Proyecto

"Estudio de Modelos Teóricos de tipo NP-completos en el Laboratorio Nacional de GRIDS de Súper Cómputo, Utilizando Algoritmos Evolutivos de Optimización con Técnicas de Procesamiento Distribuido"

AVANCES DE PROYECTO

Integrantes del proyecto

Universidad Autónoma del Estado de Morelos	Instituto Tecnológico de Veracruz
Cuerpo Académico: Optimización y Software **Integrantes:** **Investigador Líder** Dr. Marco Antonio Cruz Chávez **Investigadores Colaboradores** M.C. Martín Heriberto Cruz Rosales Dr. José Crispín Zavala Díaz **Estudiantes:** M.C. Fredy Juárez Pérez M.C. Erika Yesenia Ávila Melgar L.I. Alina Martínez Oropeza	**Cuerpo Académico: Cómputo Intensivo Aplicado a la Ingeniería** **Integrantes:** **Investigador Principal** Dr. Abelardo Rodríguez León **Investigadores Colaboradores** M.C. Rafael Rivera López **Estudiantes:** Tania I. Araus Pérez Marcela González Tinoco Olivia Suárez Toribio

CONTENIDO.

1. Introducción

En base a las dificultades que se tuvieron para poder acceder al Laboratorio Nacional de GRID de Súper Cómputo, debido a que éste aun no está disponible para dar servicio puesto que está en etapa de pruebas y acorde a la necesidad del proyecto de trabajar en una GRID, se decidió por parte del grupo de trabajo, implementar una MiniGRID UAEM-ITVer, de cómputo de alto rendimiento con comunicación a través de Internet 2. En esta GRID llamada Tarántula, se realizaron y aun está realizándose el estudio de dos modelos teóricos de tipo NP completos [Papadimitriou and Steiglitz, 82]. El primer modelo es el Problema de una Cadena de Suministros con Ventanas de Tiempo [Toth and Vigo, 01]. El segundo modelo es el Problema de Calendarización de Trabajos en Talleres de Manufactura [Roy and Sussman, 64]. Los resultados que se han obtenido hasta el momento en las pruebas experimentales, son muy alentadores en cuanto al uso de la GRID. Se espera que al finalizar el presente proyecto, se tenga el producto tecnológico de los dos modelos en estudio ya terminado, para que su implementación en el Laboratorio Nacional de GRID sea transparente una vez que se pueda tener acceso al servicio.

Mediante un estudio del flujo de información en las dependencias de las meta heurísticas más conocidas para el trato de estos problemas de tipo NP-completos, se decido crear dos algoritmos de tipo multi-poblacional, uno para cada problema con el fin de que afectara en lo mínimo la latencia encontrada en la MiniGRID Tarantula. Se desarrollaron dos algoritmos multi-poblacionales y en cada uno de ellos se abordó la debilidad que tiene este tipo de meta heurísticas. Los algoritmos que aplican población genética trabajan de forma eficiente haciendo exploración en el espacio de soluciones para encontrar la mejor solución, pero tienen una deficiencia, la explotación que hacen en el espacio de soluciones es muy débil debido a que no se preocupan por encontrar la mejor solución en vecindades (óptimo local). Para mejorar la explotación se realizó una hibridación en la fase de mutación. Para uno de los algoritmos se aplicó un mutador inteligente y para el otro se aplicó Recocido Simulado. Con las ventajas que nos da la GRID, se mandó la ejecución del proceso de mutación de los individuos de la multi-población a cada nodo de la GRID en un porcentaje proporcional, en cada generación del algoritmo de cada problema a resolver. El proceso de mutación es el que consume la mayor cantidad de tiempo en el algoritmo. La eficiencia del algoritmo dependerá del número de nodos disponibles en la MiniGRID Tarantula, a mayor número de nodos, mayor será el tamaño de la población que pueda soportar el algoritmo para dar una mejor solución al problema de manera más eficiente debido a que se podrá explorar un mayor espacio de soluciones con la posibilidad de encontrar el óptimo global del problema en un tiempo más corto.

La MiniGRID de alto rendimiento que se construyó en el Centro de Investigación en Ingeniería y Ciencias Aplicadas (CIICAp) de la Universidad Autónoma del Estado de Morelos y el Instituto Tecnológico de Veracruz (ITVer), funciona uniendo dos Clusters de alto rendimiento pertenecientes a diferentes dominios e integrados como una sola máquina paralela. El escenario para la integración de los Clusters es que se encuentran alejados geográficamente, además de ser administrados localmente y son por si mismo entidades independientes.

Este panorama de integración presentó múltiples dificultades, la más difícil de ellas, fue como resolver la comunicación entre los diferentes nodos que no pertenecen al mismo dominio, esto es, que no pertenecen al mismo Cluster y poder responder a la pregunta ¿como alcanzar los nodos de otro Cluster que se encuentra alejado geográficamente.

La idea fue poder ver en forma transparente los Clusters unidos como uno sólo y poder ejecutar programas MPI en esta GRID. Existieron muchos desafíos, ya que la infraestructura de cómputo y de comunicación ideal no existe actualmente, se tuvieron problemas con el ancho de banda y equipo de cómputo que no es de primer nivel, más sin embargo se logró hacer la integración.
En este documento se muestra de forma general dos modelos de estudio de tipo NP-Completo y resultados parciales de la investigación realizada de estos modelos teóricos en la plataforma experimental propuesta, que ejecuta programas MPI a través de la MiniGRID Tarántula. Se presentan pruebas de latencia entre clusters con comunicación en I2, así también describimos las técnicas utilizadas para unir infraestructura de cómputo de alto rendimiento a través de una red privada virtual basada en Software (MiniGRID Tarántula), también se presentan diferentes técnicas para desplegar Clusters y el aporte en experiencia para hacer frente a esta problemática. Finalmente se presentan las conclusiones y los trabajos futuros.

Objetivo general

Desarrollo de algoritmos evolutivos en plataforma Grid para problemas NP-hard aplicando el estándar MPI

Objetivos particulares

Objetivo 1.- Desarrollo y análisis de una estructura de vecindad con procesamiento distribuido para ser implantada en algoritmos evolutivos.

Objetivo 2.- Desarrollo de un algoritmo evolutivo híbrido con procesamiento distribuido y que aplique búsquedas locales también con procesamiento distribuido para la optimización de problemas de manufactura y que pueda resolver modelos de estudio tomados de la literatura.

Objetivo 3.- Desarrollo de un algoritmo evolutivo híbrido con procesamiento distribuido que aplique búsquedas locales con procesamiento distribuido para la optimización de una cadena de suministro y que pueda resolver modelos de estudio tomados de la literatura.

2. Problema del Ruteo Vehicular con Ventanas de Tiempo

El problema de ruteo de vehículos con ventana de tiempo (VRPTW, por sus siglas en inglés) es un problema basado en asignación de rutas a vehículos para atender a diferentes clientes, este problema se caracteriza por utilizar un rango de tiempo de atención asignado a cada cliente, conocido como ventana de tiempo, la ventana de tiempo incrementa el número de restricciones en el problema lo que complica la búsqueda de la solución. El modelo que representa al VRPTW [Toth and Vigo, 01] se presenta a continuación.

2.1. Modelo NP-Completo VRPTW

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} X_{ijk} \qquad (1)$$

$$\sum_{k \in K} \sum_{j \in \Delta^+(i)} x_{ijk} = 1 \qquad \forall i \in N \qquad (2)$$

$$\sum_{j \in \Delta^+(0)} x_{0jk} = 1 \qquad \forall k \in N \qquad (3)$$

$$\sum_{i \in \Delta^-(j)} x_{ijk} - \sum_{i \in \Delta^+(j)} x_{ijk} = 0 \qquad \forall k \in K, i \in N \qquad (4)$$

$$\sum_{i \in \Delta^-(n+1)} x_{i,n+1,k} = 1 \qquad \forall k \in N \qquad (5)$$

$$w_{ik} + s_i + t_{ij} - w_{jk} \leq (1 - x_{ijk}) M_{ij} \qquad \forall k \in K, (i,j) \in N \qquad (6)$$

$$a_i \sum_{j \in \Delta^+(i)} X_{ijk} \leq w_{ik} \leq b_i \sum_{j \in \Delta^+(i)} X_{ijk} \qquad \forall k \in K, i \in N \qquad (7)$$

$$E \leq w_{ik} \leq L \qquad \forall k \in K, i \in \{0, n+1\} \qquad (8)$$

$$\sum_{i \in N} d_i \sum_{j \in \Delta^+(i)} x_{ijk} \leq C \qquad \forall k \in N \qquad (9)$$

$$x_{ijk} \geq 0 \qquad \forall k \in K, (i,j) \in A \qquad (10)$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in K, (i,j) \in A \qquad (11)$$

La función objetivo que se presenta en (1), representa el costo total, el cual se puede interpretar como el tiempo de viaje o distancia recorrida total de todos los vehículos que interviene en el suministro. Se requiere encontrar la mínima distancia de recorrido total utilizando el menor número de vehículos. La variable x_{ijk}, toma valor de uno cuando el vehículo *k* atiende la ruta que va del cliente *i* al cliente *j*. El depósito se representa como *i = 0* ó *i = 1 + n*. Las restricciones que se deben cumplir se presentan de (2) a (11).

Restricciones en (2), indican que un cliente será atendido por un sólo vehículo.

Restricciones de (3) a (5), dan las características de la ruta a seguir por cada vehículo *k*.

- Restricciones en (3), indican que para cada vehículo k, sólo un cliente *j* se puede alcanzar partiendo del depósito.
- Restricciones en (4), indican que el número de vehículos que llegan a un cliente es el mismo número de vehículos que sale.
- Restricciones en (5), indica que cada ruta vehicular tiene un sólo cliente que conecta al depósito.

Restricciones (6) a (8) y (9), garantizan la factibilidad de la secuencia con respecto a condiciones de tiempo y aspectos de capacidad respectivamente.

- Restricciones en (6), indican que no se puede iniciar un servicio en cliente *j* si el cliente *i* no ha sido atendido y el vehículo no ha llegado al cliente *j*. Aquí *M* es una constante muy grande.
- Restricciones en (7), indican para cada unidad vehicular y cada cliente *i*, que el tiempo inicial del servicio w_{ik} debe iniciar dentro de la ventana de tiempo $[a_i, b_i]$.
- Restricciones en (8), indican para cada unidad vehicular y en el depósito *0*, que el tiempo inicial del servicio w_{ik} debe iniciar dentro de la ventana de tiempo $[E, L]$, el cual es el intervalo de atención del deposito.
- Restricciones en (9), indican que para todo vehículo *k*, la suma de las demandas de todos los clientes atendidos no debe exceder la capacidad del vehículo.

Restricciones (10), son restricciones de no negatividad de las variables *x*.

Restricciones (11), son restricciones que definen al modelo lineal como un modelo lineal entero binario. Estas restricciones definen la dureza del problema.

A continuación se presenta la metodología desarrollada para el presente proyecto para dar solución a ciertas instancias del VRPTW definidas por Solomon [Solomon, 1987].

2.2. Algoritmo Genético híbrido

La figura 1, presenta la metodología de solución [Díaz-Parra and Cruz-Chávez, 08] que está formada de un algoritmo genético que contiene un operador de selección "*the best*", un operador de cruzamiento "*crossover-k*" y un operador de mutación "*mutation-S*" con multi poblaciones iniciales generadas con la técnica de *k-means* comúnmente utilizada en técnicas de minería de datos. En el algoritmo genético propuesto se crean las poblaciones mediante agrupación *k-means* donde los individuos se crean factibles de acuerdo a su distribución geográfica. El operador "the best", selecciona de acuerdo a su aptitud en un porcentaje a los mejores individuos de cada población y cruza estos por población. Con el cruzamiento k, se mantiene el tamaño de cada población y permite la exploración del espacio de soluciones. El mutador inteligente optimiza la función de aptitud, de tal forma

que se reacomode cada gen en su respectivo cromosoma para llegar a un individuo-mejor-solución; lo que significa que la agrupación de genes en ese individuo es la que reportará la menor distancia total recorrida ya que la agrupación será entre genes cercanos a él en el espacio euclidiano, este mutador realiza una búsqueda local hasta mejorar la aptitud del individuo del que se parte. El procedimiento anterior se repite en cada generación multi poblacional.

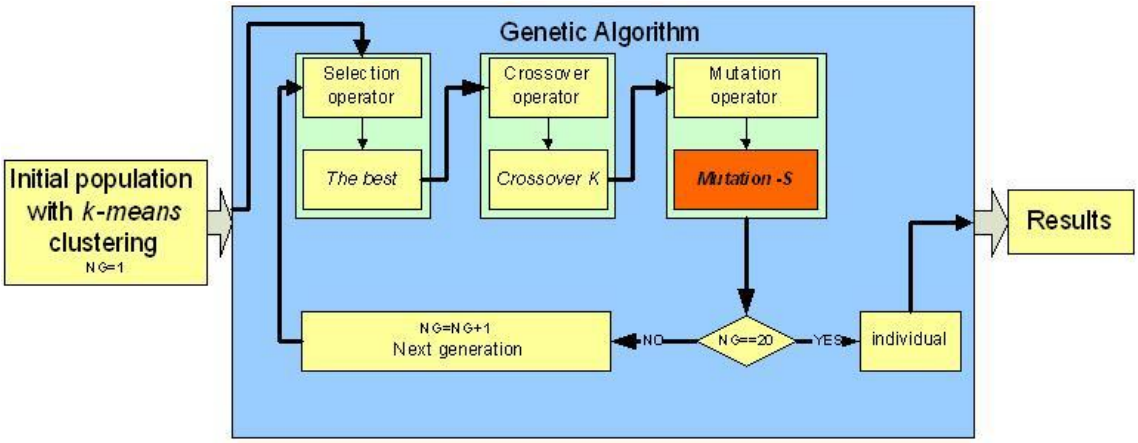


Figura 1. Metodología de solución del problema VRPTW.

2.3. Procesamiento distribuido del Algoritmo

El procesamiento distribuido aplicado al algoritmo se presenta en la figura 2. En cada nodo de la MiniGRID Tarántula, se crea una población factible. Cada nodo realiza de manera independiente la selección, cruzamiento y mutación. Antes del inicio de una nueva generación los nodos se comunican para intercambiar parte de su población. Esto es una representación de la realidad de lo que pasa con los grupos de inmigrantes, en este caso existe migración de una población a otra.

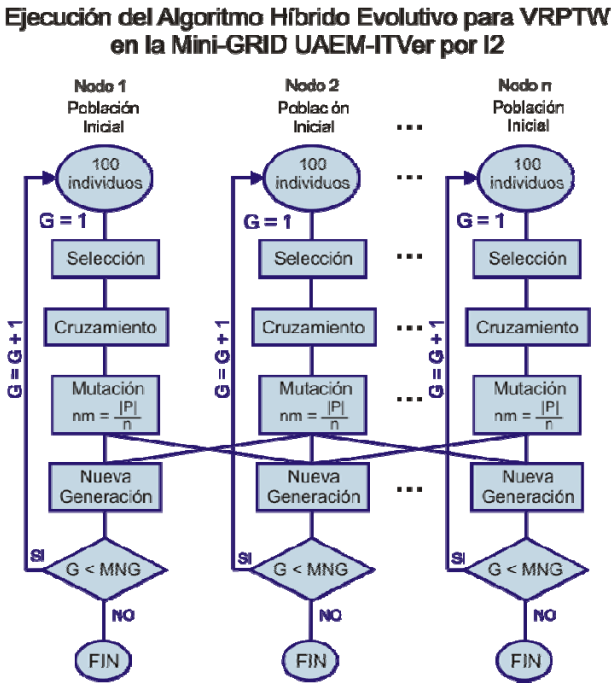


Figura 2. Ejecución del algoritmo híbrido evolutivo para VRPTW en la Mini-GRID UAEM-ITVer por I2.

3. Problema de Calendarización de Trabajos en Talleres de Manufactura

El problema de asignación de recursos en talleres de manufactura (JSSP, por sus siglas en inglés) es un problema basado en la calendarización de máquinas de un taller para la realización de un conjunto de trabajos, donde cada trabajo requiere de la ejecución de un conjunto de operaciones, este problema se caracteriza por la búsqueda de una secuencia óptima de ejecución de trabajos en cada máquina del sistema cumpliendo restricciones de precedencia en la ejecución de operaciones en cada en trabajo. El modelo que representa al JSSP [Roy and Sussman, 64] se presenta a continuación.

3.1. Modelo NP-Completo JSSP

La función objetivo en (12) se relaciona con el máximo tiempo de término de ejecución de la última operación que se realiza en el taller de manufactura. Para este problema se requiere minimizar ese tiempo. El conjunto de restricciones en (13) indica que el tiempo en el que inicia cada operación j, debe ser mayor o igual a cero. El conjunto de restricciones en (14) define las restricciones de precedencia que existe entre pares de operaciones que pertenece a un mismo trabajo. El conjunto de restricciones en (15) define las restricciones de capacidad de recursos que existe entre pares de operaciones que ejecuta la misma máquina.

$$\text{Min}\left[\max_{j \in O}\left(s_j + p_j\right)\right] \tag{12}$$

$$\forall\ j \in \boldsymbol{O} \qquad s_j \geq 0 \tag{13}$$

$$\forall\ i, j \in \boldsymbol{O},\ (i \prec j) \in J_k \qquad s_i + p_i \leq s_j \tag{14}$$

$$\forall\ i, j \in \boldsymbol{O},\ \left(i, j \in M_k\right) \qquad s_i + p_i \leq s_j \vee s_j + p_j \leq s_i \tag{15}$$

3.2. Algoritmo Genético híbrido

La figura 3, presenta la metodología de solución que está formada de un algoritmo genético que contiene un operador de selección "*the best*", un operador de cruzamiento "*SXX*" y un operador de mutación "*mutation-S*" con una población factible inicial generada de forma aleatoria. El operador "the best", selecciona de acuerdo a su aptitud en un porcentaje a los mejores individuos de cada población y cruza estos. Con el cruzamiento de intercambio subsecuente, se mantiene el tamaño de la población y permite la exploración del espacio de soluciones. El mutador que se ocupa, es un mutador iterativo [Cruz-Chávez and J. Frausto-Solís, 04] el cual optimiza la función de aptitud, de tal forma que se reacomode cada gen en su respectivo cromosoma para llegar al mejor individuo encontrado mediante búsqueda local iterada mediante el procedimiento de recocido simulado; lo que significa que la agrupación de genes en ese individuo es la que reportará el menor Makespan. El procedimiento anterior se repite en cada generación de la población.

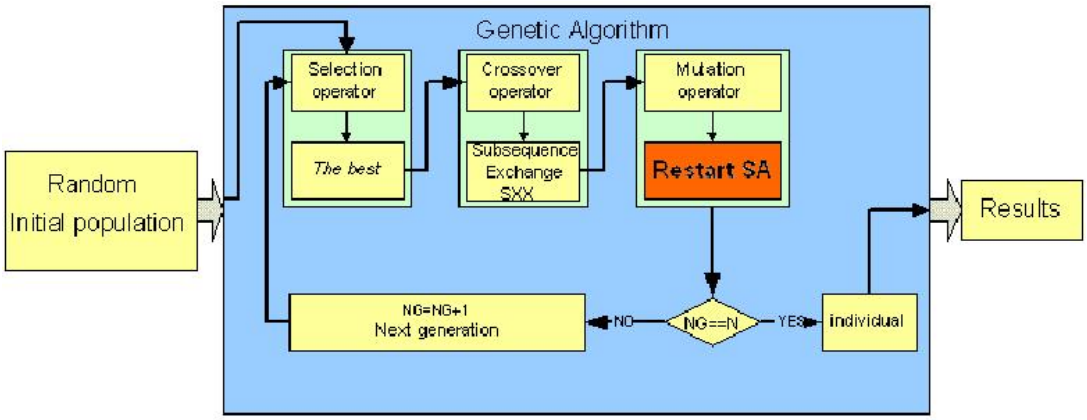


Figura 3. Metodología de solución del problema JSSP.

3.3. Procesamiento distribuido del Algoritmo

El procesamiento distribuido aplicado al algoritmo se presenta en la figura 4. En un nodo de la MiniGRID Tarántula, se crea la población inicial factible. Este nodo realiza la selección y cruzamiento. La mutación que es el procedimiento más costoso en cuanto a tiempo de cómputo se reparte en forma proporcional en todos los nodos de la MiniGRID, esto quiere

decir que se genera un número de procesos de acuerdo al número de CPU's existentes en la GRID (1:1), cada proceso ejecuta un conjunto de Recocidos Simulados en una CPU, un RS por cada individuo de un subconjunto de la población a mutar. Los nodos regresan los nuevos individuos mutados al nodo origen. La ejecución del algoritmo continúa para la nueva generación en el nodo origen con la selección y cruzamiento. Para una población definida de tamaño 100 en las pruebas y si se muta el 80% de la población. Se tiene que si la MiniGRID cuenta con 20 CPU´s, cada CPU estará realizando la mutación de 4 individuos.

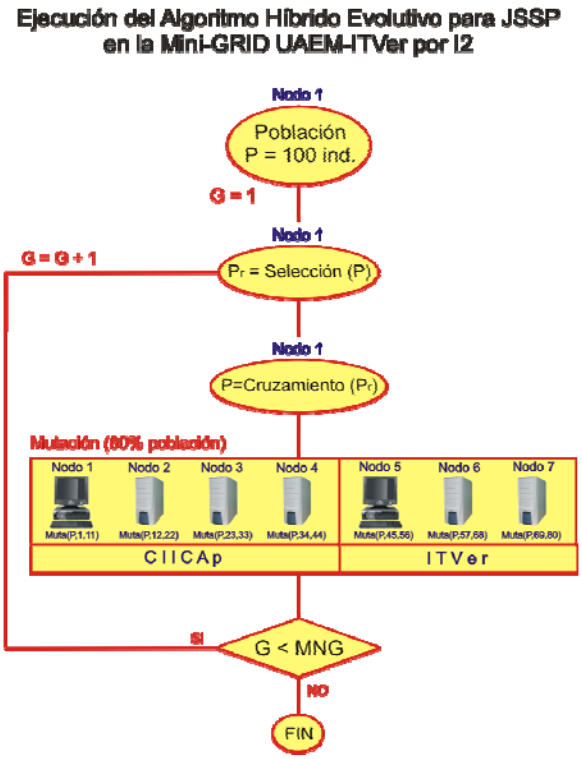


Figura 4. Ejecución del algoritmo híbrido evolutivo para JSSP en la Mini-GRID UAEM-ITVer por I2.

4. Infraestructura clusters

La infraestructura desplegada en Clusters de alto rendimiento en el CIICAp e ITVer está conformada con el mismo software, no así con el Hardware. El resultado son dos Clusters de alto rendimiento heterogéneos, debido a las diferencias de procesadores, capacidad de memoria y equipo de comunicaciones, tanto en los nodos maestros como en los nodos de procesamiento.

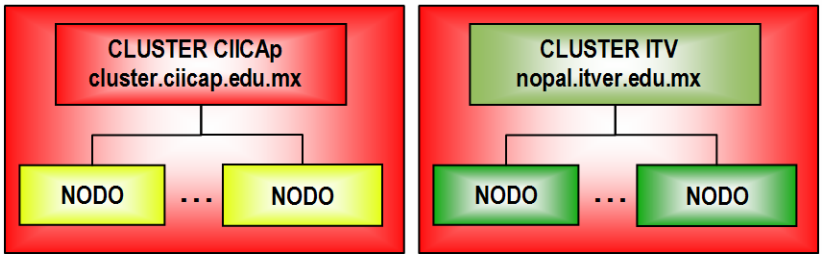


Figura 5. Clusters del CIICAp e ITV como entidades independientes.

Así, el primer panorama de esta infraestructura, es que cada entidad tiene su propia Cluster HPC, los equipos Hardware y Software se especifican en las siguientes tablas.

Cluster CIICAp.

Tabla 1. Infraestructura Hardware y Software del Clusters CIICAp.

ELEMENTO	HARDWARE	SOFTWARE
Comunicaciones	Switch Cisco C2960 24/10/100 Cableado interno nivel 5	
Nodo maestro	Pentium 4, 2793 MHz 512 MB Memoria 80 GB disco duro 2 tarjetas 10/100 Mb/s	S.O. Red Hat Enterprise Linux 4 Compilador gcc versión 3.4.3 OpenMPI 1.2.8 MPICH2-1.0.8 Ganglia 3.0.6 NIS ypserv-2.13-5 NFS nfs-utils-1.0.6-46 OpenVPN
Nodo de procesamiento 01	Pentium 2, 266 Mhz 256 MB Memoria 4 GB disco duro 1 tarjeta 10/100 Mb/s	
Nodo de procesamiento 02	Pentium 2, 133 Mhz 128 MB Memoria 4 GB disco duro 1 tarjeta 10/100 Mb/s	
Nodo de procesamiento 03 Nodo de procesamiento 04 Nodo de procesamiento 05	Intel® Pentium® Dual Core, 2800 MHz, 1GB RAM, 160GB disco duro, 1 tarjeta 10/100 Mb/s	
Nodo de procesamiento 06 Nodo de procesamiento 07 Nodo de procesamiento 08	Intel® Celeron® Dual Core, 2000MHz, 2 GB RAM, 160GB disco duro, 1 tarjeta 10/100 Mb/s	

Cluster ITVer.

Tabla 2. Infraestructura Hardware y Software del Cluster ITVer.

ELEMENTO	HARDWARE	SOFTWARE
Comunicaciones	Switch 3com 8/10/100 Cableado interno nivel 5	
Nodo maestro	Pentium 4, 2394 Mhz 512 GB Memoria 60 GB disco duro 2 tarjetas 10/100 Mb/s	S.O. Red Hat Enterprise Linux 4 Compilador gcc versión 3.4.3 OpenMPI 1.2.8 MPICH2-1.0.8 Ganglia 3.0.6 NIS ypserv-2.13-5 NFS nfs-utils-1.0.6-46 OpenVPN
Nodo de procesamiento 01	Pentium 4 Dual Core, 3200 Mhz 1 GB Memoria 80 GB disco duro 1 tarjetas 10/100 Mb/s	
Nodo de procesamiento 02	Pentium 4 Dual Core, 3201 Mhz 1 GB Memoria 80 GB disco duro 1 tarjetas 10/100 Mb/s	

5. integración de componentes Hardware y Software

Un Cluster de alto rendimiento se presenta como si fuera una sola máquina paralela, esta abstracción se debe a la integración de componentes Hardware y Software que entregan una serie de servicios al usuario final.

Los componentes Hardware son la red de comunicación que une los nodos de procesamiento con el nodo maestro y a su vez, el nodo maestro con las redes a la cuales se dará el servicio de procesamiento numérico.

Los componentes software son los servicios middleware que se ejecutan en el Cluster y nodos de procesamiento, que nos permiten enviar, procesar y monitorear los jobs (trabajos), así como observar el estado del Cluster.

6. Configuración del cluster

La configuración del Cluster se resume en las siguientes tareas básicas: instalación de Linux, configuración de redes y servicios, instalación de software adicional vía RPM y fuentes. En este documento no se darán detalles de la configuración de estas tareas.

