Reporte técnico: Implementación y creación de un sistema de reporteo de información de uso utilizando bitácoras del sistema Torque/PBS y Ganglia.

Carlos E. López Natarén, J. Ismael Fernández Martínez y Alberto Cano Ortiz

September 25, 2015

Introducción

Durante ya poco más de una década, una buena parte de la planta de investigadores y estudiantes del instituto de física han utilizado los clusters de alto rendimiento, conforme hemos adquirido experiencia en el uso y mantenimiento de estos se ha encontrado que es necesario el uso de estadísticas de uso para tomar mejores decisiones respecto a áreas de oportunidad y de crecimiento de la infraestructura de cómputo del instituto. Es por eso que se ha utilizado desde hace un año Ganglia, con esta hemos colectado información que nos dan un perfil del uso del cluster durante el transcurso de este año, además ya que la gran mayoría de los trabajos enviados al cluster es por medio de Torque/PBS (ver http://www.fisica.unam.mx/clusters/mingus_manual.php para el manual de cómo enviar trabajos en este cluster) hemos decidido hacer una versión "in house" de una herramienta que nos permita extraer información importante en forma de estadísticas, más granulares y complementarias a las que nos proporciona el propio Ganglia, hemos llamado a este proyecto Statsganglia.

Ganglia

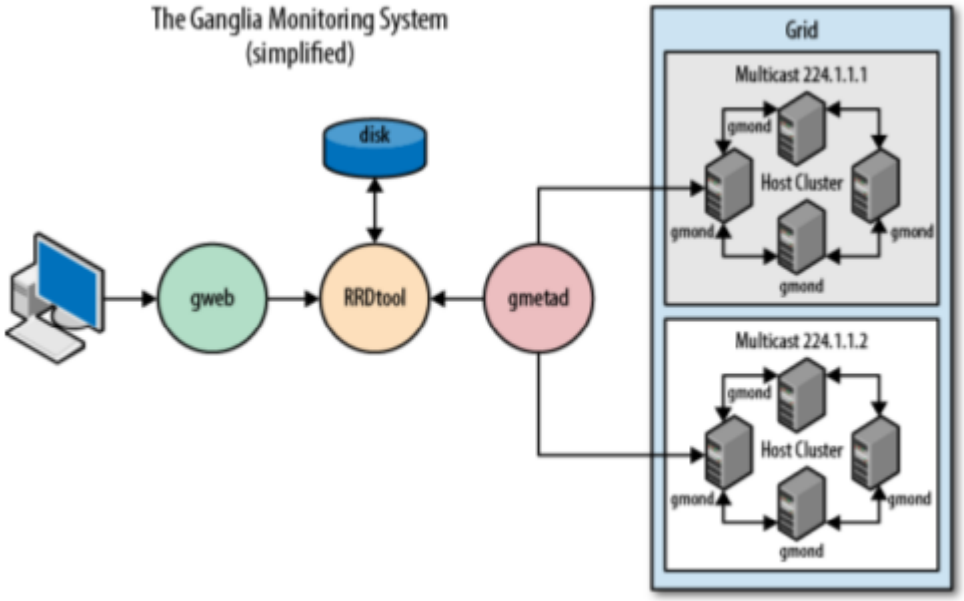
Ganglia ofrece datos de medidas de rendimiento y un monitoreo rutinario en tiempo real para redes de computadoras. Ganglia corre en todos los sitemas opertivos populares, es facilmente escalable para redes muy grandes debido a su diseño y resistente a errores. Ganglia fue diseñado para supervisar y recolectar cantidades masivas de medidas de un sistema en tiempo real.

Ganglia sirve cuando:

- Se tiene un número de computadoras con un sistema operativo de uso general y se requiere información de su desempeño en tiempo real. Asi mismo Ganglia puede monitorear componentes de una red como routers y switches.
- Los hosts puedan ser organizados en grupos.
- El sistema operativo y red no sean hostiles ante el protcolo UDP.

