ALINEAMIENTO DE SECUENCIAS BIOINFORMÁTICAS EN UNA ARQUITECTURA GRID

BIOINFORMATICS SEQUENCE ALIGNMENT ON A GRID ARCHITECTURE

AUTOR
SIMON OROZCO ARIAS
Ingeniero en Sistemas y Computación
* Universidad de Caldas
Semillero HPC Universidad de Caldas - BIOS
Facultad de Ingeniería
Simon.orozco.arias@gmail.com
COLOMBIA

AUTOR
LEONARDO SOTO AGUDELO
Ingeniero en Sistemas y Computación
* Universidad de Caldas
Semillero HPC Universidad de Caldas - BIOS
Facultad de Ingeniería
leosoto479@gmail.com
COLOMBIA

AUTOR
MARCELO HERRERA GONZÁLEZ
M.Sc. en Mecatrónica y Control
* Universidad de Caldas
Docente Departamento de Sistemas e Informática
Facultad de Ingeniería
marcelo.herrera@ucaldas.edu.co
COLOMBIA

AUTOR
GUSTAVO ISAZA ECHEVERRY
PhD en Ingeniería de Software
* Universidad de Caldas
Docente Departamento de Sistemas e Informática
Facultad de Ingeniería
gustavo.isaza@ucaldas.edu.co
COLOMBIA

***INSTITUCION**
Universidad de Caldas
Ucaldas
Universidad Pública
Calle 65 No 26 - 10
ucaldas@ucaldas.edu.co
COLOMBIA

INFORMACIÓN DE LA INVESTIGACIÓN O DEL PROYECTO: El artículo es producto del proyecto integrado por la Universidad de Caldas y el Centro de Bioinformática y Biología Computacional BIOS, desarrollado por el semillero HPC entre el 10 de agosto de 2015 y el 30 de mayo de 2016.

RECEPCIÓN: Agosto 9 de 2016 **ACEPTACIÓN:** Noviembre 2 de 2016

TEMÁTICA: Ingeniería eléctrica, electrónica, telecomunicaciones y telemática

TIPO DE ARTÍCULO: Artículo de Investigación Científica e Innovación.

Forma de citar: Orozco, Arias. S. (2016). Alineamiento de secuencias bioinformáticas en una arquitectura GRID. En R, Llamosa Villalba (Ed.). Revista Gerencia Tecnológica Informática, 15(43), 37-45. ISSN 1657-8236.

RESUMEN ANALÍTICO

Las tecnologías de la computación de altas prestaciones se han convertido en herramientas muy útiles, empleadas por diferentes empresas o centros de investigación para la ejecución de procesos y análisis de datos masivos en tiempo real, convirtiéndose en necesidades básicas para la mayoría de los procesos investigativos. Sin embargo, uno de los principales objetivos que se buscan al utilizar este tipo de sistemas es el empleo del menor tiempo posible de ejecución de procesos y una mayor precisión en los resultados. En este artículo se dará a conocer de manera general una arquitectura que satisface esas necesidades, como es el caso de una *Grid Computing*. También se presentará qué factores influyen en ella, qué elementos se deben configurar y cómo es el rendimiento al ejecutar trabajos bioinformáticos en esta arquitectura. Para ello se ejecutarán alineamientos usando NCBI-Blast en diferentes nodos de cómputo ubicados dispersos geográficamente, finalmente se evidencia el desempeño que se obtiene al finalizar las tareas.

PALABRAS CLAVE: Computación HTC, Grid, Infraestructura Cluster, Condor, Blast.

ANALYTICAL SUMMARY

Tools such as high performance computational technologies have become very useful, used by research centers for running real time analysis. This turns high performance computing into a basic need for any research process. Moreover, minimizing the time spent to run this processes and increasing the precision with which the processes can run are some of the main reasons this technology is used. This article will discuss *Grid computing* (in a general manner) which is an architecture that satisfies these need. It will also showcase the fundamental factors that influence grid computing and how the performance of bioinformatics jobs can be boosted using this type of architecture. This will be done using NCBI-Blast in computational nodes which are placed in different physical locations in order to see the obtained performance after running each job.

KEYWORDS: HTC Computing, Grid, Cluster infrastructure, Condor, Blast.

INTRODUCCIÓN

En la actualidad, uno de los aspectos fundamentales para cualquier proceso investigativo es la capacidad para procesar datos de manera rápida y efectiva, por esta razón la supercomputación ha sido considerada uno de los tres pilares de la ciencia [6].

Por lo anterior, se han realizado numerosos estudios relacionados con el mejoramiento de la capacidad de cómputo, el aprovechamiento óptimo de todos los recursos disponibles y la eficiencia en el uso de energía. Hoy en día es muy común el uso de la computación de altas prestaciones (HPC) en la bioinformática; en campos como análisis de *big data*, procesamiento de señales biomédicas, *Deep learning, machine learning,* entre otras [16].

Otras ciencias básicas, tales como la física, la química, la biología, entre otras; también han presentado la necesidad de ejecutar grandes cálculos que ameritan el uso de recursos computacionales y el empleo del menor tiempo posible de ejecución de procesos, pues demandan datos de alta dimensionalidad, complejidad y procesos intensivos, con el fin de obtener resultados confiables, de tal manera que no exista porcentaje de error o que este sea lo mínimo posible.

