Revista Educación en Ingeniería
ISSN 1900-8260
Julio a Diciembre de 2013, Vol. 8, N°. 16, pp. 106-124 • © 2013 ACOFI • http://www.educacioneningenieria.org
Recibido: 12/06/2013 • Aprobado: 24/11/2013

VENTAJAS Y APLICACIONES DE LA COMPUTACIÓN DE ALTA DISPONIBILIDAD EN CÓNDOR

ADVANTAGES AND APPLICATIONS COMPUTER OF HIGH THROUGHPUT CÓNDOR

Tatiana Blanco Rojas Frey, Alfonso Santamaría Buitrago
Universidad Pedagógica y Tecnológica de Colombia, Tunja (Colombia)

Resumen

La computación de alta disponibilidad, (High Throughput Computing - HTC), es un modelo de computación distribuida que básicamente busca crear un entorno enfocado a completar el mayor número de trabajos en un amplio periodo de tiempo, lo cual tiene como ventaja el uso eficiente de los recursos disponibles. La HTC es ideal para negocios con modelos orientados a prestar servicios competitivos, su uso también es seguro contra daños como pérdida de ingresos y oportunidades, e insatisfacción de consumidores. La alta disponibilidad como tal provee opciones de negocio por cuanto a menudo surge una creciente demanda por servicios computarizados disponibles las 24 horas del día en diversas áreas, entre ellas bancarias, financieras, de telecomunicaciones y de manejo de recursos. Cóndor es un *software* gestor de trabajos diseñado para aplicar HTC en diversos entornos utilizando adecuadamente los recursos disponibles. A lo largo de la investigación se muestra cómo Cóndor aplica alta disponibilidad factor que resulta ser muy importante en las organizaciones, no como un lujo sino como una necesidad. Igualmente se habla sobre su arquitectura, implementación y despliegue de nodos y trabajos.

Palabras claves: computación de alta disponibilidad, Cóndor, recursos, clúster.

Abstract

High Throughput computing (HTC) is a distributed computing model that basically seeks to create an environment focused on completing the highest number of jobs in a long time taking greater advantage as efficiently use available resources.

Within the HTC applications is ideal for business-oriented models provide competitive services is also safe to use against damage such as loss of income, consumer dissatisfaction and lost opportunities. The high availability itself provides business opportunities and that often come an increasing demand for computer services available 24 hours a day in various areas banking, financial, telecommunications and resource management. Finally Cóndor manager is a software application designed to work within

different environments HTC using resources appropriately, in the investigation is shown as applied Cóndor high availability and how this factor is very important in organizations not as a luxury but as a necessity, also spoke about the architecture, implementation and deployment of nodes and jobs.

Key words: High Throughput Computing (HTC), Cóndor, resources, cluster.

Introducción

La era de la globalización está equipada de útiles herramientas tecnológicas, tanto físicas como digitales, que facilitan la vida de los seres humanos y el desarrollo de sus procesos. Una de ellas es la más accesible para mayoría de personas, el computador, que resulta ser muy útil para el procesamiento de información y la ejecución de diversos trabajos.

Lamentablemente, cuando se desconoce el potencial de estas máquinas se tiende a desperdiciar recursos, espacio y tiempo de ejecución, lo que conduce al mal uso de los equipos.

Numerosas instituciones y organizaciones afrontan esta problemática de despilfarro de recursos propios e invierten en nuevos equipos sin observar que, al optimizar los que poseen, ahorran y los aprovechan para el desarrollo adecuado de los procesos, realizar investigaciones o aportar nuevo conocimiento a la sociedad.

Sin embargo, existen múltiples soluciones diseñadas para aprovechar al máximo los recursos disponibles de un equipo como la computación de alta disponibilidad, High Throughput Computing (HTC), que es un modelo de computación distribuida creado para completar el mayor número de tareas en un largo periodo de tiempo usando eficientemente los recursos disponibles [1]. En la actualidad suele ser una medida útil y eficaz para usar adecuadamente los recursos dispuestos por las máquinas. Cóndor es un *software* de sistema que crea un entorno de High Throughput Computing y utiliza eficazmente el potencia de cálculo de los computadores conectados a la red. Un usuario envía un trabajo a Cóndor y éste se encarga de buscar una máquina disponible en la red para ejecutarlo. La ventaja tanto del uso de la computación de alta disponibilidad como de Cóndor se centra en el hecho de saber si las máquinas poseen tiempos disponibles, ocupados o de inactividad con el fin de no desperdiciar recursos.

Teniendo en cuenta el problema del derroche de recursos, en la presente investigación se pretende mostrar en forma teórica y práctica las ventajas de usar computación distribuida y disponible a través del *software* Cóndor, con el fin de que las organizaciones, instituciones y personas particulares conozcan sus beneficios, los apliquen y generen buenas prácticas y nuevo conocimiento frente al paradigma de desperdicio e inactividad de las máquinas.

Finalmente, para adelantar en forma correcta la investigación, se ha desarrollado de la siguiente manera: en la primera sección se dan a conocer las ventajas y aplicaciones de la computación de alta disponibilidad, en la segunda, se identifica la arquitectura y el funcionamiento de Cóndor, en la tercera, se implementa la arquitectura de Cóndor en el nodo *máster*, en la cuarta, se instalan los nodos de despliegue *Submit* y *Execute*, y en la quinta, se visualizan los procesos en Cóndor. Por último, se presenta un análisis de resultados, conclusiones y trabajos futuros derivados del estudio.

Método de trabajo

La metodología de investigación que se aplicará para esta propuesta se basa en dos aspectos; el primero es el tipo de estudio, que para este caso es descriptivo debido a que lo que se busca analizar son los beneficios de la computación de alta disponibilidad con el fin de optimizar los recursos por medio de Cóndor.

El segundo aspecto por tratar dentro de la metodología es el diseño de la investigación. Para este caso es no experimental puesto que no se construye ninguna situación, sino que se analiza una ya existente; en este caso, el desaprovechamiento de los recursos computacionales y los beneficios de HTC desde el gestor de trabajos Cóndor [2].

Las etapas de desarrollo de la metodología de investigación son las siguientes:

- High Throughput Computing (HTC). Ventajas y aplicaciones.
- Identificación de la arquitectura y funcionamiento de Cóndor.
- Implementación de la arquitectura de Cóndor en el nodo *máster*.
- Instalación de nodos de despliegue *execute* y *submit*.
- Procesos a través de Cóndor.

Mediante las etapas anteriores se mostrarán las ventajas de la computación de alta disponibilidad y sus aplicaciones. A la vez se puede visualizar de forma teórico práctica el proceso por medio del *software* Cóndor identificando previamente su arquitectita y funcionamiento en un ambiente virtualizado.

Estado del arte y marco teórico

El término de alta disponibilidad no es un concepto nuevo, puesto que el manejo de recursos se presenta en diversos ambientes, como las bases de datos y la internet, que permiten intercambiar información desde cualquier parte del mundo. Esta transferencia de datos necesita en muchas ocasiones escenarios dispuestos las 24 horas del día, lo que genera la necesidad de la alta disponibilidad no sólo en soluciones de gestión de datos sino en el foco encaminado al aprovechamiento de los recursos disponibles [3].

