Implementación de un cluster homogéneo para la resolución de problemas de alta complejidad computacional

Implementation of a homogeneous cluster to troubleshoot high computational complexity

Andrea Mesa Múnera, Ing., John William Branch Bedoya, Ph.D.
Escuela de Ingeniería de Sistemas, Facultad de Minas, Universidad Nacional de Colombia - Sede Medellín
amesamu@unal.edu.co, amesam@unalmed.edu.co, jwbranch@unalmed.edu.co

Recibido para revisión: 30 de Abril de 2008, Aceptado: 28 de Noviembre de 2008, Versión final: 12 de Diciembre de 2008

Resumen—**A lo largo de la historia de la computación se han presentado diversos problemas de tipo complejo que en el pasado no podían ser resueltos ó que simplemente el costo de su solución era sólo alcanzable para algunos. Sin embargo, con la tecnología actual, estos problemas son blanco fácil para los investigadores y con pocos recursos se puede llegar a la solución esperada. Este artículo tiene como objetivo mostrar la importancia que tiene hoy en día el procesamiento paralelo y distribuido para la solución de diversos problemas que se presentan en la vida diaria tomando como enfoque principal el manejo de un cluster.**

Palabras Clave—**Hardware, Arquitecturas paralelas y sistemas operativos. Sistemas distribuidos, Redes y teleinformática, Cluster, Procesamiento paralelo y Computación distribuida.**

Abstract— **In the computation's story there has been presented several problems about complex problems that in the past it can't be solved or just someones can reach the solution. However, with the current technology, those problems are easy for the researchers and even more with scant resources it can get good results.This article has as objective to shown the importance now a days about the parallel and distribuited processing for solving diferents problems that are commun in regular life taking as the main approach the cluster manage.**

Keywords—**Hardware, Parallel architectures and operating systems. Distributed systems, Networks and teleinformation, Cluster, Parallel processing and distributed computing.**

I. INTRODUCCIÓN

En la actualidad y basándose en los principios de la computación se han planteado algoritmos secuenciales para tratar de dar solución a gran cantidad de problemas. A medida que estos se han tornado más y más complejos, las soluciones basadas en este tipo de algoritmos han pasado a ser ineficientes, debido a que su uso implica un consumo de tiempo considerable durante su ejecución.

Para resolver estos problemas complejos nacieron los llamados supercomputadores, los cuales cuentan con arreglos de microprocesadores que trabajan en sincronía empleando procesamiento paralelo.

En los últimos años, el personal académico de diversas universidades y centros de investigación se han dado a la tarea de aprender a construir sus propios "supercomputadores" conectando computadores personales y desarrollando software para enfrentar tales problemas. A estos "supercomputadores" se les conoce con el nombre de cluster, el cual no había sido un concepto muy difundido hasta hace poco tiempo.

Un cluster se puede definir como un grupo de múltiples computadores unidos por una red de alta velocidad, de tal forma que el conjunto puede ser visto como una única máquina, pero que por su poder de cómputo resuelve problemas que un solo equipo de escritorio no podría hacer.

El presente artículo está organizado de la siguiente manera: en primer lugar se hará una revisión del estado del arte. Luego se efectuará una contextualización sobre la computación distribuida y el procesamiento paralelo. Posteriormente se dará

a conocer el proceso usado para implementar el cluster y la información relevante para su desarrollo. Finalmente se presentarán: el análisis de los resultados, las conclusiones y el trabajo futuro.

II. ESTADO DEL ARTE

Los supercomputadores toman vida en los años 40 con la creación del ENIAC (Electronica Numeral Integrator and Computer), el cual, según [1], fue un computador electrónico digital desarrollado por John Mauchly y John Presper Eckert para fines generales a gran escala. En su época fue considerado la máquina más grande el mundo y era controlado a través de un tren de pulsos electrónicos. El ENIAC estaba dividido en 30 unidades autónomas, las cuales eran capaces de operar simultáneamente logrando así la realización de varias tareas y cálculos en paralelo. Este computador nunca pudo funcionar las 24 horas todos los días. Estuvo en funcionamiento hasta 1955 con mejoras y ampliaciones, y se dice que durante su vida operativa efectuó más cálculos matemáticos que los realizados por toda la humanidad anteriormente.

