Facultad de Ingeniería

Universidad de la República

# Proyecto Fenton - Cluster de Computadores de Alto Desempeño con Acceso Remoto (CCADAR)

## Manual de instalación

Julio 2008

Estudiantes:
Santiago Iturriaga, Paulo Maya, Damián Pintos

Supervisor:
Mg. Ing. Sergio Nesmachnow

**Resumen**

Que contiene el documento, en que consiste el proyecto...

# Índice

# 1. Sistema operativo

Se recomienda utilizar Linux 2.6 (o superior), con una arquitectura de 32-bits (x86) o 64-bits (x64). Si bien el sistema fue probado en SuSE, Ubuntu y Fedora, esto no quita que pueda realizarse una instalación en otras distribuciones (Debian, Slackware, etc.) o hasta otro sistema operativo de la familia POSIX (FreeBSD, Solaris, etc).

Durante le resto del documento se asumirá que se utiliza Linux.

# 2. Pre-instalación

**Usuario y grupo del sistema.** El sistema necesita contar con un usuario y un grupo para ejecutar. Este usuario del sistema debe ser configurado como administrador en algunos de los programas utilizados (p.ej.: Maui, TORQUE, etc.). Por defecto este usuario y este grupo se llamarán fenton.

Para crearlos es debemos ejecutar:

```
root:# addgroup fenton
root:# adduser fenton
root:# adduser fenton fenton
```

# 3. Requerimientos de software

A continuación detallaremos el software requerido por el sistema. Por simplicidad es recomendable realizar la instalación del software utilizando los paquetes de la distribución de Linux sobre la que se esté trabajando (apt-get, yast2, etc.).

A continuación veremos un breve instructivo de como realizar la instalación compilando e instalando manualmente utilizando los fuentes.

## 3.1. PostgreSQL 8.2.5 o superior [requerido]

PostgreSQL[1] es un servidor de base de datos relacional liberado bajo la licencia BSD. El sistema utiliza PostgreSQL para persistir la información sobre usuarios, trabajos, etc.

Para compilar e instalar PostgreSQL en `/usr/local/postgresql-8.2.5` se deben seguir los siguientes pasos:

```
$ ./configure --prefix=/usr/local/postgresql-8.2.5
$ make
root:# make install
```

Luego se debe crear el usuario de PostgreSQL y se debe asignar espacio físico para los datos y logs.

```
root:# adduser postgres
root:# mkdir /usr/local/postgresql-8.2.5/data
root:# chown postgres /usr/local/postgresql-8.2.5/data
postgres:$ cd /usr/local/postgresql-8.2.5/bin
postgres:$ ./initdb -D /usr/local/postgresql-8.2.5/data
```

Finalmente iniciamos el servicio:

```
postgres:$ cd /usr/local/postgresql-8.2.5/bin
postgres:$ ./pg_ctl -D /usr/local/postgresql-8.2.5/data
```

El sistema requiere que la base de datos debe sea accesible desde todos los nodos del cluster (tanto el nodo maestro como los nodos de computo). Para habilitar el acceso externo a la base de datos se deben editar los siguientes archivos de configuración:

```
/net/local/postgresql-8.2.5/data/pg_hba.conf
/net/local/postgresql-8.2.5/data/postgresql.conf
```

## 3.2. Apache 2.0 o superior [requerido]

El servidor HTTP Apache[2] es un servidor HTTP de código abierto. Fenton utiliza un interfaz de usuario web para la ejecución de trabajos y la administración del cluster por lo que es necesario contar con un servidor HTTP.

En la actualidad todas las distribuciones de Linux disponen de un paquete de Apache 2.0 o superior. Recomendamos utilizar este paquete para la instalación.

## 3.3. PHP 5.2.5 o superior [requerido]

PHP[3] es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Fenton se encuentra casi completamente implementando en PHP y requiere que este se encuentre instalado en todos los nodos del cluster. En el nodo maestro es utilizado para la ejecución de la interfaz web y la lógica del sistema. En los nodos de computo es utilizado para ejecutar scripts de pre-ejecución y post-ejecución de trabajos en el cluster.