Existen diversos trabajos relacionados con la computación de alta disponibilidad, orientados a demostrar que la aplicación de la misma en entornos específicos ayuda a mejor el rendimiento de procesamiento. A través de una interesante trabajo que cita la HTC, denominado "Bridging the gap between high-throughput computing and high-performance computing", se buscó la forma de cerrar la brecha entre la computación de alto rendimiento y la de alta disponibilidad debido a que en algunos casos se toman dos tipos de computación diferentes sin tener en cuenta que ambos necesitan compaginarse para aprovechar los recursos y lograr un trabajo ideal continuo, sin fallas. Las tareas que se ejecutan con los dos tipos de computación pueden estar orientadas a multiprocesos o monoprocesos y necesitar cálculos grandes o pequeños, pero lo ideal es que tanto los tiempos como los recursos usados para cada tarea sean óptimos [4].

La computación de alta disponibilidad también se aplica dentro de las redes bioquímicas. En el artículo "Cóndor – Copasi: high throughput computing for biochemical networks", se hace referencia a cómo los clústeres de procesamiento en un entorno de alta disponibilidad ayudan a proporcionar los recursos necesarios para analizar modelos matemáticos complejos. En esta investigación se usa una herramienta (Cóndor – Copasi) en la que se integra Copasi (herramienta bilógica) a Cóndor (*software* para la computación de alta disponibilidad), se proporciona una red basada en interfaz, ejecutando un número de modelos de simulación y tareas en paralelo, las cuales se dividen en tareas más pequeñas ejecutadas transparentemente en Cóndor. La herramienta ofrece un entorno informático para poder utilizar efectivamente un *software* de alta disponibilidad y lograr ganancias significativas para un número de modelos de simulación y tareas de análisis especificas [5].

Por otra parte, en el contexto del aprovechamiento de los recursos disponibles, en una investigación centrada en este ámbito, "Memory modeling and design for high throughput computing", se puede observar cómo el diseño de memorias basadas en alta disponibilidad resulta ser una tarea compleja al dejar todo en manos de las memorias tradicionales que en algunas ocasiones no son escalables. Existen ejemplos concretos que tratan de soluciona parte de la problemática, pero que al final de cuentas tienen desventajas. Las memorias CAM (memorias de acceso por contenidos) [6] proporcionan un alto rendimiento pero consumen gran cantidad de energía, lo que no las hace tan útiles a la hora de ejecutar procesos largos. Las SRAM (memorias estáticas de acceso aleatorio) [7] requieren mucho cuidado, el análisis está limitado en el ancho de banda global y no es inmune a los problemas de consumo de energía tampoco. Las DRAM (memorias dinámicas de acceso aleatorio) [8] pueden entregar una gran cantidad de rendimiento y disponibilidad, pero la banca de memoria complica el análisis y se necesita de algoritmos especializados para asegurarse de que determinados tipos de patrones de acceso estén libres de conflicto.

Al analizar las ventajas y desventajas de las memorias conocidas tradicionalmente, los pioneros de esta investigación decidieron aplicar la computación de alta disponibilidad mediante nuevos métodos de diseño

que no afecten los procesos, tiempos, rendimientos y recursos de las memorias.

Teniendo en cuenta la introducción encaminada al estado del arte de los estudios relacionados con la presente investigación, es necesario definir adecuadamente los términos que son útiles para llevar a cabo adecuadamente el proceso de generación de nuevo conocimiento. Como primera medida, la computación de alta disponibilidad "utiliza sistemas que están diseñados y administrados para operar con el mínimo *downtime* (tiempo de baja del sistema), tanto planeado como no planeado. Los entornos HTC están enfocados a completar el mayor número de trabajos (conjunto de tareas) en un periodo largo de tiempo. La clave en los entornos HTC es el uso eficiente de los recursos disponibles" [1].

La HTC se puede aplicar a sistemas de clúster de procesamiento. Un clúster es un "sistema distribuido compuesto por un conjunto de computadoras autónomas, interconectadas trabajando juntas en forma cooperativa como un único recurso integrado" [9]. Los clústeres pueden experimentar tres formas de recibir los procesos para su ejecución: la primera, por medio de un recurso que opera a nivel de nodo y es responsable de proveer un servicio, la segunda, por medio de un *service group* que permite que uno o más nodos brinden un servicio, y la tercera y última *failover,* en que un nodo asume la responsabilidad de otro importante recursos.

En el contexto de *software* de HTC, Cóndor es sin duda uno de los mejores cuando se trata de aplicar computación de alta disponibilidad, "Cóndor crea un entorno High Throughput Computing y utiliza eficazmente la potencia de cálculo de las estaciones conectadas en la red. Cóndor puede administrar clústeres dedicados de las estaciones" [10].

El modo de funcionamiento de Cóndor es sencillo, un usuario envía un trabajo y Cóndor busca una máquina disponible en la red para ejecutar el trabajo en ella, la herramienta tiene la capacidad de detectar si una máquina que estaba ejecutando un trabajo deja de estar disponible un tiempo considerable. En la tabla 1 se pueden ver las características de Cóndor que se convierten en grandes ventajas frente a otros gestores de trabajos.

Tabla 1. Características de Cóndor.

Características	Descripción
Checkpoint y migración	Cuando una de las máquinas en las se está ejecutando un trabajo deja de estar disponible, el *checkpoint* busca otra para migrar y continuar el proceso.
Sistema de llamadas remotas	Se crea este entorno de ejecución en el que el usuario no debe preocuparse sobre los sistemas de ficheros disponibles en las estaciones ni de tener cuentas de usuario en éstas para el respectivo acceso.
No es necesario modificar el código fuente de las aplicaciones	No se requiere de un leguaje de programación específico puesto que Cóndor es capaz de ejecutar programas interactivos y no interactivos.
Los trabajos se pueden ordenar	Se usan grafos dirigidos acíclicos con los que es posible determinar el orden de ejecución de los trabajos.
Cóndor permite computación Grid	La técnica de *glidein* permite ejecutar los trabajos enviados a Cóndor en sistemas grid.
ClassAds (diseño limpio que simplifica el envío de trabajos de los usuarios)	Los usuarios pueden pedir recursos necesarios y deseados para ejecutar los trabajos.

Fuente: adaptado de Cóndor - Centro informático de Andalucia [10].

Cóndor define diversos universos para crear un ambiente de ejecución, el cual depende del tipo de trabajo que se desee ejecutar. El universo debe especificarse en el fichero de descripción de envío. Si no se especifica algún universo, Cóndor toma por defecto el universo estándar.

Tabla 2. Universos de Cóndor.

Universos	Descripción
Estándar	Es el universo que se utiliza por defecto en Cóndor; el trabajo se enlaza a través de éste y se compila con las librerías Cóndor.
Vanilla	Es un ambiente de ejecución para trabajos que no necesitan ser enlazados con las librerías de Cóndor.
Globus	Destinado para proveer una interfaz estándar de Cóndor a usuarios que deseen ejecutar trabajos de Globus desde dicho *software*.
Parallel	Usado para enviar trabajos Message Passing Interface (MPI) interfaz de paso de mensajes.
Java	Se utiliza para enviar programas escritos en Java.

Fuente: adaptado de sistema experimental orientado a la computación paralela utilizando el gestor de carga de trabajo Cóndor [11].

Cóndor también se compone de diversos nodos de trabajo. Es posible que cada uno de los nodos (*máster, submit, execute*) trabaje combinado o de forma individual, teniendo en cuenta que sólo puede existir un nodo *máster* por cada grupo creado en Cóndor. En la figura 1 se muestran los nodos de Cóndor dentro del sistema [12]. En la presente investigación es necesario tener en cuenta que el nodo *máster* es el que destina el *software* para coordinar las operaciones en el clúster.

El nodo *submit* acepta los trabajos de los usuarios y pasa información sobre el nodo maestro para programar el envío de tareas al nodo *execute* donde se ejecutan los trabajos transmitidos por *submit* [13].