Ganglia esta compuesto de 3 demonios: gmond, gmetad y gweb. Operacionalmente cada demonio es autonomo, necesitando solo su propio archivo de configuración para operar. Cada uno de ellos comenzara y correra aun con la aucensia de alguno de los otros dos, aunque los tres demonios con copperativos. Además se requieren los tres para tener una instalación útil.

Toda la información es intercambiada en XML y XDR, esto asegura maxima extensibilidad y portabilidad.



- gmond (Ganglia Monitoring Daemon):

  Es un demonio multithread el cual corre en cada nodo perteneciente al cluster que se desea monitorear. Gmond es responsable de la interacción con el sistema operativo anfitrion para adquirir medidas sobre metricas tales como la carga del CPU y la capacidad de disco. Gmond tiene un diseño modular, contando con plug-ins especificos del sistema operativo escritos en C para tomar mediciones. Gmond viene incluido con la mayoria de las metricas que puedan interesar. Es importante hacer notar que se pueden añadir plugins para extender las metricas que se monitorean. Gmond no espera que a que algo externo le diga cuanto tomar medidas, ni pasa los resultados directamente a encuestador centralizado, en lugar de eso gmond encuesta de cauerdo con su propio calendario, como se encuentra definido en su archivo local de configuraci´on. Las medidas son compartidas con los pares (peers) usando un simple protcolo via XDR (External Data Representation), estos anuncios son hechos via multicast (el cluster esta compuesto por hosts que comparten la misma dirección multicast). Gmond solo hace multicast de una metrica que esta monitoriando por dos razones:

  - Un cambio en el valor de la metrica excede el valor del umbral. El umbral nos asegura que gmond solo haga multicast cuando es realmente necesario.
  - No ha hecho un multicast de la metrica en cierto tiempo.

  Estas restricciones reducen la saturación del canal multicast. Cada gmond host debe guardar las metricas que recibe de sus pares (peers). Con esto se puede pedir un XML con el estado copleto del cluster. Aunque gmond solo compartira información de los hosts especificados en el parametro trusted_hosts en el archivo de configuración (gmond.conf). El puerto 8649 es el puerto por defecto para para escuchar por peticiones XML (sobre TCP). Se puede ver una descripción usando telenet:

$telnet localhost 8649.

Gmond se auto-organiza y asegura que todos los gmonds esten en sicronia, con esto se solo se tiene que consultar un nodo por cluster para conocer el estado completo del cluster.

Las configuraciones por defecto trabajaran en la mayoria de los clusters. Ademas gmond es muy fexible y altamente personalizable.

- gmetad (Ganglia Meta Daemon):

Gmetad extrae la descripcion XML de fuentes de datos de ganglia (gmonds o incluso gmetads) via rutas unicast. El comportameinto de gmetad es controlado por un archivo de configuración (gmetad.conf). Gmetad es el back-end de del front-end de ganglia. Gmetad almacena la información en bases de datos Round-Robin, esta información es usada por el front-end (resumenes exportados en XML).

- gweb (Ganglia Web Frontend):

El frontend es escrito en PHP. Las paginas web son muy dinamicas cualquier cambio en la infromación de Ganglia aparece inmediatamente en el sitio (Sitio muy responsivo). El front-end depende de la existencia de Gmetad el cual le provee de la información de varias fuentes Ganglia. El front-end espera recibir un arbol XML de Ganglia por el puerto 8651.

La mayoria de los paarmaetros de configuracion se encuentran en gmetad-webfrontend/conf.php. Las partes estaticas de sitio pueden ser modificadas, es decir hay elemntos como etiquetas, links, imagenes, etc pueden ser modificadas. Existe una variable $template_name donde se nombra un directorio dond ese encuentra el actual tema (Ganglia usa TemplatePower para implementar temas). Ademas se pueden modificar valores como los rangos de fechas o que metricas desplegar.