Para compilar e instalar PHP en el directorio `/usr/local/php-5.2.5`:

```
$ ./configure --prefix=/usr/local/php-5.2.5 \
> --with-pgsql=/usr/local/postgresql-8.2.5 --with-gd
$ make
root:# make install
```

Es necesario incluir soporte para PostgreSQL y para la biblioteca gráfica GD (utilizada por Ganglia). Por último es necesario configurar el servidor HTTP Apache para poder ejecutar archivos PHP.

## 3.4. Ganglia 3.0.5 o superior [requerido]

Ganglia es utilizado para monitorear el estado del cluster: carga de memoria de los nodos, tráfico de red, utilización de CPU, etc.

Ganglia cuenta con tres componentes que deben ser instalados:

### 3.4.1. gmond

Gmond es un demonio encargado de recolectar y enviar información del estado del nodo sobre el que esta ejecutando. Por esta razón debe correr en cada una de los nodos del cluster que se desea monitorear.

Para compilar e instalar gmond se debe ejecutar:

```
$ ./configure --prefix=/usr/local/ganglia-3.0.7
$ make
root:# make install
```

Dentro del directorio `ganglia-3.0.7-src/gmond` podremos encontrar algunos scripts de ejemplo para ejecutar gmond al inicio del sistema como un servicio.

Por último es necesario configurar gmond. Para esto creamos el archivo de configuración `/etc/gmond.conf` con valores por defecto ejecutando:

```
root:# cd /usr/local/ganglia-3.0.7/sbin
root:# ./gmond --default_config > /etc/gmond.conf
```

Una vez creado con valores por defecto podemos configurarlo a gusto.

### 3.4.2. gmetad

Gmetad es un demonio al igual que gmond. Pero en lugar de enviar datos, oficia de servidor recolectando la información enviada por gmond desde los nodos del cluster. Solamente una instancia de este demonio debe estar en ejecución. Para compilar e instalar gmetad y gmond:

```
$ ./configure --prefix=/usr/local/ganglia-3.0.7 --with-gmetad
$ make
root:# make install
```

Dentro del directorio `ganglia-3.0.7-src/gmetad` podremos encontrar algunos scripts de ejemplo para iniciar gmetad como servicio al inicio del sistema. A diferencia de gmond, gmetad viene además con un archivo de configuración por defecto que se encuentra en `ganglia-3.0.7-src/gmetad/gmetad.conf` y que debemos copiar a `/etc/gmetad.conf` y luego editar.

Finalmente debemos asignar espacio en disco en donde gmetad guardará la información que recopile. La ubicación por defecto es `/var/lib/ganglia/rrds` y su owner debe ser el usuario que fue configurado en `gmetad.conf` como usuario de ejecución.

### 3.4.3. Interfaz web

Ganglia cuenta con una interfaz web implementada en PHP donde se presenta visualmente la información del cluster. La interfaz web de Ganglia debe estar alojada en el mismo nodo que ejecuta el demonio gmetad.

Para realizar la instalación de la interfaz web es necesario copiar el directorio `ganglia-3.0.7-src/web` al directorio `DocumentRoot` de su instalación del servidor Apache, por ejemplo: `/var/www/ganglia`.

## 3.5. TORQUE 2.3.0 o superior [requerido]

TORQUE[5] es un manejador de recursos distribuidos (DRM) y es utilizado para administrar los recursos disponibles en el cluster: procesadores, memoria, tiempo de cómputo, etc. La arquitectura de un cluster TORQUE cuenta con un nodo maestro y muchos nodos de cómputo. El nodo maestro debe ejecutar el demonio `qserverd` y los nodos de cómputo deben ejecutar el demonio `qnoded`. A continuación veremos como instalar estos demonios.