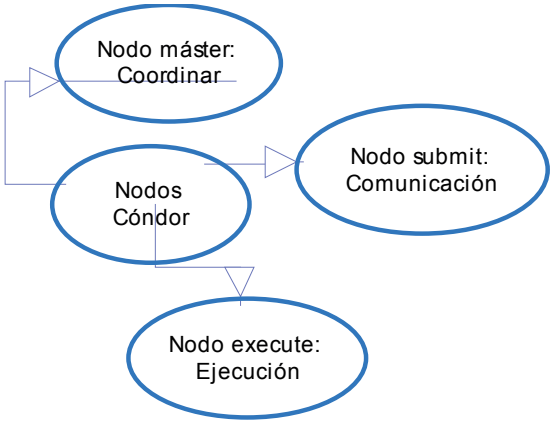

Figura 1. Nodos de Cóndor. Fuente: autor.

La computación distribuida es la base para la aplicación de HTC y Cóndor, puesto que la HTC es un modelo perteneciente a ella. La computación distribuida es un patrón en el que interviene una colección de computadores que pueden o no estar situados en distintos lugares y pertenecer a distintos dominios de administración sobre una red distribuida. Utilizan estándares abiertos para llevar a cabo una tarea u objetivo común. Esta colección de computadores lo que hace básicamente es dividir el trabajo por realizar en pequeñas tareas individuales. Reciben los datos necesarios para esa tarea, la hacen y devuelven los datos para unirlos en el resultado final [14]. Los modelos más importantes de la computación distribuida se pueden observar en la figura 2.

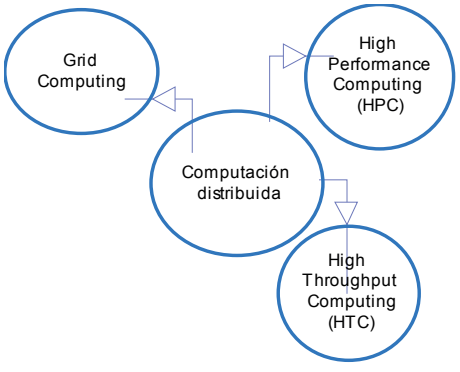

Figura 2: Modelos de computación distribuida. Fuente: autor.

Computación de alta disponibilidad dentro de Cóndor: arquitectura, ventajas y aplicaciones

La computación de alta disponibilidad, conocida como HTC por sus siglas en inglés, está relacionada con los clúster de procesamiento, que son arreglos de computadores conectados [15] para dar solución a un problema que se desglosa en problemas más pequeños y consta de dos o más nodos. A continuación se estudiarán las especificaciones de la HTC junto con sus ventajas y se mostrará la instalación e implementación de los nodos *máster, submit y execute* en el sistema Cóndor.

High Throughput Computing (HTC): ventajas y aplicaciones

El aprovechamiento de recursos por medio de la computación de alta disponibilidad no influye en el uso cotidiano de los computadores [3] sino que proporciona las siguientes ventajas (figura 3):

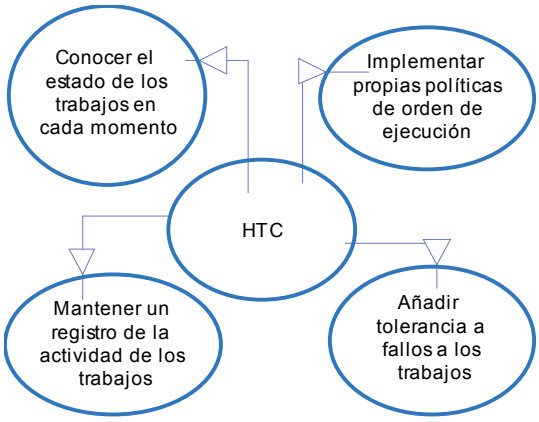

Figura 3. Ventajas de la computación de alta disponibilidad. Fuente: autor.

Identificación de la arquitectura y funcionamiento de Cóndor.

Cóndor es un sistema gestor de carga de trabajos que de forma independiente no proporciona ambientes paralelos pero sí los relacionados con la computación de alta disponibilidad.

Arquitectura del pool de Cóndor.

El *pool* de Cóndor tiene infinitas bondades de recursos que se pueden usar mediante la virtualización de máquinas. Es el caso de la presente investigación, en la que el pool se convierte en un pequeño clúster de procesamiento.

Dentro del gestor de carga Cóndor se hacen diversas peticiones de recursos por parte de los trabajos o "Jobs" teniendo en cuenta los recursos disponibles por cada máquina en ejecución. Esto se logra debido a que los miembros del *pool* envían actualizaciones periódicas al nodo *máster* con el fin de informar acerca del estado del *pool*.

El *pool* está compuesto por tres nodos [16]: el primero, llamado *máster* o administrador central, es único y recolecta la información de los recursos disponibles con el fin de negociar con los *jobs* las peticiones de los mismos. El segundo se denomina "de ejecución", es el más liviano puesto que cualquier máquina puede ser configurada para enviar trabajos y es posible que existan una o más de las mismas según la disponibilidad de la red y de los recursos.

En el tercero conocido como *submit* o máquina de envío, el pool se configura mediante un determinado archivo para enviar *jobs*, esta máquina recibe el *job* que solicita los recursos para ejecutarse y lo envía al *máster* o negociador para que éste le asigne los recursos disponibles.

Cóndor actúa como un agente seleccionando, clasificando y comparando los recursos ofrecidos por las máquinas del entorno con los de la petición del usuario, asegurando que se cubre el cupo de recursos del usuario. Es posible priorizar unos recursos frente a otros [10].

En la figura 4 se observa el funcionamiento del *pool* de Cóndor con sus respectivos nodos anidados. Se debe tener en cuenta que en la presente investigación Sólo se hizo uso de un nodo de ejecución debido a que la aplicación se orienta a mostrar como tal la conexión de los nodos y la explicación y análisis de la máquina de envíos.

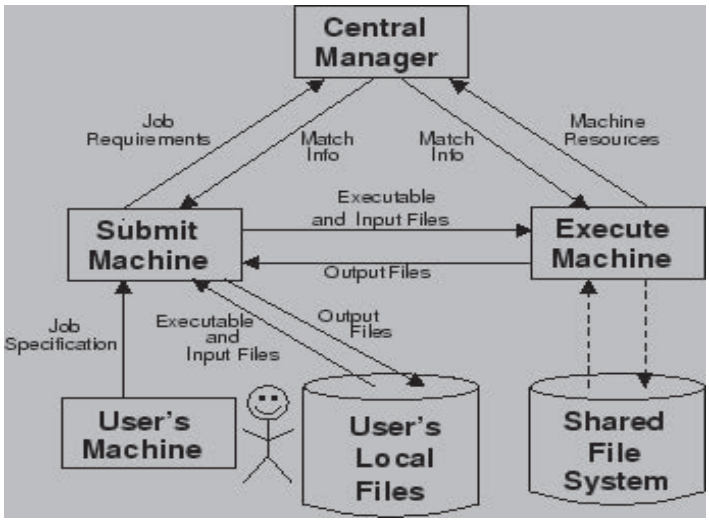

Figura 4. Arquitectura del *pool* de Cóndor.
Fuente: Transparently Gathering Provenance with Provenance Aware Condor [17].

Implementación de la arquitectura de Cóndor en el nodo máster.

