorio principal y ejecutamos para crear el archivo `makefile`.  
`# ./configure --with-gmetad`
- Compilamos la aplicación con el comando `make`:  
`# make`

- Instalamos la aplicación:  
`# make install`

Opcionalmente podemos instalar `gmetad` a través de un repositorio con:  
`# sudo apt-get install gmetad`

Otra opción para instalar `gmetad` es a través de un paquete `.deb`, el cual podemos descargar desde:  
<http://packages.debian.org/sid/i386/gmetad/download>

Esta última opción es la más recomendada.

### Instalación de la interfaz web.

- Instalamos PHP5, Apache2  
`# apt-get install php5-gd apache2`
- Instalamos la librería de PHP para Apache.  
`# apt-get install libapache2-mod-php5`
- Habilita el modulo PHP5 en Apache.  
`# a2enmod php5`
- Copiamos la carpeta web de los archivos de instalación a `/var/www/ganglia`  
`# cp -r web /var/www/ganglia/`
- Reiniciamos el servidor Apache:  
`#/etc/init.d/apache2 restart`

## V. CONFIGURACIÓN

### Configuración de cliente (gmond.conf)

Para esta instalación usaremos los archivos de configuración mínima, dejando muchas opciones por omisión, que después podremos explorar y en su caso utilizar.

- Obtenemos el archivo de configuración por omisión para el cliente:  
`# gmond --default_config > /etc/ganglia/gmond.conf`

El archivo de configuración final se muestra en el anexo.

Los hosts deben estar registrados con la dirección IP correspondiente en el archivo `/etc/hosts`

### Configuración de servidor (gmetad.conf)

Para esta instalación usaremos los archivos de configuración mínima, dejando muchas opciones por omisión, que después podremos explorar y en su caso utilizar.

```
data_source "CinvesCluster" node1 node2 node3 node4
```

El archivo de configuración final se muestra en el anexo.

Los hosts deben estar registrados con la dirección IP correspondiente en el archivo `/etc/hosts`.

## VI. PARAMETRIZACIÓN

Para establecer los parámetros de operación en el cliente, se debe realizar a través del archivo de

configuración `/etc/ganglia/gmond.conf`.

```
collection_group {
  collect_once = yes
  time_threshold = 20
  metric {
    name = "heartbeat"
  }
}
```

Se tiene una estructura denominada `collection_group`, donde se especifica, cada cuanto tiempo se recolectara y la métrica a recolectar. En el ejemplo anterior, solo se recolectara una vez y se enviara la métrica `heartbeat`, cada 20 segundos.

```
collection_group {
  collect_every = 1800
  time_threshold = 3600
  metric {
    name = "disk_total"
    value_threshold = 1.0
    title = "Total Disk Space"
  }
}
```

Para el ejemplo anterior, la métrica `disk_total`, se recolectara cada 1800 segundos y se enviara cada 3600 segundos.

En la instalación actual, se utilizaron los valores por omisión del programa, los cuales se muestran en el archivo de configuración de cada cliente.

## VII. ETAPA DE PRUEBAS

### Pruebas de funcionalidad

Esta etapa tiene el propósito de verificar que la herramienta *Ganglia* funcione de manera adecuada. Este conjunto de pruebas se ejecuta sobre un ambiente controlado con el propósito de revisar que la herramienta funciona como se espera.

Por un ambiente controlado entendemos que el tráfico generado, los eventos sucedidos y los servicios que se ofrecen son iniciados y/o terminados en cualquier momento que la entidad que prueba el sistema lo considere necesario.

En la siguiente sección se presentan las pruebas de funcionalidad que pueden ser ejecutadas en un sistema que tenga la herramienta *Ganglia* instalada.

Las pruebas de funcionalidad que se realizan sobre un sistema con la herramienta *Ganglia* se muestran en la tabla 1.

Prueba	Descripción
Boovertime	La última ocasión que el sistema volvió a iniciar.
bytes_in	Número de bytes recibidos por segundo.
bytes_out	Número de bytes enviados por segundo.
cpu_idle	Porcentaje de tiempo del CPU ocioso desde el arranque.
cpu_nice	Porcentaje de tiempo del CPU mientras estaba ocioso y el sistema no atendía operaciones de E/S.
cpu_num	Porcentaje de utilización del CPU mientras se ejecutaba en nivel de usuario con prioridad agradable.
cpu_report	Número total de CPUs.
cpu_speed	Reporte de los CPUs.
cpu_system	Reporte de los CPUs.
cpu_user	Velocidad de los CPUs.
cpu_wio	Porcentaje de utilización del CPU mientras se ejecutaba en modo sistema.
disk_free	Porcentaje de utilización del CPU mientras se ejecutaba en modo usuario.
disk_total	Porcentaje de tiempo que el CPU estuvo inactivo mientras el sistema realizaba operaciones de E/S.
gexec	Espacio disponible en disco.
load_fifteen	Espacio total en disco.
load_five	Estado de Gexec.
load_one	Promedio de carga en los últimos 15 minutos.
load_report	Promedio de carga en los últimos 5 minutos.
machine_type	Promedio de carga en el último minuto.
mem_buffers	Reporte de carga.
mem_cached	Arquitectura de la maquina.
mem_free	Cantidad de memoria en buffers.
mem_report	Cantidad de memoria cache.
mem_shared	Cantidad de memoria disponible.
mem_total	Reporte de memoria.
network_report	Cantidad de memoria compartida.
os_name	Cantidad total de memoria en KB.
os_release	Reporte de red.
packet_report	Nombre del Sistema operativo.
part_max_used	Versión del sistema operativo.
pkts_in	Reporte de paquetes.
pkts_out	Porcentaje máximo de disco ocupado para todas las particiones.
proc_run	Número de paquetes recibidos por segundo.
proc_total	Número de paquetes enviados por segundo.
swap_free	Número total de procesos ejecutándose.
swap_total	Número total de procesos.
	Cantidad disponible de memoria de intercambio.
	Cantidad total de memoria de intercambio.