### 3.5.1. Nodo maestro: qserverd

Para instalar el servidor de TORQUE en `/usr/local/torque-2.3.0`:

```
$ ./configure --prefix=/usr/local/torque-2.3.0
$ make
root:# make install
```

Una vez que termina la instalación del servidor, es necesario configurar el demonio qserverd:

```
root:# export PATH=$PATH:/usr/local/torque-2.3.0/bin
root:# export PATH=$PATH:/usr/local/torque-2.3.0/sbin
root:# torque.setup fenton
```

Donde `fenton` es el usuario que se desempeñará como administrador de TORQUE. Este script se encarga de crear un entorno básico de trabajo con una configuración por defecto para poder iniciar el servidor. Finalmente se deben configurar los nodos de computo editando `/var/spool/torque/server_priv/nodes`, p. ej.:

```
nodo001.fing.edu.uy
nodo002.fing.edu.uy
nodo003.fing.edu.uy
```

O mediante la interfaz de consola de torque:

```
$ qmgr
qmgr: create node nodo001.fing.edu.uy
```

### 3.5.2. Nodo de cómputo: qnoded

Una vez compilado torque es posible crear paquetes de instalación distribuibles para instalar en los nodos de cómputo. Esto simplifica la instalación de los nodos de computo si se trata de un cluster homogéneo, en cambio si los nodos del cluster cuentan con sistemas operativos diferentes o arquitecturas diferentes será necesario compilar torque para cada nodo. Asumiremos que se trata de un cluster homogéneo, para crear los paquetes de instalación distribuibles se debe ejecutar:

```
$ make packages
```

Y luego en cada nodo de computo:

```
root:# ./torque-package-clients-linux-i686.sh --install
root:# ./torque-package-mom-linux-i686.sh --install
```

El servidor MOM para cada nodo de computo debe ser configurado para confiar en el servidor maestro de TORQUE editando el archivo:

```
/var/spool/torque/mom_priv/config
```

**Ejecución de los demonios.** Es necesario que el servidor de TORQUE (qserverd) y los demonios qnoded se encuentren en ejecución permanente en el cluster. Para esto es recomendable que sean agregados como demonios en el sistema creando scripts de ejecución en `/etc/init.d`. Para esto existen dos scripts que pueden tomarse de ejemplo:

```
scripts/qserverd.example.sh
scripts/qnoded.example.sh
```

## 3.6. Maui 3.2.6p19 o superior [requerido]

Maui[6] es un despachador (scheduler) de código abierto de trabajos para clusters que soporta varias políticas de despacho, prioridades dinámicas, reservas de recursos, etc.

**Compilación, instalación y configuración.** Existe un bug en el script de configuración e instalación de Maui por lo que recomendamos realizar la instalación en `/usr/local/maui` (su ubicación por defecto). Además es necesario estar logueado con el usuario de Linux que se desempeñará como administrador del scheduler al momento de la compilación. Este usuario debe ser también administrador de TORQUE, por lo que recomendamos utilizar el mismo usuario se asignó durante el paso anterior como administrador de TORQUE (fenton en nuestro ejemplo). Una vez con este usuario, se debe ejecutar:

```
fenton:$ ./configure --prefix=/usr/local/maui \
> --with-pbs=/usr/local/torque-2.3.0
fenton:$ make
root:# mkdir -p /usr/local/maui
root:# chown fenton.fenton /usr/local/maui
fenton:$ make install
```

Es posible que ocurra un error durante la creación de directorios en la instalación. Si sucede esto es necesario crear manualmente los directorios: 'log', 'traces', 'stats', 'spool' y 'tools' en el directorio de instalación y luego volver a ejecutar `make install`.

Debemos asegurarnos que el usuario administrador de Maui tenga permisos totales sobre el directorio de instalación. Una posibilidad para asegurar esto es asignarlo como owner ejecutando:

```
root:# chown -R fenton /usr/local/maui
```

Luego de esto damos por finalizada la instalación. Ahora es necesario configurar el scheduler editando `/usr/local/maui/maui.cfg`:

```
RMCFG[SERVIDOR.FING.EDU.UY] TYPE=PBS
```

Para clusters pequeños quizás sea recomendable disminuir el tiempo de polling para tener una respuesta mas rápida por parte del scheduler modificando el parámetro `RMPOLLINTERVAL`, p.ej.:

```
RMPOLLINTERVAL 00:00:05
```

**Ejecución del demonio.** Al igual que TORQUE, también es necesario que Maui se encuentre en ejecución constante en el cluster. Y de la misma forma que antes, también existe un script que puede tomarse de ejemplo para esto en: `scripts/maui.example.sh`.