Cuando se tiene clara la arquitectura de Cóndor y el funcionamiento de este gestor de trabajo se procede a instalar un sistema operativo sobre el cual se implemente Cóndor. Para el caso de estudio se virtualizó Linux Mint 14 Nadia Cinnamon [18] sobre VMware Workstation 8 [19]. En este entorno se procedió a instalar Cóndor con sus respectivos nodos (*máster, submit y execute*).

- Instalación de Cóndor 7.9.6

Se descarga Cóndor directamente desde los repositorios Debian:

Sudo apt-get install Cóndor

El paquete que se acaba de instalar de Cóndor crea automáticamente un directorio que lleva ese nombre, para dirigirse allí: cd /etc/condor

En esta carpeta se busca condor_config. Para acceder a la distribución de Linux Mint trabajada es necesario dar permisos de modo que se prosigue así: sudo nano condor_config. Ésta es una prueba para rectificar que Cóndor tiene dentro de su carpeta los archivos pertinentes a la configuración.

Se procede a configurar el hostname [20] y el Cóndor *host* de la máquina trabajada. Para este caso las configuraciones se están haciendo en el nodo *máster*. Entonces primero se le asigna el nombre "principal" al maestro sudo nano /etc/hosts (figura 5).

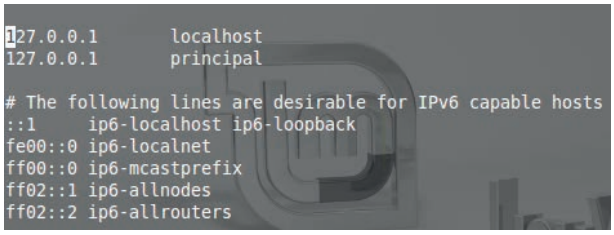

Figura 5. Configuración de hosts. Fuente: autor.

Luego se ingresa al archivo cluster.conf (sudo nano /etc/condor/config.d/cluster.conf). Allí se dejará el archivo de la siguiente manera:

" ## Cóndor configuration for OSG Clusters
For more detial please see
http://www.cs.wisc.edu/condor/manual/v7.8/3_3Configuration.html
# The following should be your cluster domain. This is an arbitrary string used by Condor, not n$
UID_DOMAIN = principal.uptc.edu.co
# Human readable name for your Condor pool
COLLECTOR_NAME = "OSG Cluster Condor at $(UID_DOMAIN)"

```
# A shared file system (NFS), e.g. job dir, is assumed if the name is the same
FILESYSTEM_DOMAIN = $(UID_DOMAIN)
# Here you have to use your network domain, or any comma separated list of hostnames and IP addr$
# condor hosts. * can be used as wildcard
ALLOW_WRITE = *.principal.uptc.edu.co
CONDOR_ADMIN = root@$(FULL_HOSTNAME)
# The following should be the full name of the head node (Condor central manager)
CONDOR_HOST = principal
# Port range should be opened in the firewall (can be different on different machines)
# This 9000-9999 is coherent with the iptables configuration in the Firewall documentation
IN_HIGHPORT = 9999
IN_LOWPORT = 9000
# This is to enforce password authentication
SEC_DAEMON_AUTHENTICATION = required
SEC_DAEMON_AUTHENTICATION_METHODS = password
SEC_CLIENT_AUTHENTICATION_METHODS = password,fs,gsi
SEC_PASSWORD_FILE = /var/lib/condor/condor_credential
ALLOW_DAEMON = condor_pool@*
##  Sets how often the condor_negotiator starts a negotiation cycle
CONDOR_HOST = principal
# Port range should be opened in the firewall (can be different on different machines)
# This 9000-9999 is coherent with the iptables configuration in the Firewall documentation
IN_HIGHPORT = 9999
IN_LOWPORT = 9000
# This is to enforce password authentication
SEC_DAEMON_AUTHENTICATION = required
SEC_DAEMON_AUTHENTICATION_METHODS = password
SEC_CLIENT_AUTHENTICATION_METHODS = password,fs,gsi
SEC_PASSWORD_FILE = /var/lib/condor/condor_credential
ALLOW_DAEMON = condor_pool@*
##  Sets how often the condor_negotiator starts a negotiation cycle
##  for negotiator and schedd).
# It is defined in seconds and defaults to 60 (1 minute), default is 300.
NEGOTIATOR_INTERVAL = 20
##  Scheduling parameters for the startd
TRUST_UID_DOMAIN = TRUE
# start as available and do not suspend, preempt or kill
START = TRUE
SUSPEND = FALSE
PREEMPT = FALSE
KILL = FALSE [21]"
```

Las partes del documento anterior que se subrayan de color azul es donde se debe especificar el hostname de la máquina, para el caso de estudio se le asignó principal.uptc.edu.co, luego se asigna (color amarillo) el Condor_host "principal" asignado anteriormente con la configuración de host (figura 5).

Instalación y configuración DNS

El sistema de nombres de dominio (Domain Name System) se instala con el fin de traducirlos [22] (para este caso principal.uptc.edu.co) a la dirección IP correspondiente.

Esto se hace con el fin de facilitar el trabajo puesto que es complicado memorizar las IP y resulta más efectivo trabajar directamente con el nombre de dominio.

La instalación se realiza en un sistema debían. Por tal razón se parte del comando: sudo apt-get install bind9 [23]. Se debe especificar que se necesita conexión a internet para acceder a los repositorios directos de debian.

Se transforma la máquina usada como servidor DNS. Para ello se modifica el archivo -/etc/resolv.conf modificando el nameserver (principal) por 127.0.0.1. Luego de este paso se debe reiniciar el servidor con /etc/init.d/bind9 restart. Para verificar que el servicio de DNS haya quedado instalado y esté activo se ejecuta el comando nslookup www.debian.org, mediante esto se puede ver si el servidor está activo y cuál fue la IP asignada.

Debe dirigirse a la carpeta de instalación del programa mediante cd /etc/bind, Entrado en este directorio se debe editar el archivo gedit named.conf.local, así:

```
// Do any local configuration here
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
```

```
zone “principal.uptc.edu.co” {
        type master;
        file “/etc/bind/db.servidor.dns”;
};
zone “1.168.192.in-addr.arpa”{
        type master;
        file “/etc/bind/db.1.168.192”;
};
```

Luego de modificar este archivo se debe guardar con named-checkconf. Se procede a copiar la estructura del archivo previamente creado y se modifica de la siguiente forma: sudo cp db.local db.servidor.dns. Se ingresa así: sudo gedit db.servidor.dns y se cambia de la siguiente manera:

```
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
        2               ; Serial
        604800          ; Refresh
         86400          ; Retry
        2419200         ; Expire
         604800)        ; Negative Cache TTL
;
@       IN      NS      localhost.
@       IN      A       127.0.0.1
@       IN      AAAA    ::1
```

El TTl Time to Live se refiere a la validez en tiempo (segundos) de la consulta realizada. El campo *Refresh* indica el intervalo de tiempo en que los DNS secundarios deben refrescar la información del archivo de zona si ha habido cambios. *Retry* indica el intervalo de tiempo que los DNS secundarios deben reintentar actualizar la información si el DNS primario no responde. *Expire* indica el tiempo que el DNS secundario expira como servidor de nombres de la zona en caso de que el DNS primario no responda a requerimiento de actualización. *Negative cahe* TTL arroja una respuesta negativa si la consulta sea errónea [24].