En búsqueda de mejores prácticas, se han implementado diferentes infraestructuras computacionales que logren dar soporte a grandes volúmenes de datos y que de una forma ágil y efectiva proporcionen información precisa y veraz del trabajo que se esté realizando. En años anteriores surgieron máquinas computacionales de gran tamaño para cálculos matemáticos simples, pero gracias al avance constante de la tecnología se escaló al desarrollo de tecnologías móviles y pervasivas como celulares inteligentes o *Smartphones* logrando llevar a cabo algunos procesos computacionales en dispositivos más pequeños que los acostumbrados. Sin embargo, el avance tecnológico no ha parado allí, también se están desarrollando infraestructuras que puedan reunir los

recursos de varias máquinas (servidores, computadores y *smartphones)* [5] e incluso grupos de trabajo o investigación y utilizarlos de una manera integrada y de la forma más óptima posible. Una de estas soluciones son los *Clusters,* que se entienden como la integración de computadores utilizando una red *LAN* (red local), con el objetivo de ofrecer mayores capacidades de procesamiento que en una sola máquina [15]; y también, dando la posibilidad de tener en diferentes espacios físicos, diferentes arquitecturas de computo, se han implementado las *Grid Computing* "permitiendo que los ordenadores compartan a través de internet u otras redes de telecomunicaciones no solo información, sino también poder de cálculo (*Grid Computing*) y capacidad de almacenamiento (*Grid Data*). Es decir, en el *Grid* no sólo se comparten contenidos, sino también capacidad de procesamiento, aplicaciones e incluso dispositivos totalmente heterogéneos (sensores, redes, ordenadores, etc.)." [20].

En estos tiempos, se trabaja en diferentes partes del mundo, proyectos conjuntos entre diferentes Universidades y centros de investigación para el procesamiento de grandes cantidades de datos en temáticas de ámbito social, tecnológico y económico, como la forma de reproducción de alguna bacteria, crecimiento acelerado de enfermedades, o incluso el cálculo a gran escala de un valor estadístico [2]. Todo esto con el fin de reducir tiempos de respuestas y optar por una solución más rápida y efectiva. Otras temáticas que se realizan con las *Grid Computing* son el trabajo con las altas energías, previsión de análisis de tiempo, bioinformática, redes de transporte, visualización del espacio, E-learning, entre otras [9].

En correspondencia, en este artículo se presenta la implementación de una Tecnología *Grid* que usa un modelo computacional *HTC (Hight Throughput Computing)* [13] entre la Universidad de Caldas y el Centro de Bioinformática y Biología Computacional BIOS para proveer una comunicación entre cuatro máquinas de cómputo para el análisis de datos, ejecución de procesos bioinformáticos, evaluación del rendimiento y resultados de las máquinas al realizar un alineamiento de secuencias bioinformáticas con el uso del algoritmo de Blast [1], para posteriormente concluir que aspectos ayudan a los investigadores en una ejecución más óptima de sus procesos de estudio, focalizando en cómo lograr una reducción de tiempos en el procesamiento de datos con bajos costos administrativos de los recursos computacionales, fundamentalmente, para aproximar una solución de aceleración de analítica de grandes volúmenes de datos genómicos y metagenómicos en entornos Grid.

1. METODOLOGÍA

Con el fin de implementar una tecnología *Grid Computing* entre ambas instituciones, se realizó un estudio del estado del arte de HPC y HTC y pruebas locales utilizando recursos propios. En el paso siguiente, se llevó a cabo la identificación y análisis de la topología de red de ambas instituciones. Se analizó qué elementos de hardware y software se debían utilizar y se realizó finalmente la puesta a punto de los *Clusters*, en principio vía MPI por el comportamiento de paralelismo del entorno de NCBI-BLAST, usando memoria distribuida en su algoritmo, como herramienta para la paralelización de tareas y luego la configuración de la infraestructura Grid.

Con la infraestructura Grid configurada se ejecutaron procesos bioinformáticos utilizando como referencia una base de datos que contiene la información de todos los organismos que son no redundantes (cuyo objetivo es representar moléculas biológicas distintas que se observan para un organismo, cepa o haplotipo [18], esto se mostrará más en detalle en el punto 2.6.