Durante los años 50 el considerado padre de la supercomputación Seymour Cray trabajó en ERA (Engineering Research Associates). Allí fue uno de los responsables del diseño del UNIVAC 1103 (Universal Automatic Computer), el cual fue el primer computador fabricado para fines comerciales y el primero en utilizar un compilador para traducir lenguaje de programa a lenguaje de máquinas. El UNIVAC 1103 tenía la capacidad de hacer interrupciones [2] y [3].

Basados en [4], en la década de los 60 aparecen las primeras máquinas paralelas, aunque con muy poco éxito comercial. La universidad de Illinois desarrolla el ILLIAC IV, el cual poseía 64 elementos de cálculo, que disponían cada uno de su propia memoria y estaban todos físicamente unidos por medio de una red de interconexión propia del sistema. En la década de los 60 se crea además una auténtica industria informática alrededor de los computadores de la época evolucionados respecto a los primitivos y ello propiciado por el progreso de la electrónica en cuanto al tamaño de los circuitos, su costo de fabricación y su rapidez. La mayor parte de las máquinas paralelas de esta época, máquinas Cyber, se conocían como calculadores vectoriales y poseían un solo procesador.

De [5], a principios de los años 70 la aplicación predominante del computador era el procesamiento de datos administrativos. Los banqueros, los administradores de universidades y los ejecutivos publicitarios se sorprendían ante la velocidad sensacional con que los grandes computadores de millones de dólares procesaban datos. Los ingenieros y científicos se mostraban agradecidos por este tremendo logro tecnológico, pero distaban de estar satisfechos.

Cuando los ejecutivos empresariales hablaban acerca de la capacidad ilimitada, los ingenieros y científicos sabían que deberían esperar futuros avances antes de que pudieran usar los computadores para manejar problemas complicados. Los ingenieros automotrices aún no podían construir prototipos tridimensionales de automóviles en una computadora. Los físicos no podían investigar las actividades de un átomo durante una explosión nuclear. Las comunidades de ingenieros y científicos tenían una necesidad apremiante de computadores más potentes. En respuesta a esa necesidad, los diseñadores de máquinas empezaron a trabajar en lo que ahora se conoce como Supercomputadores.

De [6], en el año 1972 Seymour Cray fundó Cray Research en Wisconsin, con el compromiso de dedicarse a construir exclusivamente supercomputadores y además de uno en uno por encargo. El primer sistema Cray-1 fue instalado en el laboratorio "Los Alamos" en 1976 y era el único en su diseño ya que incorporaba el primer ejemplo práctico en funcionamiento de procesador vectorial, junto con el procesador escalar más rápido del momento.

En la investigación espacial, la utilización de computadores se convirtió en esencial. La nave Voyager 2, que fue lanzada el 20 de agosto de 1977 con la misión de explorar los planetas exteriores al sistema solar, iba equipada con seis máquinas diferentes, con capacidad de 540 Megas, algo portentoso para la época [5].

A finales de los 70 aparecen los sistemas paralelos multiprocesador, que introducen varias categorías: máquinas vectoriales multiprocesador, multiprocesadores de memoria distribuida y las máquinas asíncronas [4].

Así continúa el desarrollo de los supercomputadores con el transcurrir del tiempo, los cuales fueron adquiriendo costos bastante elevados que no todas las entidades investigativas podían incurrir; por lo tanto se vieron forzados a buscar alternativas más asequibles en cuanto a costos y que ofrecieran los mismos resultados o incluso mejores.

De acuerdo a [7], en 1994, se integró el primer cluster de computadores en el Centro de Vuelos Espaciales Goddard de la NASA, para resolver problemas computacionales que aparecen en las ciencias de la tierra y el espacio. Los pioneros de este proyecto fueron Thomas Sterling, Donald Becker y otros científicos de la NASA. El cluster de PCs desarrollado tuvo una eficiencia de 70 megaflops (millones de operaciones de punto flotante por segundo). Los investigadores de la NASA le dieron el nombre de Beowulf a este cluster.

En 1996, hubo también otros dos sucesores del proyecto Beowulf de la NASA. Uno de ellos es el proyecto Hyglac desarrollado por investigadores del Instituto Tecnológico de California (CalTech) y el Laboratorio de Propulsión Jet (JPL), y el otro, el proyecto Loki construido en el Laboratorio Nacional de Los Alamos, en Nuevo México. Cada cluster se integró con 16 microprocesadores Intel Pentium Pro y tuvieron un rendimiento sostenido de más de un gigaflop con un costo menor a $50,000 dólares.