Tabla 1. Pruebas de funcionalidad en un sistema *Ganglia*.

### Pruebas de rendimiento

Esta etapa tiene el propósito de verificar el desempeño de la herramienta a través de pruebas de rendimiento. Para el caso de nuestra herramienta, este conjunto de pruebas va relacionado con la configuración de la instalación y de los parámetros de operación de *Ganglia*.

Estas pruebas se realizan sobre un ambiente controlado, tal y como se describió en la sección 4 de este documento.

En la siguiente sección se presentan las pruebas de rendimiento que pueden ser ejecutadas en un sistema que tenga la herramienta *Ganglia* instalada.

Las pruebas de rendimiento que se realizan sobre un sistema con la herramienta *Ganglia* se muestran en la tabla 2.

Prueba	Descripción
Utilización del CPU	Porcentaje de utilización del procesador.
Utilización de memoria	Porcentaje de utilización de la memoria.
Utilización de memoria de intercambio	Porcentaje de utilización de la memoria de intercambio
Bytes enviados	Cantidad de bytes enviados por segundo.
Bytes recibidos	Cantidad de bytes recibidos por segundo.
Periodo de muestreo	Cantidad de veces por segundo que se muestrea la variable.
Carga media para el último minuto	Promedio de carga en los últimos 15 minutos.
Carga media para los últimos 5 minutos	Promedio de carga en los últimos 5 minutos.
Carga media para los últimos 15 minutos	Promedio de carga en el último minuto.

Tabla 2. Pruebas de rendimiento en un sistema *Ganglia*.

## VIII. RESULTADOS

### Pruebas de funcionalidad

Se eligieron 6 pruebas de funcionalidad, las cuales se describen y presentan sus resultados a continuación.

### Prueba # 1

La primer prueba consistió en transferir un archivo de video, entre una maquina externa al *cluster* desde un servidor *HTTP* dentro del *cluster*. El tamaño del archivo es de aproximadamente 60 MB.

En esta prueba esperamos ver una modificación en los parámetros: CPU\_User, CPU\_System, Bytes\_Sent, Network Last Hour. Mientras que los parámetros restantes los cambios que esperamos ver son mínimos, debidos principalmente a otros procesos.

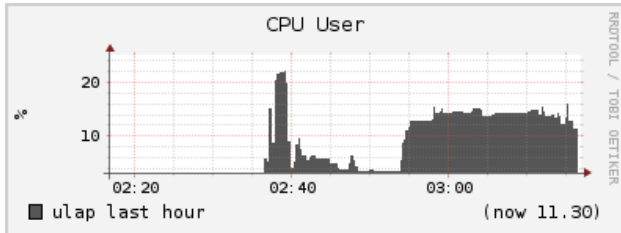


Figura 4. Prueba 1 – Uso de CPU para procesos de usuario.

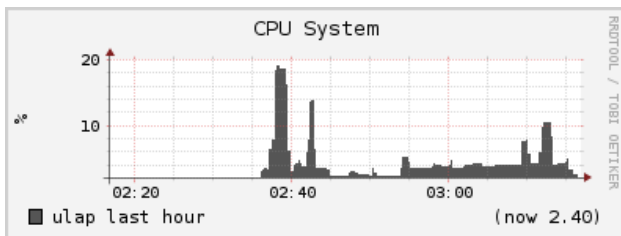


Figura 5. Prueba 1 – Uso de CPU para procesos de sistema.

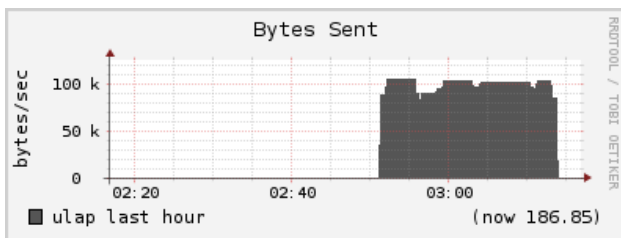


Figura 6. Prueba 1 – Cantidad de bytes enviados.

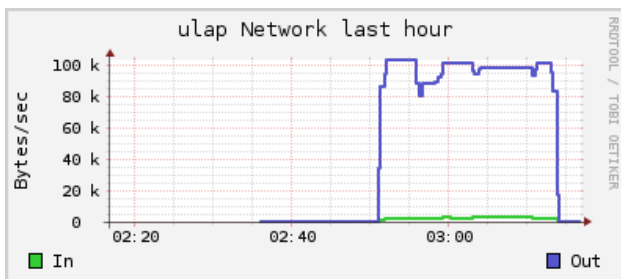


Figura 7. Prueba 1 – Vista general de la actividad de red.