1.1 ACONDICIONAMIENTO DEL CLUSTER LOCAL

Se instaló inicialmente *Centos* 6.5 en 3 máquinas virtuales, una de ellas correspondía al *Head Node (también conocido como nodo maestro)* y las otras dos *Worker Nodes (o nodos de trabajo)*. Se prosiguió a realizar una configuración de red básica, para que todos los *hosts* estuvieran en un mismo segmento, se siguieron los siguientes pasos:

- instalación del *Network File System* ó NFS, con el fin de compartir el directorio *Home* entre varios servidores. Al realizar la configuración NFS, se hace más efectivo para configuraciones permanentes que deberían estar siempre accesibles [17].
- Instalación y configuración del *Network Information System* o NIS, el cual corresponde a un protocolo de servicios cliente–servidor utilizado para el envío de datos de configuración en sistemas distribuidos como son el caso de nombres de usuarios, *Host* entre computadores de la red, etc. Además, maneja la identificación de los clientes en un punto central llamado servidor [22].
- Configuración de *Secure Shell* o SSH; este es también llamado intérprete de orden seguro y corresponde a un protocolo y el programa que lo implementa, con el fin de acceder a máquinas remotas a través de una red. Este permite administrar computadores mediante un intérprete de comandos. También permite copiar datos de forma segura [23].
- Instalación y configuración de *la* interfaz de paso de

mensajes (MPI), que es una interfaz o estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca diseñada para ser usada en programas que explotan la existencia de múltiples procesadores [7]. Este programa de bibliotecas de desarrollo es una norma o estándar para aplicaciones de memoria distribuida que utilizan computación paralela. Este proporciona un compilador para trabajar con los programas tanto distribuidos como paralelos [14]. En MPI el número de procesos requeridos se asigna antes de la ejecución del programa, y no se crean procesos adicionales mientras la aplicación se ejecuta. A cada proceso se le asigna una variable que se denomina *Rank*, la cual identifica a cada proceso en el rango de 0 a p-1, donde p es el número total de procesos.

El control de la ejecución del programa se realiza mediante la variable *Rank*, la variable *Rank* permite determinar qué proceso ejecuta determinada porción de código. En MPI se define un *Comunicator* como una colección de procesos, los cuales envian mensajes el uno al otro; el *Comunicator* básico se denomina *MPI_COMM_WORLD* y se define mediante un macro del lenguaje C. *MPI_COMM_WORLD* agrupa a todos los procesos activos durante la ejecución de una aplicación [3].

1.2 TOPOLOGÍA DE RED DE LA GRID COMPUTING

En las Figuras 1 y 2, se presentan las topologías físicas de red usadas por la Grid entre BIOS y la Universidad de Caldas respectivamente. Ambas instituciones estarán comunicadas a través de la red Renata.

FIGURA 1. Topología de red BIOS.

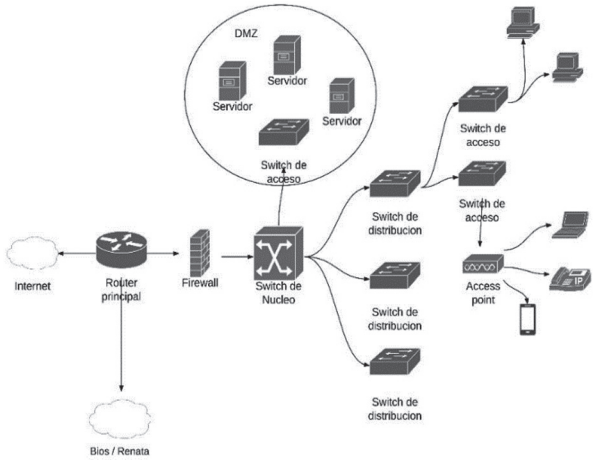
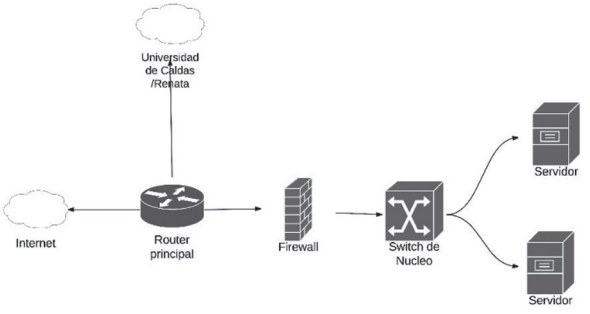


FIGURA 2. Topología de red Universidad de Caldas.



1.3 MATERIALES Y MÉTODOS (HARDWARE Y SOFTWARE)

Para la implementación de la *Grid Computing* se utilizaron distintos dispositivos y aplicaciones de *Software* para llevar a cabo la puesta a punto del mismo.

Para el prototipo construido se utilizaron los siguientes recursos:

CentOS, una distribución del Sistema Operativo Linux basada en las fuentes libremente disponibles por red *Hat Enterprise Linux* [4], *Slurm,* una arquitectura basada en un nodo cabeza, el cual gestiona todos los nodos de cómputo, representados por demonios de cómputo, los cuales son llamados *Slurmd [24].* Entre la cabeza y los nodos de trabajo existe una comunicación jerárquica, la cual es tolerante a fallos. Todos estos demonios son manejados por *Slurmctld*, que se ejecuta en el nodo cabeza. *Slurm* alcanza su máximo rendimiento con 500 trabajos por segundo.

Slurm cuenta con 3 funciones claves:

- Asigna acceso exclusivo y/o no exclusivo de recursos (nodos de cómputo) a usuarios por alguna duración de tiempo.
- Provee un *Framework* para comenzar, ejecutar y monitorear trabajo (normalmente trabajo paralelo) en el conjunto de nodos.
- Controla los recursos manejando colas de trabajo pendiente.