Hoy en día, la existencia de superordenadores que trabajen en tiempo real, se ha convertido en una necesidad [5]. Se busca entonces que los nuevos supercomputadores alcancen velocidades cada vez mayores con el fin de que se puedan solucionar fácilmente diversas aplicaciones del mundo real.

III. PROCESAMIENTO PARALELO

De [8], la computación paralela o procesamiento en paralelo consiste en acelerar la ejecución de un programa mediante su descomposición en fragmentos que pueden ejecutarse de forma simultánea, cada uno en su propia unidad de proceso. Surge así, de forma natural, la idea de la computación paralela, que genéricamente consiste en utilizar n computadores para multiplicar, idealmente por n la velocidad computacional obtenida de un único computador. Por supuesto, esta es una situación ideal que muy rara vez se consigue en la práctica. Normalmente, los problemas no pueden dividirse perfectamente en partes totalmente independientes y se necesita, por tanto, una interacción entre ellas que ocasiona una disminución de la velocidad computacional. En este sentido se habla de mayor o menor grado de paralelismo en la medida en que un algoritmo sea más o menos divisible en partes independientes con igual costo computacional.

IV. COMPUTACIÓN DISTRIBUIDA

Según [9] y [10], la computación distribuida es un modelo para resolver problemas de computación masiva utilizando un gran número de computadores organizados en cluster incrustados en una infraestructura de telecomunicaciones distribuida.

Para resolver estos problemas se debe considerar que se pueden solucionar bajo una arquitectura que funciona a partir de procesamiento paralelo, el cual no es más que la ejecución simultánea de una misma tarea (dividida y adaptada especialmente) en múltiples procesadores, con el fin de obtener resultados en menor tiempo y de manera más eficiente.

La computación paralela y distribuida maneja dos tipos de arquitecturas principales: Cluster y Grid, las cuales hacen posible la solución de problemas frecuentes en alto desempeño computacional.

A. Cluster

De [11], los Cluster están conformados por una colección de PC's autónomos interconectados trabajando unidos como un solo recurso de computación integrado.

El concepto de Cluster nació cuando los pioneros de la supercomputación intentaban difundir diferentes procesos entre varios computadores, para luego poder recoger los resultados que dichos procesos debían producir. Con un hardware más barato y fácil de conseguir se pudo perfilar que podrían conseguirse resultados muy parecidos a los obtenidos con aquellas máquinas mucho más costosas [12].

Un cluster se puede definir como el trabajo realizado por dos o más computadores que en conjunto se encargan de proveer una determinada solución. Tiene como finalidad agrupar el poder de cómputo de los nodos implicados para proporcionar una mayor escalabilidad, disponibilidad y fiabilidad [13] y [14]. La escalabilidad es la capacidad de un equipo para hacer frente a volúmenes de trabajo cada vez mayores sin, por ello, dejar de prestar un nivel de rendimiento aceptable. La disponibilidad y la fiabilidad, se encuentran bastante relacionadas, aunque difieren ligeramente en algunos aspectos. La disponibilidad es la calidad de estar presente y listo para su uso, mientras que la fiabilidad es la probabilidad de un correcto funcionamiento.

Según [15], la idea básica de un cluster es simple, se trata de un conjunto de computadores conectados a través de una red, trabajando en un gran problema de cómputo que ha sido dividido en varios subproblemas pequeños. Existen diferentes configuraciones de hardware y mucho software disponible para trabajar en clusters, dividiendo los problemas y aprovechando al máximo los recursos disponibles. Todos estos paradigmas de resolución de problemas tienen sus bases en el "cómputo paralelo". El paralelismo consiste en poder dividir una tarea en partes que trabajen independientemente en lugar de poseer una única tarea en la que todos sus procesos se encadenan uno tras otro, necesitando de los resultados del anterior para poder comenzar.

En la actualidad existen algunas variables que afectan las características esenciales de los cluster, las cuales se encuentran directamente ligadas al rendimiento que un cluster puede presentar o a la forma en que trabaja éste (migración de procesos, latencia de red, balanceo de carga, gestión de memoria y monitorización).