El archivo anterior se guarda y se ejecuta /etc/init.d/bind9 restart. Después se hace lo mismo pero para la resolución inversa, así: sudo cp db.127 db.1.168.192 y se edita -sudo gedit db.1.168.192-, ingresar al archivo. Este se modifica así:

```
;
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ;
Refresh
                          86400         ; Retry
                        2419200         ;
Expire
                         604800 )       ; Negative
Cache TTL
;
@       IN      NS      localhost.
1.0.0   IN      PTR     localhost.
```

Se verifica que toda la configuración esté funcionando mediante: named-checkzone 1.168.192.in-addr.arpa /etc/bind/db.1.168.192 y se reinicia el servicio /etc/init.d/bind9 restart. (figura 6).

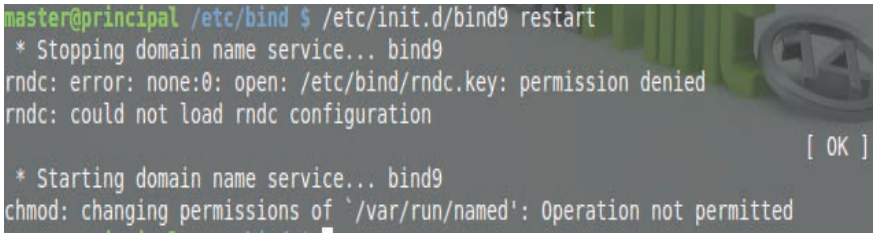

Figura 6. Reiniciando el servicio de DNS. Fuente: autor.

Cuando se tenga creada toda la configuración pertinente se modifica el archivo -/etc/resolv.conf así:

```
Domain principal.uptc.edu.co
Search principal.uptc.edu.co
Principal 127.0.0.1
```

Se prueba todo el procedimiento con sudo nslookup server o con la IP para comprobar que el servicio esté activo y montado dentro de la máquina pertinente.

Instalación y configuración de SSH

SSH – Secure Shell, también conocido como intérprete de órdenes seguras, es un protocolo enfocado a la seguridad de la red. Funciona de la siguiente manera: cuando los datos se envían a la red SSH hace un encriptado automáticamente. Una vez los datos llegan a su destino, los descifra automáticamente. Como resultado de este proceso se obtiene una encriptación transparente en la que los usuarios pueden trabajar con total normalidad sin saber que su comunicación con otros se cifra con seguridad para que viaje a través de la red [25].

Primero se deben instalar los servicios de cliente y servidor para el protocolo SSH de la siguiente forma: sudo apt-get install openssh openssh-client openssh-server. Se rectifica el servicio dirigiéndose al directorio cd /etc/ssh/ y al archivo de configuración -sudo nano sshd_config. Este archivo contiene la información necesaria para que el protocolo SSH pueda realizar la encriptación y cifrado de los datos que viajan a través de los nodos. A continuación se presenta su estructura:

“Package generated configuration file
See the sshd_config(5) manpage for details
What ports, IPs and protocols we listen for
Port 22
Use these options to restrict which interfaces/ protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

Logging
SyslogFacility AUTH
LogLevel INFO

Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
MaxAuthTries 5
Logging
SyslogFacility AUTH
LogLevel INFO

Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
MaxAuthTries 5

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile %h/.ssh/authorized_keys

Don’t read the user’s ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
similar for protocol version 2
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile %h/.ssh/authorized_keys

Don’t read the user’s ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
similar for protocol version 2
HostbasedAuthentication no
Uncomment if you don’t trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

```
# To enable empty passwords, change to yes (NOT
RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response pas-
swords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts
for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT
RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response pas-
swords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
 X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
 X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*
 Subsystem sftp /usr/lib/openssh/sftp-server
# Set this to 'yes' to enable PAM authentication,
account processing,
#MaxStartups 10:30:60
#Banner /etc/issue.net
# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server
# Set this to 'yes' to enable PAM authentication,
account processing,
# and session processing. If this is enabled, PAM
authentication will
# be allowed through the
ChallengeResponseAuthentication and
# PasswordAuthentication. Depending on your PAM
configuration,
# PAM authentication via
ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session
checks to run without
# PAM authentication, then enable this but set
PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes"
```

Este archivo resulta ser de vital importancia en el proceso de configuración del protocolo SSH. Por eso se debe estudiar minuciosamente con el fin de comprender su relevancia en el proceso de comunicación de los nodos.

Después de configurar el archivo de SSH, se procede a darle permisos al puerto 22 para poder comunicar los nodos entre sí, esto se logra de la siguiente manera:

Primero, se deben otorgar los permisos para entrada en cada nodo sudo iptables -A INPUT -p tcp --dport

22 -j ACCEPT, segundo, dar los permisos de salida en cada nodo sudp iptables -A OUTPUT -p udp --sport 22 -j ACCEPT, tercero, guardar los cambios sudo iptables-save, y finalmente se reinicia la máquina sudo reboot now.

Cóndor

Teniendo en cuenta las configuraciones anteriores, se inicia el servicio de gestor de carga Cóndor mediante el comando "sudo service Cóndor start" [26]. Debe aparecer un mensaje que confirme que el servicio ha iniciado (figura7).

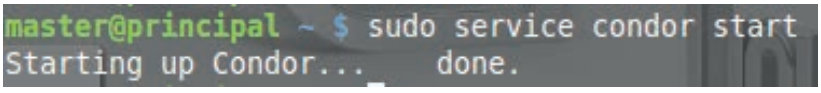

Figura 7. Inicio del servicio de Cóndor. Fuente: autor.

El servicio de Cóndor también se puede detener en el momento que se desee (figura 8).

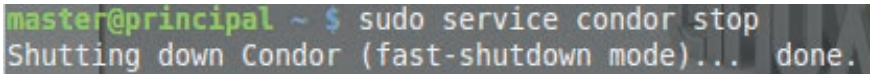

Figura8. Detener el servicio de Cóndor. Fuente: autor.

Cada vez que se desee probar un servicio diferente al gestor de carga Cóndor, se recomienda detener el servicio con el fin de que no existan conflictos internos en las máquinas o nodos trabajados.

Instalación de nodos de despliegue execute y submit.

Dentro del nodo *submit* se enviarán los mensajes para llevar a cabo los trabajos requeridos por el nodo *execute*, se debe tener en cuenta que el nodo *máster* se comporta como negociador para asignar los recursos disponibles a cada trabajo o *job* que lo solicite.

Dentro del nodo *submit* y el nodo *execute* es necesario realizar el mismo proceso de instalación de Cóndor, DNS y SSH con el fin de que exista una comunicación continua entre cada uno de los nodos. En este punto lo que cambia es la función que va a desempeñar cada uno dentro del clúster de procesamiento o como tal dentro de las máquinas conectadas al *pool* de Cóndor.

Para comprobar que los servicios estén activos al terminar la instalación y configuración de los mismos, se puede proceder mediante un *ping* de una máquina a otra ya sea con la IP o con el hostname asignado según las configuraciones previas (figura 9).

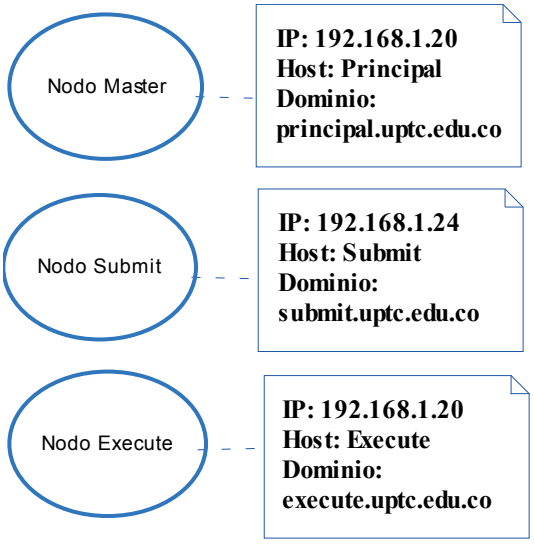

Figura 9. IP, hosts y dominios de cada máquina. Fuente: autor.