Las entidades que son administradas por estos demonios pueden estar conformadas por nodos de cómputo, particiones que agrupan los nodos dentro de conjuntos lógicos *Jobs* o trabajos que son asignaciones de recursos por un tiempo determinado y *Jobs Steps*, las cuales son las tareas a realizar para cada *Job*. A cada trabajo se le asignan nodos dentro de una partición,

cuando un *Job* es asignado a un conjunto de nodos, el usuario tiene la capacidad de inicializar el trabajo en paralelo en forma de *Jobs Steps* [24] y *HTCondor*, es un proyecto de la Universidad de Wisconsin-Madison (UWMadison) y está ideado para aprovechar al máximo la capacidad computacional de una red de computadores. HTCondor pone a disposición toda la capacidad de cálculo presente en cualquier institución, incrementando considerablemente los recursos disponibles [8].

En el trabajo de [15], se dice que HTCondor cuenta con una arquitectura de 3 componentes claves:

- Un nodo de administración, el cual se encarga de recibir las solicitudes de ejecución de trabajos, determinar cuál es el nodo ideal para ejecutarlas, encolar y desencolar trabajos y consolidar los estados y respuestas recibidas por parte de los Worker Nodes.
- Nodos de ejecución, son los que ejecutan los trabajos.
- Nodos de envío de trabajos, son los únicos que están autorizados para que a través de ellos los usuarios envíen trabajos al Cluster.

Para la paralelización de un trabajo en Condor, se envía una petición al nodo administrador o negociador; este pone las tareas en una cola y gestiona la disponibilidad de las diferentes máquinas disponibles para procesar ese trabajo. Si hay disponibilidad de procesamiento, este nodo envía a ejecución una parte del trabajo que se está realizando. A medida que los diferentes recursos se van desocupando, se continúa con los procesos que estén en cola, revisando constantemente si han llegado más peticiones de trabajos y si es del caso, los sigue colocando en la cola de tareas. Al finalizar toda la ejecución de las tareas, el nodo administrador organiza los diferentes resultados entregados por los Worker Nodes y lo devuelve finalmente al usuario que realizó la petición a Condor.

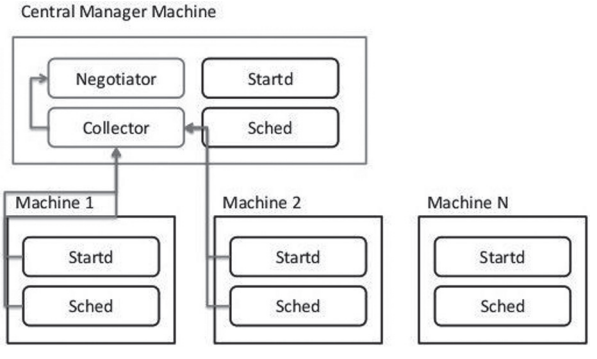
En la Figura 3 se observa la arquitectura de HTCondor y cuáles son sus principales componentes en un ambiente de supercomputación.

HTCondor también cuenta con 9 roles para la administración y ejecución de tareas, entre ellas se encuentras condor_master, condor_collector y condor_negotiator. Además HTCondor divide sus funcionalidades en varios universos, algunos de ellos son [15]:

- Universo Vanilla, se utiliza para ejecutar Scripts, comandos del sistema operativo y aplicaciones en general de las cuales no se tenga acceso al código fuente. Su funcionalidad es limitada y no cuenta con soporte para puntos de revisión.
- Universo Standard, requiere que las aplicaciones sobre las cuales se basan los trabajos a ejecutarse en el cluster se recompilen con las librerías de Condor. Cuenta con soporte para puntos de revisión, invocación a llamados remotos e incluye limitaciones técnicas (Fork, Sockets, entre otras).
- Universo Parallel, permite la ejecución de aplicaciones en paralelo en cluster, incluyendo a aplicaciones basadas en MPI.
- Universo Grid, permite la ejecución de trabajos en el nodo Grid a través del encapsulamiento de las interfaces provistas por el Globus Toolkit.

En Hardware se utilizaron 4 nodos de cómputo con las características de la Figura 4 Características físicas de los nodos. Fuente: Autoría propia.

FIGURA 3. Arquitectura HTCondor. Fuente: Tomado de [10]



1.4 MONTAJE Y PUESTA A PUNTO DEL CLUSTER CON OPEN MPI

Este proceso secuencial requirió de un conjunto de pasos como instalación y adaptación del sistema operativo, configuración de la red, instalación del sistema de archivos (NFS) y sistema de usuarios (NIS), hasta la configuración de aplicaciones y finalmente MPI [10].

En la Figura 5 Arquitectura general de la Grid, se podrá observar la arquitectura general de una Grid Computing, y en específico, la utilizada de referencia para la realización del presente artículo.