Los componentes básicos para construir un Cluster de alto rendimiento con soporte para programación con OpenMPI y MPICH esta basado en la instalación y configuración del siguiente software y servicios:

- Configuración de la red privada.
- Configuración del sistema de archivos de red usando Network File System (NFS)
- Configuración de los servicios de información de red usando Network Information Service (NIS).
- Configuración de llaves públicas para habilitar la ejecución de trabajos remotos.
- Configuración de MPI para usar de OpenMPI y MPICH que es la implementación de código abierto de los estándares MPI-1 y MPI-2.
- Configuración de Ganglia para el monitoreo en tiempo real del Cluster.
- Configuración de Torque para administrar lo recursos de Cluster.
- Configuración de MAUI para la planificación de trabajos en el Cluster.

La integración de todos estos componentes es lo que da vida a un Cluster de alto rendimiento como se muestra en la figura 6.

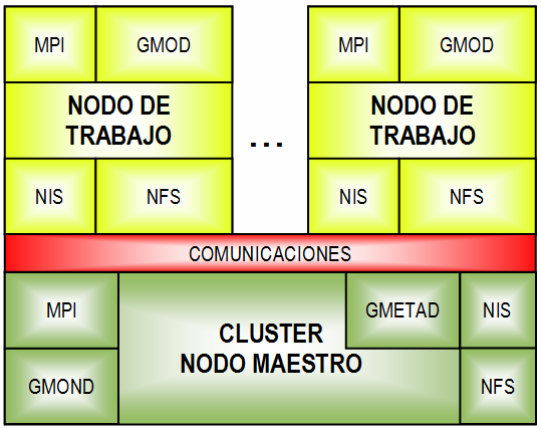


Figura 6. Componentes necesarios para implementar un cluster de alto rendimiento.

Este software puede variar dependiendo de las necesidades que se tengan, para el presente proyecto no utilizamos Torque + Maui, la razón, es que al tener un Cluster heterogéneo, la elección de los nodos para la ejecución de los trabajos no se basa en la disponibilidad, si no en los mas rápidos. Es por eso que hacemos una selección manual de los nodos.

Cada uno de los componentes de software que se instalaron requiere de dos procesos básicos que son instalación y configuración, estas tareas se llevan a cabo casi siempre sobre una terminal utilizando línea de comandos, no se darán detalles de estas tareas, pero si se explicará su funcionamiento a continuación.

Configuración básica de la Red. Se basa en la comunicación del nodo maestro a los nodos de procesamiento, esto es comúnmente una red privada con topología tipo estrella, la cuál se encuentra aislada de la red local para evitar conflictos de red y tráfico excesivo que haga más lenta la comunicación entre los nodos y sus procesos.

La puerta de entrada al Cluster es el nodo maestro, desde este punto se puede tener acceso a cualquier nodo de trabajo, pero sobre todo se pueden enviar trabajos para ser ejecutados en

el Cluster. Para usar el cluster, es necesario ir hasta el nodo maestro, hacer la copia de los programas y ejecutarlos.

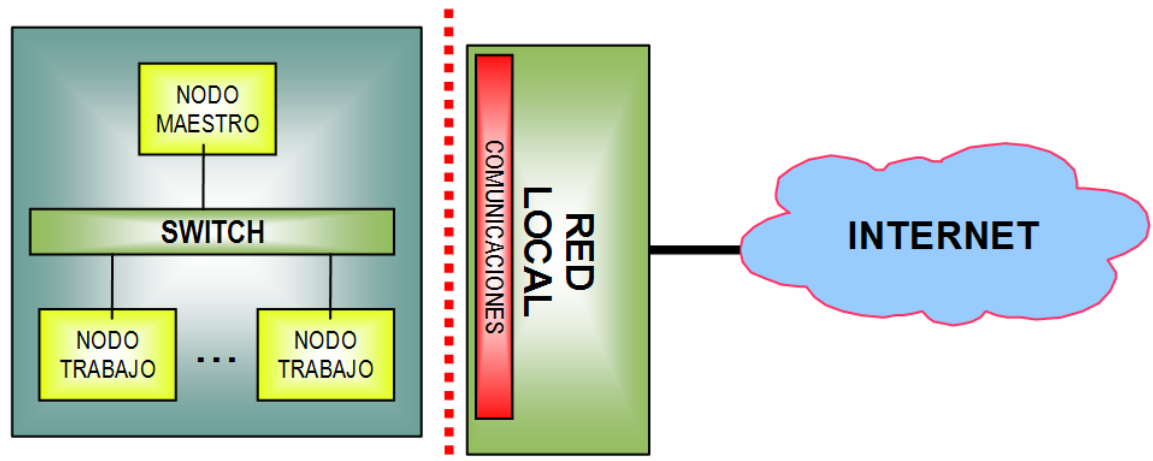


Figura 7. Configuración de la red privada, aislada del Clusters a la red local.

Esta configuración otorga la máxima seguridad, debido a que sólo tendrán acceso aquellas personas que físicamente puedan llegar hasta el nodo principal.

Configuración de la red para el Cluster. La construcción de un Cluster exige definir una subred privada, que a su vez este aislada de la red local.

La primera tarea es definir un rango de direcciones IP mediante la aplicación de una máscara de red, en nuestro Cluster el rango de direcciones va desde la 192.168.100.1 a 192.168.25.254. Este rango se tiene para escalar (incrementar) el número de nodos.

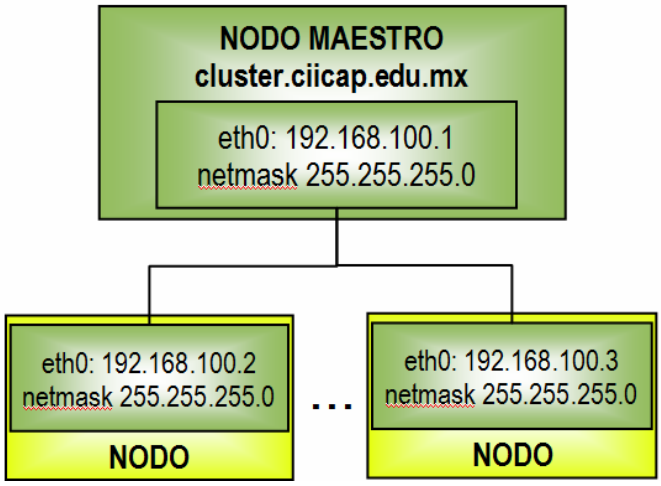


Figura 8. Configuración de la red.

Una vez configurada y probada la red, la siguiente tarea es asignarles nombres a los nodos, esto se realiza modificando el archivo /etc/hosts, en el cual se agregan tanto el nombre del

nodo maestro como de cada uno de los clientes.

```
[root@cluster ~]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost

192.168.100.1   cluster.ciicap.edu.mx   masterciicap
192.168.100.2   nodo01.ciicap.edu.mx    nodo01ciicap
192.168.100.3   nodo02.ciicap.edu.mx    nodo02ciicap
[root@cluster ~]#
```

Figura 9. Vista del contenido de los archivos /etc/hosts.

Este nombrado de nodos se lleva a cabo debido a la falta de un servidor DNS que haría la tarea de asignar nombres a los nodos en base a su dirección IP.

Configuración del Sistema de Archivos de Red. El servicio de archivos NFS, es un middleware que se encarga de compartir directorios base a todo un grupo de máquinas, su tarea principal es cuidar la integridad de los datos y mantener actualizado el sistema de archivos en todas las máquinas conectadas al servidor NFS.

Las características de este servicio lo hacen idóneo para su implementación en Clusters de alto rendimiento, debido a que el sistema de archivos de un Cluster debe estar conformado por un sólo directorio para todos los nodos del Cluster; esto es, cuando un archivo llegue al nodo maestro para su ejecución, automáticamente debe de estar disponible para todos los nodos del Cluster. Esta es una necesidad básica, el compartir un directorio base que contenga todos los usuarios del Cluster tiene como propósito la centralización que facilite la administración. Este puede mostrar un retardo en la réplica de datos para un número muy grande de nodos. Existe un problema que radica cuando el número de nodos aumenta a 1000, ya que éste número sobrepasa los límites permitidos por el NFS y será necesario implementar otro sistema de archivos.

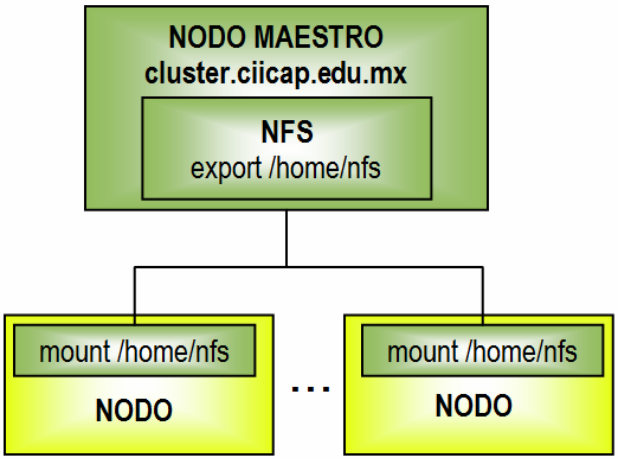


Figura 10. Configuración del sistema de archivos de red.

Configuración de los Servicios de Información de Red. El Network Information Service (NIS), es un middleware para la Autentificación de Usuarios de Red, su tarea principal es otorgar el acceso mediante una clave de usuario (login) y password.

Este servicio esta basado en un middleware cliente/servidor, lo cual indica que en el nodo maestro se ejecutará el servidor NIS, así como en todos los nodos cliente donde se encuentren configurados los servicios de NIS; estos servicios se conocen como *ypserv* y *ypbind* y están incluidos en casi todas las versiones de Linux. Una vez instalado y configurado se debe verificar que su arranque sea automático, para ello, es necesario revisar los niveles de ejecución de los servicios. Estas tareas rutinarias deben de hacerse sólo una.

Para gestionar un usuario se requieren dos cosas: 1) Crear el usuario con su password y 2) designar un directorio de trabajo. Estos dos elementos están presentes en el nodo maestro, para poder replicarlos en los nodos se requieren dos servicios: 1) NIS para autentificación de usuarios en el Cluster y 2) NFS para exportar el directorio de trabajo. Una vez configurados estos dos elementos, se tiene como resultado una gestión centralizada de usuarios del Cluster, lo que facilita su administración y mantenimiento.

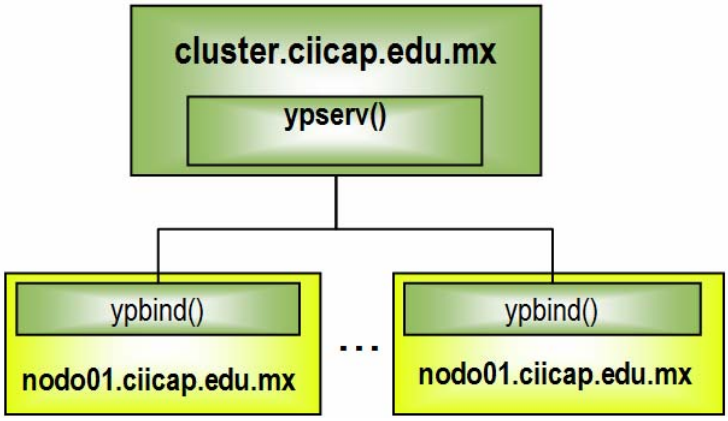


Figura 11. Configuración de los servicios de información de red.

Esta configuración permite crear un usuario que se encuentre disponible para todo el Cluster; como se observa en la figura 12 (usuario MPICH), para ello utilizamos el comando *adduser* y le indicamos que su directorio de trabajo es */home/nfs*, este hecho asegura que el directorio del usuario está disponible para todo el Cluster. Después de crear el usuario se requiere actualizar la información que utiliza el servidor NIS para que este disponible para todos los nodos del Cluster, tal y como se observa en la figura 07; para ello entramos al directorio /var/yp y utilizamos el comando *make*. Una vez realizado esto, es necesario acceder a la cuenta de usuario previamente realizada, para lo cuál tendremos que acceder al usuario MPICH para después mostrar el nombre del nodo donde estamos actualmente que en este caso es el nodo maestro (cluster.ciicap.edu.mx), paso seguido nos intentamos

conectar vía SSH al primer nodo (nodo01.ciicap.edu.mx), al realizar esto, el nodo utiliza los servicios de NIS para confirmar el password que se tecleó, como dicho usuario ya existe en el nodo maestro, NIS permite el acceso. Ahora se muestra el nombre del nodo donde se está trabajando que en este caso es nodo01.ciicap.edu.mx y por último se muestra su directorio de trabajo que es /home/nfs/mpich, que en realidad es el directorio exportado por el nodo maestro en modo lectura/escritura y al cuál tienen acceso todos los nodos que lo han montado previamente.

```
[root@cluster ~]$ adduser -h /home/nfs/mpich -m -p mpich
[root@cluster ~]$ cd /var/yp
[root@cluster ~]$ make
[root@cluster  ~]$ su mpich
[mpich@cluster root]$ hostname
cluster.ciicap.edu.mx
[mpich@cluster root]$ pwd
/home/nfs/mpich
[mpich@cluster root]$ ssh nodo01.ciicap.edu.mx
mpich@nodo01.ciicap.edu.mx's password:
Last login: Thu Apr  2 12:43:25 2009 from cluster.ciicap.edu.mx
[mpich@nodo01 ~]$ hostname
nodo01.ciicap.edu.mx
[mpich@nodo01 ~]$ pwd
/home/nfs/mpich
```

Figura 12. Shell para la creación de un usuario en el Cluster.

De esta forma es como se integran los servicios NIS + NFS para la gestión centralizada de usuarios que se utiliza en prácticamente cualquier tipo de Cluster de Alto Rendimiento. Los esquemas de autentificación y los sistemas de archivos pueden variar, pero la lógica de funcionamiento sigue siendo la misma: compartir y autentificar.

Configuración de llaves publicas. El Cluster se muestra al usuario final como una sola máquina paralela con múltiples procesadores, pero para que esto suceda, lo que el Cluster necesita es lanzar programas en forma remota hacia todos los nodos del Cluster sin necesidad de pedir autorización (login y password). Para lograr esto es necesario configurar el uso de llaves publica.

OpenSSH dispone de varios métodos para verificar la identidad de un usuario remoto, uno de ellos es el uso de contraseñas de usuarios; otro de los métodos se basa en la autentificación RSA, en donde se dispone de un juego de llaves privada/pública que garantiza la identidad del usuario que intenta conectarse al equipo remoto (nodos). El juego de llaves tiene la propiedad de que lo que se encripta con una llave privada, sólo se puede desencriptar con una pública; sin embargo, a partir de una llave pública no se puede derivar una privada.

Para asegurarse de que un usuario sea el autorizado para acceder al cluster se utiliza criptografía pública OpenSSH.

Para generar las llaves públicas se utiliza el comando *ssh-keygen* el cuál genera una llave privada y otra pública con los nombres de *id_rsa* e *id_rsa.pub* respectivamente; estos dos archivos son depositados en la carpeta *.ssh* del directorio */home* del usuario como se

observa en la figura 13.

```
[mpich@cluster ~]$ ls .ssh -la
drwx------  2 mpich mpich 4096 mar  1 13:24 .
drwxr-xr-x  6 mpich mpich 4096 mar  2 12:45 ..
-rw-------  1 mpich mpich  471 feb 26 12:30 authorized_keys
-rwx------  1 mpich mpich  887 feb 26 12:28 id_rsa
-rwx------  1 mpich mpich  237 feb 26 12:28 id_rsa.pub
-rwx------  1 mpich mpich 1670 mar  2 09:25 known_hosts
```

Figura 13. Shell para la creación de un usuario en el Cluster.

Una vez que se ha obtenido la llave pública, sólo es cuestión de agregar su contenido al archivo *authorized_keys* y modificar los permisos para que sean de sólo lectura y acceso exclusivo al usuario propietario. Este archivo debe de copiarse a cada uno de los nodos del Cluster para asegurar el acceso sin necesidad de proporcionar password. La carpeta *home/nfs/user/.sh* es parte del directorio */home/nfs* que exporta el servicio NFS, tal y como se puede observar en la figura 14. De ésta forma es replicado automáticamente a todos los nodos del Cluster. De esta manera aseguramos el acceso confiable al Cluster, para poder ejecutar algoritmos en forma remota como lo exige MPI.

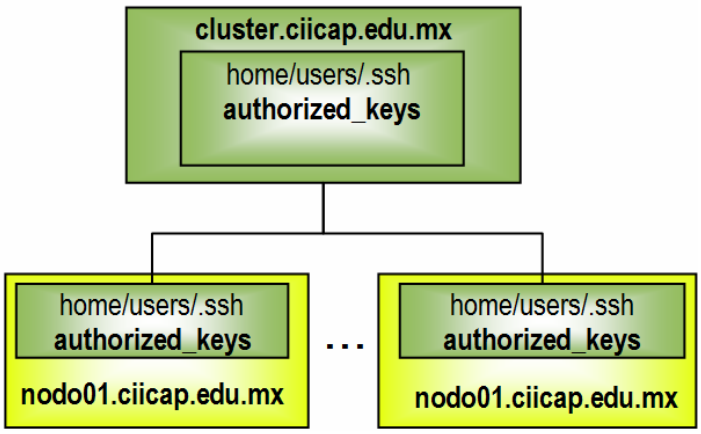


Figura 14. Compartiendo llaves públicas en authorized_keys.

La figura 15 muestra una terminal abierta en el nodo maestro (cluster.ciicap.edu.mx), misma que realiza una conexión remota vía SSH al primer nodo (nodo01.ciicap.edu.mx), después hace un acceso remoto hacia el segundo nodo (nodo02.ciicap.edu.mx) y por último hacia el nodo maestro (cluster.ciicap.edu.mx). En ninguna de las conexiones se pide password, debido a que la autentificación se basa en llaves públicas RSA.

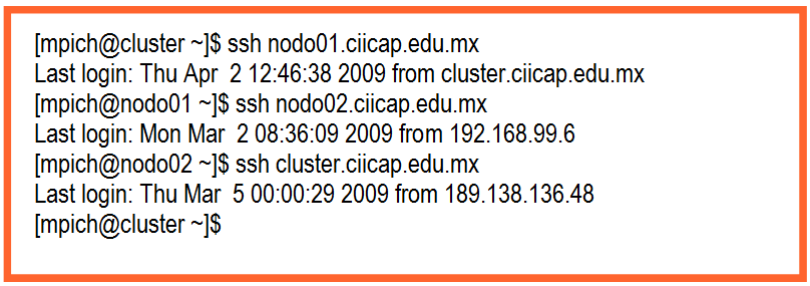


Figura 15. Shell para comprobar el acceso mediante llaves públicas en authorized_keys.

Configuración de MPI (Message Passing Interface) o Interfaz de Paso de Mensajes, es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores en múltiples maquinas. El paso de mensajes es una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua, de manera similar a como se hace con los semáforos, monitores, etc. Su principal característica es que no precisa de memoria compartida, por lo que es muy importante en la programación para sistemas distribuidos.

En este Cluster se usan dos implementaciones de código abierto de los estándares MPI-1 y MPI-2 que son OpenMPI y MPICH; estas dos versiones están disponibles para ser descargadas de Internet para compilarlas e instalarlas en el Cluster. Para realizar la instalación es necesario seguir las tres siguientes etapas:

- Instalación y configuración de MPI que incluye tareas de descarga, descompresión de archivos, compilación y generación de binarios.
- Configuración de los usuarios para hacer uso de MPI.
- Pruebas.

Existen muchas opciones de compilación para los fuentes de OpenMPI y MPICH, la opción que se utilizó fue en direccionamiento de los binarios hacia un directorio de instalación utilizando la directiva *–prefix* para *configure*, seguida de *make* y *make install*. Con estos pasos obtenemos un directorio que contendrá tanto los binarios como las librerías de MPI.

Obsérvese que estas tareas pueden hacerse con el usuario *root* y sin la directiva *–prefix*, esto asegura que los binarios y librerías están disponibles para todos los usuarios que se crean en el Cluster, con la única observación de que los binarios para versiones de MPI se llaman exactamente igual; entre ellos se encuentran los comandos para compilar *mpicc* y de ejecución *mpiexec* y *mpirun*.

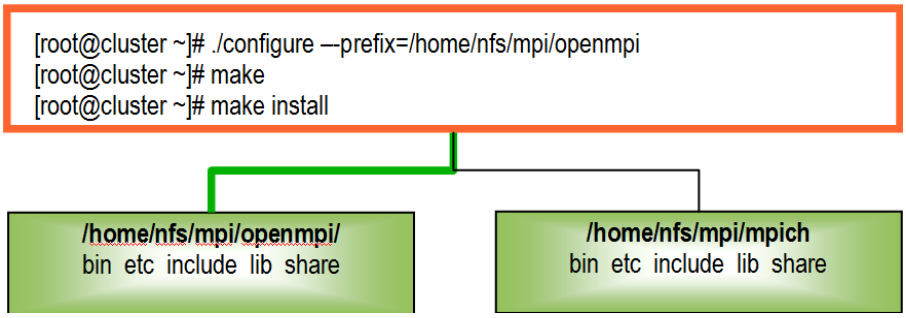


Figura 16. Generación de binarios a partir de compilar los fuentes MPI.

Debido a esto, se optó por la generación de binarios en directorios diferentes, donde cada directorio corresponde a una versión de MPI en particular (Figura 16), lo que conlleva a la necesidad de configurar las variables de entorno específicas para cada usuario del Cluster.

Configuración de usuarios. Cada usuario creado en el Cluster debe poder trabajar con una versión particular del MPI, por ello es necesario configurar las variables de entorno que corresponda al tipo de shell que el usuario elija, generalmente se usa *bash.* Para el tipo de shell que se está utilizando, se modifican *.bashrc* y *.bash_profile* del */home* de cada usuario, también es posible modificar */etc/bashrc* y */etc/profile* para afectar las variables de entorno de todos los usuarios.

Ya sea que se decida modificar las variables de entorno para un solo usuario o para todos, MPI requiere de configurar el camino de búsqueda hacia sus binarios y adicionalmente configurar una variable de entorno para las librerías, estas variables son *PATH* y *LD_LIBRARY_PATH*.

Los archivos a modificar y posibles escenarios para establecer las variables de entorno se enlistan a continuación:

- **Para todos los usuarios**. Se requiere permisos de *root* para editar los archivos (parte izquierda de la figura 17).
    - /etc/profile --> Se ejecuta cuando cualquier usuario inicia la sesión.
    - /etc/bashrc --> Se ejecuta cada vez que cualquier usuario ejecuta el *bash*.

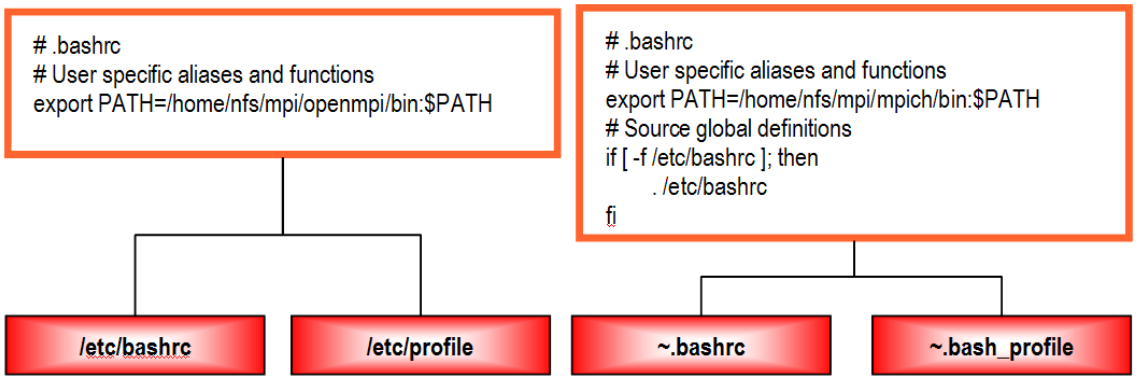


Figura 17. Modificación de las variables de entorno.

- **Para un usuario específico**. Cada usuario edita los archivos de su directorio */home* (parte derecha de la figura 13).
    - ~/.bash_profile --> Se ejecuta el *.bash_profile* de un usuario cuando el usuario inicia su sesión.
    - ~/.bashrc --> Se ejecuta el *.bashrc* del usuario cuando el usuario ejecuta el programa bash.

En este caso se modificaron las variables de entorno de cada usuario del Cluster, que trae como consecuencia que algunos usuarios no puedan usar MPI, lo que resulta idóneo para crear usuarios cuyo trabajo sea ajeno a ejecutar programas MPI, tales como mantenimiento, administradores, cursos, etc.

Monitor de redes Ganglia. Ganglia es un sistema de monitoreo distribuido para Sistemas de Cómputo de Alto Rendimiento y Grids. Está diseñado para tener un consumo muy bajo de recursos por nodo y una alta concurrencia; actualmente se encuentra en uso en miles de sistemas alrededor del mundo y soporta un manejo de hasta 2000 nodos.

Ganglia se ejecuta con dos demonios: el *Ganglia Monitoring Daemon* (***gmond***) y el *Ganglia Meta Daemon* (***gmetad***), además del *Ganglia Web Frontend*.

El *Ganglia Monitoring Daemon (gmond)*, es un servicio multi-thread, el cual corre en cada uno de los nodos del Cluster que se desea monitorear (fig. 18). No se tiene porque tener un sistema de archivo NFS en común, ni una base de datos especifica. *Gmond* tiene su propia base de datos distribuida y su propia redundancia.

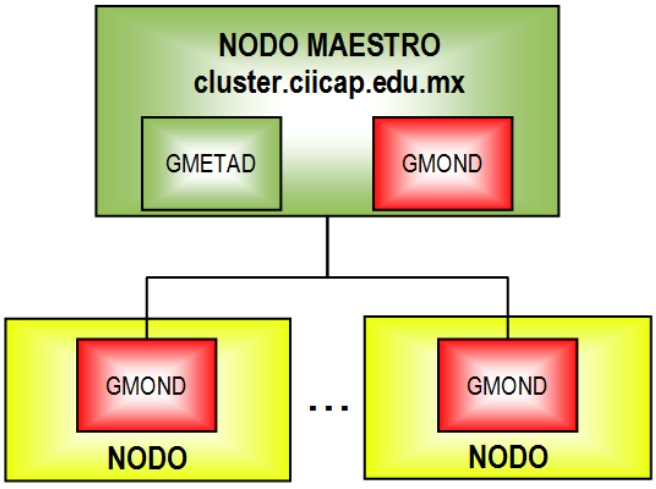


Figura 18. Pruebas realizadas con MPICH.

El *Ganglia Meta Daemon (gmetad)*, es un servicio que permite recopilar la información a través de los servicios de *gmond*, el cuál se ejecuta en cada nodo a monitorear (fig. 18), los datos son enviados en un formato XML en intervalos regulares de tiempo desde los nodos,

que es almacenada por *gmetad* en una base de datos propia. Esta información recopilada en tiempo real es leída y mostrada mediante el sistema *Web de Ganglia*.

Ganglia Web Frontend: Es un sistema Web basado en PHP que requiere de un contenedor de aplicaciones Web como Apache. Provee una serie de vistas con información dinámica del estado de salud de todo el Cluster o Grid en tiempo real, lo que resulta especialmente útil para administradores y usuarios del sistema.