Se procede a hacer las pruebas de comunicación entre los nodos. Esto se logra mediante el protocolo SSH configurado con anterioridad y la respectiva IP de cada nodo "SSH IP del nodo a conectar" (figura 10).

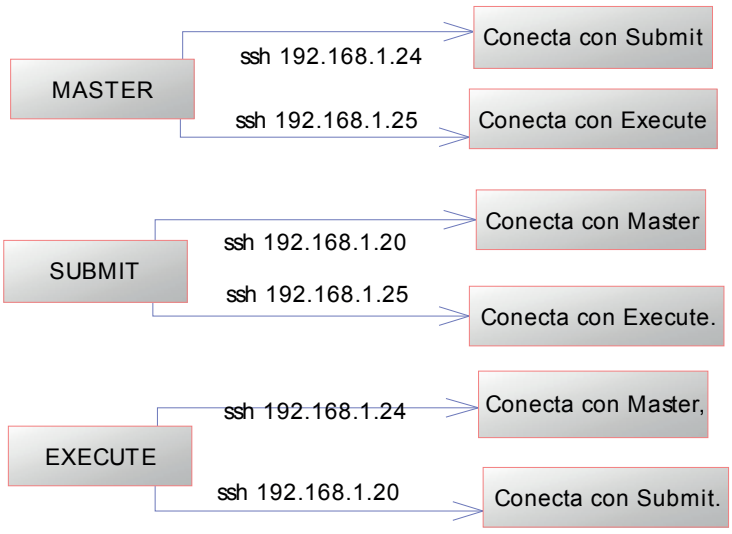

Figura 10. Conexión -SSH- para cada nodo. Fuente: autor.

Procesos a través de Cóndor

Un proceso es un conjunto de procedimientos o funciones que tienen uno o más objetivos. Los programas y aplicaciones como Cóndor pueden ejecutar más de un proceso simultáneamente [27].

Dentro del sistema Cóndor existen diversos procesos para la gestión y carga de trabajos (tabla 3).

Tabla 3. Procesos en Cóndor. Fuente: adaptado de sistema experimental orientado a la computación paralela, utilizando el gestor de carga de trabajo Cóndor[11].

Cóndor máster	Responsable de mantener el resto de demonios ejecutándose sobre cada máquina del pool de Cóndor.
Cóndor status	Monitorea y consulta el estado del *pool* Cóndor
Cóndor submit	Envía los trabajos o *jobs* para que sean ejecutadas en el *pool*mde Cóndor. Requiere un archivo de descripción de envío de la que contiene especificaciones necesarias para enviar el *job*.
Cóndor q	Muestra el estado de los *jobs* que se encuentran en la cola de *jobs* de Cóndor o en ejecución.
Cóndor rm	Elimina uno o más *jobs* de Cóndor.
Cóndor schedd	Relaciona las máquinas disponibles con los *jobs* que se encuentran esperando recursos en Cóndor.
Cóndor collector	Recoge información de todos los otros demonios en el *pool*. Cada demonio envía una actualización periódica llamada ClassAd.
Cóndor negotiator	Obtiene información sobre todas las máquinas disponibles y los *jobs* que se encuentran en la cola de *jobs* de Cóndor.
Cóndor startd	Responsable de iniciar, suspender o detener un *job*.
Cóndor shadow	Actúa como administrador de la petición por parte de un *job*, y se comunica con la máquina donde se ejecutará éste.

Los procesos mencionados anteriormente interactúan con cada uno de los nodos cuando se realiza el envió de un *job* (figura 11). Es importante observar a qué nodo pertenece cada proceso y cómo cada uno de ellos interactúa con los demás del clúster de procesamiento.

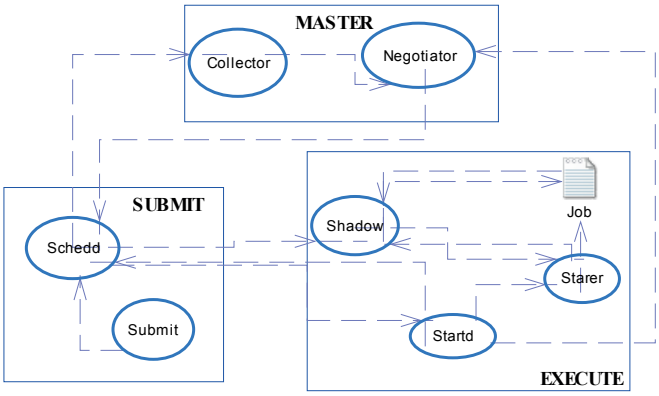

Figura 11. Interacción de los procesos de Cóndor.
Fuente: estudio, configuración y prueba de un entorno de computación Cóndor [28].

Resultados

El proceso empleado para montar cada uno de los nodos, la conexión y comunicación de los mismos fue indispensable para observar los resultados de la presente investigación.

La primera evidencia relacionada con los resultados está en probar el estado de Cóndor en alguno de los nodos trabajados. Para el caso se muestra la prueba (figura 12) de Cóndor status en el nodo *execute* donde se pueden ver valores relacionados con las actividades de la máquina.

```
master@execute /etc/condor/config.d $ condor_status

Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

execute            LINUX      INTEL  Unclaimed Idle     0.420   681  0+00:00:04
                        Total Owner Claimed Unclaimed Matched Preempting Backfill

        INTEL/LINUX     1     0       0         1       0          0        0

              Total     1     0       0         1       0          0        0
master@execute /etc/condor/config.d $
```

Figura 12. Proceso Cóndor status en el nodo *execute*. Fuente: autor.

A través de Cóndor status se puede observar y monitorear el *pool* de Cóndor. Para el caso, la consulta se hace directamente en el nodo *execute*.

Dentro de los resultados de la investigación también se puede observar la conexión de cada uno de los nodos (figura 13) y la forma en que accede determinado nodo del clúster a uno cercano por medio de SSH.

```
master@submit ~ $ ssh 192.168.1.20
Welcome to Linux Mint 14 Nadia (GNU/Linux 3.5.0-17-generic i686)

Welcome to Linux Mint
 * Documentation:  http://www.linuxmint.com
Last login: Tue Jun 11 15:08:50 2013 from submit.local
master@principal ~ $ exit
logout
Connection to 192.168.1.20 closed.
master@submit ~ $
```

Figura 13. Acceso desde el nodo *submit* al nodo principal.
Fuente: autor.

Este proceso de conexión se puede realizar de un nodo a otro las veces que se desee. Se hace con el fin de mantener comunicada toda la estructura del clúster de procesamiento. En el momento en que se quiera salir de la conexión se aplicará el comando exit y se terminará la sesión iniciada con determinado nodo.

Para el acceso a los demás nodos por medio de la IP se sigue el mismo procedimiento de nombrar el protocolo SSH y en seguida la IP del nodo al cual se quiere establecer el proceso de comunicación. Se debe asegurar que los nodos estén concatenados para poder desplegarlos. En este caso útil el comando cat /etc/host (figura 14).