1.5 MONTAJE Y EJECUCIÓN DE PROCESOS BIOINFORMÁTICOS EN LA GRID

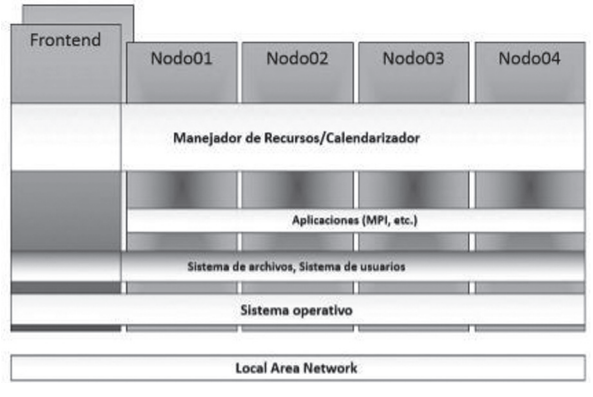
Se parametrizó el Cluster en la Universidad de Caldas, y se implementó HTCondor para la comunicación de procesos entre ambas instituciones. El último paso llevado a cabo consistió en la ejecución de alineamiento de secuencias usando NCBI-Blast para observar el comportamiento de las máquinas de cómputo dentro de la Arquitectura Grid.

Blast (*Basic Local Aligment Search Tool*) es un algoritmo para comparación (alineamiento) de secuencias ya sea de ADN, ARN o de proteínas. Más exactamente se encuentra clasificado dentro de los algoritmos para alineamiento local.

FIGURA 4 Características físicas de los nodos.

	Cluster Ucaldas		Cluster Bios	
Nodo	c-head	c-wn1	nodo1gridbc	nodo2gridbc
Arquitectura	4 Processors Intel(R) Xeon(R) CPU X5550 a 2.67GHz	1 Processor Intel(R) Xeon(R) CPU X5550 a 2.67GHz	8 cores Intel Xeon E5-2670 a 2.60 GHz	8 cores Intel Xeon E5-2670 a 2.60 GHz
Memoria Ram	8 GB	3,68 GB	64 GB	64 GB
Disco duro	150 GB	150 GB	350 GB	150 GB
Velocidad puerto de red	Conexión a 1 Gbps	Conexión a 1 Gbps	Conexión a 1 Gbps	Conexión a 1 Gbps

FIGURA 5. Arquitectura general de la Grid.



El algoritmo encuentra las secuencias de la base de datos que tienen mayor parecido a la secuencia problema.

BLAST usa un algoritmo heurístico por lo que no garantiza que ha encontrado la solución correcta. Sin embargo, BLAST es capaz de calcular la significación de sus resultados, por lo que provee un parámetro para juzgar los resultados que se obtienen [19].

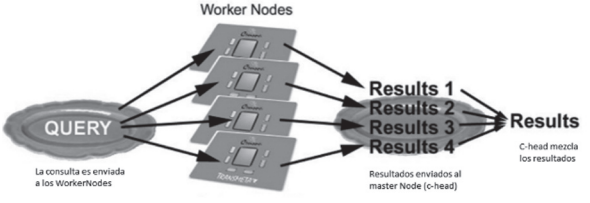
Existen varias implementaciones de este algoritmo, una de las más conocidas es la realizada por el NCBI-Blast. El NCBI-Blast es el programa/algoritmo que usa por defecto el NCBI para realizar búsquedas de secuencias en sus bases de datos.

Existen varios tipos de Blast como: Blastn, Blastx, TBlastn, TBlastX, Bl2seq y el utilizado en el presente trabajo Blastp, que compara una secuencia protéica con una base de datos de proteínas. Este programa, dada una proteína de consulta, retorna las secuencias de proteínas que el usuario especifique [11].

Para lograr esta ejecución de Blast se utilizó una secuencia de proteínas llamada alu.a y una base de datos de referencia llamada *env_nr.01* también de proteínas. Ambos ficheros fueron descargados del NCBI.

El objetivo de estos dos elementos es revisar en qué partes de las cadenas hay una gran similitud y obtener la información pertinente del estudio. Se usó la base de datos mencionada, debido a su pequeña dimensión y a que en los archivos nr están contenidos datos de todos los organismos que son *no redundantes*. En la figura 6 se observa el proceso de paralelización de BLAST.

FIGURA 6. Paralelización de BLAST.



2. RESULTADOS Y DISCUSIÓN

Para evaluar el rendimiento y comportamiento de la Grid, se validaron algunos recursos para determinar el grado de viabilidad de esta arquitectura en procesos bioinformáticos, entre ellos el NCBI-Blast, variando la cantidad de tareas lanzadas con el fin de obtener resultados de dos variables que son de suma importancia en este tipo de arquitecturas: La latencia presentada entre los diferentes nodos y el tiempo de ejecución de trabajos.

2.1 MEDICIÓN DE LATENCIA EN LA GRID

La latencia es definida como la suma de retardos temporales dentro de una red. Estos retardos son producidos por las demoras en la propagación y transmisión de paquetes dentro de la red [12].

Esta variable evaluada tiene un papel muy importante en este tipo de tecnologías Grid debido a que si se tiene esta infraestructura dentro de una red donde hay afluencia constante de datos, hará que los paquetes y procesos que se envíen entre los diferentes nodos de cómputo tengan cierto retardo y por ende retrasa la unificación total de procesos.

En las siguientes 4 figuras se aprecian los resultados de la medición de la latencia en todos los nodos de trabajo con respecto al nodo maestro de la Grid.