Una vez instalado provee una serie de páginas que muestran la adquisición de diferentes parámetros como niveles de carga, memoria, velocidad, tiempo de ejecución, entre otros.

La figura 19 presenta la vista principal de la MiniGRID Tarantula. En esta vista, se presenta el monitoreo del número de procesos en ejecución de la GRID y el número de clusters que componen GRID, en este caso se tienen dos clusters. Se puede observar en la figura que el número de CPU's totales de la MiniGRID es de 20 repartidos en 12 nodos. 15 CPU's pertenecen al cluster CIICAp y 5 al cluster Nopal. También se observa el monitoreo de forma independiente de cada Cluster que forma la GRID.

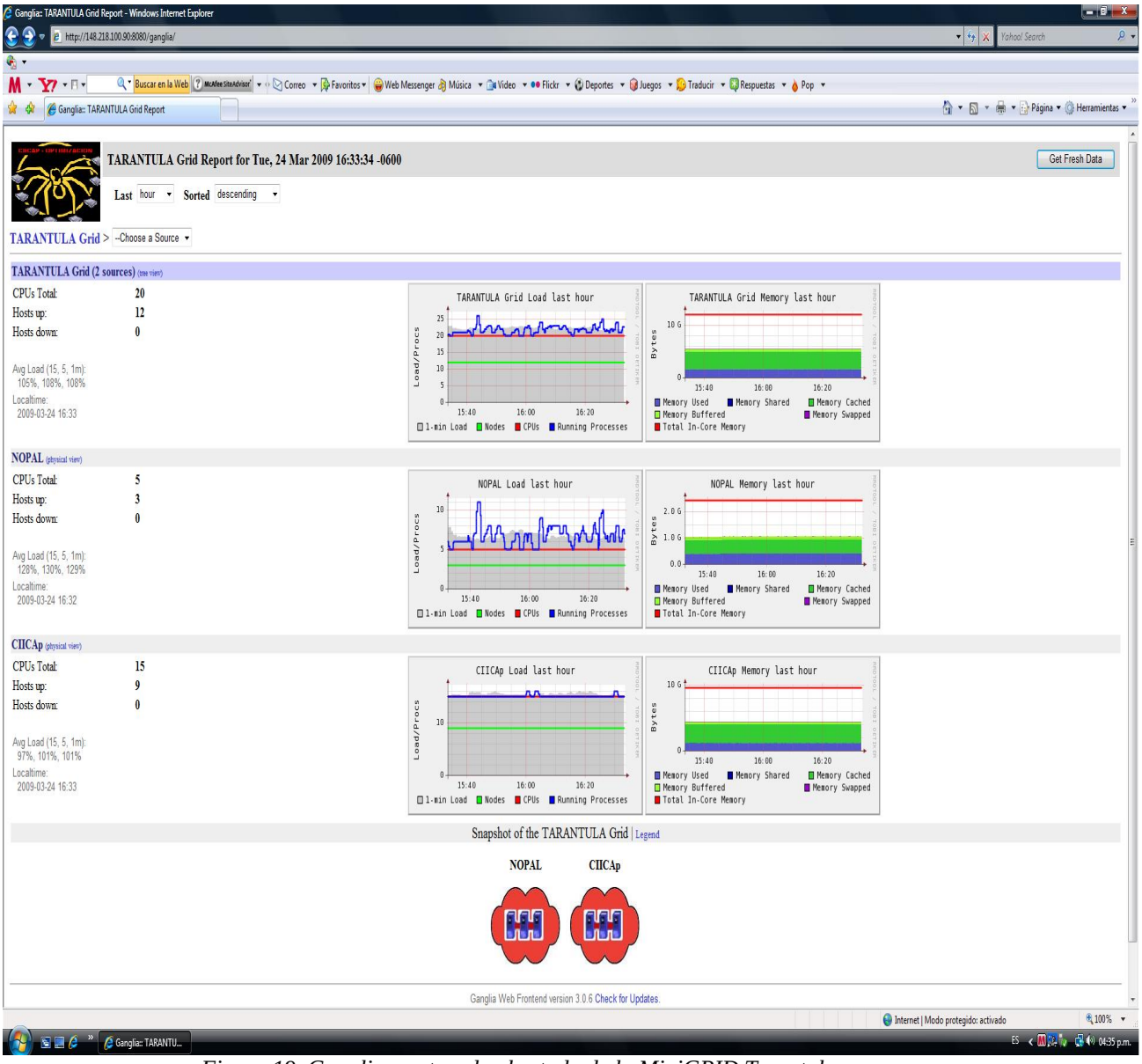


Figura 19. Ganglia mostrando el estado de la MiniGRID Tarantula.

En la ejecución de los programas MPI, sólo se utiliza un número de nodos pequeño y nodos heterogéneos, debido a esto la selección de nodos no se basa en la disponibilidad, si no en los más rápidos. Es por eso que se realiza una selección manual de los nodos directamente sobre las sentencias *mpirun* y *mpiexec* al usar OpenMPI y MPICH (no se utiliza Torque y Maui). A futuro, al contar con una infraestructura más robusta y homogénea se pondrán a punto estos dos elementos.

7. Pruebas en el cluster

Pruebas MPI. Las pruebas que se realizan en el Cluster para comprobar la correcta

configuración de MPI, consisten en compilar y ejecutar un programa (Fig. 20) que haga uso de las librerías de MPI.

Posteriormente se envía el programa a todos los nodos, lo que también permite probar la configuración del NIS + NFS, en caso de dar una salida de error denotaría la mala configuración del Cluster.

```
#include <stdio.h>
#include <unistd.h>
#include "mpi.h"

int main(int argc, char *argv[]){
int rank; int size; char hostname [126]; int len = 126;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);

gethostname(hostname, len);
printf("Hola desde %d en la maquina:%s de un total de: %d\n", rank, hostname, size);
MPI_Finalize();
return 0;
}
```

Figura 20. Programa MPI de prueba.

En la primer prueba mostrada se utiliza OpenMPI con dos variantes, la primera se basa en la obtención de los nodos destino a partir de un archivo previamente configurado (Fig. 21A) y la segunda se especifican los nodos explícitamente sobre la línea de comandos (Fig. 21B).

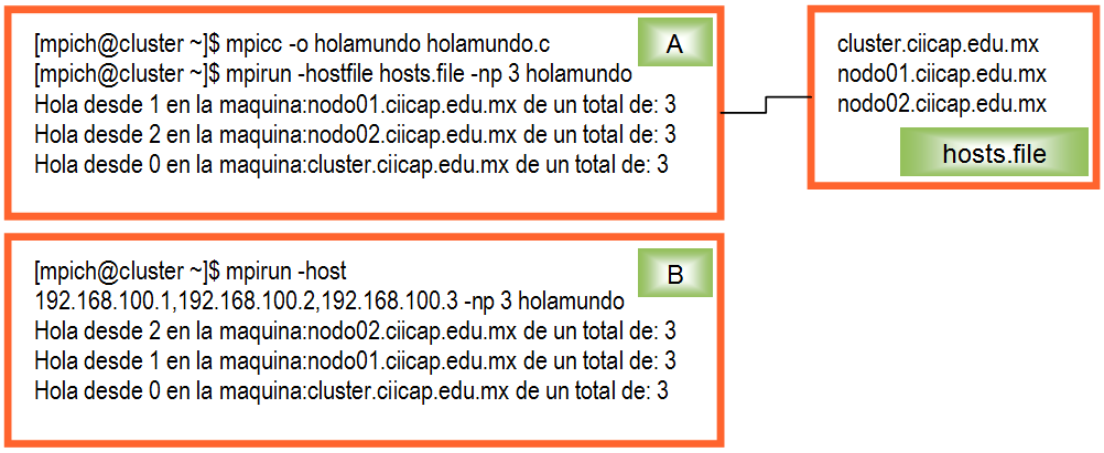


Figura 21. Pruebas realizadas con MPI.

En la segunda prueba mostrada se está utilizando MPICH, también con dos variantes, y que al igual que la primera prueba se basa en un archivo previamente configurado, mientras que la siguiente se da explícitamente sobre al línea de comandos (Fig. 22A y 22B

respectivamente).

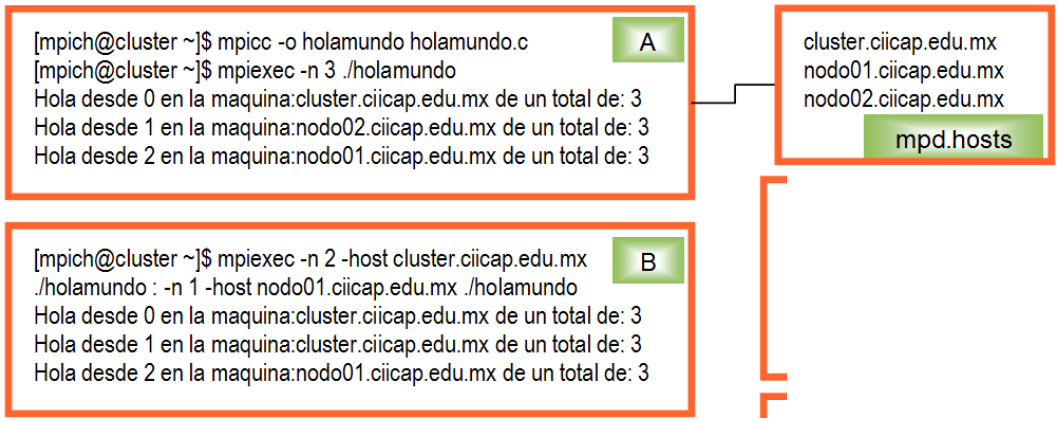


Figura 22. Pruebas realizadas con MPICH.

Las dos versiones de MPI (OpenMPI y MPICH) comparten muchas características semejantes como compilación y ejecución de trabajos, MPICH requiere la ejecución previa de su ambiente de trabajo con *mpdboot*; pero trabajando en ambientes Grid ambos imponen ciertas restricciones a sus entornos de ejecución, siendo más flexible MPICH, por lo que en el presente proyecto se decidió utilizar MPICH para la programación de los algoritmos.

Cálculo del ancho de banda [Sloan, 2001]. Se puede utilizar ping para calcular una estimación aproximada del rendimiento de una conexión. El comando ping es la herramienta por excelencia para comprobación de conectividad a nivel de red. Ping envía un paquete al nodo destino solicitando al emisor que regrese una respuesta con un paquete del mismo tamaño, a este tiempo se le conoce como Round Trip Time (RTT) o tiempo que tarda un paquete en ir del nodo fuente al nodo destino y del nodo destino al nodo fuente.

El procedimiento que se sigue es enviar con un ping dos paquetes de diferente tamaño cuatro veces y tomar la media. Esto se hace con la opción -c -s, que especifica el número de envíos y el tamaño del paquete en bytes respectivamente, el tamaño se ve afectado con 8 bytes adicionales para el encabezado de control. La diferencia en tiempos nos da una idea de cuánto tiempo se tarda en enviar los datos adicionales en el paquete más grande. Por ejemplo, si un ping con 100 bytes (92 de datos + 8 de encabezados) tarda 30 milisegundos y 60 milisegundos con 1100 bytes (1092 de datos + 8 de encabezado), entonces, tiene un viaje de ida y vuelta de 30 milisegundos o 15 milisegundos para enviar los 1000 bytes adicionales u 8000 bits en una dirección. El rendimiento es de aproximadamente 8000 bits por 15 milisegundos o 540,000 bps. La diferencia entre dos mediciones, se utiliza para eliminar la sobrecarga (suma de la transmisión, la propagación y cola retrasos).

Prueba sin tráfico de mensajes en la MiniGRID Tarántula

Se realizaron pruebas sin carga de procesos que se deriven de la ejecución de los algoritmos evolutivos.

Para realizar el cálculo del nodo maestro (cluster.ciicap.du.mx) al primer nodo de procesamiento (nodo01.ciicap.edu.mx) se utiliza el comando *ping* con un tamaño de 100 bytes cuatro veces, lo que da una media del tiempo de ida y vuelta (RTT) de 0.173 milisegundos (figura 23a) y con un paquete de 1100 bytes da una media de 0.511 milisegundos (figura 23b), la diferencia entre estos dos tiempos nos da 0.338 milisegundos que tarda en transmitir 1000 bytes u 8000 bits de un extremo a otro de ida y vuelta o 0.169 milisegundos sólo de ida.
Para calcular la velocidad en Mbps., lo primero que hacemos es convertir 0.169ms a segundos, que nos daría el número de paquetes que se pueden enviar en un segundo, a continuación lo multiplicamos por 8000 bits que corresponde al contenido de cada paquete, que nos da la tasa de transferencia en bps. Este resultado lo dividimos entre 1024 para convertirlo en Kbps y se vuelve a dividir por 1024 para convertirlo en Mbps. El resultado final es una tasa de transferencia de 45.2 Mbps.

$DA = (PG - PP)8$
$t_{ida} = (RTTPG - RTTPP)/2$
$AB = DA(1000)/[(t_{ida})(1024)^2]$

$DA = (1100 – 100)8 = 8000$ bits (fig23a y fig 23b)
$t_{ida} = (0.511 – 0.173)/2 = 0.169$ ms (fig23a y fig 23b)
$AB = 8000(1000)/[(0.169)(1024)^2] = 45.2$ Mbps

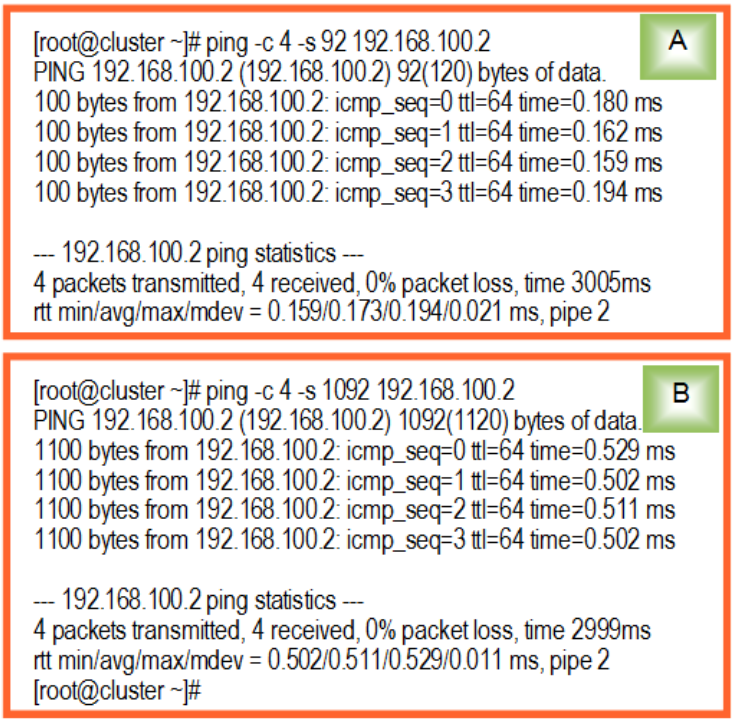


Figura 23. Pruebas realizadas con MPICH.

RTT = tiempo que tarda un paquete en ir de la fuente al destino y del destino a la fuente, bytes
RTTPG = tiempo que tarda un paquete en ir de la fuente al destino y del destino a la fuente, bytes
RTTPP = tiempo que tarda un paquete en ir de la fuente al destino y del destino a la fuente, bytes
DA = Datos adicionales del paquete más grande, bits
PG = Paquete grande, bytes
PP = Paquete pequeño, bytes
t_{ida} = tiempo de envío de datos adicionales el paquete más grande sólo de ida, ms
AB = Ancho de banda, Mbps

Los cálculos completos se enlistan a continuación, observe que los Clusters están aislados en una red privada, las tasas de transferencia entre sus nodos sólo manejan el tráfico de ellos mismos, por lo que no se ven afectados por el tráfico externo y para ellos sólo fue necesario realizar una sola medición.

Cluster CIICAp.

NODO FUENTE	NODO DESTINO	TIEMPO 100 bytes	TIEMPO 1100 bytes	TIEMPO DIFERENCIA	ANCHO DE BANDA
cluster.ciicap.edu.mx	nodo01.ciicap.edu.mx	0.173 ms	0.511 ms	0.338 ms	45.2 Mbps
cluster.ciicap.edu.mx	nodo02.ciicap.edu.mx	0.242 ms	0.558 ms	0.316 ms	50.6 Mbps
nodo01.ciicap.edu.mx	nodo02.ciicap.edu.mx	0.275 ms	0.655 ms	0.400 ms	38.2 Mbps

Cluster ITVer.

NODO FUENTE	NODO DESTINO	TIEMPO 100 bytes	TIEMPO 1100 bytes	TIEMPO DIFERENCIA	ANCHO DE BANDA
nopal.itver.edu.mx	nodo01.itver.edu.mx	2.065 ms	0.442 ms	1.623 ms	9.4 Mbps
nopal.itver.edu.mx	nodo02.itver.edu.mx	2.067 ms	0.451 ms	1.614 ms	9.5 Mbps
nodo01.itver.edu.mx	nodo02.itver.edu.mx	2.047 ms	0.435 ms	1.612 ms	9.4 Mbps

Observe que las velocidades de transferencia entre un Punto A y B es la misma que entre B y A, por tanto se cubren todas las combinaciones para este caso donde la red esta aislada

Cálculo de latencia. La latencia es el retardo en el envió de un paquete de tamaño ***n*** del nodo fuente al nodo destino que incluye además el tiempo de transmisión, propagación y cola de retrasos.

Primeramente el procedimiento que se sigue es el mismo que el utilizado anteriormente para calcular el ancho de banda. En el caso del cálculo entre el cluster.ciicap.edu.mx y el nodo01.ciicap.edu.mx, el tiempo real que se necesita para enviar 1000 bytes de un extremo a otro y de regreso es de 0.338 ms. De esta manera podemos calcular que el tiempo real para 100 bytes sería 0.0338 ms. (0.338 ms. entre 10).

Para calcular el tiempo de transmisión, propagación y cola de retrasos sería restar al RTT para 100 bytes, el tiempo real de enviar 100 bytes (0.173 menos 0.0338 ms) resulta ser de **0.1392 ms**, de igual manera el RTT para enviar 1100 bytes es 0.511 ms menos el tiempo real de enviar 1100 bytes (0.0338ms por 11ms) es igual a **0.1392 ms**, ambos dan el mismo retraso.

Observe que de los 1000 bytes enviados, 992 son de datos y 8 del encabezado, por tanto la latencia para enviar 992 bytes de ida y vuelta es el tiempo real de enviar 1000 bytes (0.0338 ms. por 10) más el tiempo de transmisión, propagación y cola de retrasos (0.1392 ms) lo que nos da una latencia de **0.4772 ms**. o **0.2386 ms.** en un sólo sentido. Lo mismo para 100 bytes (92 reales + 8 encabezado) sería 0.0338 ms más el tiempo de transmisión, propagación y cola de retrasos (0.1392 ms) lo que nos da una latencia de **0.173 ms o 0.0865 ms** en un sólo sentido.

Cluster CIICAp.

NODO FUENTE	NODO DESTINO	TIEMPO 1000 bytes	RETRASO 1000 bytes	LATENCIA 1000 bytes
cluster.ciicap.edu.mx	nodo01.ciicap.edu.mx	0.338 ms	0.139 ms	0.477 ms
cluster.ciicap.edu.mx	nodo02.ciicap.edu.mx	0,316 ms	0.210 ms	0.526 ms
nodo01.ciicap.edu.mx	nodo02.ciicap.edu.mx	0,400 ms	0,235 ms	0.635 ms

Cluster ITVer.

NODO FUENTE	NODO DESTINO	TIEMPO 1000 bytes	RETRASO 1000 bytes	LATENCIA 1000 bytes
nopal.itver.edu.mx	nodo01.itver.edu.mx	1.623 ms	1.902 ms	3.525 ms
nopal.itver.edu.mx	nodo02.itver.edu.mx	1.614 ms	1.9057 ms	3.519 ms
nodo01.itver.edu.mx	nodo02.itver.edu.mx	1.612 ms	1.885 ms	3.497 ms

Sobre esta base, se puede hacer el cálculo para enviar un paquete con un tamaño específico de bytes, mediante una simple división del tamaño de los datos a enviar en bytes, entre 992 bytes que es el tamaño real de datos, el resultado multiplicarlo por 0.338 que es el tiempo real en transmitir 992 bytes y agregarle el tiempo de transmisión, propagación y cola de retrasos que es de 0.1392 (TR).

Así con libre tráfico, para el peor de los casos en las pruebas del algoritmo evolutivo para JSSP, en el problema mayor de prueba (20x20) y como se observó en la sección 3.3, se envía sólo 4 individuos por cada proceso, entonces de acuerdo al tipo de dato utilizado en el algoritmo, el cual es entero de 4 bytes. El flujo de datos a enviar en cada proceso depende del tamaño de la estructura, en este caso, el tamaño de la estructura que representa a un individuo es de 12800 bytes, por lo que en cada CPU del cluster CIICAp se enviará un flujo de 51200 bytes de forma local, el cálculo de latencia de forma local lanzando el algoritmo desde el cluster CIICAp sería:

t_{real} para enviar 100 bytes = $t_{ida}/10$
TR = Tiempo de Retraso = RTT para enviar 100 bytes - t_{real} para enviar 100 bytes
TR = 0.173-0.338/10 = 0.1393 ms
Latencia = TRTP + TR
TRTP = Tiempo Real de Transmisión de un paquete de n datos en ms.

Latencia de un paquete de 992 bytes = 0.338 + 0.1392 = 0.4772 ms

Latencia de un paquete mayor (PM)
Latencia = PM*Latencia/992 = 51200bytes*0.4772/992bytes
Latencia = 24.63 ms
Debido a que la comunicación entre nodos debe ser en ambos sentidos la latencia generada en cada iteración del algoritmo (cada generación) será.
Latencia = 2*24.63 ms.
Latencia = 49.26 ms.

8. Configuración de servicios al usuario final

Una vez construido y probado, el Cluster se encuentra potenciado con una serie de servicios cuyos consumidores finales se encuentran divididos en tres escenarios que son: 1) usuarios privados, 2) usuarios de redes locales y 3) usuarios globales.

Una vez construido el Cluster, los servicios de procesamiento numérico deben ser proporcionados a los potenciales usuarios, para ello es necesario configurar la red de servicios al usuario final y que esta compuesta de las siguientes tareas:

1. Configuración de la red para usuarios locales.
2. Configuración de al red para usuarios locales y globales.

Para garantizar el acceso a estos usuarios, la construcción de un Cluster debe darse en forma escalar y atendiendo las necesidades de conectividad de los usuarios a cualquier hora en cualquier lugar, a continuación se definen estos tres escenarios.

Configuración de red para usuarios locales. Para poder extender la cobertura de los servicios del Cluster a la red local, es necesario que el nodo maestro se encuentre dentro de la red, de esta manera los usuarios aseguran el acceso al nodo maestro y a la ejecución de trabajos.

Observe que esta configuración esta entre dos subredes diferentes A y B, es decir, es necesario contar con dos tarjetas de red, una configurada para la *subred A*, que en este caso sería el área oscura de la red privada y otra para la *subred B* que sería la subred local marcada con azul. Así el acceso hacia la red Privada A se logra a través de la red local B.

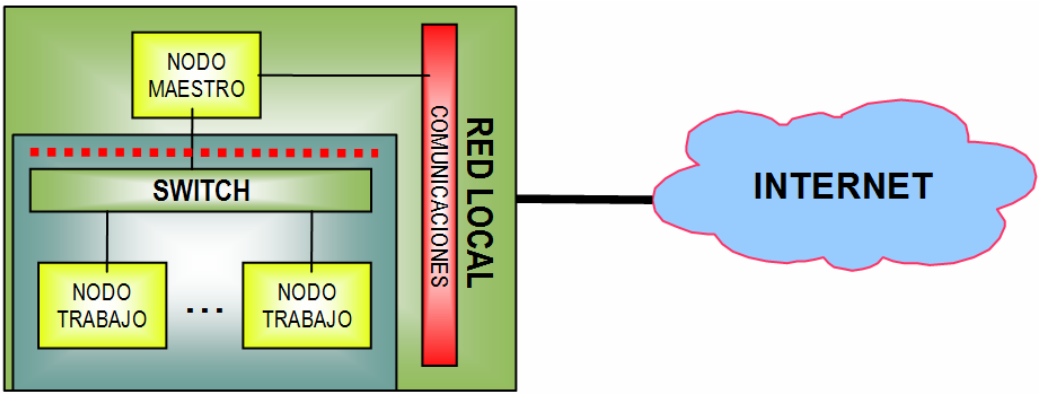


Figura 24. Configuración del Cluster para dar servicio a la red local.

En esta configuración observamos que el nivel de seguridad es un tanto mas bajo, debido a que ahora todos los usuarios dentro de la red local, en teoría, podrían tener acceso. En este punto es necesario tener controles de seguridad no tan estrictos, ya que aún no representa un grado de peligrosidad alto. Nuestro mayor temor es que alguna persona con conocimientos avanzados de Unix pudiera tener acceso al cluster y modificar archivos de configuración o cuentas.

De esta forma es posible extender los servicios a una red local o redes locales dentro de un campus, estableciendo puentes de enlace entre diferentes subredes.

Una ultima configuración es cuando necesitamos extender los servicios a la red global, es decir a Internet y de esta manera asegurar el acceso desde cualquier lugar del mundo en donde se disponga de una conexión a Internet.

Configuración de red para usuarios globales. Para asegurar el acceso al Cluster desde cualquier lugar, es necesario que el nodo maestro tenga un dirección IP Pública adicional a las dos anteriores definidas, en teoría se requiere de tres tarjetas de red, aunque en la práctica existen maneras de virtualizar al menos dos tarjetas.

Para asegurar el acceso a lugares donde se cuente con algún departamento de redes centralizado, es necesaria la apertura de puertos específicos para la dirección Pública, esta parte puede demorar una gran cantidad de tiempo si no se tiene el conocimiento o la actitud de servicio.

El ofrecer los servicios de un Cluster a la red Global, representa un gran riesgo de seguridad debido a que cualquier persona en cualquier parte del mundo pudiera acceder al nodo maestro y de ahí acceder a todas las redes o subredes a que se tiene acceso mediante el nodo maestro, por ello es imperativo implantar los niveles de seguridad mas altos.

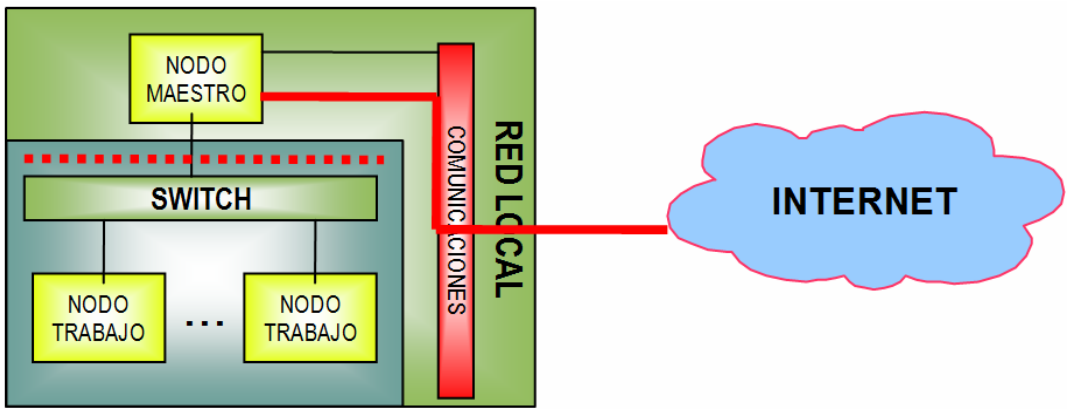


Figura 25. Configuración del Cluster para dar servicio a la red local.