Torque/PBS (Carlos: Descripción de cómo funciona en general)

El uso de Torque/PBS ha sido documentado en varias partes del manual de mingus, principalmente en la página de clusters del instituto, en la página de manuales (http://www.fisica.unam.mx/clusters/mingus_manual.php la configuración actual del scheduler es la siguiente:

```
Max open servers: 10239
Qmgr: p s
# # Create queues and set their attributes.
#
#
# Create and define queue medium
#
create queue medium
set queue medium queue_type = Execution
set queue medium max_user_queuable = 174
set queue medium resources_max.ncpus = 176
set queue medium resources_max.nodect = 200
set queue medium resources_min.cput = 00:00:01
```

```
set queue medium resources_min.ncpus = 1
set queue medium resources_min.nodect = 1
set queue medium resources_min.walltime = 00:00:01
set queue medium resources_default.cput = 1920:00:00
set queue medium resources_default.ncpus = 1
set queue medium resources_default.neednodes = medium
set queue medium resources_default.nodect = 1
set queue medium resources_default.walltime = 24:00:00
set queue medium resources_available.ncpus = 176
set queue medium resources_available.nodect = 200
set queue medium max_user_run = 174
set queue medium enabled = True
set queue medium started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_hosts = mingus
set server managers = root@mingus
set server operators = root@mingus
set server default_queue = medium
set server log_events = 511
set server mail_from = adm
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 6
set server mom_job_sync = True
set server keep_completed = 60
set server next_job_number = 40953
set server record_job_info = True
set server record_job_script = True
set server job_log_keep_days = 780
```

Archivos de bitácoras de Torque/PBS (Carlos)

R/Shiny (Ismael: Cómo funciona)

Shiny es un paquete de R que hace fácil el crear aplicaciones web interactivas llamadas apps directamente desde R. Para instalar shiny se debe abrir una sesión en R, estar conectado a internet y correr:

- *install.packages("shiny")*

Las apps de Shiny están compuestas por:

- *ui.R*: un script usuario-interfaz (controla el diseño y la apariencia de la app).

- *server.R*: un script servidor (contiene las instrucciones que la computadora necesita para construir la app).

Es importante notar que se puede añadir contenido HTML a las apps de Shiny. Se puede crear una app de Shiny al crear un nuevo directorio y guardar los archivos *ui.R* y *server.R* en él. Para ejecutar una app se debe dar el nombre de su directorio como argumento a la funcion *runApp*, por ejemplo si nuestra aplicación se encuentra en el directorio llamado *my_app,* la ejecutariamos desde una sesión de R de la siguiente manera:

1. *library(shiny)*
2. *runApp("my_app")*

Una app de Shiny es simplemente una aplicación web creada en R. Shiny esta basado en un modelo de programación reactivo, similar al de las hojas de cálculo. Con Shiny se pueden hacer ánalisis de datos reactivos, accesibles a cualquiera con navegador web.

Creación de las estadísticas generales (Ismael)

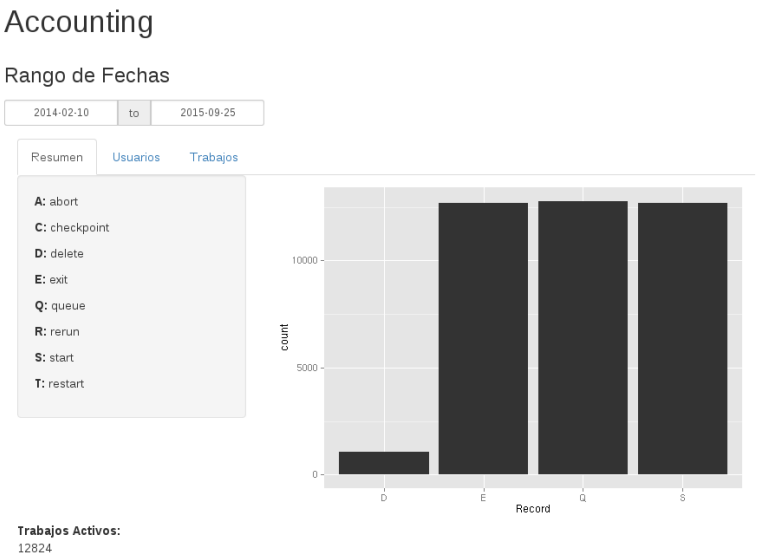
Primero se limpio la base datos, es decir se creo un tabla, donde cada renglón reprensenta un trabajo y toda su información, es decir toda la información obtenida de los archivos de bitácoras. Para realizar esto se contruyeron expresiones regulares:

- *kNameExpression <- "[a-zA-Z.0-9\|_\|\-@]+"* (Expresión para nombres)
- *kFileExpression <- "[a-zA-Z.0-9\|_\|\-\|\/@]+"* (Expresión para nombres de archivos y directorios)
- *kNumberExpression <- "[0-9]+"* (Expresión para números)
- *KTimeExpression <- "[0-9]+:[0-9]+:[0-9]+"* (Expresión para cantidades de tiempo)
- *KMemAmountExpression <- "[0-9]+(m|M|k|K|g|G|t|T)*(b|B)"* (Expresión para cantidades de memoria)

Las cuales se usuaron para extraer los campos *Jobid, Jobname, User, Group, Owner, Queue, Start, End, ctime, qtime, etime, execHost, listCPUT, listMem, listNCPUs, listNeedNode, listNodeCT, listWalltime, session, exitStatus, usedCPUT, usedMem, usedVMem, usedWalltime, requestor, dtime.* Debido a que el *Jobid* es el identificador único para los trabajos se uso como llave para manejar la información, es decir manipular la base de datos.

Una vez limpios los datos se diseño una interfaz para desplegar la información de manera interactiva (*ui.R*) y se contruyeron las funciones necesarias para manipular la información en *server.R* y la funciones para realizar cálculos y graficas en *utils.R*. Se hizo uso de expressiones reactivas para prevenir calculos innecesarios y dar mayor rápidez y fluidez a la aplicación.

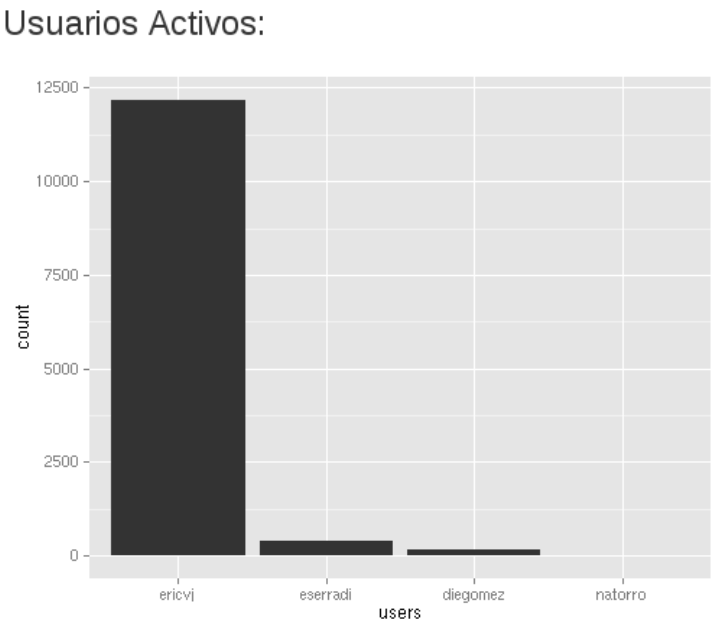
En la interfaz se contruyo un panel con tres pestañas: *"Resumen", "Usuario", "Trabajo".*



La información desplegada en esos paneles será en función de los rangos de fecha seleccionados en la parte superior.