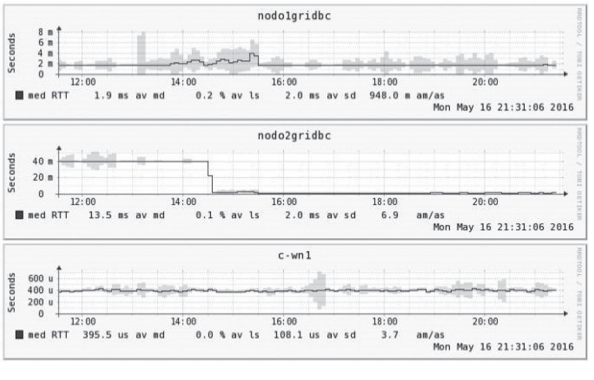
En Figura 7 se evidencia la latencia de red desde el nodo maestro hacia los nodos esclavos, se presenta de manera general cómo es la fluctuación de paquetes del nodo cabeza con los demás nodos de trabajo en un rango de 8 horas (12:00 y 20:00 aproximadamente) en la cual los diferentes nodos se ven afectados por una gran petición de tareas por parte de usuarios que interactúan con los nodos de la Grid. Para el caso específico del documento, en los días en que se realizaron las pruebas de latencia se estaban ejecutando a la par diferentes tareas en los nodos de cómputo de la universidad de Caldas lo que afectó en cierta medida el tiempo de finalización de las pruebas.

En la Figura 8 Latencia entre Nodo maestro y Nodo Cwn-1, se observa la latencia encontrada entre el nodo cabeza C-head y el worker node Cwn-1. En la Figura 10 Latencia entre el Nodo maestro y el Nodo Node1gridbc se encuentra la latencia encontrada entre el nodo cabeza C-head y el *worker node* nodo1gridbc de BIOS. Finalmente en la Figura 11 Latencia entre el Nodo maestro y el Nodo Node1gridbc se analiza la latencia encontrada entre el nodo cabeza C-head y el *worker node* nodo2gridbc de BIOS. Todas las mediciones corresponden a las últimas 30 horas tomando como referencia las 9:31 pm (hora Colombiana).

2.2 MEDICIÓN DEL TIEMPO DE EJECUCIÓN DE LOS TRABAJOS

Con el fin de revisar el comportamiento de la Grid al ejecutar procesos, se decidió por ejecutar de distintas maneras cierta cantidad de procesos NCBI-Blast en la Grid para revisar cual era el tiempo empleado por cada CPU al procesar un job o trabajo recibido y finalmente calcular el tiempo total de ejecución de cada prueba.

FIGURA 7. Latencia de red desde el nodo maestro hacia los nodos esclavos



Esta otra variable evaluada permite analizar el performance o rendimiento que tiene este tipo de arquitecturas dentro de un contexto con las características usadas; es así que gracias a la administración y distribución de tareas por parte de un manejador o nodo cabeza se logra tener tiempos muy parecidos al lanzar una variada cantidad de procesos al mismo tiempo como se ve a continuación.

Es importante resaltar que el tiempo de finalización de los distintos procesos va acompañado de la latencia que tenga en ese momento la red y del tipo de procesos paralelos que también se estén ejecutando en las distintas máquinas que son ajenas a los procesos del actual estudio.

FIGURA 8. Latencia entre Nodo maestro y Nodo Cwn-1

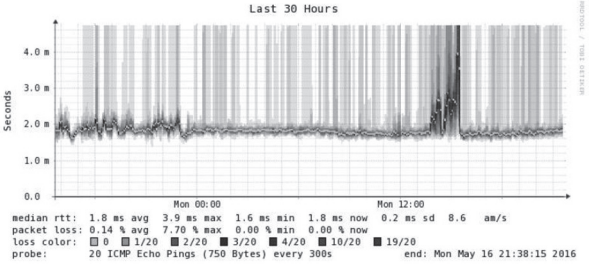


FIGURA 9. Latencia entre el Nodo maestro y el Nodo Node1gridbc

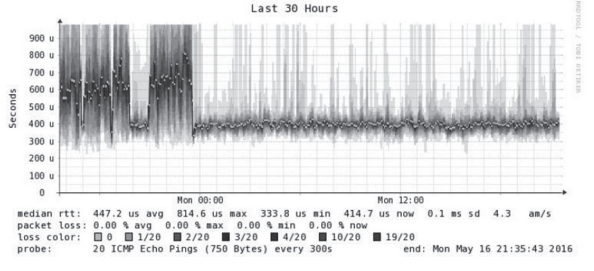
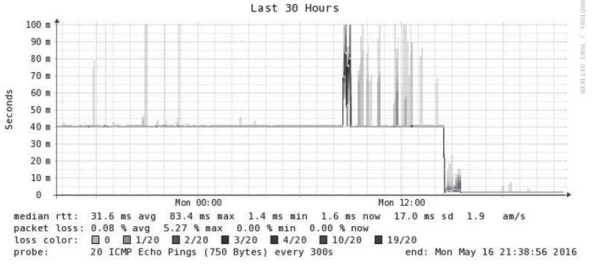