El método común de acceso es el uso mediante los servicios de Secure Shell (SSH) por el puerto 22, un método seguro es cambiar este puerto por otro, una vez mas, esperando no depender de agentes externos devoradores de tiempo. El habilitar el acceso desde cualquier lugar es también una comodidad de trabajo, ya que prácticamente podemos trabajar en el Cluster en cualquier momento y en cualquier lugar que disponga de una conexión a Internet.

Por otro lado también es importante tomar en cuenta que cuanto mas nos alejamos del nodo principal mas se incrementa la latencia, normalmente no deberíamos tener problemas en trabajar estando en la red local, pero sí estando conectados desde otra ciudad o incluso en otro país, algunas localidades, ciudades o países no cuentan con servicios de alta velocidad.

Observemos que esto no afecta en el desempeño del Cluster, solo afecta la transferencia de los trabajos desde un punto externo hacia el nodo maestro, así mismo la transferencia del flujo de datos hacia la consola vía SSH. Con esto hemos cubierto los tres niveles de servicio de un Cluster, primero un Cluster totalmente privado, después habilitando el servicios hacia una red local y por último uno totalmente abierto a la red global de Internet.

9. Configuración de la GRID

La Grid esta conformada por dos Clusters de alto rendimiento, alejados geográficamente y que se unen a través de una Red Privada Virtual red-a-red utilizando OpenVPN en lugar de usar routers (vía Hardware).

En principio, los Clusters contienen una diferente subred, el Cluster del CIICAp se configuró como una subred 192.168.100.0 que denotaremos de aquí en adelante como subred A y el Instituto Tecnológico de Veracruz con una subred 192.168.101.0 que denotaremos Cluster B (fig. 26).

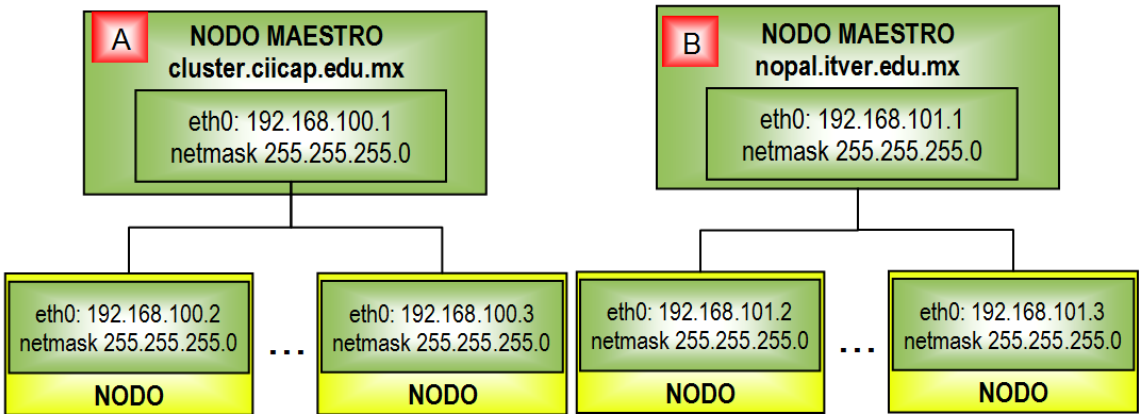


Figura 26. Dos Clusters separados geográficamente A y B.

Configuración de OpenVPN. OpenVPN es una solución de conectividad basada en software que ofrece conectividad punto a punto a través de Internet con validación de usuarios y hosts conectados remotamente. OpenVPN es una implementación vía software para construir una red privada virtual sin necesidad de routers vía Hardware, soporta diferentes medios de autenticación como certificados, usuarios y contraseñas.

La configuración de OpenVPN se puede resumir en tres tipos:

- **Maquina a Maquina.** Es el método más simple, nos permite encriptar la comunicación entre dos PC las cuales deberán solamente tener conexión; es decir: ambas PC deben poderse enviar paquetes directamente ya sea porque estén conectadas en la misma red local, o porque ambas estén conectadas a Internet y sean alcanzables entre sí.
- **Road Warrior.** Es una de las formas más utilizadas y solicitadas para conectarse a una red, permitir que una máquina externa de nuestra red pueda comunicarse con el servidor OpenVPN y una vez autenticado pueda acceder a los recursos de nuestra red local.
- **Red a Red.** Mediante ésta forma, dos redes separadas y alejadas geográficamente pueden unirse y alcanzar mutuamente sus recursos, la comunicación entre ambas redes viajará encriptada una vez que salga de los servidores de OpenVPN y hasta que llegue al otro extremo (destino).

La idea esta basada en poder usar un único canal de comunicación llamado *túnel* (figura 27) para enviar todas las comunicaciones de un extremo a otro y viceversa, permitiendo ver las maquinas conectadas al otro extremo como si estuvieran unidas físicamente a través de un cable. De esto se deduce que para unir dos cluster se requiere de unir dos redes configurando OpenVPN utilizando para ello Internet 2.

Para poder configurar una conexión red a red es necesario definir quien será servidor y quien cliente, nosotros usamos a cluster.ciicap.edu.mx como servidor y nopal.itver.edu.mx como cliente, observe que las únicas dos maquinas conectadas son los nodos maestros. Para

poder alcanzar los nodos que están atrás de los extremos de la VPN es necesario implementar encaminamiento de paquetes.

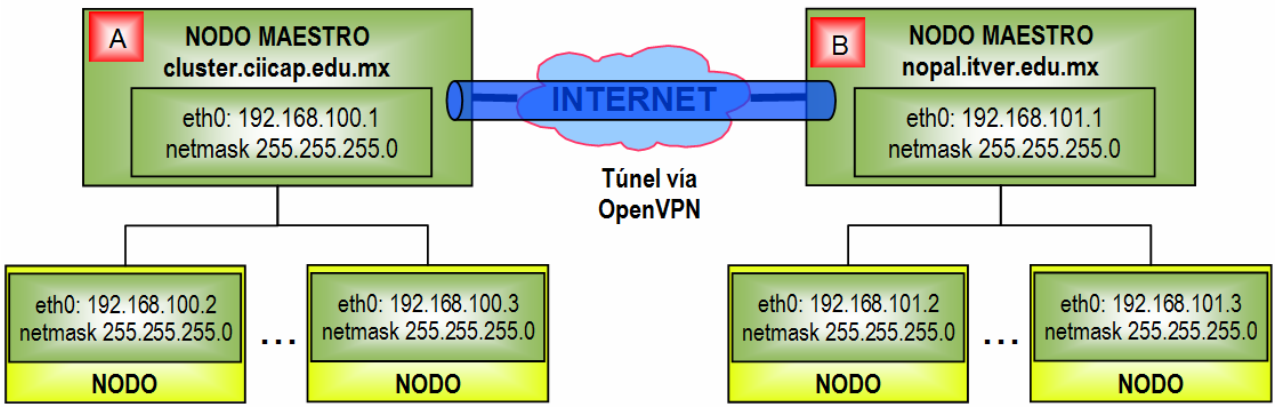


Figura 27. Dos Clusters separados geográficamente A y B unidos a través de una VPN.

Configuración servidor. La instalación del middleware OpenVPN se puede hacer vía compilación de los archivos fuente o vía instalación de paquete RPM. Nuestra elección se basó en la descarga, desempaquetado, compilado, construcción de binarios y edición del archivo de configuración (fig. 28).

```
[root@cluster ~]# cat /etc/openvpn/server.conf
port 1723
proto tcp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh1024.pem
server 192.168.99.0 255.255.255.0
ifconfig-pool-persist ipp.txt
client-config-dir ccd
route 192.168.101.0 255.255.255.0
client-to-client
push "route 192.168.101.0 255.255.255.0"
push "route 192.168.100.0 255.255.255.0"
keepalive 10 120
comp-lzo
user nobody
group nobody
persist-key
persist-tun
status openvpn-status.log
verb 4
```

Figura 28. Archivo de configuración para el servidor OpenVPN.

Posteriormente es necesario configurar la ejecución del middleware de forma que sea automático durante el arranque de sistema. En este punto ya esta listo para esperar las conexiones de parte del cliente.

Configuración cliente. La instalación del middleware del lado del cliente es igual al procedimiento usado en el servidor, con la excepción de que el archivo de configuración es diferente (fig. 29), ya que este indica a donde debe de conectarse mientras que el lado servidor indica en donde estará esperando recibir conexiones.

```
root@nopal:~> cat /etc/openvpn/client1.conf
client
dev tun
proto tcp
remote 148.218.100.90 8080
resolv-retry infinite
nobind
#Las dos siguientes opciones no van en windows
user nobody
group nobody

persist-key
persist-tun
ca ca.crt
cert client1.crt
key client1.key
comp-lzo
verb 4
```

Figura 29. Archivo de configuración para el cliente OpenVPN.

10. Pruebas en la GRID

Pruebas de conectividad. La manera de comprobar si una red VPN está bien configurada es muy sencilla, bastará con usar el comando *ping* para ver si podemos alcanzar las máquinas del otro extremo desde cualquier punto.

Desde cualquier punto del Cluster (cluster.ciicap.edu.mx, nodo01.cluster.ciicap.edu.mx, nodo02.ciicap.edu.mx, …, nodo0n.ciicap.edu.mx) se debe poder establecer la comunicación, utilizando el comando *ping*, lo mismo sucede para cualquiera de los nodos pertenecientes al cluster del Instituto Tecnológico de Veracruz, (nopal.itver.edu.mx, nodo01.itver.edu.mx, …, nodo0n.itver.edu.mx), incluidos los nodos maestros.

Pruebas en la Grid con OpenMPI y MPICH

Cálculo del ancho de banda en la Grid. Para realizar el cálculo del ancho de banda para el Cluster, se hace uso de la VPN, esta abarca los nodos maestros y sus nodos de trabajo.

Debido a que el tráfico afecta directamente las comunicaciones, se realizaron varias mediciones en un día, estas pruebas de repitieron por tres días. Se aplicó el procedimiento presentado en la sección 7 para el cálculo del ancho de banda.
Observe que no se cubrieron todas las combinaciones debido a que las velocidades de transferencia se ajustan a la velocidad que alcance la VPN entre los puntos cluster.ciicap.edu.mx y nopal.itver.edu.mx, por tanto las transferencias entre los nodos de los Clusters se ajustan a esta velocidad.
Esto quiere decir que al momento de enviar los datos desde los nodos, estos son enviados a través de los nodos maestros y que son los que se conectan a través de la VPN, por tanto su ancho de banda y su latencia afectan a los nodos en ambos extremos de la VPN. Aun así se incluyen las medidas de ancho de banda y latencia entre varios nodos.
La figura 30, presenta la medición experimental del ancho de banda entre los clusters CIICAp y Nopal con conexión I2. Específicamente la medida se realiza entre varios pares de nodos, uno correspondiente al cluster CIICAp y otro correspondiente al cluster Nopal, como se observa en la figura. Se puede ver que el ancho de banda tomado a diferente hora del día 23 de marzo oscila entre 0.1 y 1 Mbps, tomando como referencia varios pares de nodos de la MiniGRID Tarántula, de hecho la mayoría de las mediciones esta entre 0.1 y 0.5 Mbps. La figura presenta un pico entre 4.5 y 4 Mbps a las 18 hr. Se puede observar que el ancho de banda en cualquier par de nodos nunca permanece estable y que es muy inferior al máximo ancho de banda reportado en I2 para la conexión de la MiniGRID el cual es de 34Mbps. Se puede observar un crecimiento en el ancho de banda conforme avanza el día, esto se puede deber a que el uso de I2 está mas libre por la tarde.

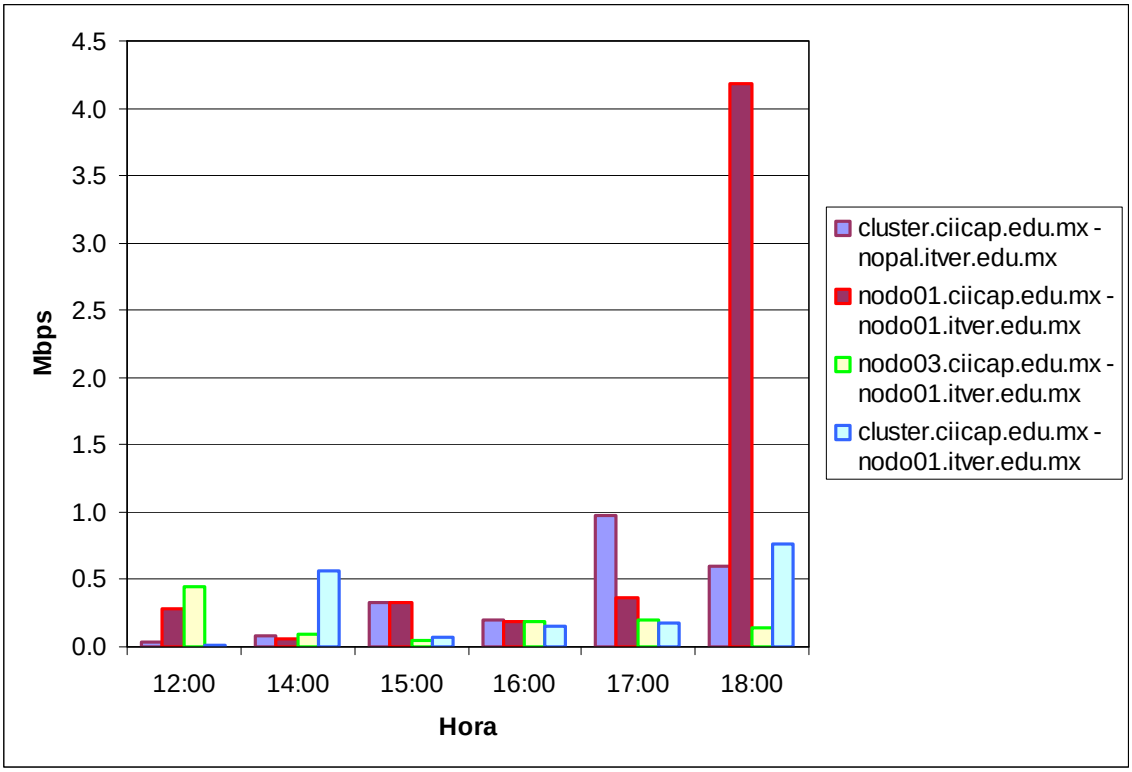


Figura 30. Ancho de banda (Mbps), monitoreado el 23 de marzo del 2009

La figura 31, presenta la medición experimental del ancho de banda entre los clusters CIICAp y Nopal con conexión I2. Específicamente la medida se realiza entre varios pares de nodos, uno correspondiente al cluster CIICAp y otro correspondiente al cluster Nopal, como se observa en la figura. Se puede ver que el ancho de banda tomado a diferente hora del día 24 de marzo oscila entre 0.1 y 0.4 Mbps, tomando como referencia varios pares de nodos de la MiniGRID Tarántula. La figura presenta un pico entre 1.2 y 1.4 Mbps a las 18 hrs. También se puede observar que el ancho de banda en cualquier par de nodos nunca permanece estable y que es muy inferior al máximo ancho de banda reportado en I2 para la conexión de la MiniGRID el cual es de 34Mbps. Se puede observar un decremento del ancho de banda de 12 a 14 hrs y después comienza un incremento entre las 14 y 18 hrs. Esto se puede deber a que el uso de I2 se pudo haber ocupado ese día y en el intervalo de 13 a 15 hr con mayor carga.

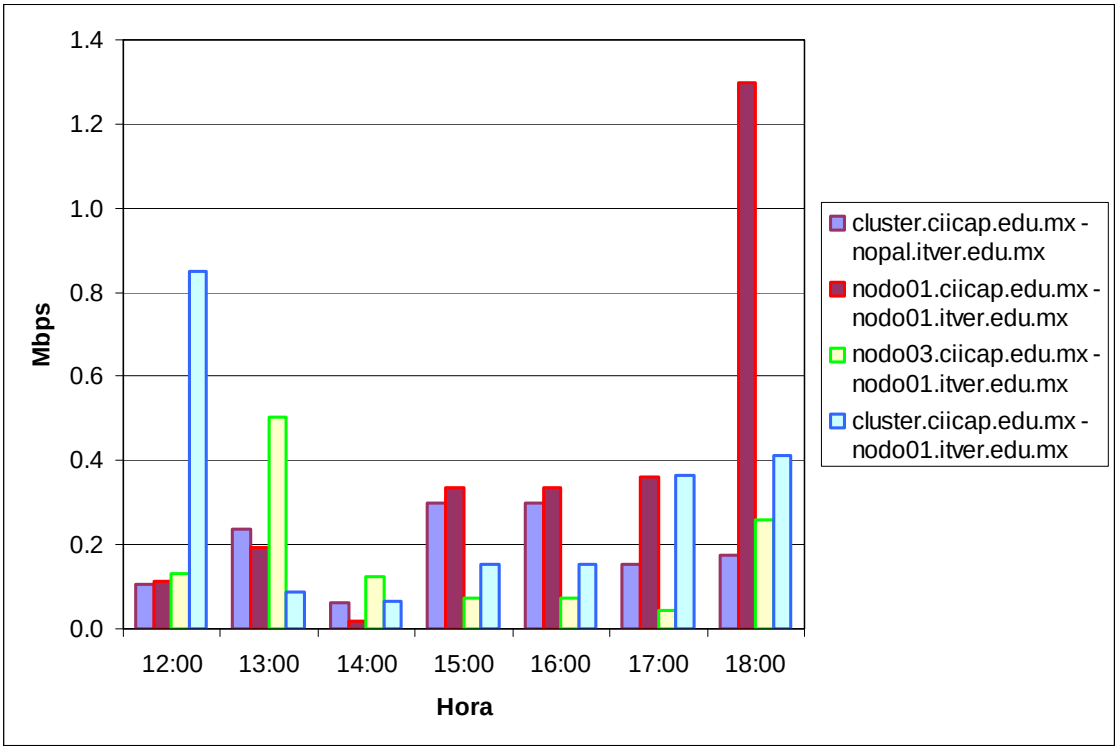


Figura 31. Ancho de banda (Mbps), monitoreado el 24 de marzo del 2009

La figura 32, presenta la medición experimental del ancho de banda entre los clusters

CIICAp y Nopal con conexión I2. Específicamente la medida se realiza entre varios pares de nodos, uno correspondiente al cluster CIICAp y otro correspondiente al cluster Nopal, como se observa en la figura. Se puede ver que el ancho de banda tomado a diferente hora del día 25 de marzo oscila entre 0.1 y 0.5 Mbps, tomando como referencia varios pares de nodos de la MiniGRID Tarántula. La figura presenta un pico entre 3 y 3.5 Mbps a las 14 hrs. También se puede observar que el ancho de banda en cualquier par de nodos nunca permanece estable y que es muy inferior al máximo ancho de banda reportado en I2 para la conexión de la MiniGRID el cual es de 34Mbps. Se puede observar que el ancho de banda tiene una tendencia a incrementarse conforme avanzan las horas del día. Esto se puede deber a que el uso de I2 se ocupa menos por la tarde.

Comparando los tres días de medición en las figuras 30, 31 y 32, se puede ver que los picos mayores de ancho de banda encontrados se presentan por la tarde del día a partir de las 14 hrs. También se puede ver lo contrario, que las medidas de ancho de banda más pequeñas encontradas se presentan por la mañana antes de las 14 hrs.

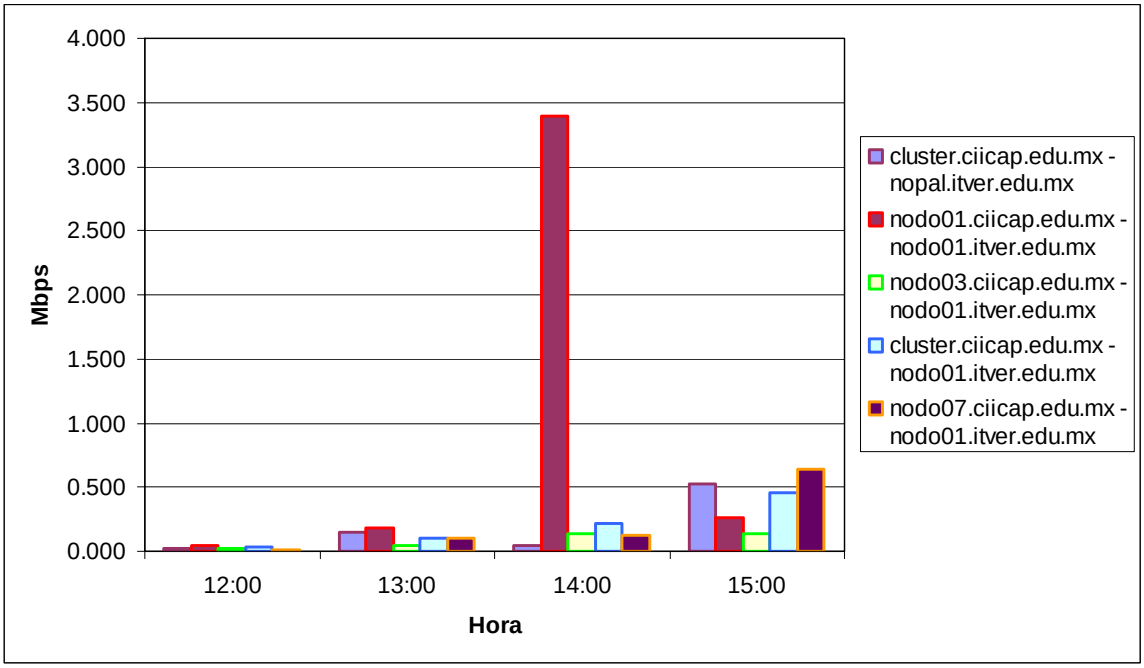


Figura 32. Ancho de banda (Mbps), monitoreado el 25 de marzo del 2009

La figura 33, presenta el ancho de banda promedio del día 23 de marzo entre pares de nodos. Se puede observar que el par de nodos que presenta el menor ancho de banda promedio es el par nodo03.ciicap.edu.mx – nodo01.itver.edu.mx y el par de nodos que presenta el mayor ancho de banda promedio es el par nodo01.ciicap.edu.mx – nodo01.itver.edu.mx. El ancho de banda oscila entre 0.2 y 0.9 Mbps.

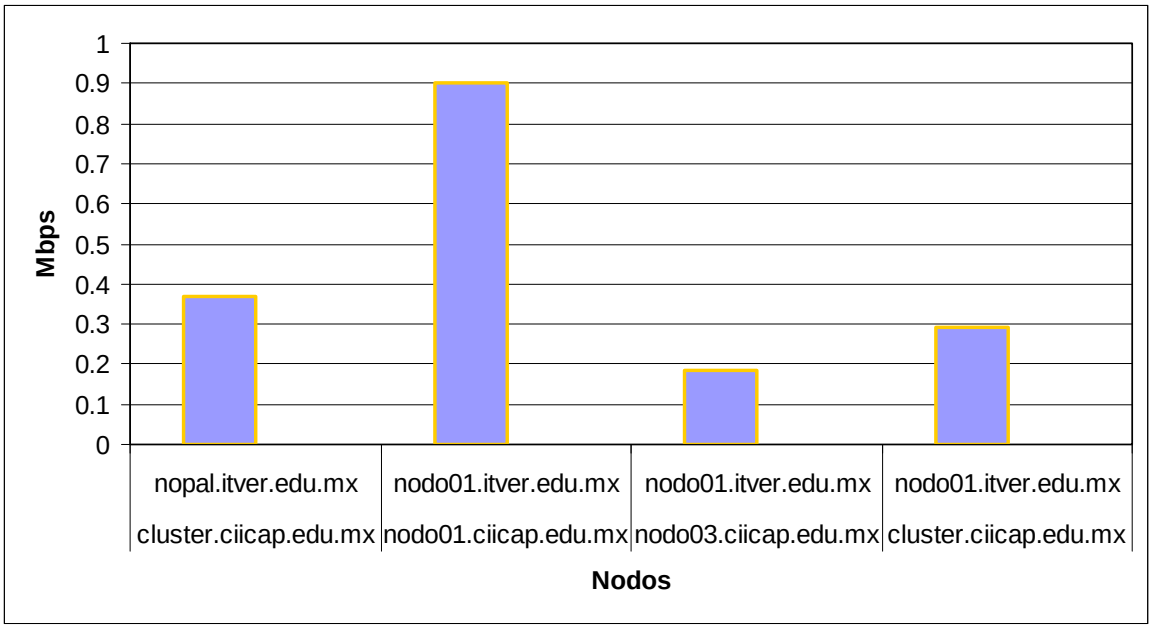


Figura 33. Ancho de banda promedio (Mbps), monitoreado el 23 de marzo del 2009

La figura 34, presenta el ancho de banda promedio del día 24 de marzo entre pares de nodos. Se puede observar que el par de nodos que presenta el menor ancho de banda promedio es el par de nodos maestros, cluster.ciicap.edu.mx – nopal.itver.edu.mx y el par de nodos que presenta el mayor ancho de banda promedio es el par nodo01.ciicap.edu.mx – nodo01.itver.edu.mx. El ancho de banda oscila entre 0.15 y 0.4 Mbps.