La figura 4 muestra un incremento en el uso del CPU por parte de un programa de usuario, en este caso Apache2, dado que se está realizando la transferencia de un archivo vía *HTTP*, además la figura 6, muestra que súbitamente se ha comenzado a enviar datos por la red a una de transferencia promedio de 90KB.

La figura 5 muestra el uso del CPU para procesos de sistema, el cual no ha tenido mayor influencia bajo esta prueba.

Finalmente la figura 7 muestra la actividad de la red en la última hora, por un lado confirma la información presentada en la figura 6 (trazo azul), mientras que nos informa que la cantidad de información recibida fue mínima, como podemos apreciarlo en el trazo de color rojo de la figura 7.

### Prueba # 2

La segunda prueba consistió en transferir un archivo de video con un tamaño aproximado de 700 MB a una memoria USB.

En esta prueba esperamos ver una modificación en los parámetros: CPU\_User, CPU\_System, CPU\_wio, Bytes\_Sent, Network Last Hour. Mientras que los parámetros restantes los cambios que esperamos ver son mínimos, debidos principalmente a otros procesos.

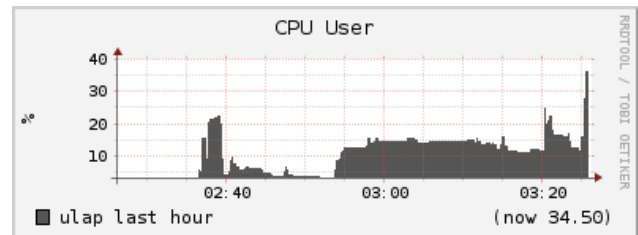


Figura 8. Prueba 2 – Uso de CPU para procesos de usuario.

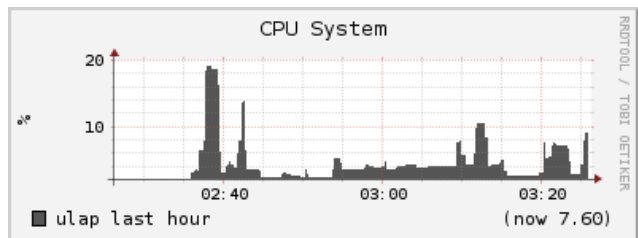


Figura 9. Prueba 2 – Uso de CPU para procesos de sistema

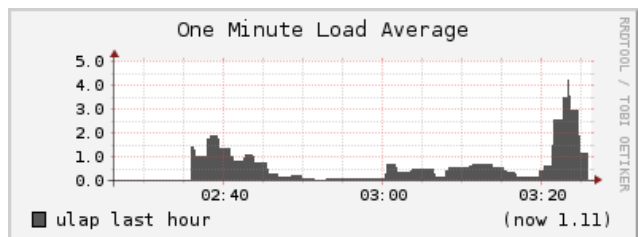


Figura 10. Prueba 2 – Carga promedio en el último minuto.

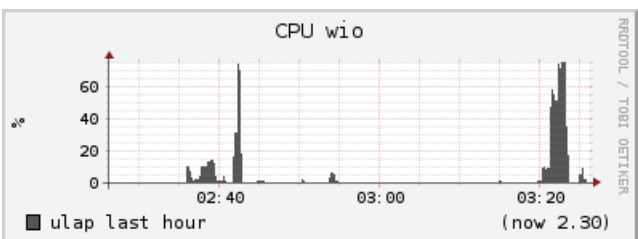


Figura 11. Prueba 2 – Porcentaje de CPU inactivo por E/S.

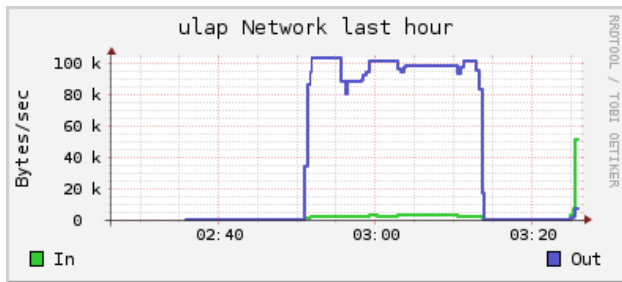


Figura 12. Prueba 2 – Vista general de la actividad de red.

La figura 8, muestra un incremento en la actividad del CPU, tal y como era esperado dado un programa de usuario, mientras tanto el CPU para programas de sistema incrementa su actividad (Figura 9), pero en menor proporción que la figura 8.

Otro punto interesante de mencionar es lo que podemos apreciar en la figura 11, el porcentaje del tiempo que el CPU permaneció en espera dado operaciones de E/S, tales como el USB.

Adicionalmente la carga promedio del CPU aumenta en el último minuto tal y como lo muestra la figura 8. Por último, tenemos la figura 11, la cual muestra que la actividad de red a partir de las 3:20, que fue cuando inicio la transferencia del archivo, es prácticamente nula.

### Prueba # 3

La tercer prueba consistió en ejecutar un programa que haga uso intensivo del CPU, en este caso elegimos un programa que simula gráficamente (a través de la consola y la librería *ncurses*) la resolución del problema de las Torres de Hanoi de forma recursiva.

En esta prueba esperamos ver una modificación en los parámetros: CPU\_User, CPU\_System, Memoria de intercambio. Mientras que los parámetros restantes los cambios que esperemos ver son mínimos, debidos principalmente a otros procesos.