FIGURA 10. Latencia entre el Nodo maestro y el Nodo Node2gridbc

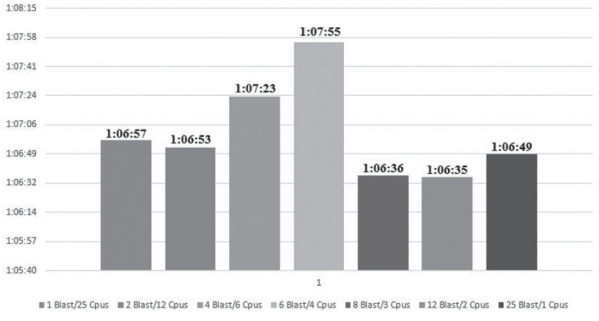


Luego de la ejecución de los NCBI-Blast en la Grid, se llevó a cabo un post procesamiento para ordenar la salida y eliminar las repeticiones extrayendo las secuencias con las que se alineó el query inicial con la base de datos. Esto, debido a que se utilizó el formato estándar de NCBI-Blast para la salida del alineamiento. Posterior

a esto los resultados de todos los alineamientos fueron iguales.

En la figura 11 se observa cuál fue el tiempo total de la prueba cuando se ejecutó 1 solo Blast en 25 CPUs, 2 Blast en 12 CPUs c/u, 4 Blast en 6 CPUs c/u, 6 Blast en 4 CPUs c/u, 8 Blast en 3 CPUs c/u, 12 Blast en 2 CPUs c/u y finalmente 25 Blast en 1 CPU c/u.

FIGURA 11. Tiempos de ejecución de NCBI-Blast sobre Grid.



3. CONCLUSIONES

La forma en que se ejecuten trabajos en una Grid y la cantidad de procesos lanzados dentro de los distintos nodos que la componen, aportarán un tiempo razonable de ejecución de los trabajos, logrando también resultados más precisos y reducir en cierta medida el porcentaje de error que se pueda obtener a simple vista si se evaluara de forma manual y/o en máquinas con bajo rendimiento de cómputo.

Gracias a la distribución de trabajos que se tiene en una Grid, a la sincronización en tiempos de ejecución y al control de todos los subprocesos por un nodo maestro, se logra tener una retroalimentación constante del estado en que se encuentran los diferentes nodos que están ejecutando sus trabajos en paralelo, para posteriormente hacer la integración de los resultados. Este nivel de integración buscará a su vez una mejora en los diferentes procesos efectuados en la Grid y la asignación de nuevas tareas a medida que los nodos en ejecución vayan terminando sus trabajos.

Se confirmó que debido a la heterogeneidad a la que está sometida una Grid, el rendimiento de esta se ve sometida al rendimiento del nodo de menor capacidad computacional.

Con el uso de *Smokeping* [21] se llevó a cabo pruebas de latencia desde el nodo cabeza hasta los demás nodos de cómputo y se demostró que la Grid depende en gran medida del tráfico de la red. Esto como factor clave en la ejecución de procesos bioinformáticos dentro de una arquitectura Grid.

Basado en los resultados obtenidos, se confirmó que es más recomendable correr una gran cantidad de procesos independientes que se ejecuten en la menor cantidad de nodos posibles que lanzar un solo proceso en toda la Grid. Debido a que se tienen diferentes velocidades de procesamiento en los nodos, además del manejo de la cola de tareas que utiliza HTCondor; ya que serán lanzados procesos cada vez que haya recursos disponibles y finalmente porque cada proceso es independiente en su ejecución y se ve menos afectado por la alta latencia de la red.

Para la realización de las pruebas se utilizó como apoyo metodológico una base de datos de proteínas llamada env_nr.01, la cual se comparó con una secuencia de proteínas llamada alu.a con el fin de comparar el alineamiento de secuencias que se puedan tener entre diferentes especies en una misma región y calcular el tiempo de ejecución de las distintas pruebas realizadas.

Al ejecutar NCBI-Blast de forma distribuida en varios nodos de cómputo, la salida tiende a tener duplicaciones, por lo tanto no se recomienda ejecutar este tipo de aplicaciones sobre muchos nodos. Si es así, se debe realizar un post procesamiento de la salida eliminando las repeticiones, lo cual se puede lograr a través de *shell scripting*. Además se recomienda usar el formato de salida del alineamiento en tabular, para facilitar el adecuamiento posterior del NCBI-Blast.

Gracias a los esfuerzos colaborativos entre diferentes instituciones y a los bajos costos que acarrean las infraestructuras Grid frente a los Cluster, estas pueden ser de mucha utilidad a la hora de ejecutar grandes cantidades de tareas, todas ordenadas y encoladas gracias a *HTCondor*, además, teniendo en cuenta las características y funcionamiento de una Grid, se puede optimizar mucho tiempo en investigaciones bioinformáticas que la requieran.

Los costos administrativos por nodo que acarrean las arquitecturas Grid son menores en comparación a los de un clúster, esto debido a que los equipos físicos se encuentran bajo la coordinación de distintas organizaciones, dividiendo también los gastos de estos. Se continuará probando con secuencias reales de proyectos afines en los grupos de investigación y se integrará el entorno GRID como contenedores de la plataforma GITIRBio [25].