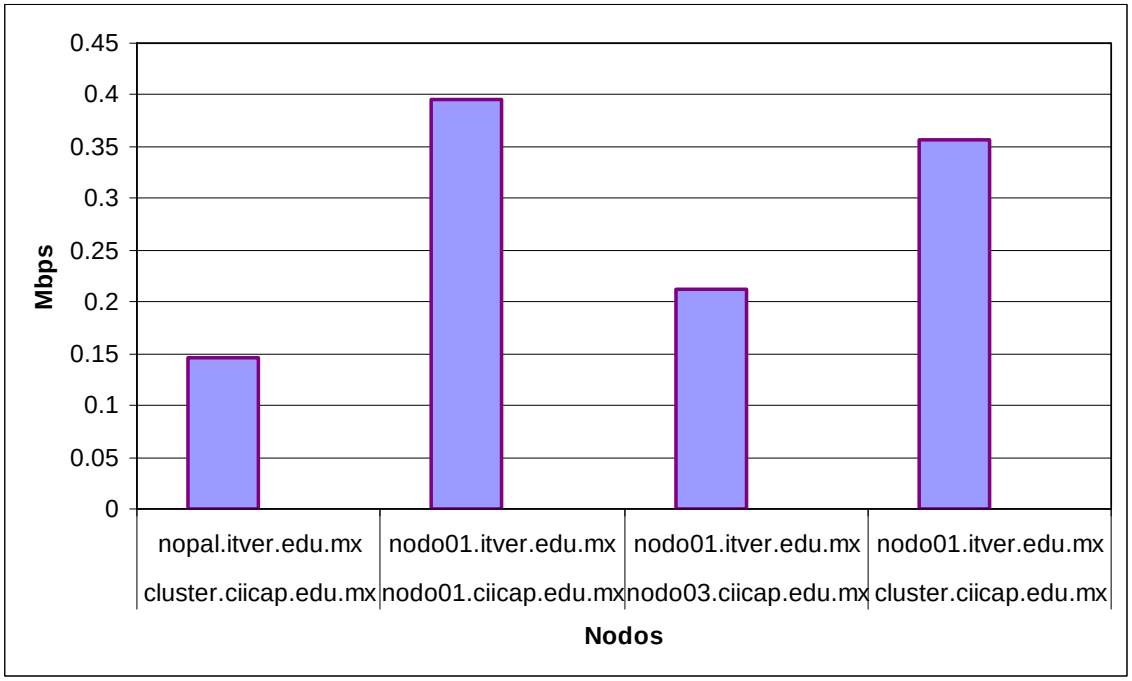


Figura 34. Ancho de banda promedio (Mbps), monitoreado el 24 de marzo del 2009

La figura 35, presenta el ancho de banda promedio del día 25 de marzo entre pares de nodos. Se puede observar que el par de nodos que presenta el menor ancho de banda

promedio es el par de nodos nodo03.ciicap.edu.mx – nodo01.itver.edu.mx y el par de nodos que presenta el mayor ancho de banda promedio es el par nodo01.ciicap.edu.mx – nodo01.itver.edu.mx. El ancho de banda oscila entre 0.05 y 0.98 Mbps. Se puede observar de las figuras 33, 34 y 35 que el par de nodos nodo01.ciicap.edu.mx – nodo01.itver.edu.mx en los tres días, siempre obtuvo el mayor ancho de banda y el par nodo03.ciicap.edu.mx – nodo01.itver.edu.mx obtuvo el menor en dos de tres días.

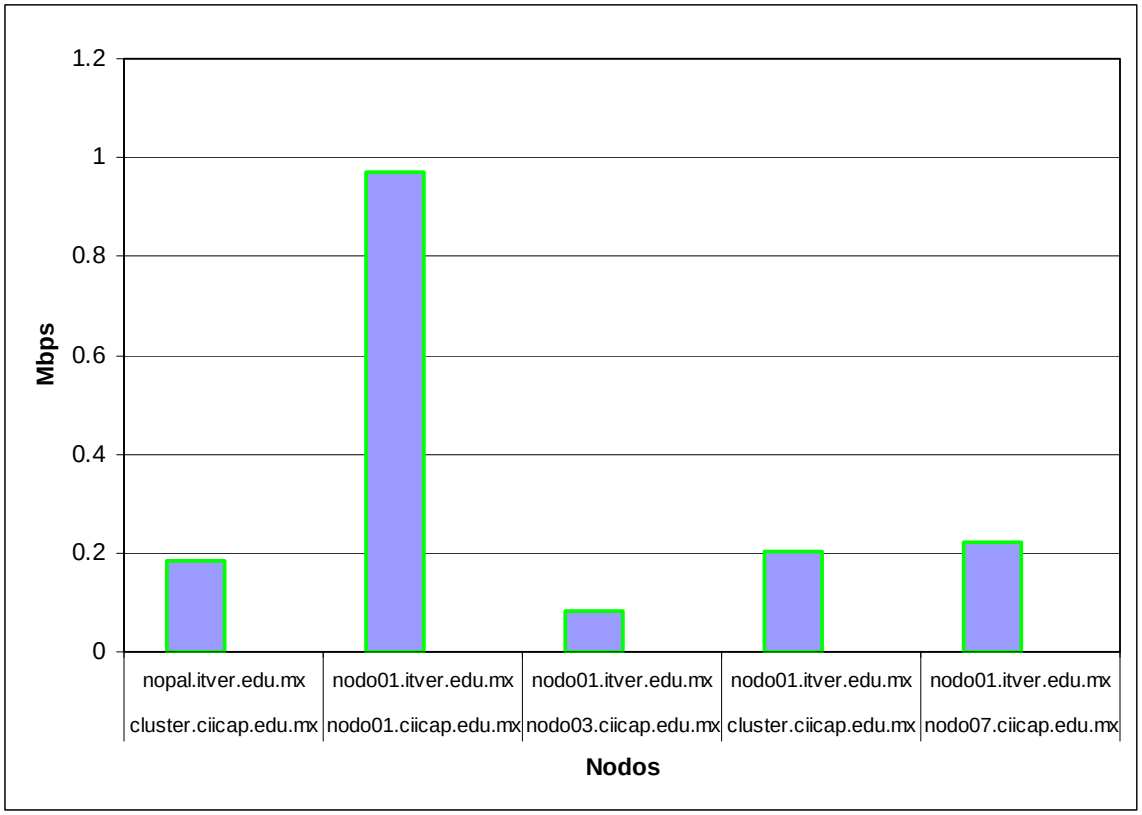


Figura 35. Ancho de banda promedio (Mbps), monitoreado el 25 de marzo del 2009

Cálculo de latencia en la Grid. El cálculo de la latencia se basa en las velocidades de transferencia, las cuales se ajustan a la velocidad de la VPN entre los puntos cluster.ciicap.edu.mx y nopal.itver.edu.mx, así la latencias entre los nodos de los clusters se ajustan a las latencias de los nodos maestros. Se aplicó el procedimiento visto en la sección 7 para el cálculo de la latencia. Para el peor de los casos en las pruebas del algoritmo evolutivo para JSSP, en el problema mayor de prueba (20x20) y como se observó en la sección 3.3, se envían sólo 4 individuos por cada proceso. De acuerdo al tipo de dato utilizado en el algoritmo, el cual es entero de 4 bytes. El flujo de datos a enviar en cada proceso depende del tamaño de la estructura, en este caso, el tamaño de la estructura que representa a un individuo es de 12800 bytes, por lo que en cada CPU del cluster CIICAp se enviará un flujo de 51200 bytes, el cálculo de latencia en la MiniGRID Tarántula lanzando el algoritmo desde el cluster CIICAp, requerirá enviar un flujo de datos al cluster Nopal de 204,800 bytes (200kb) de ida y vuelta, en cada iteración del algoritmo genético a través de I2, de acuerdo a este flujo de datos, la latencia calculada en la MiniGRID Tarántula es la presentada en las figuras 36 a la 39.

La figura 36, presenta las pruebas experimentales de latencia obtenidas el 23 de marzo entre las 12 a las 18 hrs. Se observa que la latencia va disminuyendo conforme avanza el día, inicia a las 12 hr con la mayor latencia medida entre 45 y 40 seg y termina a las 18 hr con la menor latencia medida, menor de 0.4 seg. Se puede observar que en cada par de nodos la latencia tiende normalmente a disminuir conforme pasa el día. Esto concuerda con lo que se observó en las figuras de ancho de banda, aparentemente el uso de I2 disminuye por la tarde y esto hace que la transferencia de datos sea más eficiente.

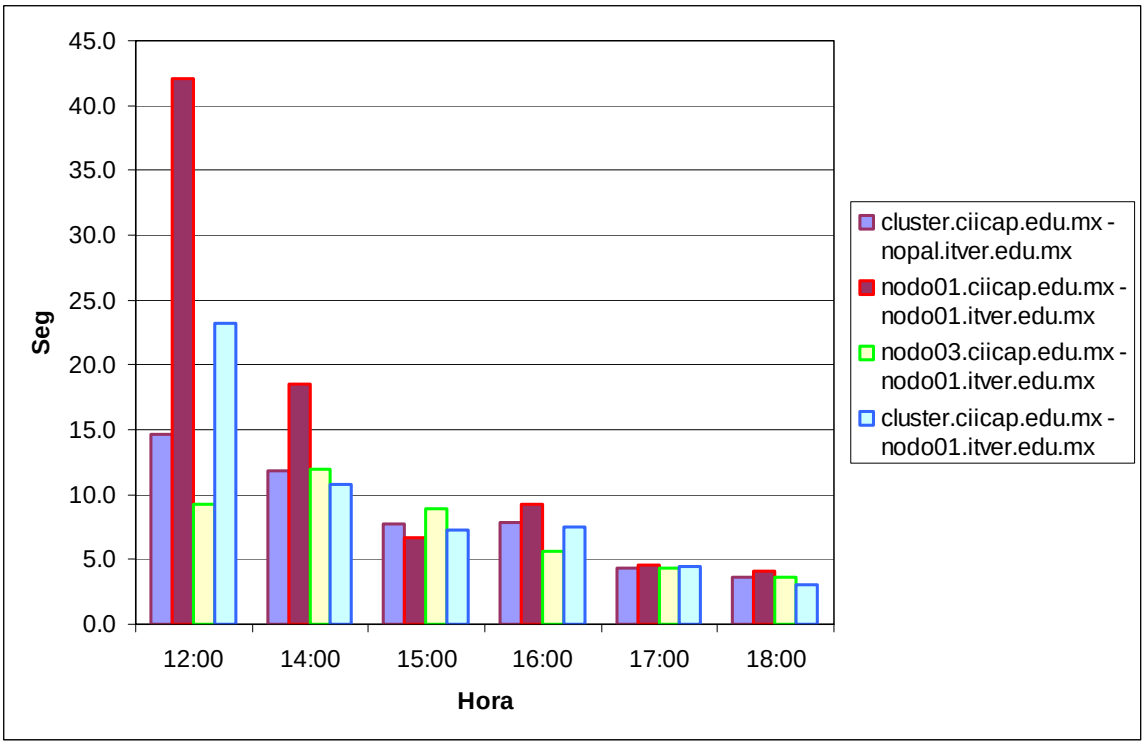


Figura 36. Latencia monitoreada el 23 de marzo del 2009

La figura 37, presenta las pruebas experimentales de latencia obtenidas el 24 de marzo entre las 12 a las 18 hrs. Se observa que la latencia no presenta un comportamiento de incremento o decremento conforme avanza el día, a las 14 hr y a las 16 hr se presenta la mayor latencia en todos los pares de nodos en promedio. La mayor latencia se mide a las 16 hrs para la mayoría de los pares de nodos, estando ésta entre 60 y 70 seg. Se tiene la menor latencia a las 18 hrs para todos los pares de nodos. Al igual que en la figura 36, la menor latencia se presenta a las 18 hrs.

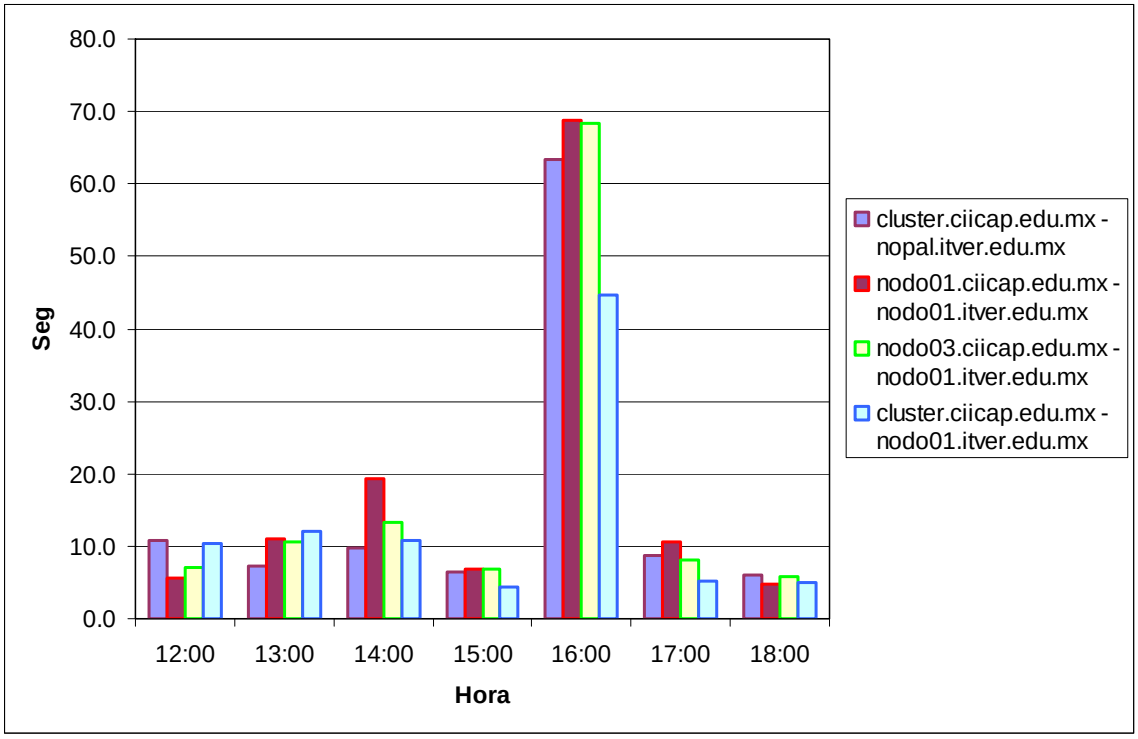


Figura 37. Latencia monitoreada el 24 de marzo del 2009

La figura 38, presenta las pruebas experimentales de latencia obtenidas el 25 de marzo entre las 12 a las 18 hrs. Se observa que la latencia va disminuyendo conforme avanza el día, inicia a las 12 hr con la mayor latencia medida entre 70 y 75 seg y termina a las 18 hr con la menor latencia medida, menor de 0.8 seg. Se puede observar que en cada par de nodos la latencia tiende normalmente a disminuir conforme pasa el día. Esto concuerda con lo que se observó en las figuras de ancho de banda, aparentemente el uso de I2 disminuye por la tarde y esto hace que la transferencia de datos sea más eficiente.

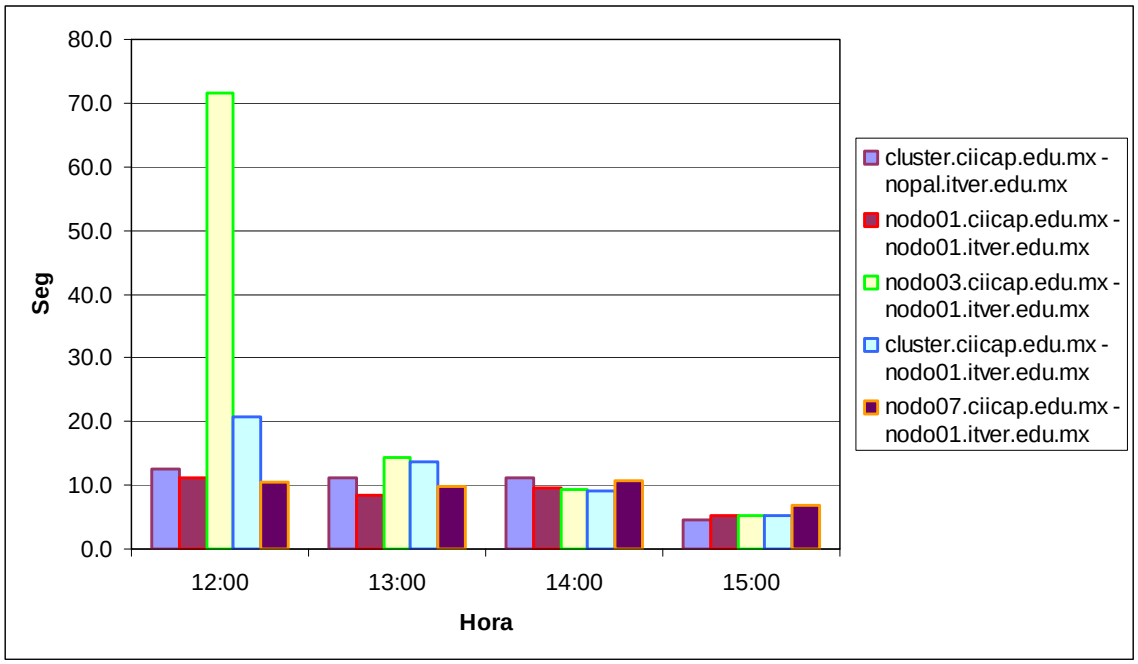


Figura 38. Latencia, monitoreada el 25 de marzo del 2009

La figura 39, presenta la latencia promedio del día 23 de marzo entre pares de nodos. Se puede observar que el par de nodos que presenta la menor latencia promedio es el par nodo03.ciicap.edu.mx – nodo01.itver.edu.mx y el par de nodos que presenta la mayor latencia promedio es el par nodo01.ciicap.edu.mx – nodo01.itver.edu.mx. La latencia del día oscila entre 6 y 14 seg.

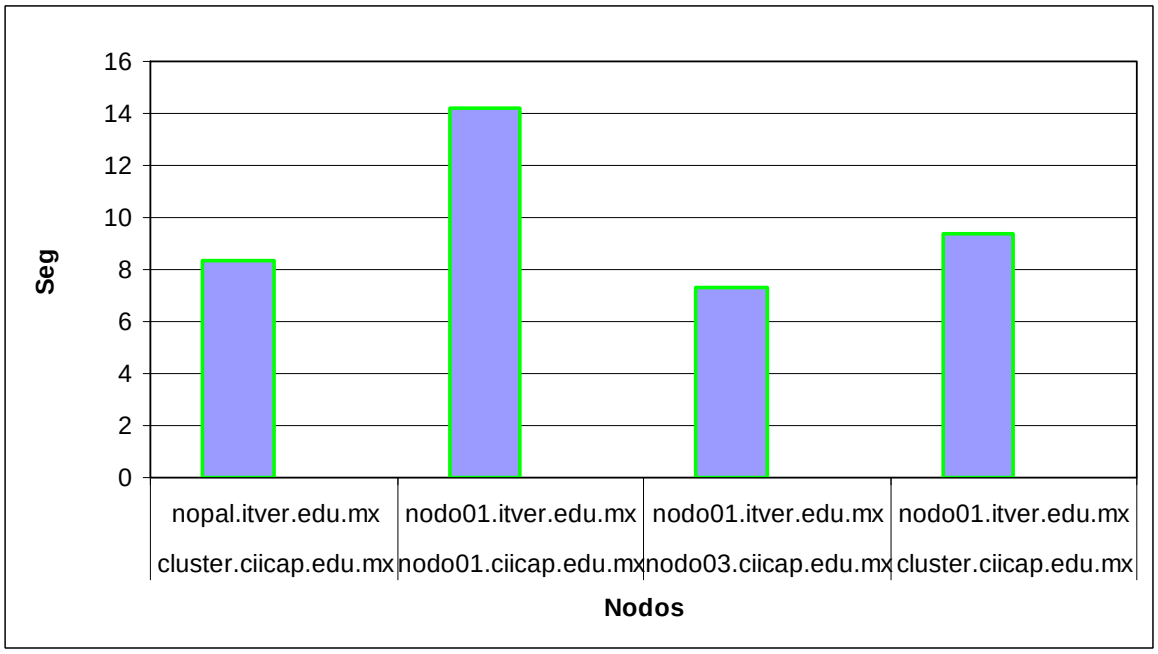


Figura 39. Latencia promedio (seg), monitoreada el 23 de marzo del 2009

La figura 40, presenta la latencia promedio del día 24 de marzo entre pares de nodos. Se puede observar que el par de nodos que presenta la menor latencia promedio es el par cluster.ciicap.edu.mx – nodo01.itver.edu.mx y el par de nodos que presenta la mayor latencia promedio es el par nodo01.ciicap.edu.mx – nodo01.itver.edu.mx. La latencia del día oscila entre 12 y 18 seg.

La figura 41, presenta la latencia promedio del día 25 de marzo entre pares de nodos. Se puede observar que el par de nodos que presenta la menor latencia promedio es el par nodo01.ciicap.edu.mx – nodo01.itver.edu.mx y el par de nodos que presenta la mayor latencia promedio es el par nodo03.ciicap.edu.mx – nodo01.itver.edu.mx. La latencia del día oscila entre 8 y 25 seg.
Se puede observar de las figuras 39, 40 y 41 que el par de nodos nodo01.ciicap.edu.mx – nodo01.itver.edu.mx en dos de tres días obtuvo la mayor latencia y para la menor latencia no hay distinción en pares de nodos.

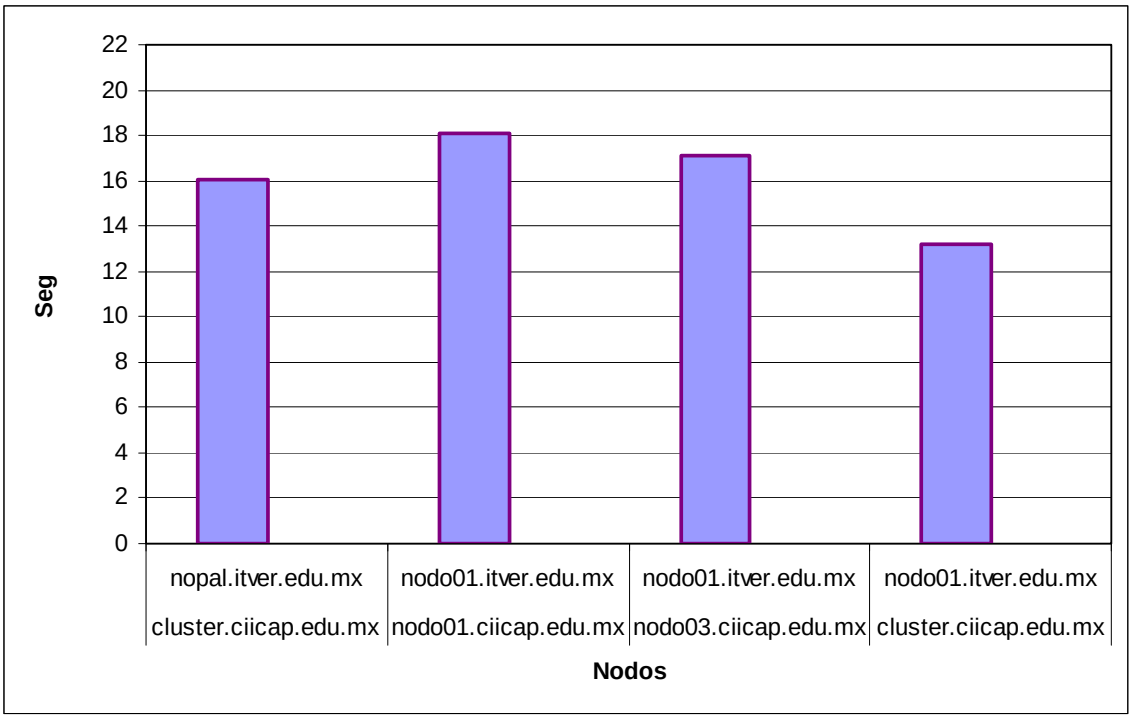


Figura 40. Latencia promedio (seg), monitoreada el 24 de marzo del 2009

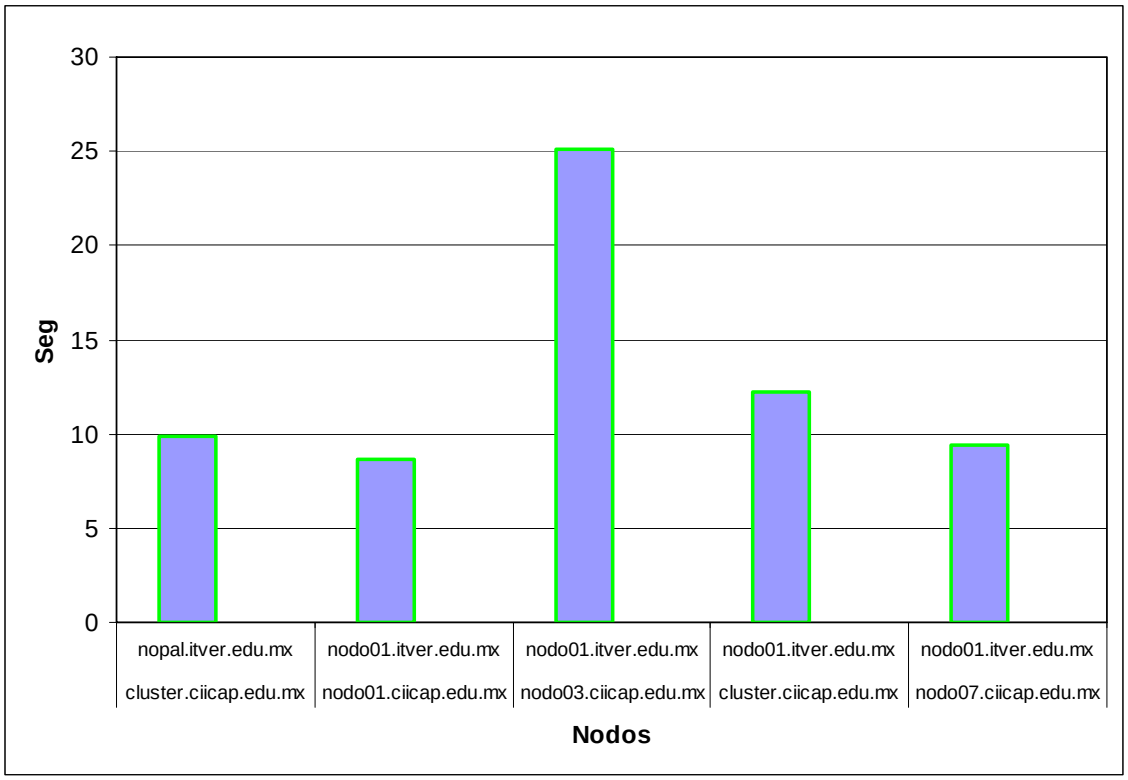


Figura 41. Latencia promedio (seg), monitoreada el 25 de marzo del 2009

Las pruebas experimentales que se presentan en las gráficas de la figura 30 a la 41, se presentan en los anexos 1, 2 y 3.

También se tomó el tiempo de respuesta que se obtiene al enviar un archivo de 200kb del nodo maestro del CIICAp al nodo maestro del Nopal (FrontEnds), de ida y vuelta. Se tomó la medición del tiempo de término de escritura del archivo. Estas mediciones fueron realizadas por tres días seguidos entre las 9 y 18 hrs. Las figuras de la 42 a la 43, presentan el tiempo requerido para la transferencia de un archivo de 200kb. Se observa en todas las gráficas el mismo comportamiento. La transferencia se hace más rápida del cluster nopal al cluster CIICAp. La latencia se hace más notoria por la mañana y disminuye por la tarde.