## 3.7. MPI [opcional]

Para ejecutar programas paralelos utilizando MPI es necesario contar con una implementación del estándar. A continuación veremos las implementaciones que fueron probadas en el sistema.

Cualquiera sea la implementación elegida, es necesario que sea instalada (o se encuentra accesible) en todos los nodos del cluster.

### 3.7.1. MPICH 1.2.7p1 o superior

MPICH1[7] es una implementación del estándar MPI versión 1. Para compilar e instalar MPICH1 en `/usr/local/mpich-1.2.7p1` se debe ejecutar:

```
$ ./configure --with-device=ch_p4 \
> --prefix=/usr/local/mpich-1.2.7p1 \
> --with-common-prefix=/usr/local/mpich-1.2.7p1
$ make
root:# make install
```

### 3.7.2. Mpiexec 0.83 o superior

Mpiexec[8] es un programa que reemplaza al script mpirun del paquete MPICH y es utilizado para inicializar un trabajo paralelo desde TORQUE. Si bien no es obligatorio utilizar Mpiexec, este facilita la integración de MPICH con TORQUE y es muy recomendable utilizarlo.

```
$ ./configure --prefix=/usr/local/mpiexec-0.83 \
> --with-default-comm=mpich-p4 \
> --with-pbs=/usr/local/torque-2.3.0 \
> --with-mpicc=/usr/local/mpich-1.2.7p1/bin/mpicc
$ make
root:# make install
```

NOTA: Mpiexec no es necesario si se utiliza OpenMPI.

### 3.7.3. OpenMPI 1.2.6 o superior [recomendado]

OpenMPI[9] es una implementación del estándar MPI versión 2. Para compilar e instalar OpenMPI se debe ejecutar:

```
$ ./configure --prefix=/usr/local/openmpi-1.2.6 \
> --with-tm=/usr/local/torque-2.3.0
> --disable-shared --enable-static
$ make
root:# make install
```

Compilar OpenMPI de forma estática tiene la ventaja de que evitamos problemas de bibliotecas en los diferentes nodos. Aunque tiene dos desventajas: los ejecutables tienen un tamaño mayor y si las bibliotecas de OpenMPI son actualizadas es necesario re-compilar los proyectos y generar nuevos ejecutables para utilizar la nueva versión.

# 4. Instalación de Fenton

## 4.1. Repositorio del sistema

Es necesario crear el repositorio raíz donde el sistema almacenará la estructura de clientes y trabajos para que los usuarios realicen ejecuciones y almacenen sus resultados. Por ejemplo, creamos el repositorio en `/home/fenton/repositorio` y luego le asignamos los permisos adecuados.

```
fenton:$ mkdir -p /home/fenton/repositorio
fenton:$ chmod g+w+r /home/fenton/repositorio
```

El grupo del directorio debe ser `fenton`. En caso de que sea otro debemos cambiarlo con el comando:

```
root:# chgrp fenton /home/fenton/repositorio
```

Finalmente debemos crear un espacio para el repositorio interno del sistema. Aquí el sistema almacenará logs de ejecuciones y otros datos que no estarán directamente disponibles a los usuarios. La ubicación recomendada para este directorio es `/home/fenton/repositorio/sistema`. Para crearlo:

```
fenton:$ mkdir -p /home/fenton/repositorio/sistema
fenton:$ chmod g+w+r /home/fenton/repositorio/sistema
```

Al igual que en el directorio anterior, debemos asegurarnos que el grupo del directorio sea `fenton`.

El usuario y el grupo `fenton` deben existir en todos los nodos del cluster. Y ambos directorios también deben ser accesibles desde cualquier nodo del cluster.

Debido a que el repositorio del sistema y la web (como veremos mas adelante) se encuentran en el directorio home del usuario `fenton`, debemos verificar que cualquier usuario tenga acceso de lectura al directorio `/home/fenton`.

```
fenton:$ chmod a+r+x fenton
```

## 4.2. Acceso SSH con autenticación RSA

En el nodo de ejecución de la aplicación web, debe ser posible realizar un SSH sin password desde el usuario que ejecuta la aplicación web (el usuario configurado en el servidor Apache) al usuario del sistema (por defecto `fenton`) en el nodo maestro.