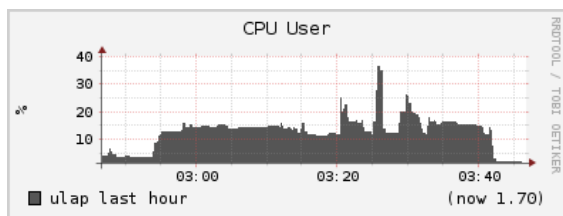


Figura 13. Prueba 3 – Uso de CPU para procesos de usuario.

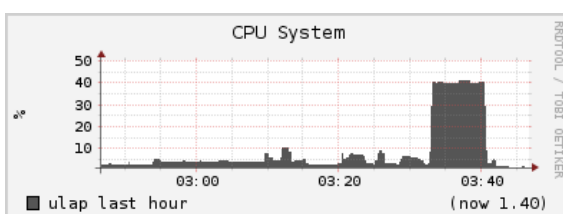


Figura 14. Prueba 3 – Uso de CPU para procesos de sistema.

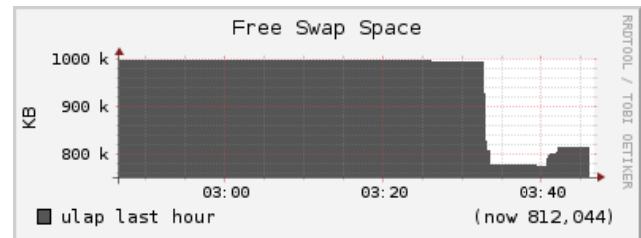


Figura 15. Prueba 3 – Memoria de intercambio libre.

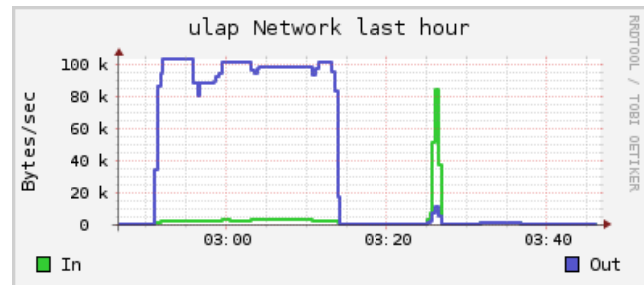


Figura 16. Prueba 3 – Vista general de la actividad de red.

La figura 13, muestra el comportamiento del CPU para los procesos de usuario, el cual incrementa ligeramente aproximadamente a las 3.32 hora en que comienza la ejecución del programa para la prueba 3.

Sin embargo, el comportamiento del CPU para los procesos de sistema incrementa considerablemente, hasta casi el 40%, tal y como se muestra en la figura 14. Este tipo de aplicación hace uso máximo del procesador, sin embargo solo vemos valores cercanos al 50% (resultado de la suma de CPU\_User y CPU\_System), debido a que la aplicación se ejecuta sobre un sistema que tiene un procesador con 2 núcleos, y que la aplicación no ha sido programada para usar ambos.

Como lo podemos apreciar en la figura 15, hay una disminución en la memoria de intercambio libre, teniendo una reducción de casi el 20%. Por último, en la figura 16, mostramos que la actividad de la red durante esta prueba es casi nula.

### Prueba # 4

La cuarta prueba de funcionalidad consistió en comprimir un archivo de 1.4 GB en una maquina del *cluster*. El proceso inicio a las 10.30 y tardo 7 min.

En esta prueba esperamos ver una modificación en los parámetros: CPU\_User, CPU\_System, CPU\_wio y carga.

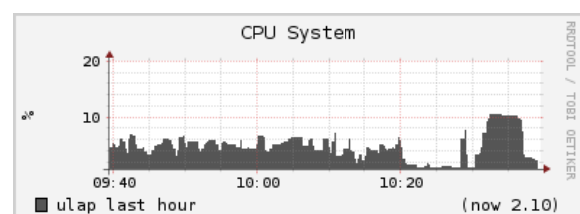


Figura 17. Prueba 4 – Uso de CPU para procesos de sistema.

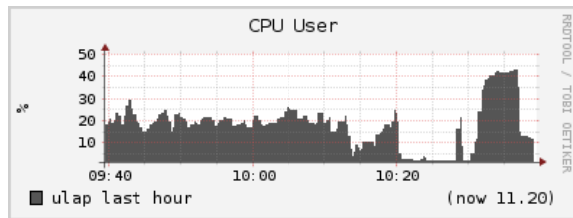


Figura 18. Prueba 4 – Uso de CPU para procesos de usuario.

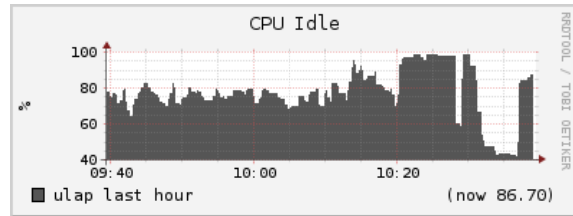


Figura 19. Prueba 4 – Porcentaje de CPU ocioso.