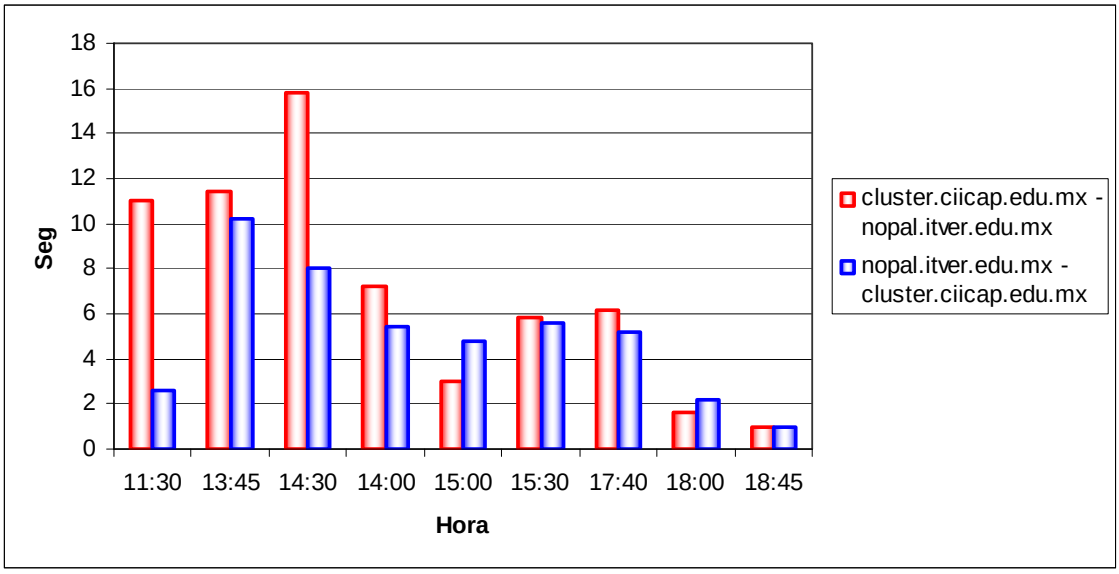


Figura 42. Latencia, monitoreada el 31 de marzo del 2009

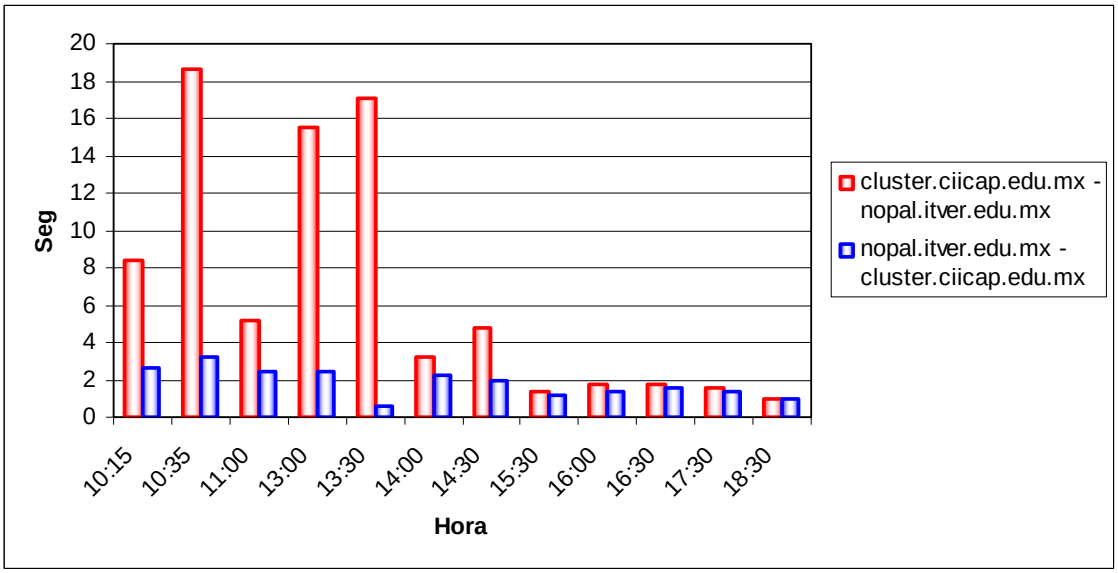


Figura 43. Latencia, monitoreada el 1 de abril del 2009

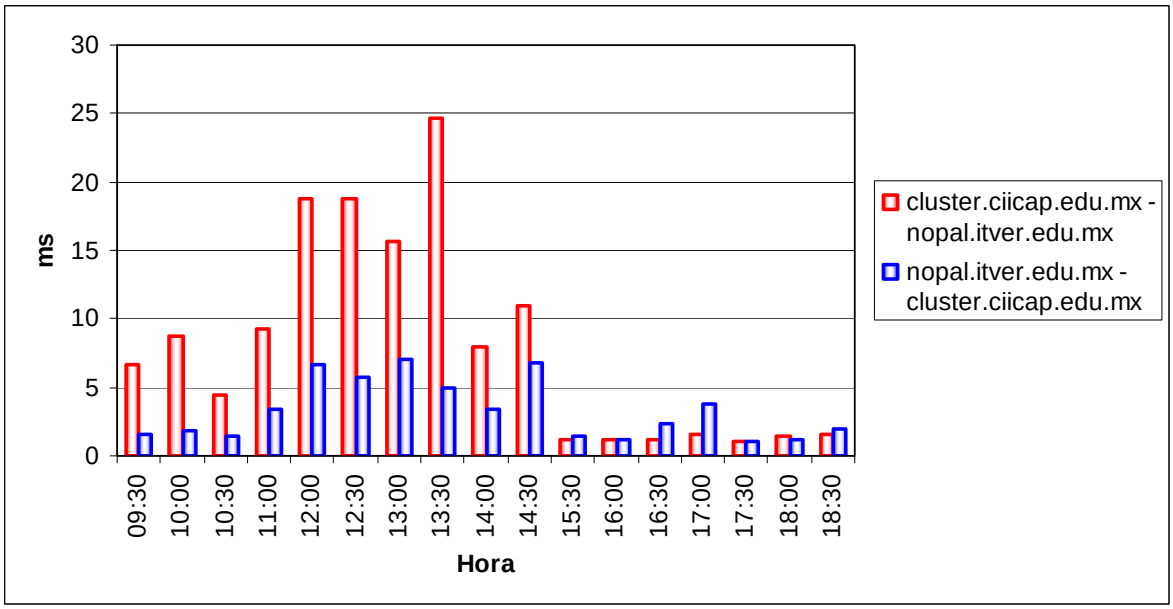


Figura 44. Latencia, monitoreada el 2 de abril del 2009

11. Resultados experimentales en GRID del modelo JSSP

El algoritmo genético híbrido en su primera versión para el JSSP, se ejecuto para dos benchmarks tomados de la OR Library [Beasley, 03].
El primer problema ejecutado desde el nodo maestro del cluster CIICAp por el algoritmo genético, es el problema LA40 que consta de 15 trabajos por 15 máquinas, con un total de 225 operaciones en el problema. El criterio de paro es el número de generaciones. La tabla 3, presenta los datos de entrada al algoritmo.

Tabla 3. Datos de entrada al algoritmo genético híbrido para el problema LA40

Variable	Valor
Valor inicial parámetro de control RS	25
Valor final parámetro de control RS	1.0
Coeficiente de control RS	0.99
Longitud cadena de Markov	210
Cota superior conocida del problema	1222
Reinicios en RS por cada individuo	20
Población buena seleccionada (%)	40
Población mala seleccionada (%)	5
Población a mutar (%)	80
Número de generaciones	35

Para el envío de mensajes en el modelo JSSP se utilizó la librería de paso de mensajes MPI. Actualmente esta librería permite la transferencia de tipos de datos complejos. Sin embargo no cuenta con mecanismos que permitan el envío de datos dinámicos. Para ello se debe crear un procedimiento de conversión de datos ya que, por sus características, MPI no permite transferir dichos datos. MPI únicamente reconoce los tipos de datos primitivos del

lenguaje C. Por ejemplo, para los datos en lenguaje C de tipo int, char, long, double, entre otros, existe un tipo de dato equivalente en MPI: MPI_INT, MPI_CHAR, MPI_LONG, MPI_DOUBLE. Por el contrario, para tipos de datos dinámicos no existe en MPI un tipo de dato que sea equivalente. Se debe realizar un procedimiento de conversión.

La siguiente figura muestra las estructuras complejas y dinámicas utilizadas en el programa JSSP.

```
struct maq
{
      int j;
      int t;
      int s;
      int nop;
};

struct trab
{
      int m;
      int t;
      int s;
      int nop;
};

struct sched
{
      maq **opm;
      maq **MOPM;
      maq **mopm;
      maq **opm_tmp;
      trab **opj;
      trab **MOPJ;
      trab **mopj;
      trab **opj_tmp;
};
```

Las estructuras *maq* y *trab* son estructuras de datos complejas que usan tipos de datos básicos del lenguaje C (tipo int). Como puede observarse los datos que manejan son datos primitivos y no hacen uso de apuntadores.

La estructura *sched* es una estructuras de datos dinámica ya que, como puede observarse en la figura, utilizan apuntadores para referirse a otros tipos de datos.

El procedimiento para que sea posible transferir datos se conoce en algunas plataformas como serialización y de-serialización. Consiste en serializar la estructura dinámica, enviarla, recibirla y reconstituirla. Este proceso de conversión es complejo y laborioso si se realiza de forma manual. Sin embargo existen herramientas que permiten la conversión automática de tipos de datos dinámicos para que puedan ser enviados fácilmente por el sistema de pase de mensajes MPI.

Automap es una herramienta que permite crear tipos de datos en MPI a partir de tipos de datos definidos por el usuario en lenguaje C. Es un compilador que hace uso del analizador yacc++ para analizar las estructuras definidas por el usuario y convertirlas a tipos de datos que MPI pueda reconocer. Para hacer uso de automap se requiere de un archivo que contenga los tipos de datos definidos por el usuario. Se deben colocar unos símbolos (comentarios /**/) que sirven a AutoMap para identificar el inicio y fin de los datos que se de desean convertir. A partir del archivo de entrada AutoMap genera un conjunto de archivos que contienen la definición de los tipos de datos en MPI. También contiene algunos procedimientos que deben invocarse desde el código del algoritmo paralelo. Algunos son para la creación de los tipos de datos otros sirven para reservar y liberar memoria.

De acuerdo al diseño del algoritmo, se tiene que el flujo de información que se envía a cada CPU, es cada vez que la población termina su cruzamiento (Figura 4). Una parte de la nueva generación sufre el proceso de mutación. El subconjunto de la población a mutar se reparte en parte proporcional en todos los CPU's de la MiniGrid Tarántula. Cada CPU realiza la mutación de los individuos que le toca. En la repartición de los individuos en cada CPU del cluster Nopal, existe una latencia. La latencia promedio detectada para el flujo de datos que se envía desde el Cluster CIICAp al cluster Nopal está en un intervalo de 7 a 25 segundos (Figuras 39, 49, 41). Esto indica, en el peor de los casos, que la latencia total en el tiempo de ejecución del algoritmo será de 875 segundos de acuerdo al número de generaciones. La tabla 4 presenta los resultados de una ejecución del algoritmo. Se presentan los resultados obtenidos de cada procesador de la GRID en la ejecución de la mutación por Recocido Simulado, al término de la generación 35. Se presenta el tiempo de ejecución total por cada nodo de la GRID y el valor encontrado de la función objetivo.

Como medida de comparación, el algoritmo también se corrió con los mismos datos de entrada (Tabla 3) en una PC con Windows Vista a 3.16Ghz, 4GB RAM, se utilizó el compilador de Visual C++ 2008. El tiempo de ejecución secuencial fue de 55,728 segundos con MS de 1233. Lo interesante aquí es que el tiempo de ejecución que da la GRID en el peor de los casos (sin tomar nodos 1 y 2 de CIICAp), es sólo el 29.2% de lo que se tarda en forma secuencial. Es importante notar que si la GRID contara con un mayor número de procesadores, entonces se podría aumentar el tamaño de la población, teniendo una eficiencia muy parecida en tiempo. Esto haría que el algoritmo pudiera realizar una mayor exploración en el espacio de soluciones en un tiempo parecido y poder encontrar de una manera más rápida una mejor solución. Esto da evidencia de la importancia de tener una GRID que cuente con un gran número de recursos en cuanto a CPU´s se refiere para aplicación de cómputo de alto rendimiento, como lo es el caso que se presenta.

Tabla 4. Resultados de la ejecución del algoritmo genético híbrido para el problema LA40.

NODO	Tiempo en segundos	MS
nodo03.ciicap.edu.mx	11403	1247
nodo03.ciicap.edu.mx	11347	1246
nodo04.ciicap.edu.mx	11347	1246
nodo04.ciicap.edu.mx	11375	1248
nodo05.ciicap.edu.mx	11571	1241
nodo05.ciicap.edu.mx	11319	1242
nodo01.ciicap.edu.mx	91399	1256
nodo02.ciicap.edu.mx	186431	1246
cluster.ciicap.edu.mx	11571	1246
nodo01. itver.edu.mx	15939	1246
nodo01. itver.edu.mx	16275	1250
nodo02. itver.edu.mx	15911	1250
nodo02. itver.edu.mx	16247	1245
nopal. itver.edu.mx	12691	1243
nodo06.ciicap.edu.mx	14567	1241
nodo06.ciicap.edu.mx	14455	1246
nodo07.ciicap.edu.mx	15323	1245
nodo07.ciicap.edu.mx	15547	1234
nodo08.ciicap.edu.mx	12131	1245
nodo08.ciicap.edu.mx	12075	1245

En la tabla 3, se indica para el algoritmo que el número de reinicios de Recocido Simulado por cada individuo es de 20. De acuerdo a este número de reinicios, en el peor de los casos el tiempo promedio que tarda en ejecutarse el recocido simulado en cada procesador CPU es de 345 segundos contra los 25 segundos de la peor latencia registrada por I2 en la Grid. De acuerdo a esto, se puede afirmar que la latencia pasa a segundo término ya que el tiempo en el uso de los procesadores es mucho mayor que el tiempo de comunicación entre clusters (93.2% vs. 6.8%).

El segundo problema ejecutado desde el nodo maestro del cluster CIICAp por el algoritmo genético, es el problema YN1 que consta de 20 trabajos por 20 máquinas, con un total de 400 operaciones en el problema. El criterio de paro es el número de generaciones. La tabla 5, presenta los datos de entrada al algoritmo.

Tabla 5. Datos de entrada al algoritmo genético híbrido para el problema YN1

Variable	Valor
Valor inicial parámetro de control RS	25
Valor final parámetro de control RS	1.0
Coeficiente de control RS	0.955
Longitud cadena de Markov	5000
Cota superior conocida del problema	885
Reinicios en RS por cada individuo	20
Población buena seleccionada (%)	40

Población mala seleccionada (%)	5
Población a mutar (%)	80
Número de generaciones	6

De acuerdo al diseño del algoritmo, se tiene que el flujo de información que se envía a cada CPU, es cada vez que la población termina su cruzamiento (Figura 4). Una parte de la nueva generación sufre el proceso de mutación. El subconjunto de la población a mutar se reparte en parte proporcional en todos los CPU's de la MiniGrid Tarántula. Cada CPU realiza la mutación de los individuos que le toca. En la repartición de los individuos en cada CPU del cluster Nopal, existe una latencia. La latencia promedio detectada para el flujo de datos que se envía desde el Cluster CIICAp al cluster Nopal está en un intervalo de 7 a 25 segundos (Figuras 39, 49, 41). Esto indica, en el peor de los casos, que la latencia total en el tiempo de ejecución del algoritmo será de 150 segundos de acuerdo al número de generaciones. La tabla 6 presenta los resultados de una ejecución del algoritmo. Se presentan los resultados obtenidos de cada procesador de la GRID en la ejecución de la mutación por Recocido Simulado, al término de la generación 35. Se presenta el tiempo de ejecución total por cada nodo de la GRID y el valor encontrado de la función objetivo.

Tabla 6. Resultados de la ejecución del algoritmo genético híbrido para el problema YN1.

NODO	Tiempo en segundos	MS
nodo03.ciicap.edu.mx	18568	934
nodo03.ciicap.edu.mx	18828	936
nodo04.ciicap.edu.mx	18664	942
nodo04.ciicap.edu.mx	18539	935
nodo05.ciicap.edu.mx	18644	947
nodo05.ciicap.edu.mx	18606	936
nodo01.ciicap.edu.mx	--	--
nodo02.ciicap.edu.mx	--	--
cluster.ciicap.edu.mx	18755	939
nodo01. itver.edu.mx	27347	939
nodo01. itver.edu.mx	26636	936
nodo02. itver.edu.mx	27404	935
nodo02. itver.edu.mx	26564	937
nopal. itver.edu.mx	20132	936
nodo06.ciicap.edu.mx	19854	935
nodo06.ciicap.edu.mx	19820	938
nodo07.ciicap.edu.mx	20147	940
nodo07.ciicap.edu.mx	20041	933
nodo08.ciicap.edu.mx	19508	939
nodo08.ciicap.edu.mx	19561	935

En la tabla 3, se indica para el algoritmo que el número de reinicios de Recocido Simulado por cada individuo es de 20. De acuerdo a este número de reinicios, en el peor de los casos el tiempo promedio que tarda en ejecutarse el recocido simulado en cada procesador CPU es de 4,542 segundos contra los 25 segundos de la peor latencia registrada por I2 en la Grid.

De acuerdo a esto, se puede afirmar que la latencia pasa a segundo término ya que el tiempo en el uso de los procesadores es mayor que el tiempo de comunicación entre clusters (99.5% vs. 0.5%).

Como medida de comparación, el algoritmo también se corrió en una PC con Windows Vista a 3.16Ghz, 4GB RAM, se utilizó el compilador de Visual C++ 2008. El tiempo de ejecución secuencial fue de 100,292 segundos con MS de 1233. Lo interesante aquí es que el tiempo de ejecución que da la GRID en el peor de los casos para esta corrida (sin tomar nodos 1 y 2 de CIICAp), es sólo el 27% de lo que se tarda en forma secuencial. Es importante notar que si la GRID contara con un mayor número de procesadores, entonces se podría aumentar el tamaño de la población, teniendo una eficiencia muy parecida en tiempo. Esto haría que el algoritmo pudiera realizar una mayor exploración en el espacio de soluciones en un tiempo parecido y poder encontrar de una manera más rápida una mejor solución. Esto da evidencia de la importancia de tener una GRID que cuente con un gran número de recursos en cuanto a CPU´s se refiere para aplicación de cómputo de alto rendimiento, como lo es el caso que se presenta.

12. Resultados experimentales en GRID del modelo VRPTW

Para aprovechar los recursos de la MiniGRID y acelerar el algoritmo evolutivo multi-poblacional se aplicó mutación sobre la mitad de la población (para un par de nodos de la GRID) y posteriormente se realizo intercambio de la población mutada, con el nodo complementario. El intercambio se lleva a cabo a través del envió de información vía un archivo temporal con la mitad de la población faltante en el nodo. Una vez completada la población, se procede a continuar el proceso de evaluación de aptitud de cada individuo y a iniciar una nueva generación.

Las pruebas de la versión paralela se realizaron de la siguiente forma:

- Entre nodos del cluster nopal.itver.edu.mx (comunicación Ethernet 10 Mbs).
- Entre procesadores SMP de un nodo del cluster nopal.itver.edu.mx (comunicaciones por motherboard).
- Entre nodos del cluster.ciicap.edu.mx (comunicaciones Ethernet 100 Mbs).
- Entre los frontend de los clusters nopal.itver.edu.mx y cluster.ciicap.edu.mx (comunicaciones por Internet 2).

Los tiempos promedios obtenidos para 5 generaciones son los siguientes:

Generación	Nopal-Sec	Nopal-Nodos	Nopal-SMP	Eficiencia Nopal-Nodos	Eficiencia Nopal-SMP
1	23.55	12.21	11.96	96.00%	98.00%
2	46.73	24.39	24.48	96.00%	95.00%
3	70.16	36.57	36.56	96.00%	96.00%
4	93.13	49.15	48.53	95.00%	96.00%
5	117.24	61.71	61.01	95.00%	96.00%

Se llevo a cabo el profile para el código secuencial de Ocotlán Díaz-Parra para evaluar las funciones o que parte del código es posible paralelizar con OpenMP.

Se determinó la paralelización de las funciones ordenamayores() y ordenamenores() pertenecientes a la operación de mutación(véase anexo 7).

Para poder establecer una conclusión acerca de lo elaborado, por tanto las tablas presentadas a continuación representan algunos resultados favorables.

En la realización del profile el algoritmo genético secuencial solo con la función mutate(). El resultado es el siguiente:

Flat profile:

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
99.97	72.58	72.58	5	14.52	14.52	mutate()
0.03	72.60	0.02	99	0.00	0.00	Permutaciones(int*, int)
0.00	72.60	0.00	100	0.00	0.00	CALCULARUTAS()
0.00	72.60	0.00	5	0.00	0.00	seccionbest()
0.00	72.60	0.00	5	0.00	0.00	calculadisttotal()
0.00	72.60	0.00	5	0.00	0.00	crossover()
0.00	72.60	0.00	1	0.00	0.00	global constructors keyed to archivo
0.00	72.60	0.00	1	0.00	0.00	rutakmeans()
0.00	72.60	0.00	1	0.00	0.02	genpobinialgo0()
0.00	72.60	0.00	1	0.00	72.60	algoritmogeneticoKOKO()
0.00	72.60	0.00	1	0.00	0.00	__static_initialization_and_destruction_0(int, int)
0.00	72.60	0.00	1	0.00	0.00	std::operator|(std::_Ios_Openmode, std::_Ios_Openmode)

Figura 41. Análisis con solo la función mutate

Usando funciones ordenamayores, ordenamenores y mutate:

Flat profile:

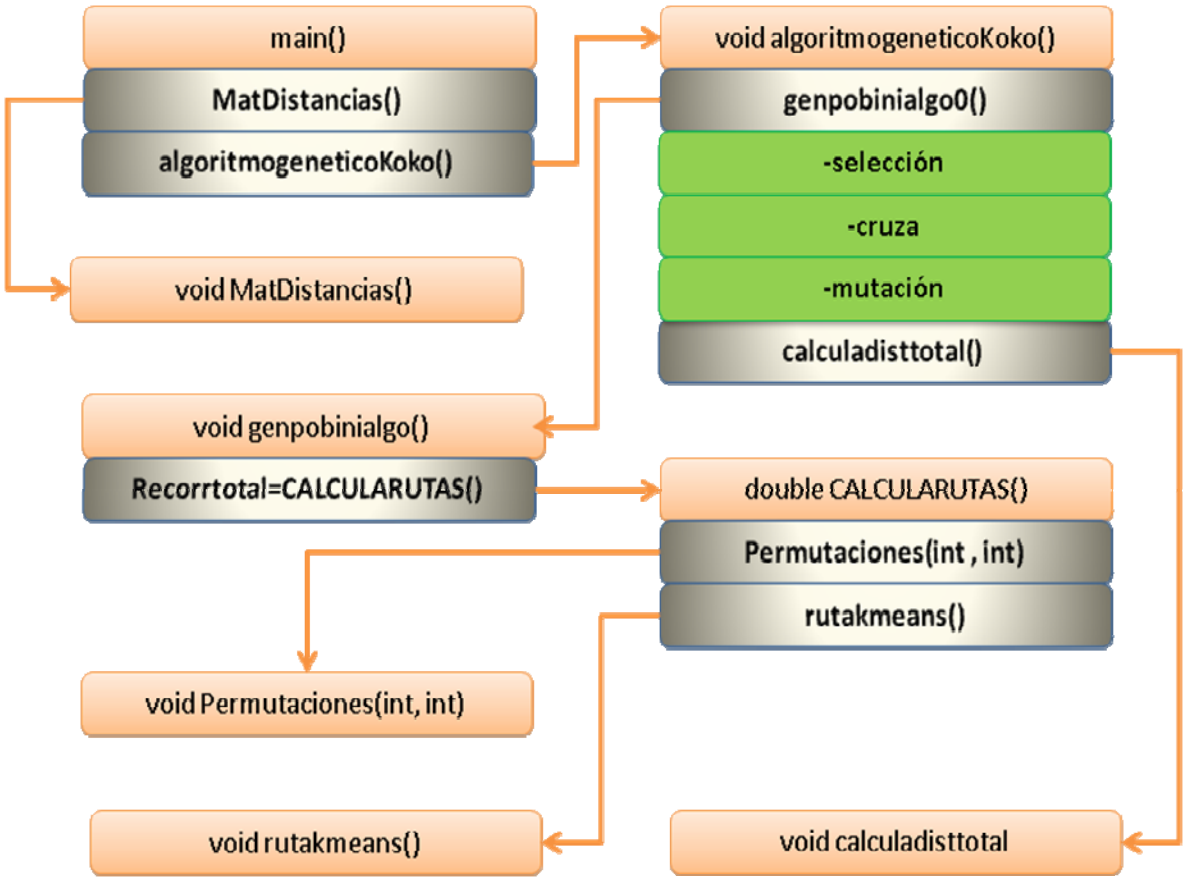
Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
55.93	42.70	42.70	1770000	0.00	0.00	ordenamayores()
37.60	71.41	28.71	1770000	0.00	0.00	ordenamenores()
6.46	76.34	4.93	5	0.99	15.27	mutate()
0.01	76.35	0.01	99	0.00	0.00	Permutaciones(int*, int)
0.00	76.35	0.00	100	0.00	0.00	CALCULARUTAS()
0.00	76.35	0.00	5	0.00	0.00	seccionbest()
0.00	76.35	0.00	5	0.00	0.00	calculadisttotal()
0.00	76.35	0.00	5	0.00	0.00	crossover()
0.00	76.35	0.00	1	0.00	0.00	global constructors keyed to archivo
0.00	76.35	0.00	1	0.00	0.00	rutakmeans()
0.00	76.35	0.00	1	0.00	0.01	genpobinialgo0()
0.00	76.35	0.00	1	0.00	76.35	algoritmogeneticoKOKO()
0.00	76.35	0.00	1	0.00	0.00	__static_initialization_and_destruction_0(int, int)
0.00	76.35	0.00	1	0.00	0.00	std::operator|(std::_Ios_Openmode, std::_Ios_Openmode)

5.1. Análisis del código secuencial aplicado al problema VPTRW

La tabla 7, expone en manera de abstracción y relación las funciones empleadas en el código VRPTW.