El primer paso para configurar este acceso es averiguar el usuario de ejecución de los scripts en Apache. Este usuario varía de distribución en distribución. En nuestro sistema es `www-data`.

Luego debemos asignar a este usuario un directorio home. Para esto es necesario editar `/etc/passwd`:

```
www-data:x:33:33:www-data:/home/www-data:/bin/sh
```

De esta manera asignamos el directorio `/home/www-data` como home del usuario. Seguramente este directorio no exista, por lo que deberemos crearlo y asignarle `www-data` como owner.

A continuación creamos la clave publica y la privada, estas claves son necesarias para la autenticación:

```
root:# su - www-data
www-data:$ ssh-keygen -t dsa
```

Finalmente debemos agregar a las claves publicas del usuario `fenton` la recién creada clave de `www-data`.

```
fenton:$ mkdir -p /home/fenton/.ssh
root:# cd /home/fenton/.ssh
root:# cat /home/www-data/.ssh/id_dsa.pub > authorized_keys2
root:# chown fenton.fenton /home/fenton/.ssh/authorized_keys2
```

Es necesario probar al menos una vez el acceso SSH al nodo maestro para agregar el nodo maestro a los hosts conocidos de SSH. Suponiendo que el servidor apache se encuentra en el nodo maestro ejecutamos:

```
www-data:$ ssh fenton@localhost
```

## 4.3. Base de datos

Es necesario crear un usuario para el sistema en la base de datos, asignarle una contraseña y crear las tablas de datos. A continuación detallaremos como crear todo esto desde la linea de comando. Como primer paso debemos crear el usuario de la base de datos:

```
root:# su - postgres
postgres:$ createuser fentondb
```

Luego creamos el esquema de datos y las tablas:

```
$ createdb fentondb
$ psql fentondb
psql=# \i postgres/database.sql
```

Finalmente asignamos una contraseña al nuevo usuario:

```
psql=# alter user fentondb password 'fentondb';
```

## 4.4. Aplicación web

Para instalar la aplicación web simplemente debemos copiar el directorio `web/` a la ubicación deseada. Por ejemplo `/home/fenton/web`, y luego crear un alias para la aplicación. En la mayoría de los casos basta con copiar el archivo `apache/fenton.conf` a `/etc/apache2/conf.d`.

La aplicación web debe ser accesible desde cualquier nodo del cluster. Esto es necesario debido a que los scripts de prologo y epilogo (ver siguiente paso) requieren ejecutar scripts PHP de la aplicación.

Es necesario que el sistema cuente con un interprete de linea de comando de PHP y que este se encuentre en `/usr/bin/php`. Existen varios scripts PHP que se encuentran en `/home/fenton/web/bin` y que esperan encontrar el interprete en la hubicación mencionada. En caso de que el inteprete de PHP no se encuentre en esa hubicación o tenga otro nombre (p.ej. `/usr/bin/php5`) puede solucionarse facilmente creando un link al nombre deseado. En nuestro ejemplo sería ejecutando:

```
ln -s /usr/bin/php5 /usr/bin/php
```

Por último debemos configurar el script de control de cuota para que ejecute periodicamente en el nodo maestro. Para esto podemos utilizar el comando cron de Linux.

El script de control se encuentra en `/home/fenton/web/lib/Vigilante.php` y para ejecutarlo periódicamente debemos editar la confguración del cron ejecutando:

```
root:# crontab -e
```

Este comando nos abrirá para edición el archivo de configuración del cron del usuario. Supongamos que se quiere ejecutar el script de control una vez cada hora, para esto editamos la configuración agregando la siguiente linea:

```
01  * * * * php /home/fenton/web/lib/Vigilante.php
```

## 4.5. Configuración de TORQUE

Es necesario que TORQUE notifique al sistema cuando un trabajo es sacado de la cola de espera para iniciar su ejecución, y cuando un trabajo termina de ejecutar (ya sea de forma exitosa o no). Para esto debemos editar y copiar el script de pre-ejecución (prologo) y el script de post-ejecución (epilogo). Estos scripts deben estar disponibles en todos los nodos del cluster (tanto el maestro como los nodos de cómputo).