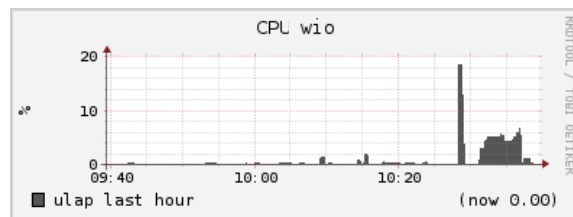


Figura 20. Prueba 4 – Porcentaje de CPU inactivo por E/S.

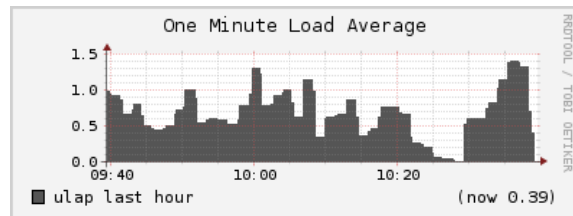


Figura 21. Prueba 4 – Carga promedio en el último minuto.

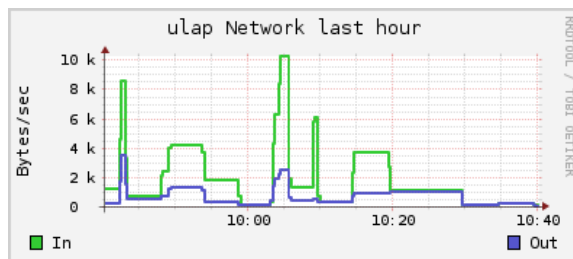


Figura 22. Prueba 4 – Estado de la red en la última hora.

La figura 17 y 18 muestran como el CPU fue utilizado por la tarea encargada de comprimir el archivo, así mismo en la figura 19, se muestra como el CPU paso de un estado mayormente ocioso a un estado de trabajo.

La figura 21, muestra la carga promedio para este equipo durante el último minuto, la cual fue incrementando considerablemente cada minuto.

Además la figura 20, muestra como fue el bloqueo del CPU por operaciones de entrada/salida, tal como escribir a un archivo.

Finalmente la figura 22 muestra el estado de la red durante la operación de compresión, donde se muestra con una actividad mínima.

### Prueba # 5

La quinta prueba de funcionalidad consistió en enviar 50 correos electrónicos de puro texto con mensajes de prueba desde el servidor *sendmail* instalado en un equipo del *cluster*.

La prueba comenzó a las 10:15 y tuvo una duración de 5 minutos.

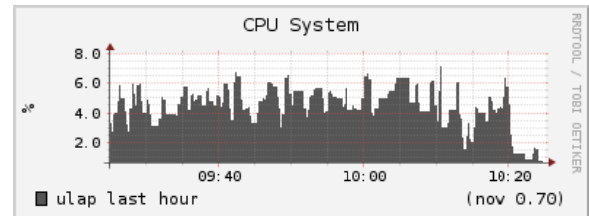


Figura 23. Prueba 5 - Uso de CPU para procesos de sistema.

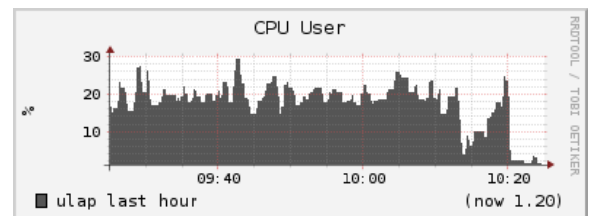


Figura 24. Prueba 5 – Uso de CPU para procesos de usuario.

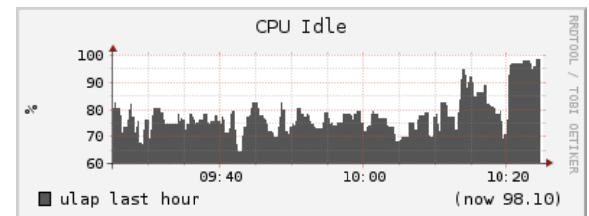


Figura 25. Prueba 5 – Porcentaje de CPU ocioso.

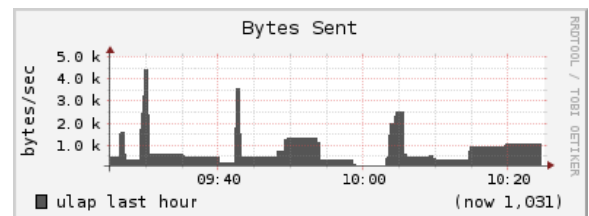


Figura 26. Prueba 5 – Cantidad de bytes enviados.

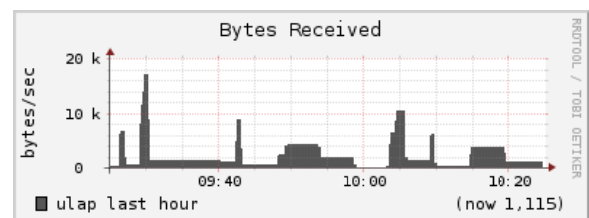


Figura 27. Prueba 5 – Cantidad de bytes recibidos.

La figura 23 y 24, muestran como fue el incremento en el uso de CPU al momento de enviar los correos electrónicos de prueba, destacando un mayor uso para procesos de usuario.

La figura 25 muestra el porcentaje de CPU ocioso, donde se puede ver que durante el tiempo que se enviaron los correos, el CPU tuvo mayor carga de trabajo, sin embargo esta fue baja en comparación a otros servicios como HTTP.