Tabla 7 Mecanismo de vecindad con búsqueda local y algoritmo genético para el problema de transporte con ventanas de tiempo



Pseudocódigo

main()

```
-Inicio
        - Abre archivo, "ResultadosGA.txt"
        - MAT_DISTANCIAS()
        - algoritmogeneticoKoko()
- Fin
```

Figura 46 Función main()

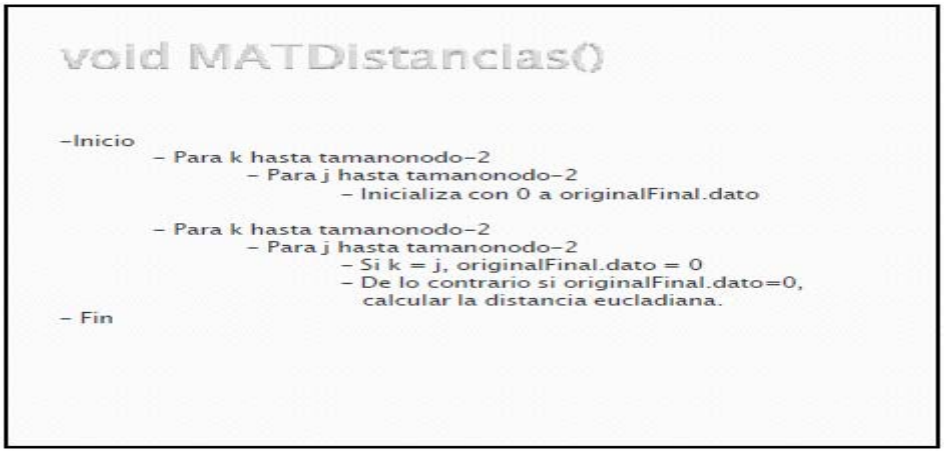


Figura 47 Calcula las distancias de las rutas de nodo a nodo mediante la función Euclidiana

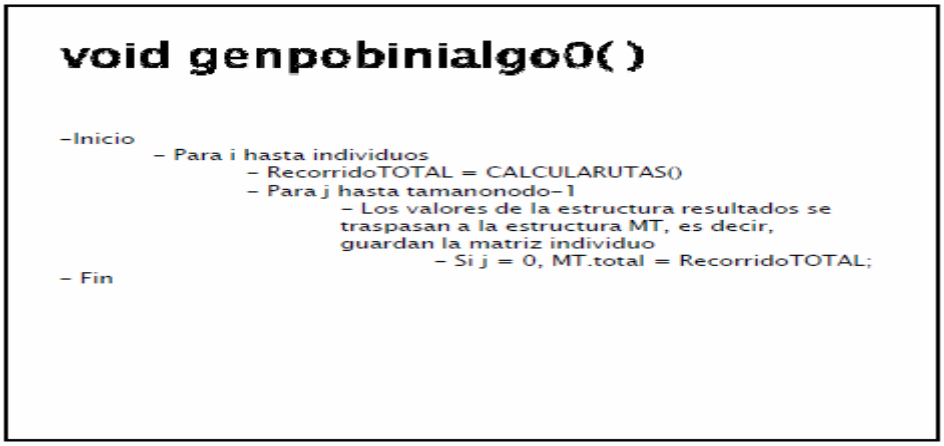


Figura 48 Generación de la población inicial, determinada por el cálculo de las distancias de las rutas.

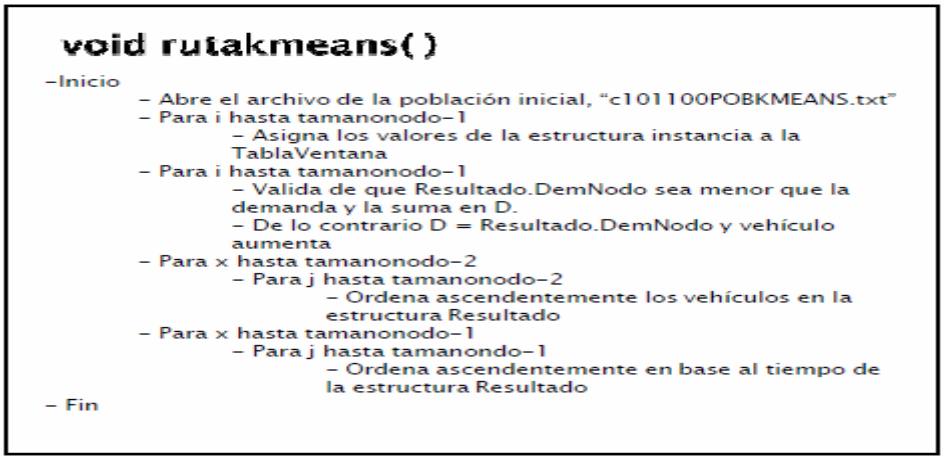
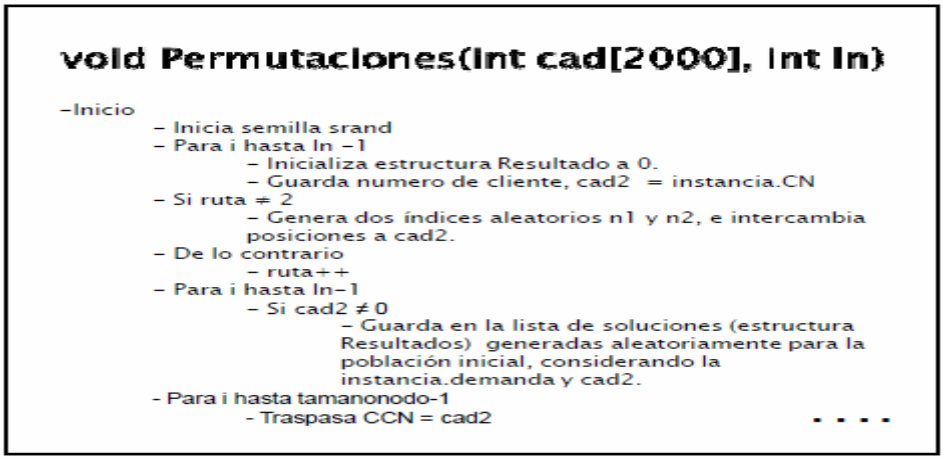


Figura 19 Algoritmo de ruta k-means, genera las primeras rutas agrupando por customers.

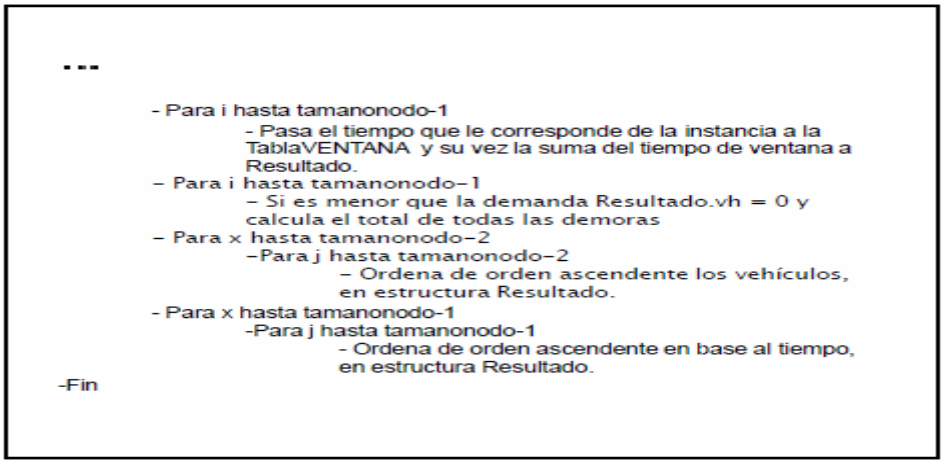


void Permutaciones(Int cad[2000], Int In)

-Inicio
- Inicia semilla srand
- Para i hasta In -1
  - Inicializa estructura Resultado a 0.
  - Guarda numero de cliente, cad2 = instancia.CN
- Si ruta = 2
  - Genera dos índices aleatorios n1 y n2, e intercambia posiciones a cad2.
- De lo contrario
  - ruta++
- Para i hasta In-1
  - Si cad2 ≠ 0
    - Guarda en la lista de soluciones (estructura Resultados) generadas aleatoriamente para la población inicial, considerando la instancia.demanda y cad2.
- Para i hasta tamanonodo-1
  - Traspasa CCN = cad2

. . . .

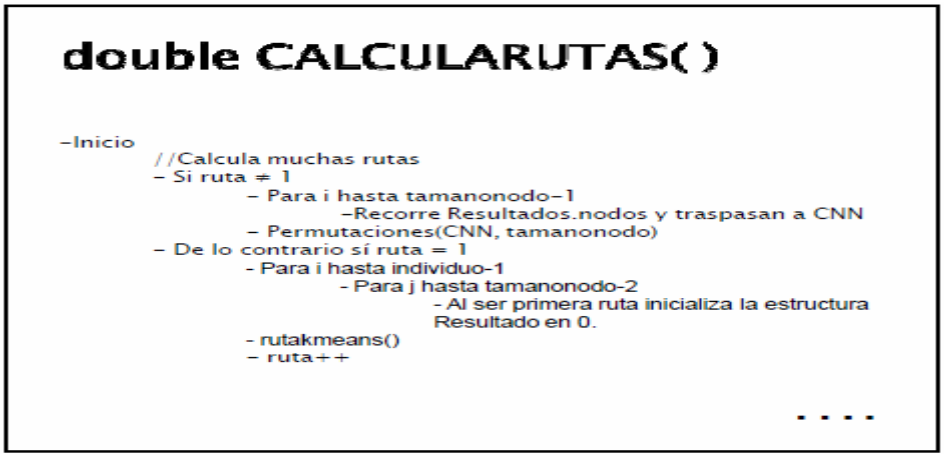
Figura 50 Considerando la cadena de la ruta, permuta dependendiendo la demanda del nodo.



- Para i hasta tamanonodo-1
  - Pasa el tiempo que le corresponde de la instancia a la TablaVENTANA y su vez la suma del tiempo de ventana a Resultado.
- Para i hasta tamanonodo-1
  - Si es menor que la demanda Resultado.vh = 0 y calcula el total de todas las demoras
- Para x hasta tamanonodo-2
  - Para j hasta tamanonodo-2
    - Ordena de orden ascendente los vehículos, en estructura Resultado.
- Para x hasta tamanonodo-1
  - Para j hasta tamanonodo-1
    - Ordena de orden ascendente en base al tiempo, en estructura Resultado.

-Fin

Figura 51 Continuación de la función de permutación.



double CALCULARUTAS( )

-Inicio
//Calcula muchas rutas
- Si ruta = 1
  - Para i hasta tamanonodo-1
    - Recorre Resultados.nodos y traspasan a CNN
  - Permutaciones(CNN, tamanonodo)
- De lo contrario sí ruta = 1
  - Para i hasta individuo-1
    - Para j hasta tamanonodo-2
      - Al ser primera ruta inicializa la estructura Resultado en 0.
  - rutakmeans()
  - ruta++

. . . .

Figura 52 Calcula rutas

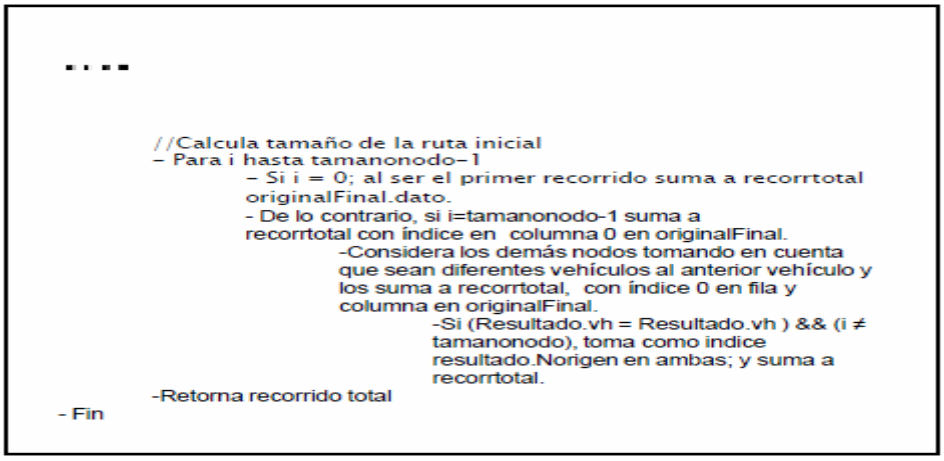


Figura 53 Función calcula rutas, determina mediante restricciones la primera ruta.

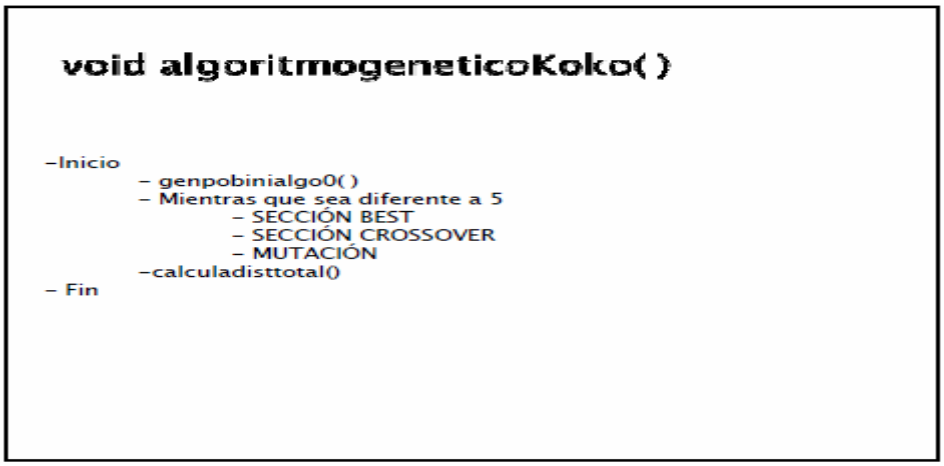


Figura 54 Función del algoritmo genético, el cual obtendrá 5 generaciones.

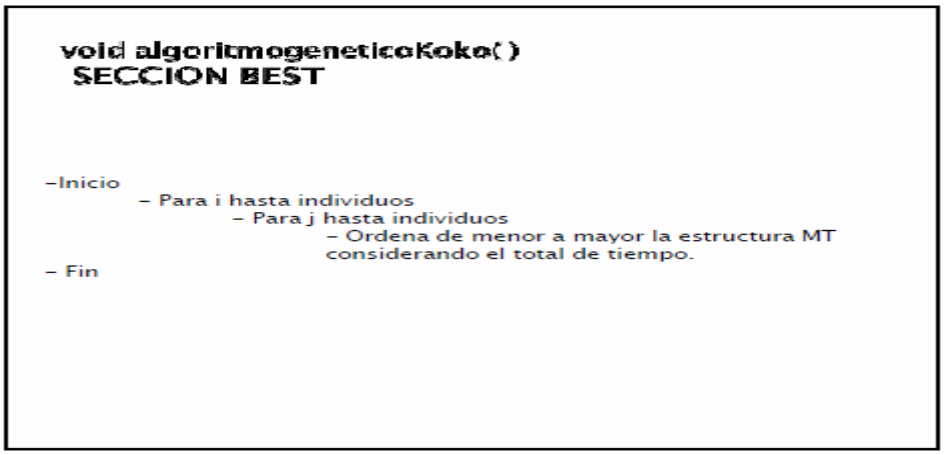


Figura 55 Sección Best, selección de los mejores individuos considerando que tengan el menor tiempo.

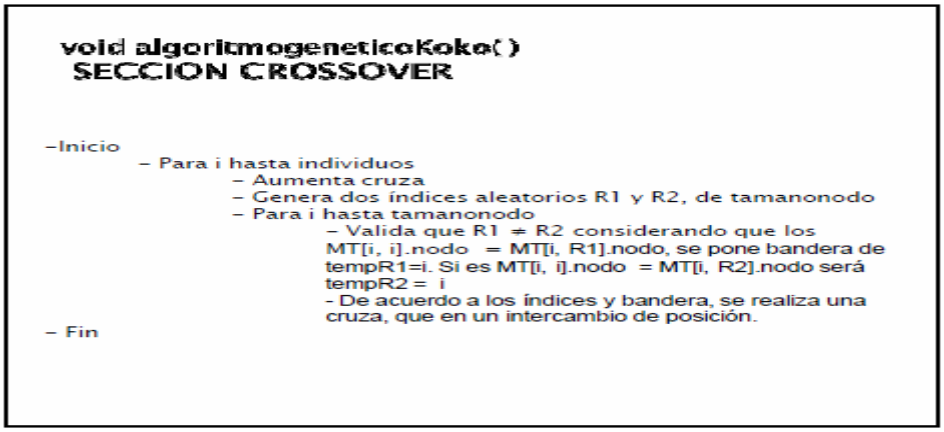


Figura 56 Sección Crossover, cruza dentro de los mejores individuos.

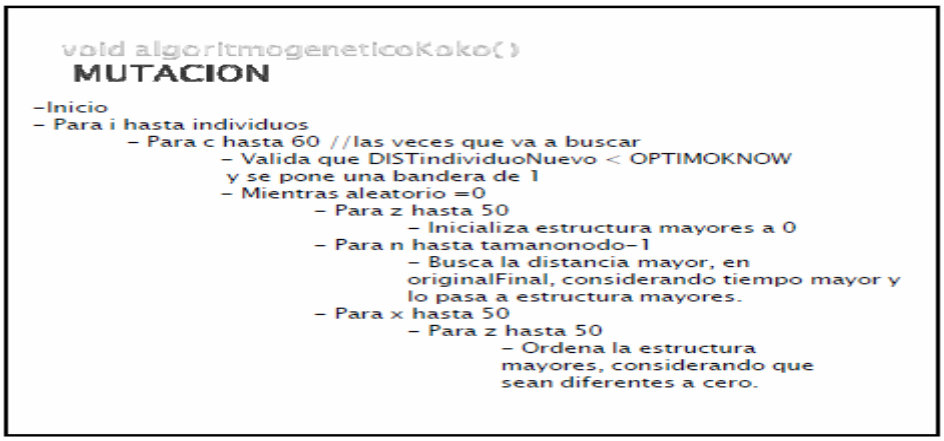


Figura 57 Sección Mutación, altera los individuos considerando restricciones establecidas.

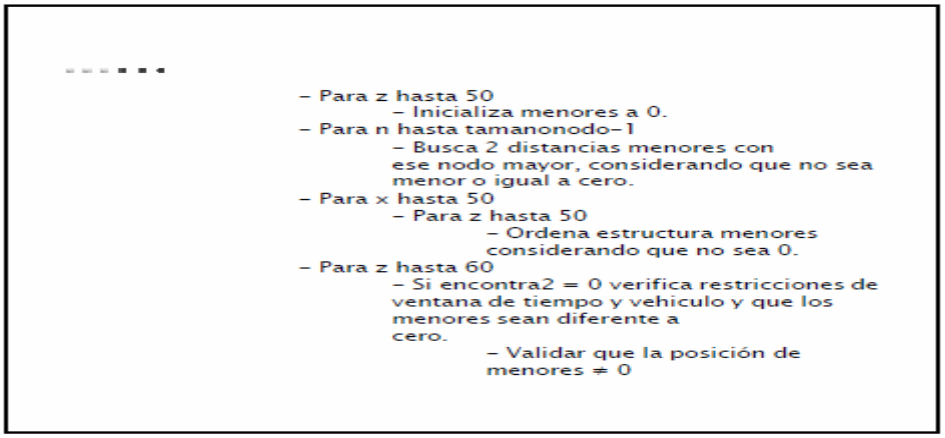


Figura 58..Continuación de sección mutación.

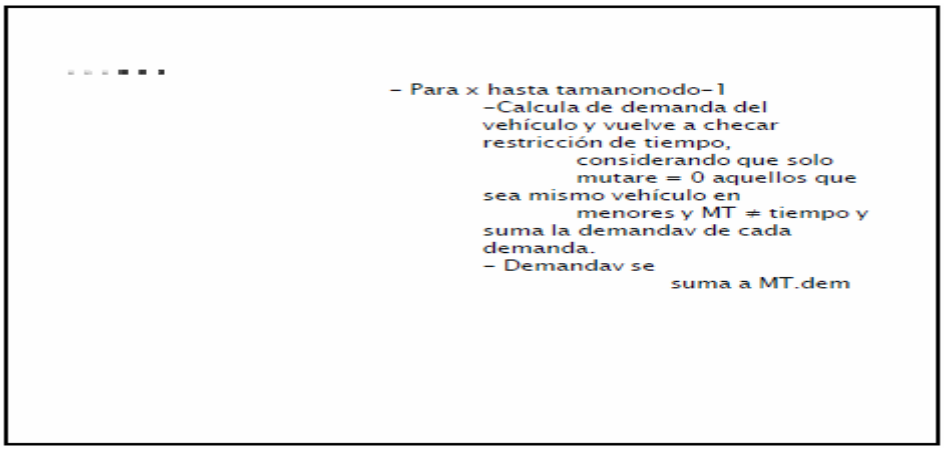


Figura 59 .continuación de sección mutación.

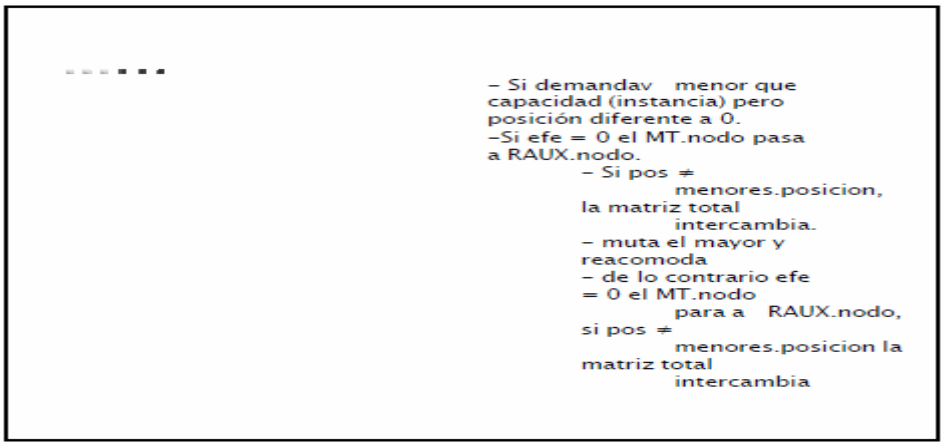


Figura 60..continuación de sección mutación.

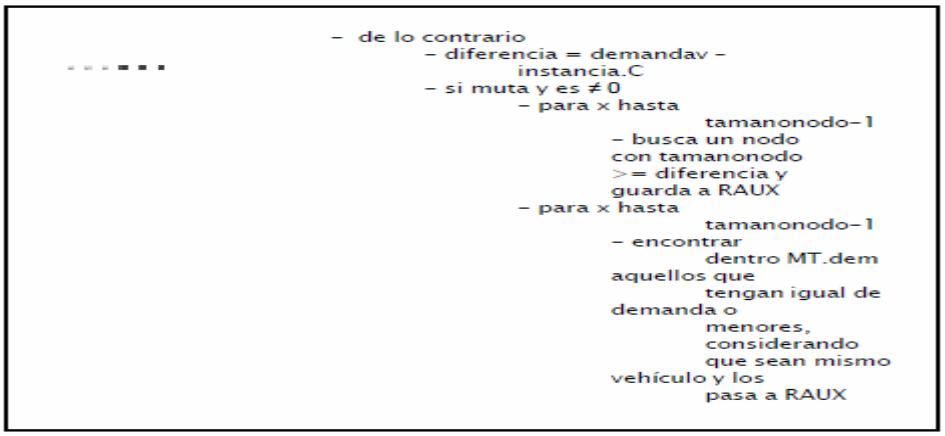


Figura 61..continuación de sección mutación.

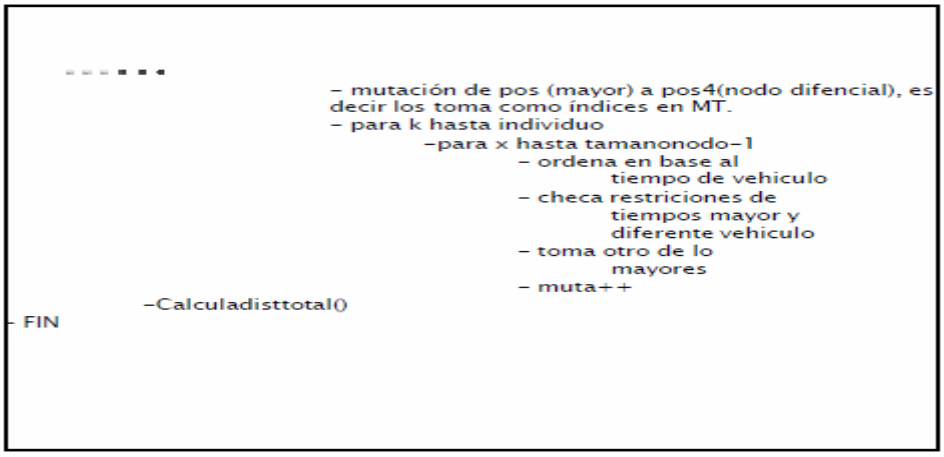


Figura 62..continuación de sección mutación.

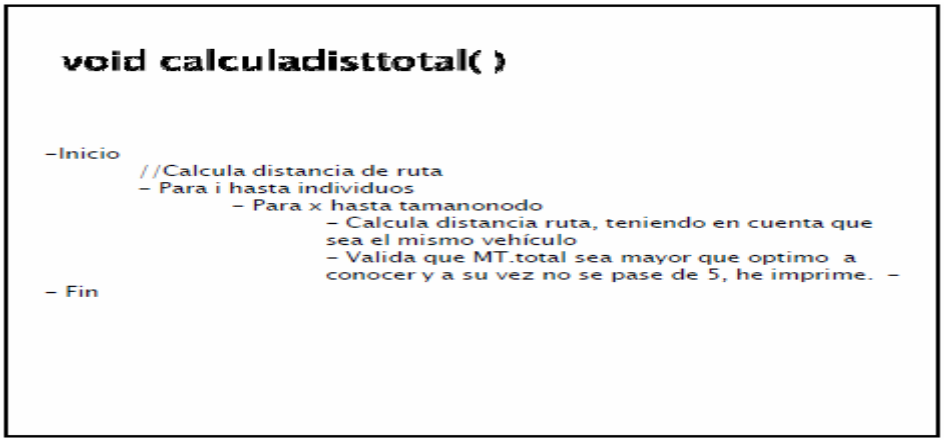
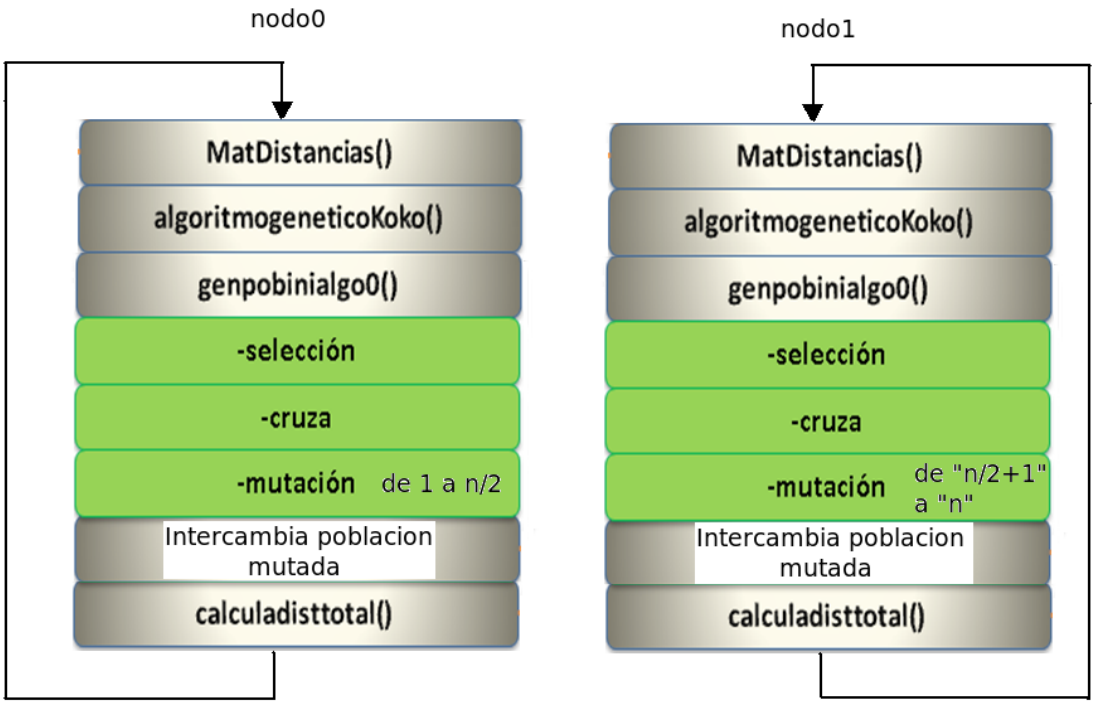


Figura 63 Calcula la distancia total, obtenida de cada generación.