4. AGRADECIMIENTOS

Agradecemos a la Universidad de Caldas y al Centro de Bioinformática y Biología Computacional de Colombia BIOS por facilitar los recursos computacionales y humanos para el desarrollo del trabajo anteriormente

presentado, sin su colaboración esta investigación no hubiera sido posible.

5. REFERENCIAS

[1] Altschul, S. F., W. Gish, et al (1990). Basic local alignment search tool.. Revista Journal of molecular biology 215(3): 403-410.

[2] Baker, M., Buyya, R., & Laforenza, D (2002). Grids and Grid technologies for wide-area distributed computing. Revista S*oftware: Practice and Experience*, 32(15), 1437-1466.

[3] Barker, B (2015). *Message passing interface (mpi)*. Paper presented at the Workshop: High Performance Computing on Stampede.

[4] Beloglazov, A., Piraghaj, S. F., Alrokayan, M., & Buyya, R (2012). Deploying OpenStack on CentOS using the KVM Hypervisor and GlusterFS distributed file system. *University of Melbourne*

[5] Franco, Cesar. 2016 Procesamiento y Visualización Distribuida de Relaciones Funcionales de genes en ambientes Ubicuos. "Tesis de Maestría en Ingeniería Computacional no publicada" Universidad de Caldas, Manizales, Colombia.

[6] González, Á. F., Rosillo, R., Dávila, J. Á. M., & Olivera, V. M (2015). Historical review and future challenges in Supercomputing and Networks of Scientific Communication. Revista The Journal of Supercomputing

[7] Gropp, W., Lusk, E., Doss, N., & Skjellum, A (1996). A high-performance, portable implementation of the MPI message passing interface standard. Revista Parallel computing, 22(6), 789-828.

[8] Hernández, E. A. Z., & Ordoñez, J. S (2015). Una herramienta para el soporte a la computación distribuida.

[9] Hernández, J. T., Díaz, E., Figueroa, P., & De la Rosa, F (2007). El desarrollo de aplicaciones colaborativas de alta calidad: una realidad sobre la Red Académica de Alto Desempeño (Renata). *Revista de Ingeniería,* 26, 22-28.

[10] University of Winsconsin-madison (2016) HTCondor High Throughput Computing HTCondor Manuals. Recuperado (2016, noviembre 05) de http://research.cs.wisc.edu/htcondor/manual/

[11] Johnson, M., Zaretskaya, I., Raytselis, Y., Merezhuk, Y., McGinnis, S., & Madden, T. L (2008). NCBI BLAST: a better web interface. *Nucleic acids research*, 36(suppl 2), W5-W9.

[12] Kay, R (2009). Pragmatic network latency engineering fundamental facts and analysis. *cPacket Networks, White Paper*, 1-31.

[13] Liarte López, M. R (2008). Estudio, implementación y evaluación de entornos de computación de alto rendimiento HTC.

[14] Lonarkar, M. G., & Pandey, Y (2013). Real Time & Secure Video Transmission Using OpenMPI.

[15] Meza Martínez, J. I., & Uribe Hurtado, A. L (2013). Implementación de dos nodos grid basados en clusters e integrados a grid Colombia a través de *Renata, utilizando software libre*.

[16] Min, S., Lee, B., & Yoon, S (2016). Deep Learning in Bioinformatics. *arXiv preprint arXiv:1603.06430*.

[17] Palevich, J. H., & Taillefer, M (2008). Network file system: Google Patents.

[18] Pruitt, K. D., Tatusova, T., & Maglott, D. R (2007). NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Revista *Nucleic acids research,* 35(suppl 1), D61-D65.

[19] Ramstad, J (2015). Protein Alignment on the Intel Xeon Phi Coprocessor.

[20] Tejedor, R. J. M (2007). Grid Computing. *Manual formativo* de ACTA(43), 17-22.

[21] Tobias Oetiker (2015). About SmokePing. Recuperado (2016, abril 16) de http://oss.oetiker.ch/smokeping/

[22] V. Kalusivalingam (2004). Network Information Service (NIS),Configuration Options for Dynamic Host. Configuration Protocol for IPv6 (DHCPv6). Cisco Systems (India) Private Limited. Recuperado (2016, Marzo 22) de https://tools.ietf.org/html/rfc3898

[23] Ylonen, T., & Lonvick, C (2006). The secure shell (SSH) protocol architecture.

[24] Zhou, X., Chen, H., Wang, K., Lang, M., & Raicu, I (2013). Exploring Distributed Resource Allocation Techniques in the SLURM Job Management System. Illinois Institute of Technology, Department of Computer Science, Technical Report.

[25] Castillo, L. F., López-Gartner, G., Isaza, G. A., Sánchez, M., Arango, J., Agudelo-Valencia, D., & Castaño, S. (2015). GITIRBio: A Semantic and Distributed Service Oriented-Architecture for Bioinformatics Pipeline. Journal of Integrative Bioinformatics, 12(1), 255.