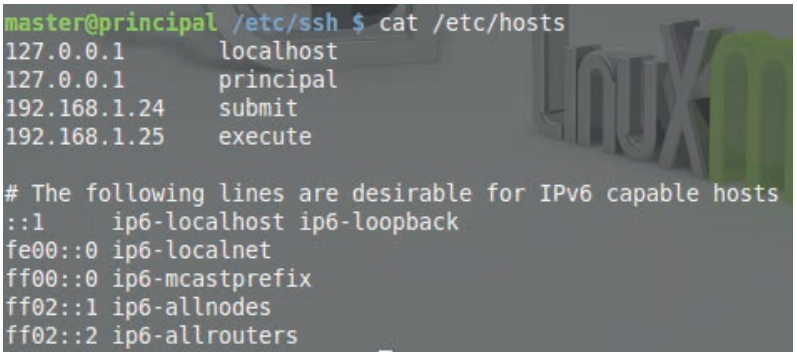

Figura 14. Cat, concatenación entre nodos. Fuente: autor.

En último resultado que se obtuvo en el proceso de investigación está orientado a los trabajos o *jobs* dentro de Cóndor. Para la demostración fue necesario ejecutar un *scrip* de envío en el nodo *submit.*

"##
Test Submit File
##
# Use: Cóndor submit testTask. Cóndor
should_transfer_files = Yes
when_to_transfer_output = ON_EXIT_OR_EVICT
Executable = /bin/hostname
Universe = vanilla
Log = logfileTest
Output = hostOut.$(Process)
Error = hostErr.$(Process)
#Requirements = Arch == "INTEL"
#Requirements = Arch == "X86_64"
Queue 12"

Mediante este scrip fue posible la creación de jobs para realizar su respectivo envío y seguimiento dentro del pool de Cóndor. Primero se le asigno un nombre de prueba, luego se le autorizó para la transferencia de archivos, después se le asigno un universo (para este caso el universo seleccionado es el vanilla debido a que no se requiere de ninguna librería de Cóndor para realizar esta prueba), también se creó un archivo de registro y seguimiento de los jobs y un aviso de errores y salidas que ocurran en el proceso de ejecución, los trabajos para la prueba fueron en total 12.

Se debe buscar un lugar dentro de la máquina para ejecutar el scrip de envío, en este caso se ubicó en un directorio denominado imágenes, allí se guardó y se ejecutó el archivo con el comando Cóndor *submit* testTask.Cóndor junto a los *jobs* (figura 15).

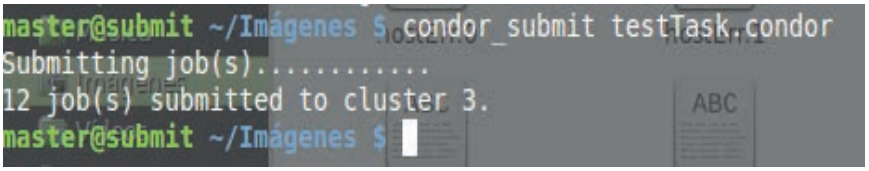

Figura 15. Ejecución del archivo de prueba testTask.condor. Fuente: autor.

Cuando se ejecuta el archivo de prueba mediante el comando condor_submit [29] se crean vario registros (figura 16) de las salidas, y los errores y uno consolidado de la ejecución de los 12 *jobs*.

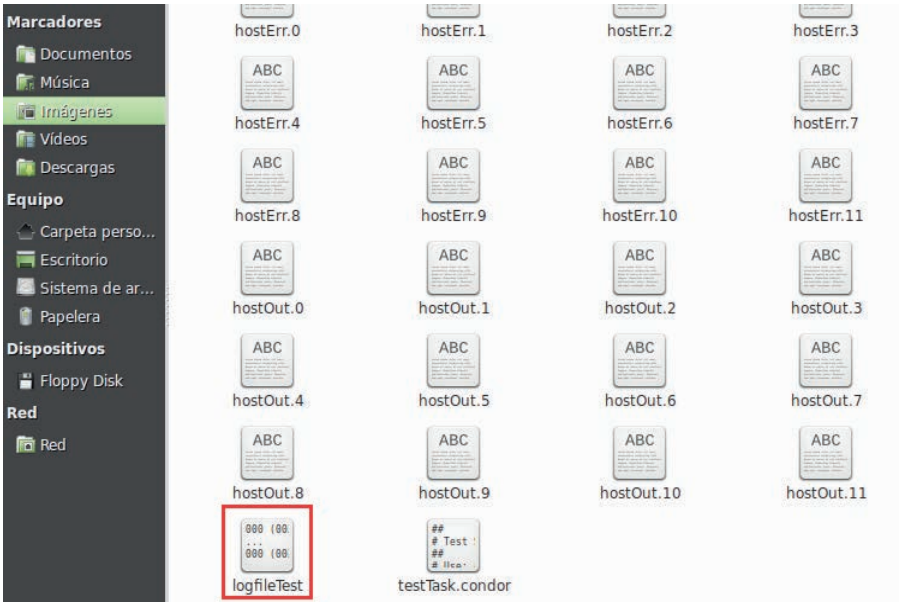

Figura 16. Registro de los *jobs* ejecutados. Fuente: autor.

El archivo logfileTest es el registro total de la ejecución de los *jobs* [30]. En la figura 16 de puede observar señalado con rojo, sin embargo, la estructura interna contiene información indispensable para el seguimiento del *jobs* ejecutados (figura 17).

logfileTest

```
000 (003.000.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.001.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.002.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.003.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.004.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.005.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.006.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.007.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.008.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.009.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.010.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
000 (003.011.000) 06/11 16:48:57 Job submitted from host: <127.0.0.1:9772>
...
```

Figura 17. Registro final logfileTest. Fuente: autor.

El registro presentado anteriormente es útil para el seguimiento de cada proceso pues denota los tiempos de ejecución, lugares y orden en que cada trabajo se llevó a cabo dentro del nodo.

Conclusiones

Las conclusiones derivadas de esta investigación se pueden dividir en dos partes: la primera, relacionada con las configuraciones pertinentes al proceso. En este punto se puede ver lo esencial de la aplicación de los conocimientos básicos en redes, como la configuración e instalación de servicios, por ejemplo el sistema de nombre de dominio (DNS) y el intérprete de órdenes seguras (SSH), los cuales aportan las características de seguridad y comunicación, indispensables para los procesos que se deben llevar a cabo en un clúster de procesamiento.

La segunda referente al sistema gestor de carga Cóndor y la computación de alta disponibilidad (HTC, que se deriva de la computación distribuida. En este aspecto resultó que la HTC se aplica dentro de muchas organizaciones para sistemas que trabajan de forma continua, con el fin de usar todos los recursos disponibles en un determinado momento, en forma óptima, para mejorar procesos internos y externos en las organizaciones.

Cóndor es un sistema que permitió mostrar -la aplicación de la computación de alta disponibilidad -puesto que es un gestor de carga orientado a distribuir y negociar correctamente los recursos disponibles.

Se realizó un ejemplo orientado a la comunicación entre nodos *(máster, submit y execute)* y al envío de trabajos, con su respectivo registro y seguimiento, el cual permitió visualizar de forma real las ventajas de la computación distribuida y de la HTC en entornos pequeños.

Trabajos futuros

Se proponen dos trabajos futuros que pueden proporcionar ventajas y soluciones en diversos ambientes:

- La aplicación de la computación de alta disponibilidad para aprovechar al máximo los recursos, eliminando los tiempos de inactividad Cóndor.
- Profundización en el tema con el fin de montar en el futuro mallas computacionales que permitan aprovechar los recursos de las máquinas en nuevas investigaciones, como la digitalización y análisis de imágenes robustas.

Referencias

[1] P. S. Weygant. (01 de abril). *Clusters for High Availability*. Recuperado de: http://gestionareasistemas.wikispaces.com/Conceptos+de+Alta+Disponibilidad

[2] Fernandez Collado,C., Hernández Sampieri, R. & Baptista Lucio, P., (1997) *Metodología de la investigación*, México: McGraw - Hill Interamericana.