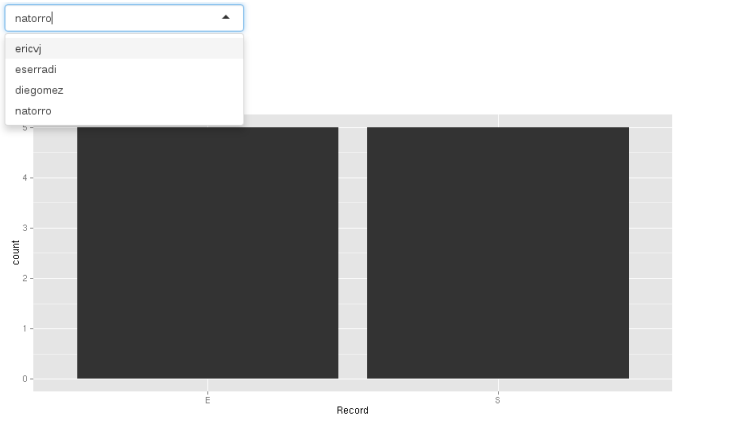
- Resumen:

La pestaña resumen muestra los tipos mensajes de las bitácoras en el rango de fecha especificado, además muestra el número de trabajos activos en ese periodo y un histograma de los usuarios activos en ese periodo.



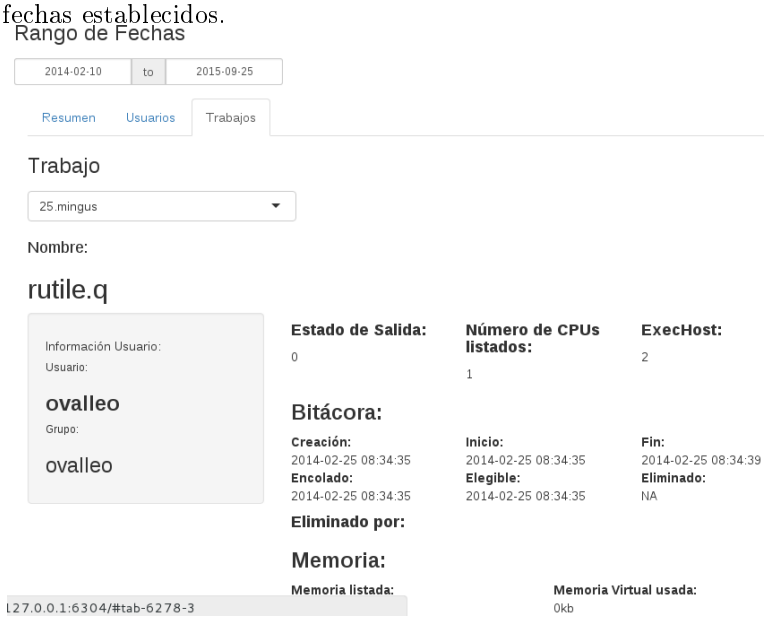
- Usuarios:

La pestaña usuarios muestra la actividad de cada usuario, el usuario es elegido en un selector y muestra los mensajes asociados a cada usuario, además de el número de trabajos pertenecientes a ese usuario, así como toda la información de cada trabajo de ese usuario. Los usuarios disponibles en el selector, asi como la información despeglada de cada usuario van en función de los rangos de fechas seleccionados.



- Trabajos:

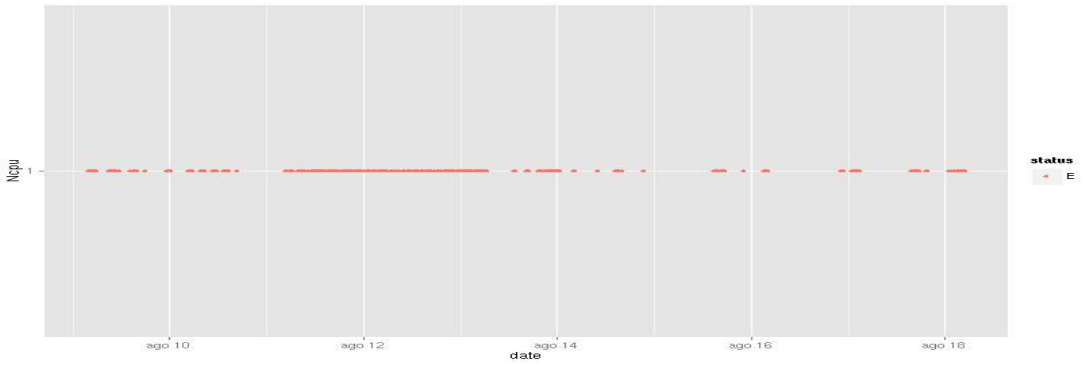
La pestaña Trabajos muestra la información de cada trabajo, en la parte superior se encuetra un selector el cual nos permite selectionar todos los trabajos si su actividad se encuentra en el rango de fechas establecidos.