Los scripts se encuentran en `torque/prologue` y `torque/epilogue`, y se deben copiar a `/var/spool/torque/mom_priv`. Luego de copiados es necesario editar estos scripts configurando la variable `PATH_PHP` según nuestra instalación, p. ej.: `/home/fenton/web/bin`.

## 4.6. Configuración final del sistema

Como último paso, debemos realizar los ajustes finales y configurar la aplicación web con la ubicación de los diferentes componentes del sistema. Para esto debemos editar el archivo de configuración `/home/fenton/web/lib/Constantes.php`.

Si toda la instalación fue realizada utilizando las ubicaciones sugeridas las modificaciones al archivo de configuración deberían ser mínimas. A continuación veremos algunos de los parámetros de configuración:

- Ubicación del repositorio y de los archivos temporales.

```
define("RAIZ","/home/fenton/repositorio\char");
define("RAIZ_SISTEMA","/home/fenton/repositorio/sistema");
define("TMP", "/tmp");
```

- Conexión a la base de datos.

```
define("CONEXION_HOST","localhost");
define("CONEXION_PORT","5432");
define("CONEXION_USUARIO","fentondb");
define("CONEXION_PASSWORD","fentondb");
define("CONEXION_BASE","fentondb");
```

- Configuración del launcher MPI.

```
define("MPIEXEC","/usr/local/openmpi-1.2.6/bin/mpiexec");
```

- URL de la instalación de Ganglia.

```
define("GANGLIA_URL","http://localhost/ganglia");
```

- Configuración del usuario y el host utilizado por el sistema para ejecutar tareas administrativas.

```
define("SSH","/usr/bin/ssh");
define("USERNAME","fenton");
define("HOST","localhost");
```

- Opciones generales del sistema.

```
define("GRUPOFENTON","fenton");
```

  Grupo al que pertenecen todos los usuarios del sistema.

```
define("REDIRECCION_SALIDA",
 "/home/fenton/web/bin/redireccion_salida.php");
define("OUTPUT","salida\_extra");
define("EJECUTABLE","plantillas/archivos/qsub.script");
define("LOG_EJECUCIONES","../log/ejecuciones.log");
define("TIEMPO_REFRESH_RESULTADOS","5");

define("COMANDOS_EJECUCION",
 "make=make&mpicc=/usr/local/openmpi-1.2.6/bin/mpicc");
```

  Lista de comandos disponibles para la ejecución por parte del usuario del sistema. Deben estar separados entre ellos por un '&', y cada uno esta compuesto de dos partes separadas por un '=': la primer parte es la descripción y la segunda parte es ubicación del comando.

- Configuración de TORQUE y Maui.

```
define("PATH_TORQUE","/usr/local/torque-2.3.0");
define("PATH_MAUI","/usr/local/maui");
define("QSUB",PATH_TORQUE."/bin/qsub");
etc...
```

**Algunas de las opciones que seguramente siempre debamos configurar son:**

```
define("CONEXION_HOST","servidor.fing.edu.uy");
```

Ubicación del host ejecutando la base de datos.

```
define("GANGLIA_URL","http://servidor.fing.edu.uy/ganglia");
```

URL base de la instalación de Ganglia.

# Referencias

[1] PostgreSQL
`http://www.postgresql.org/`

[2] Apache
`http://httpd.apache.org/`

[3] PHP
`http://www.php.net/`

[4] Ganglia
`http://ganglia.info/`

[5] TORQUE
`http://www.clusterresources.com/pages/products/torque-resource-manager.php`

[6] Maui
`http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php`

[7] MPICH1
`http://www-unix.mcs.anl.gov/mpi/mpich1/`

[8] Mpiexec
`http://www.osc.edu/~pw/mpiexec/index.php`

[9] OpenMPI
`http://www.open-mpi.org/`