Por último la figura 26 y 27 muestran el uso de la red para el envío de correos electrónicos, esta muestra un uso bajo, dado que el correo electrónico solo contenía texto de prueba.

## Prueba # 6

La sexta prueba de funcionalidad consistió en una sesión SSH. La prueba duro 3 min y comenzó a las 10.45.

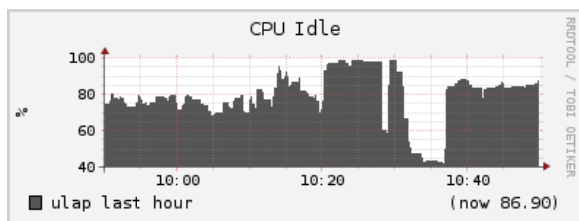


Figura 28. Prueba 6 – Porcentaje de CPU ocioso.

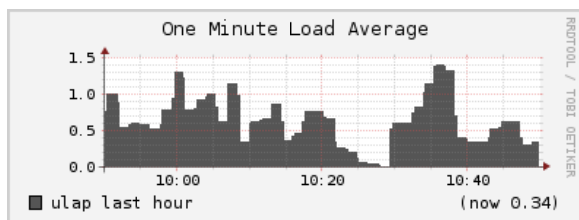


Figura 29. Prueba 6 – Carga promedio en el último minuto.

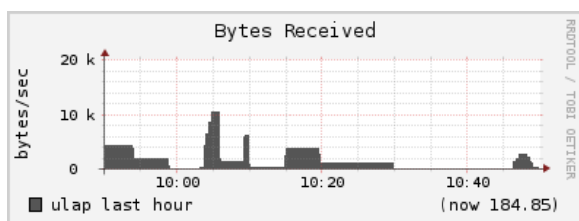


Figura 30. Prueba 6 – Cantidad de bytes recibidos.

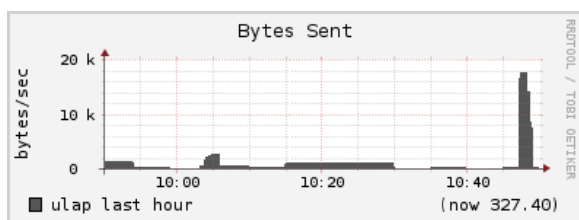


Figura 31. Prueba 6 – Cantidad de bytes enviados.

La figura 28 y 29 muestran que el uso del CPU no incremento por hacer uso del servicio de SSH. Cabe destacar que no se lanzo ninguna aplicación que consumiera CPU desde la consola ya que deseábamos ver que tanto incremento en el uso de CPU contribuía el tener funcionando el servicio de SSH.

Pudimos ver que el porcentaje que esperamos ver no se alcanza a distinguir del resto de los procesos.

La figura 30 se muestra la cantidad de bytes recibidos, los cuales son pocos, ya que estos representan los comandos enviados por el equipo remotamente.

En la figura 31 se muestra la cantidad de bytes enviados, los cuales son mayores a comparación de los bytes recibidos, ya que estos representan la respuesta del servidor al cliente.

En esta prueba, se enviaron repetidamente peticiones para explorar el sistema de archivos a través del Shell.

A manera de complemento a esta información, se proveen todas las graficas de las 6 pruebas en el anexo A.

## Pruebas de rendimiento

Se eligieron 4 pruebas de rendimiento, las cuales se describen y presentan sus resultados a continuación.

### Prueba # 1

La primera prueba consistió en monitorear en la computadora que aloja el servidor *HTTP* y cliente *Ganglia*, el uso de CPU y memoria cuando tenemos los valores de muestreo por omisión.

Los valores por omisión son:

- Periodo de muestreo para CPU = 20 segundos
- Periodo de muestreo para espacio libre en disco = 1800 segundos
- Periodo de muestreo para memoria = 40 segundos.
- Periodo de muestreo para actividad de red = 40 segundos.

Los resultados los presentamos en la figura 32.

### Prueba # 2

La siguiente prueba fue modificar los periodos de muestreo para CPU, memoria y red a 1 segundo.

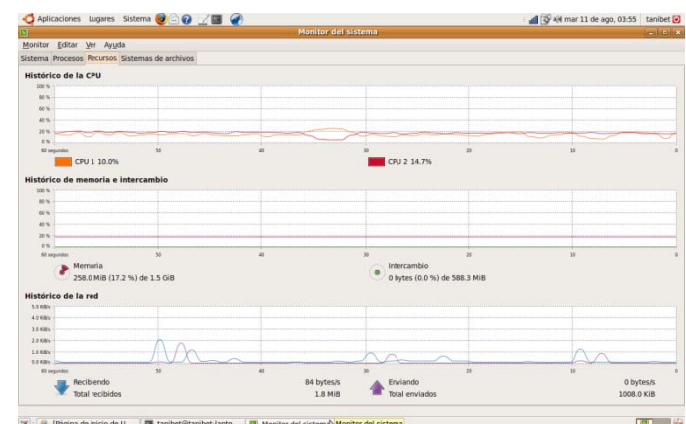


Figura 32. Resultados para prueba 1 de rendimiento.

Los resultados los presentamos en la figura 33.