5.2. Esquema de paralelización.



Después de analizar detalladamente el código, se llegó a la conclusión de que el proceso que más tiempo consumía (al rededor del 95% del tiempo de ejecución) es la mutación.
Por ello se decidió hacer la mutación sobre la mitad de la población (para dos nodos) y posteriormente intercambiar la población mutada, con el nodo complementario.
El intercambio se llevó a cabo a través del envió de información vía un archivo temporal con la mitad de la población faltante en el nodo. Una vez completada la población, se procede a continuar el proceso de cálculo de distancia y a iniciar una nueva generación.

Se implementaron dos Cluster con arquitectura HPC (los clusters nopal.itver.edu.mx y cluster.ciicap.edu.mx). Dichos cluster se interconectaron entre si por medio de una VPN sobre Internet 2, para formar así una minigrid que permita establecer secciones de confianza y lanzar programas MPI entre ambos.

Adicionalmente se uso el cluster Picolo de la Universidad Miguel Hernández de Alicante, España. Dicho cluster cuenta con 24 nodos con 2 procesadores Opteron AMD Dual Core (un total de 96 núcleos) interconectado con una red Gigabit Ethernet usando la distribución HPC Rocks.

La base experimental utilizada para medir la eficiencia del algoritmo paralelizado son las instancias de 100 clientes c101-100 y c106-100, tomados del conjunto de 56 problemas de prueba propuestos por Salomon. Estos problemas tienen en común el número de clientes y pedidos 100, cada cliente siempre hace un pedido, y el número máximo de vehículos que se pueden utilizar, 25. Además de la

distribución topológica, las diferencias vienen dadas por las cantidades de pedido y las diferentes ventanas de tiempo. Para las pruebas se manejo con 1000 individuos y 20 generaciones.

Se utilizaron diferente número de nodos, según los que se tuvieran disponibles en cada plataforma de prueba:

– Cluster ITVer máximo disponible: 4 nodos.
Cluster CIICAP máximo disponible: 8 nodos.
Cluster Picolo máximo disponible: 32 nodos
Minigrid tarántula: máximo disponible: 8 nodos.

Como se puede observar en la figura 1, el speedup obtenido para los dos benchmark usados en el cluster del ItVer es muy limitado, debido principalmente a que son pocos procesadores y que la red de interconexión interna del cluster es a 10 Mbps.

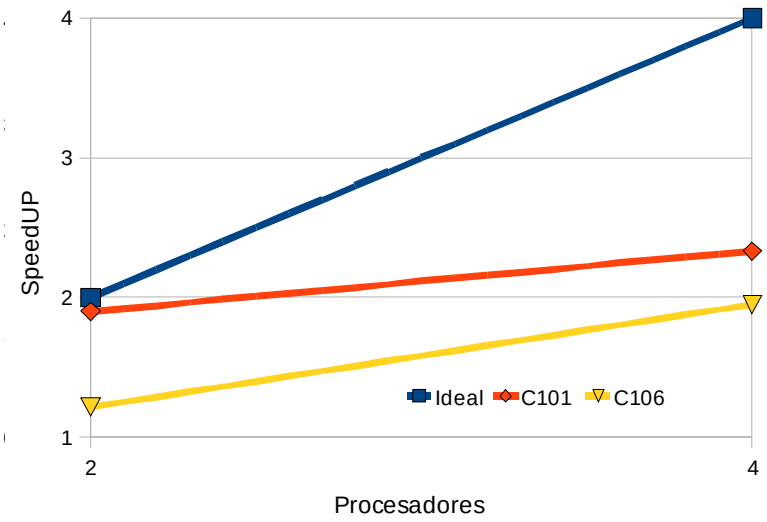


Figura 64. Speedup del las pruebas C101 y C106 en el cluster ITver

El cluster CIICAp cuenta con 16 procesadores, pero por razones técnicas no se tenían disponibles en el momento de las pruebas, por lo cual se realizaron solo pruebas con 8 procesadores. Como se puede observar la escalabilidad es buena y el comportamiento es muy cercano al ideal. Lo anterior se explica porque son maquinas nuevas con mucha memoria (2 GB cada nodo) y con una red de 1 Gbs, lo que hace el aprovechamiento sea mejor.

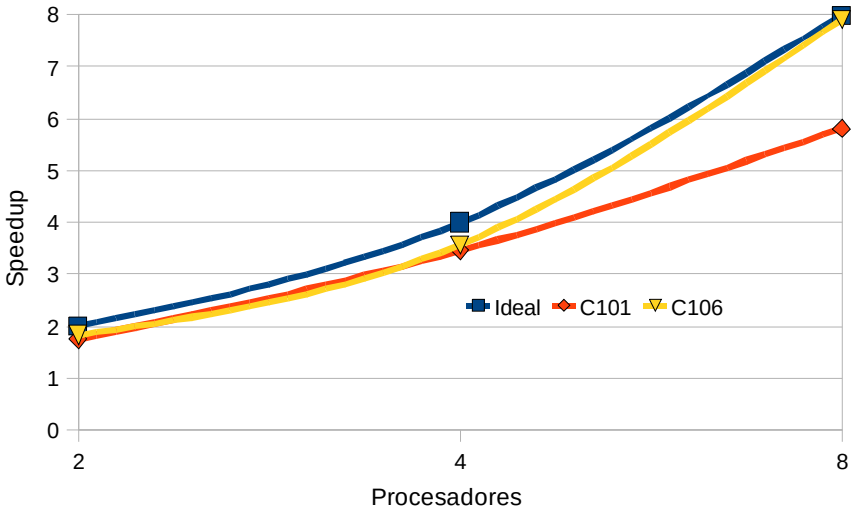


Figura 65. Speedup del las pruebas C101 y C106 en el cluster Ciicap.

En las pruebas realizadas en el cluster picolo se puede observar un efecto que no se notaba en las anteriores pruebas debido a que se tenían máximo procesadores. Picolo cuenta con 96 núcleos pero al momento de las pruebas solo estaban disponibles 56 por lo que se decidió hacer las pruebas con máximo 32 núcleos.
Como se puede observar en la figura 63, la escalabilidad es buena hasta 8 procesadores, pero en 16 y 32 procesadores casi es el mismo speedup. Lo anterior quiere decir que el algoritmo debe mejorarse buscando tener también un buen speedup para más de 8 procesadores.

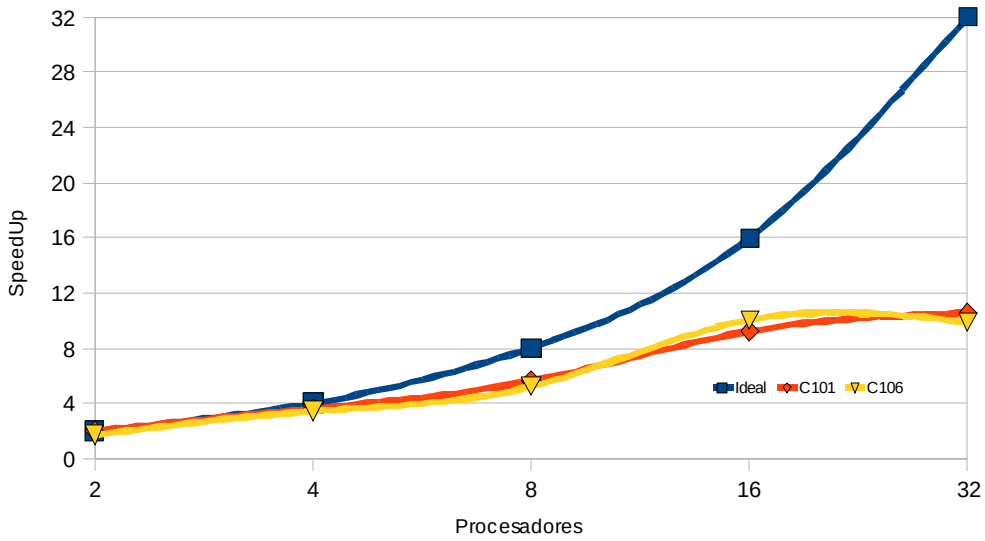


Figura 63. Speedup del las pruebas C101 y C106 en el cluster Picolo

En la minigrid se enfrentaron muchos problemas de inestabilidad en la red y de acceso a Internet 2, por lo que solo se pudieron hacer pruebas con el benchmark c101 y con 2, 4 y 8 procesadores. En estas corridas se busco que el número de núcleos de procesamiento fuera el mismo en ambos cluster para que el sistema estuviera balanceado. Además se tuvo que elaborar una versión especial que compartiera los archivos de las poblaciones entre ambos cluster, debido a que no se tiene sistema de archivos compartido mediante la grid. Lo anterior se limito ya que de haber pasado el NFS sobre I2 por la VPN esto hubiera significado meter demasiado tráfico en una red pública, lo que no es bueno.
En la figura 64 se puede observar que el comportamiento es similar al que se observa en el cluster CIICAp y en picolo, lo que quiere decir que da buenos resultados el uso de la minigrid siempre que el tráfico de datos no sea demasiado intenso.

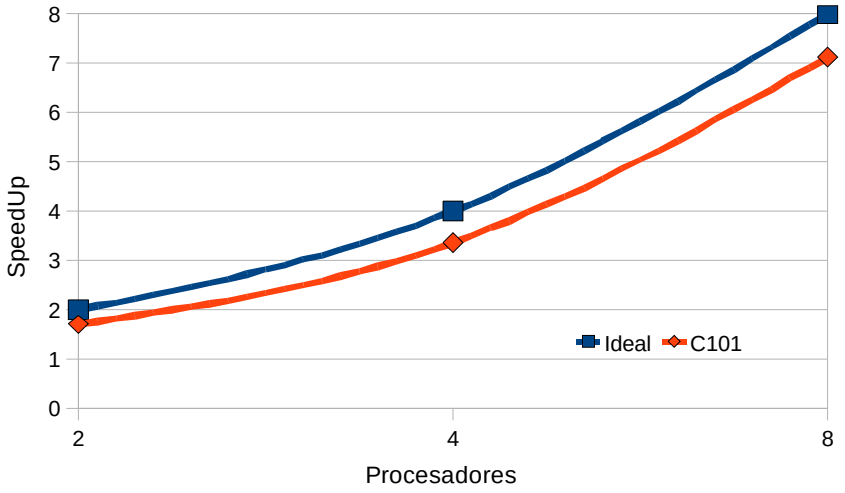


Figura 64. Speedup del las prueba C101 en la Minigrid Tarántula

En la figura 65 se puede observar una grafica en la que se ven las eficiencias promedio obtenidas en todas las configuraciones usadas de prueba.

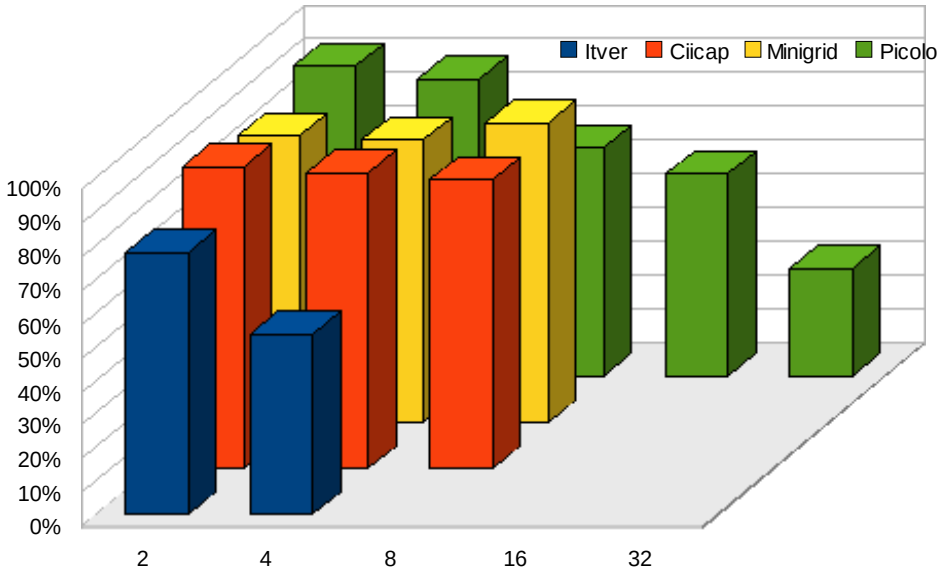


Figura 65. Eficiencias promedio en todos los clusters.

13. Monitoreo de carga de trabajo en Ganglia

La figura 66, presenta el monitoreo de la carga de procesos generadaza por algoritmo genético híbrido ejecutado para el Problema de Calendarización de Trabajos en Talleres de Manufactura. El problema presentado es el de 15 trabajos por 15 máquinas. Se puede observar en la figura que el número de CPU's totales de la MiniGRID es de 20 repartidos en 12 nodos. 15 CPU's pertenecen al cluster CIICAp y 5 al cluster Nopal. Se puede notar en Tarántula que el número de procesos en ejecución está por encima del número de CPU's. Esto es, el algoritmo genético, tiene ejecutando 20 procesos, más procesos internos de la GRID, repartidos uno en cada CPU de la GRID. También se observa que la carga de procesos supera el número de CPU's en cada cluster. Esta y todas las ejecuciones fueron realizadas con conexión de I2 entre ambos clusters. El monitoreo con ganglia

permite entender para este ejemplo que el algoritmo genético se esta ejecutando en toda la GRID por lo que hace uso te todos los recursos disponibles de CPU's. Ganglia también muestra que los requerimientos de memoria del algoritmo genético híbrido son relativamente pequeños y también constantes. Esto indica sin necesidad de evaluar la complejidad espacial del algoritmo, que ésta es lineal en el peor de los casos.

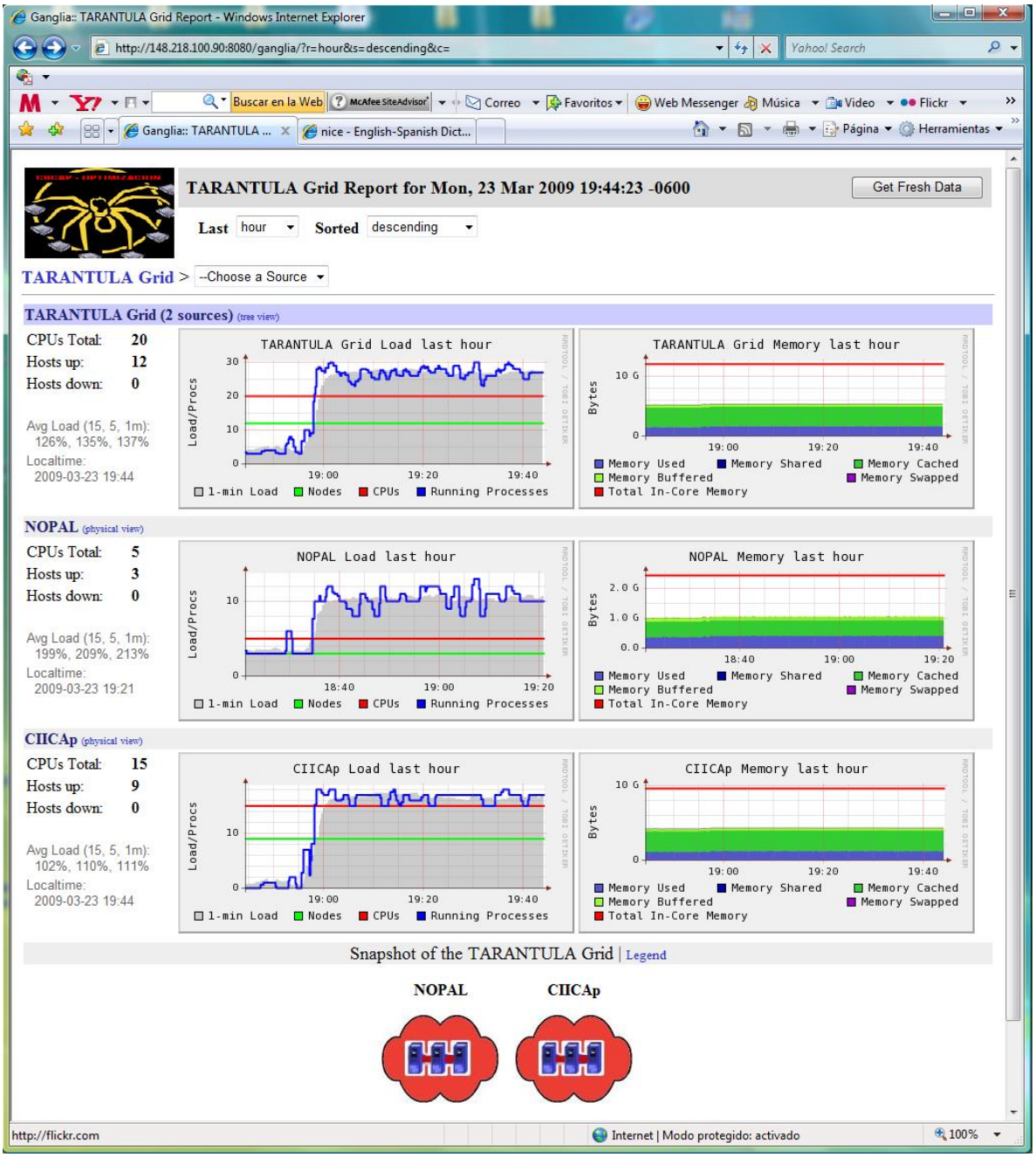


Figura 66. Monitoreo de la MiniGRID Tarántula ejecutando el algoritmo genético híbrido con carga de procesos, para el JSSP.

Se puede entender que el algoritmo podría crecer en tamaño de la población si la GRID contara con un mayor número de recursos siendo capaz de aprovechar estos al 100% para que el algoritmo pudiera explorar un mayor espacio de soluciones y explotar un mayor número de vecindades en beneficio de la búsqueda de la solución óptima global.

14. Conclusiones

El trabajo presentado en este reporte da los resultados preliminares de la ejecución de algoritmos híbridos genéticos aplicados para la solución de dos modelos de tipo NP-completos en una Grid experimental. Se puede concluir que el uso de plataforma grid para este tipo de problemas, es muy recomendable en la búsqueda de mejores cotas de solución a problemas de instancias grandes en tiempos mucho más cortos. La reducción en tiempo para encontrar mejores cotas se presenta función del número de CPU's disponibles en la grid. La eficiencia de los algoritmos se hace mayor debido a que la exploración y explotación en el espacio de soluciones aumenta conforme aumenta el número de CPU's disponibles en la grid, siendo el tiempo de ejecución del algoritmo relativamente constante.
El ancho de banda es cambiante durante todo el tiempo y tiende a mejorar por las tardes, por lo que es recomendable la ejecución de algoritmos que consumen mucho tiempo en comunicación, trabajarlos por las tardes.
La latencia es cambiante durante todo el tiempo y tiende a reducirse por las tardes, por lo que es recomendable la ejecución de algoritmos que consumen mucho tiempo en comunicación, trabajarlos por las tardes.
El uso de plataforma Grid para la ejecución de algoritmos genéticos es recomendable siempre que la comunicación entre procesos sea mínima. Por lo que se recomienda definir una estructura del algoritmo que permita en su ejecución reducir el tiempo de comunicación entre procesos. El presente trabajo presenta una estructura en la que el proceso de mutación es efectivo e independiente de otros procesos y requiere mucho tiempo de CPU. El proceso de selección y cruzamiento es muy rápido, dependiente de otros procesos y requiere de un tiempo muy corto. Definir una estructura del algoritmo donde la mutación se efectúe en los CPU's de la grid, hace que el algoritmo trabaje de forma eficiente pues no requieres de comunicación entre nodos mientras se efectúa la mutación, esto es, no afecta la latencia en el proceso más tardado del algoritmo. El algoritmo sólo requiere realizar comunicación en los procesos de selección y cruzamiento los cuales se hacen en un sólo nodo (nodo maestro), esto evita que la latencia influya de forma negativa en la ejecución del algoritmo.
El algoritmo realizado puede utilizar todos los recursos de la grid no importando el tamaño de ésta. Mientras más recursos de CPU, posea la grid de alto rendimiento, una mayor exploración y explotación se podrá generar para los problemas de tipo Np-completos en un tiempo con crecimiento polinomial. De esta manera, la Grid beneficia de forma general el campo de estudio de modelos clasificados por la teoría de la complejidad como los modelos más difíciles de resolver en el mundo, esto por que la mayoría de los modelos NP-completos pueden ser mapeados de un modelo a otro.

15. Trabajos futuros

Mejora de infraestructura. La infraestructura utilizada funciona como escenario de pruebas, pero para una infraestructura de producción, su desempeño es pobre debido a que el equipo de cómputo y el de comunicaciones no es el más adecuado, a continuación detallamos las mejoras requeridas en las cuales se buscará apoyo.

- Incrementar el numero de nodos
- Acondicionar el escenario de pruebas con aire acondicionado
- Sistema eléctrico y de tierras que soporte la grid

Incrementar el número de clusters conectados a la MiniGrid tarántula. Trabajar con la configuración de nuevos clusters dentro de los mismos campus, en unidades académicas.

Utilizar el software estándar de grids. Trabajar con herramientas de software como globos

Toolkit, que usan la mayoría de las grids y hacen más fácil la construcción de grids computacionales.

Formación de recursos humanos especializados en ambiente grid. Involucrar estudiantes de postgrado en el uso de plataformas grids dentro de su trabajo de investigación. Actualmente se tienen 2 estudiantes de doctorado y dos de maestría, todos en programas reconocidos por el PNP, ellos están haciendo su trabajo de investigación en problemas de tipo NP-completos, esta investigación dará lugar a su tesis de grado en ambiente grid.

16. Solución a problemas comunes

Uno de los problemas que nos encontramos al tratar de correr aplicaciones MPI usando los recursos de los dos Cluster, se debió a las políticas de enrutamiento de OpenMPI, debido a que la comunicación entre procesos que corren en los nodos no se permite entre nodos de subredes diferentes debido a que determina que no son enrutables.

De esta manera dado un Cluster A con una subred 192.168.100.0 y un Cluster B son una subred 192.168.101.0 OpenMPI determina lo siguiente.

> Si dos ip's hacen match después de aplicar la mascara de red, son mutuamente enrutables.
> Si dos ip's son públicas, se asume que son mutuamente enrutables.
> De otra manera no son enrutables.

Por lo tanto la solución a este problemática es usar la misma subred para los dos Cluster y hacer una segmentación en la numeración de las ip's, esto es, si usamos una subred 192.168.100.0, se determina que para el Cluster A las ip's disponibles son desde 192.168.100.1 hasta 192.168.100.100 y Cluster B desde la ip 192.168.100.101 hasta 192.168.100.200., dejando el rango 192.168.100.201 hasta 192.168.100.254 para otros servicios.

Otro manera en que solucionamos este problema es usando MPICH en lugar de OpenMPI, ésta distribución corre perfectamente sobre nodos con subredes diferentes.

17. Productos generados

- Infraestructura GRID básica
- Manuales de administración configuración de clusters
- Recursos humanos. Dos estudiantes de doctorado y dos de maestría que se encuentran en un programa de postgrado reconocido por el PNP. Desarrollarán su investigación en plataforma grid. Dos estudiantes de licenciatura que realizaron parte de su tema de tesis en este proyecto.
- Dos artículos. Se están redactando dos artículos para ser enviados a revistas indizadas.

18. Referencias

[Solomon, 1987]M. M. SOLOMON, Algorithms for vehicle routing and scheduling problem with time windows constraints. Operations research 35 (2). 1987.

[Sloan, 2001] J. D. Sloan, Network Troubleshooting Tools, ISBN 10: 0-596-00186-X, ORelly, pp. 364, 2001.

[Beasley, 03] J. E. Beasley. OR-Library: Distributing test problems by electronic mail, Journal of the Operational Research Society, Vol. 41, No. 11, 1069-1072, 1990. Last update 2003.

[Papadimitriou and Steiglitz, 82] C.H. Papadimitriou and K. Steiglitz, Combinatorial optimization: algorithms and complexity, Prentice Hall Inc., USA. ISBN 0-13-152462-3, 496 pp., 1982.

[Toth and Vigo, 01] P. Toth and D. Vigo, The Vehicle Routing Problem, Siam, ISBN 978-0898714982, 363 pp., 2001.

[Roy and Sussman, 64] Roy and Sussman, Les problemes d’ordonnancement avec contraintes disjonctives, Note D.S. no 9 bis, SEMA, Paris, France, December 1964.

[Díaz-Parra and Cruz-Chávez, 08] O. Díaz-Parra, M. A. Cruz-Chávez, Evolutionary Algorithm with Intelligent Mutation Operator that solves the Vehicle Routing Problem of Clustered Classification with Time Windows, Polish Journal of Environmental Studies, ISSN: 1230-1485, Vol. 17, No. 4C, pp. 91-95, 2008.

[Cruz-Chávez and J. Frausto-Solís, 04] M. A. Cruz-Chávez, J. Frausto-Solís, Simulated Annealing with Restart to Job Shop Scheduling Problem Using Upper Bounds, Lecture Notes in Artificial Intelligence, Springer Verlag Pub., Berlin Heidelberg, ISSN: 0302-9743, Vol. 3070, pp. 860 - 865, 2004.

19. Bibliografía consultada

Ballesteros D.P., Ballesteros P.P. (2004).Logística competitiva y administración de la cadena de suministro:Universidad Tecnológica de Pereira

Ballou, R.H.(2004). Logística. Administración de la cadena de suministros:Pearson Prentice Hall

Cruz C. M., Díaz P.O., Juárez R. D., Barreto C.E., Zavala D. C., Martínez R. M. (2008).Un mecanismo de vecindad con búsqueda local y algoritmo genético para el problema de transporte con ventanas de tiempo. Cuernavaca,Morelos: UAEM

Coello C. C. Introducción a los algoritmos genéticos

García, R. O.R., Cruz M. E.(2004).OpenMP.México D.F.:Universidad Autónoma de México

Jiménez D.(2007).Programación en memoria compartida.España:Universidad de Murcia

Goldberg. E.D. (1984).Genetic Algorithms in Search, Optimization and machine learning:Addison-wesley

Haulp R.L, Haulp S.E. (1998).Practical Genetic Algorithms.New Jersey:Willey~Interscience

Leung J. Y-T.(2004).Handbook of Scheduling: Algorithms, Models, and Performance Analysis:CRC Press

Magallón L. J. A.Programación paralela Prácticas:Univesidad de Zaragoza

Mantas L. J. M. Programación multihebra con OpenMp: Universidad de Granada

Otero T. M.(2003).Biologia+matematicas+informatica=Algoritmos genéticos:Univesidad de la Coruña.

Rodríguez L.A. (2007).Desarrollo y evaluación de algoritmos paralelos para la compresión de secuencias de video, Valencia, España: Universidad Politécnica de Valencia.
Rodríguez L.A.Programación de alto desempeño: Veracruz,Ver.:Instituto Tecnológico de Veracruz