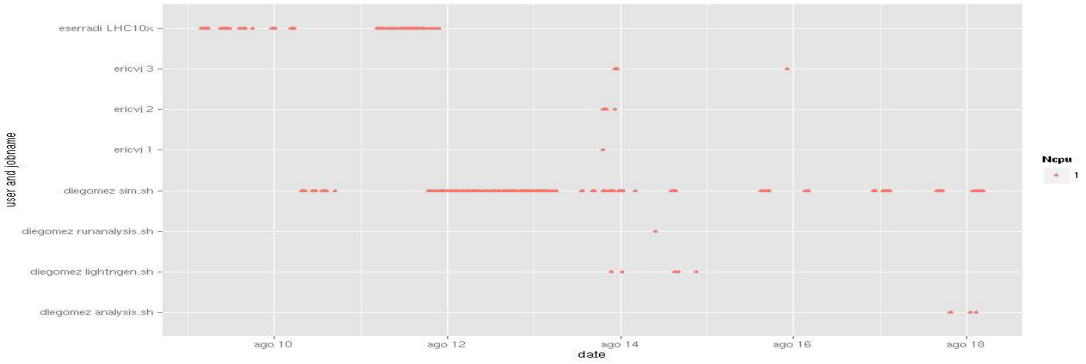
Creación de las estadísticas particulares por usuario (Alberto)

Despues de haber limpiado partes claves del mensage que es arrojado en cada unos de los registros log del calendarizador se crearon las sieguientes series de tiempo.

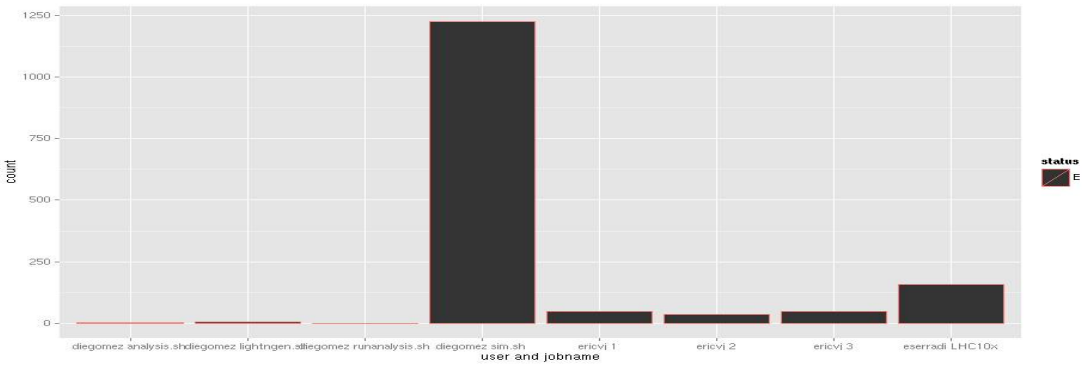
Fue creada una serie de tiempo de los nodos, donde estos ademas son agrupados por el estado en el que se encuentran; se espera que facilite el analisis de frecuencia de fallos en cada unos de los nodos.



Fue creada una serie de tiempo de los trabajos por usuario, donde ademas estos mismos son agrupados por el estado en el que se encuentran, mostrara en cada punto la frecuencia de los fallos.



fue creado un histograma que muestra el numero de trabajos de cada usuario agrupado por el estado del trabajo del usuario, complementa la información de la serie de tiempo pasada.


1250
1000
750
500
250
0
count
diegomez analysis.shdiegomez lightngen.sdiegomez runanalysis.sh diegomez sim.sh ericvi 1 ericvi 2 ericvi 3 eserradi LHC10x
user and jobname
status
E