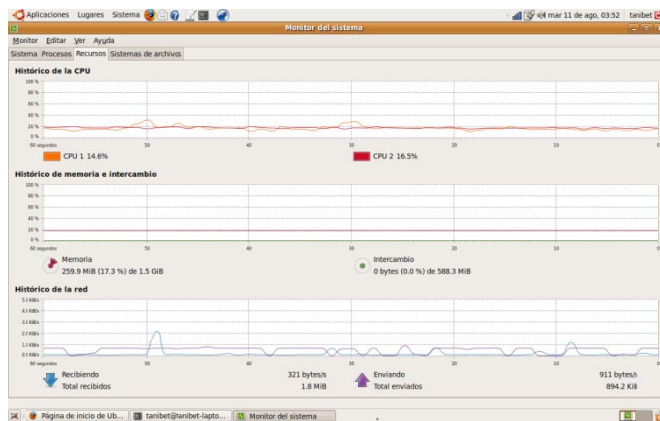


Figura 33. Resultados para la prueba 2 de rendimiento.

Como podemos apreciar, el incremento promedio tanto en uso de CPU y de memoria entre ambas pruebas es mínimo, lo cual atribuimos a las siguientes razones:

- Tiempo de muestreo: El tiempo que lleva tomar una muestra es mínimo y las instrucciones son optimizadas. Para tomar una medición, pudiéramos considerar que esta solo lleva algunas decenas de instrucciones en lenguaje ensamblador.
- Uso de la base de datos: Esta aplicación hace uso de una base de datos *round robin (rrdtool)* con tamaño fijo, por lo tanto no esperamos incrementos significativos en el uso de memoria del sistema.

El único punto donde podemos apreciar un cambio es en la actividad de la red, donde podemos apreciar que tenemos la presencia de mayor tráfico, en especial a la salida.

Estas pruebas tuvieron como objetivo evaluar el desempeño de los clientes, a continuación se muestra el desempeño del servidor, ante el mismo tipo de pruebas.

### Prueba # 3

La prueba numero 3 consistió en observar los mismos parámetros, para el caso del servidor *Ganglia*.

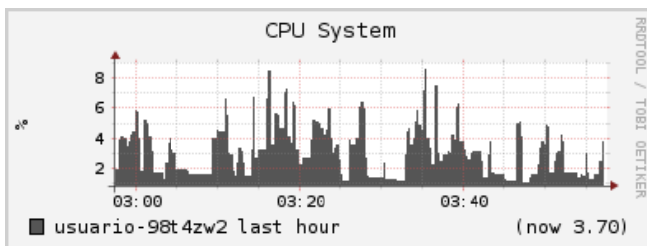


Figura 34. Uso del CPU en el servidor *Ganglia*.

En la figura 34, podemos apreciar una mayor utilización del CPU para procesos de sistema alrededor de las 3:52, que fue cuando los parámetros de muestreo fueron modificados y llevados a 1 segundo.

Minutos después, a las 3:55, cuando los parámetros fueron regresados a los valores por omisión, notamos como la utilización del CPU disminuyo en forma apenas considerable.

Los resultados de las pruebas de rendimiento han demostrado que, el rendimiento de la herramienta mantiene un comportamiento uniforme, sin importar que tanto disminuyamos o incrementemos el periodo de muestreo.

Consideramos que este comportamiento esta dado por que la frecuencia de muestreo es bastante más pequeña que la frecuencia de operación del CPU. Si tomáramos en cuenta una frecuencia de muestreo muy parecida a la frecuencia de operación del CPU, podríamos visualizar que buena parte del tiempo el CPU se encuentra tomando muestras, pero como esto no es así, apenas podemos distinguir la actividad del CPU dada la medición de variables. Incluso, gran parte de la actividad del CPU mostrado por estas graficas se deben a otros procesos que se encuentran ejecutando dentro del mismo equipo.

### Prueba # 4

Esta prueba consistió en monitorear el estado del clúster en un estado de actividad nula, para ver cuánto contribuían los procesos de la herramienta Ganglia al comportamiento del sistema en general.

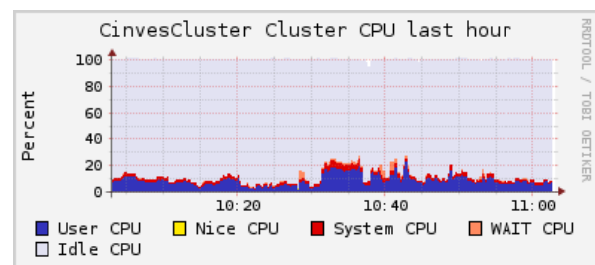


Figura 35. Estado del clúster en la última hora.

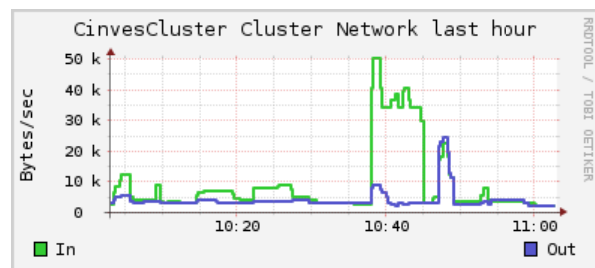


Figura 36. Estado de la red en la última hora.

La figura 35 muestra el estado del clúster en los últimos 5 minutos donde se ha tenido actividad, la utilización es menor al 10% y por lo tanto es difícil determinar cuánto contribuye la herramienta Ganglia al comportamiento global. Lo mismo sucede en la figura 36, pero en este caso para el estado de la red.