[3] J. A. A. Trujillo and J. G. Díaz. (2009), *Arquitectura Alta Disponibilidad*. Recuperado de: http://campusvirtual.unex.es/cala/epistemowikia/index.php?title=Arquitectura_alta_disponibilidad

[4] I. Raicu. (2009), Many-task computing: bridging the gap between high-throughput computing and high-performance computing 24. Recuperado de: http://search.proquest.com/docview/305056447?accountid=43790#

[5] H. S. Kent E., Mendes P., (2012)"Cóndor-COPASI: High-throughput computing for biochemical networks.

[6] UNED. (2012, 20 de abril). *Memoria CAM y RAM*. Recuperado de: http://www.uned.es/ca-bergara/ppropias/Morillo/web_et_dig/11_mem_ram_cam/transp_mem_cam_ram.pdf

[7] Ecured. 01 de junio). *Static Random Access Memory. Recuperado de:* http://www.ecured.cu/index.php/Memoria_Est%C3%A1tica_de_Acceso_Aleatorio

[8] H. V. S. Herrera Vaquero, E., Ponce Acosta, C., Sánchez Santos, V., Trujillo Jiménez, J. A. (2010, 30 de abril). *Tipos de memoria RAM*. Recuperado de: https://sites.google.com/site/electronicadigitaluvfime/5-1tipos-de-memorias-ram-rom-dram-sram

[9] G. Nudelman. Computacion de alta disponibilidad. Ed. Universidad Tecnológica Nacional.

[10] A. Á. Cano. (15 de mayo). *Cóndor*. Recuperado de: http://www.cica.es/Software/cóndor.html

[11] O. I. G. Parra. (2010) "Sistema experimental orientado a la computación paralela utilizando el gestor de carga de trabajo Cóndor". Proyecto de investigación presentado como requisito para optar al título de: ingeniero de sistemas y computación, Facultad de

Ingeniería - Escuela de Sistemas y Computación, Universidad Pedagógica y Tecnológica de Colombia, Tunja.

[12] T. d. Project. *Informatics Cóndor Pools*. Recuperado de: http://www.dice.informatics.ed.ac.uk/units/research_and_teaching/documentation/unit/beowulf/cóndor.html

[13] C. resources. (2010). *TORQUE Resource Manager*. Recuperado de: http://www.clusterresources.com/torquedocs21/p.introduction.shtml

[14] Ecured. (16 de mayo). *Computación distrubuida*. Recuparado de: http://www.ecured.cu/index.php/Computaci%C3%B3n_distribuida

[15] J. superbenja. (2006, 01 de marzo). *Implementación de un clúster OpenMosix para cómputo científico en el instituto de ingeniería*. Recuperado de: http://www.josecc.net/archivos/tesis/tesis_html1/.

[16] P. O. HTCóndor. *HTCóndor*. Recuperado de: http://research.cs.wisc.edu/htcondor/

[17] J. F. N. Christine F. Reilly. (2009, 01 de junio). *Transparently Gathering Provenance with Provenance Aware Cóndor*. Recuperado de: http://static.usenix.org/event/tapp09/tech/full_papers/reilly/reilly_html/

[18] L. Mint. (2009). *Linux mint 14 Nadia Cinnamon* Recuperado de: http://www.linuxmint.com/release.php?id=19

[19] V. workstation. (2010), *VMware workstation 8*. Recuperado de: http://www.vmware.com/products/workstation/

[20] D. A. Christenson. (2009). "TCP/IP host name resolution on a private network".

[21] O. S. Grid. (25 de mayo de 2013). *Installing HTCóndor as a Batch System*. Recuperado de: https://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallCondor

[22] ADSL-ZONE. (29 de mayo de 2012). *DNS - Domain Name System*. Recuperado de: http://www.adslzone.net/dns.html

[23] S. Kiesel. (2013). "Third-Party ALTO Server Discovery Prototype Implementation and Testbed draft-kiesel-alto-3pdisc-impl-00".

[24] M. A. Pineda. (2013). "Guía de instalacion DNS".

[25] R. S. Daniel J. Barrett, Robert G. Byrnes, Richard E. Silverman. (2001) *SSH, The Secure Shell: The Definitive Guide*. Sebastopol: O'really and Assosiates.

[26] G. F. Christophe Cérin. (2012) *Desktop Grid Computing*.

[27] Definicion.de. (2009). (7 de junio de 2013). *Procesos informáticos*. Recuperado de: http://definicion.de/proceso/

[28] P. p. Mariño. (2011) "Estudio, configuración y prueba de un entorno de computación Cóndor".

[29] htCóndor. (2013, 01 junio). *Submitting your first Cóndor job*. Recuperado de: http://research.cs.wisc.edu/htcóndor/tutorials/fermi-2005/submit_first.html

[30] T. grid. (2009). *Gestión de trabajos en Condor*. Recuperado de: http://ocw.uniovi.es/file.php/27/MSISEINF-1-002/practicas/pt02.html

Sobre los autores

Tatiana Blanco Rojas

Estudiante de Ingeniería de Sistemas y Computación (con terminación académica) de la Universidad Pedagógica y Tecnológica de Colombia. En la actualidad desarrolla su proyecto de Pregrado "Implementación de proceso migratorio de datos desde redes sociales específicas usando Business Intelligence Engineering Process" dirigido por el Ingeniero Javier Antonio Ballesteros Ricaurte. En este momento, desempeña el cargo de analista freelance para la empresa Omnitempus LTDA, donde realiza actividades de selección de personal y verificación de información. Ha realizado diversos cursos como: Mantenimiento de computadores, Arquitectura de computadores, Salud ocupacional, CRM: La administración de la relación con los clientes, Desarrollo web y móvil con HTML5, CSS3 y JQuery, Desarrollo web con PHP y MySQL, Desarrollo web con Django y Python, Incube – The networking and server online training.
tatiana.blanco@uptc.edu.co

Frey Alfonso Santamaría Buitrago

Especialista en Telemática, Ingeniero de sistemas especialista en Telemática y estudiante de último semestre de la Maestría en Ciencias de la Información y las Comunicaciones de la Universidad Francisco José de Caldas. Docente universitario, en diferentes centros educativos, Facultad de estudios a distancia y Escuela de Ingeniería de Sistemas y Computación de la Universidad Pedagógica y Tecnológica de Colombia UPTC, Universidad de

Boyacá, en la Escuela de Sistemas y el Postgrado de Computación para la Docencia en la Universidad Antonio Nariño, Administración Industrial UPTC seccional Duitama, en este momento es docente en la Escuela de Sistemas y Computación de la UPTC y Coordinador del Programa de sistemas de la UAN Tunja. En la actualidad desarrolla su proyecto de maestría "Servicio Web Basado en Técnicas de Computación Grid, para M-Learning, como miembro del Grupo Internacional de Investigación en Informática Comunicaciones y Gestión del Conocimiento – GICOGE, además es miembro fundador del Grupo de Investigación INFELCOM. Desempeño diferentes cargos públicos en el antiguo Instituto de Cultura de Boyacá, durante 13 años y actividades en el sector informático con empresas privadas y del estado. Ha realizado diplomados y curso como: Diplomado en Docencia Universitaria presencial y virtual-Universidad Jorge Tadeo Lozano, Diplomado en Gestión Pública-ESAP, Actualización en sistemas de la Información y las comunicaciones, curso seminario Oracle y Windows NT core technologies, entre otros.
frey.santamaria@uptc.edu.co