Hemos visto en estas 4 pruebas que el rendimiento del clúster es bastante alto, incluso es difícil de separar el rendimiento de

la herramienta del rendimiento del sistema en total, dado que la carga generada por la herramienta es muy baja.

Los resultados en las 4 pruebas han determinado que la carga es muy baja, y que aun variando los parámetros de muestreo esta carga se mantiene baja, por lo tanto nos ha sido difícil generar más pruebas de rendimiento, ya que hemos visto que todos mantienen un comportamiento similar.

## IX. CONCLUSIONES

La instalación de *Ganglia* de forma manual puede resultar complicada y en ocasiones difícil de completar, tal y como nos sucedió, creemos que hacer uso de la instalación manual solo debe ser considerada por usuarios que deseen controlar al máximo determinados parámetros (como la ruta de instalación). Este método solo es recomendado para usuarios avanzados.

Para instalación con los parámetros por omisión y/o usuarios con poca experiencia en la herramienta recomendamos ampliamente hacer uso de los repositorios, ya sea con algún gestor, o realizando la descarga del paquete desde el sitio adecuado (tal y como el sitio de debían).

Una desventaja de este último método es que estamos sujetos a la distribución de GNU/Linux y que tenemos pocas oportunidades de modificar parámetros de instalación, sin embargo como primera experiencia con la herramienta, es completamente recomendable.

Después de instalar la herramienta, y de tratar con diversas opciones, pudimos verla en funcionamiento, el paso realmente crítico es la instalación, después de ello, la configuración es de acuerdo a las necesidades de la aplicación.

Hemos visto que la herramienta está muy completa, que puede entregar la información en distintos formatos, es fácil de usar, y que para *clusters* con bastantes nodos se vuelve una herramienta indispensable de monitoreo.

La elaboración de pruebas de funcionalidad y de rendimiento nos ha dado una mejor perspectiva de esta herramienta.

Por un lado con las pruebas de funcionalidad hemos visto que es posible monitorear una gran cantidad de parámetros de operación del CPU, memoria, red, entre otros.

Hemos de destacar que el incrementar la frecuencia de muestreo de datos, no nos presenta beneficios significativos, ya que el objetivo del sistema es presentar información estadística (y muchas veces promedio) de los parámetros de funcionamiento de los equipos. De poco o nada nos sirve visualizar picos en la utilización de CPU o memoria, si lo que nos importa es tener un control detallado del rendimiento del sistema.

Por otro lado hemos visto el impacto de esta herramienta en el rendimiento del sistema, podemos decir que para los clientes, el rendimiento del equipo no se afecta debido a la configuración de la herramienta, para el servidor y la red, si pudiéramos tener un rendimiento menor si la cantidad de clientes es muy grande.

## X. REFERENCIAS

*Wide Area Cluster Monitoring with Ganglia*  
Sacerdoti, Federico.

*The ganglia distributed monitoring system: design, implementation, and experience*  
Massie, Matthew.

*A Distributed Monitoring Service Architecture*  
H.B. Newman, I.C.Legrand, P. Galvez, R. Voicu, C. Cirstoiu ,  
*MonALISA*.

Wikipedia, cluster,  
[http://en.wikipedia.org/wiki/Cluster\\_\(computing\)](http://en.wikipedia.org/wiki/Cluster_(computing)),  
Visitado ultima vez en 25-07-09

Eric A. Brewer , *Lessons from Giant-Scale Services*,  
IEEE COMPUTER, 2001,  
<http://www.cs.berkeley.edu/~brewer/Giant.pdf>

IBM developer Works : Wikis – AIX – ganglia,  
<http://www.ibm.com/developerworks/wikis/display/WikiPtype/ganglia>  
Visitado por última vez en 24-07-09

Ganglia Monitoring System >> What is Ganglia?  
<http://ganglia.info/>,  
Visitado por última vez en 24-07-09

Repositorio de paquetes para Debian de Ganglia-Monitor.  
<http://packages.debian.org/sid/i386/ganglia-monitor/download>  
Visitado por última vez en 01-08-09

Repositorio de paquetes para Debian de gmetad.  
<http://packages.debian.org/sid/i386/gmetad/download>  
Visitado por última vez en 01-08-09

Guía para la instalación de Ganglia en Debian.  
<http://wiki.freaks-unidos.net/ganglia-quickstart-es>  
Visitado por última vez en 31-07-09

Manual para instalación de Ganglia.  
[http://www.cecalc.ula.ve/HPCLC/slides/day\\_06/Monitoring/Exercises/Monitoring/Practica\\_de\\_GANGLIA\\_TOOLKIT.pdf](http://www.cecalc.ula.ve/HPCLC/slides/day_06/Monitoring/Exercises/Monitoring/Practica_de_GANGLIA_TOOLKIT.pdf)

Instalación y configuración de Ganglia en en Debian.  
[http://wiki.lsc.de.uba.ar/index.php/Instalaci%C3%B3n\\_y\\_configuraci%C3%B3n\\_de\\_Ganglia\\_en\\_Debian](http://wiki.lsc.de.uba.ar/index.php/Instalaci%C3%B3n_y_configuraci%C3%B3n_de_Ganglia_en_Debian)  
Visitado por última vez en 31-07-09