B. Grid

Es una infraestructura de hardware y software que puede encontrarse distribuida y cuyo objetivo es permitir gestionar y distribuir la potencia de cálculo disponible, de tal forma que los usuarios se beneficien de la potencia de computadores subutilizados que se encuentran dispersos geográficamente [16] y [17].

V. IMPLEMENTACIÓN DE UN CLUSTER

El primer paso en el diseño de un cluster es la elección del tipo de cluster que se desea implementar.

Entre los tipos de cluster más comunes se pueden encontrar, según [18], de alta disponibilidad (ambiente a prueba de fallas), cómputo de alto desempeño (los procesos o las tareas dadas se

presentan como un solo sistema virtual) y escalamiento horizontal (se utiliza para proporcionar una sola interfaz a un conjunto de recursos que pueden aumentar o disminuir arbitrariamente su tamaño en un cierto periodo de tiempo).

A. Beowulf

Beowulf es una clase de computador masivamente paralelo de altas prestaciones principalmente construido a base de cluster de componentes hardware estándar [19].

Consta de un conjunto de nodos minimalistas, unidos por un medio de comunicaciones económico. Esto quiere decir que tienen lo mínimo para ejecutar su función, de hecho los nodos por sí solos no son capaces de ejecutar ni tan siquiera un sistema operativo [11].

Se puede ver como un supercomputador paralelo construido con hardware comercial de fácil adquisición, que posee como sistema operativo Linux.

Los clusters Beowulf son extremadamente poderosos, pero no son para todas las personas. Su principal desventaja es que requieren software diseñado para poder aprovechar los recursos del cluster [20].

B. Mosix

Según [21], la arquitectura Mosix está basada en la misma ideología de la arquitectura Beowulf. Se basa en un conjunto de parches aplicados al kernel de Linux y que asignan a todo el grupo de nodos un espacio de direcciones y de procesos común, gracias al cual los procesos migran de uno a otro con el fin de equilibrar favorablemente la carga del sistema global.

Es una herramienta diseñada para realizar balanceo de carga en el cluster de forma totalmente transparente de manera tal que los nodos se comportan como una sola máquina, y así incrementar el aprovechamiento de cada uno de los nodos [22].

La principal ventaja de Mosix frente a Beowulf se basa precisamente en esto. Mosix da mejores respuestas que Beowulf frente a la caída o inserción de nodos. A parte de esto, Mosix permite un cambio constante de aplicación, o incluso, el correr aplicaciones independientes de forma simultánea.

C. OpenMosix

OpenMosix es una extensión del proyecto Mosix. De [12], la idea de este modelo es que la distribución de tareas en el cluster la determina OpenMosix de forma dinámica, conforme se van creando tareas. Cuando un nodo está demasiado cargado, las tareas que se están ejecutando pueden migrar a cualquier otro nodo del cluster. Así desde que se ejecuta una tarea hasta que ésta muere, podrá migrar de un nodo a otro, sin que el proceso sufra mayores cambios.

De acuerdo a [23], la idea de OpenMosix es que los procesos colaboren de forma que parezca que están en un mismo nodo. Como sus algoritmos son dinámicos, tiene una fuerte ventaja sobre los algoritmos estáticos de PVM/MPI, responden a las variaciones en el uso de los recursos entre los nodos migrando procesos de un nodo a otro, de forma transparente para el proceso, para balancear la carga y para evitar falta de memoria en un nodo.

Al contrario que PVM/MPI, no necesita una adaptación de la aplicación, ni siquiera que el usuario sepa nada sobre el cluster. Para aprovechar completamente PVM/MPI hay que programar con sus librerías, por tanto hay que rehacer todo el código que haya para sacar el mayor provecho del Cluster.

OpenMosix puede balancear una única aplicación si esta está dividida en procesos lo que ocurre en gran número de aplicaciones hoy en día, también puede balancear las aplicaciones entre sí, balanceando los procesos en la mínima unidad de balanceo.

Además OpenMosix funciona a nivel de kernel por tanto puede conseguir toda la información que necesite para decidir cómo está de cargado un sistema y qué pasos se deben seguir para aumentar el rendimiento, además puede realizar más funciones que cualquier aplicación a nivel de usuario.

Después de elegir el tipo de cluster ha ser utilizado, se procede a elegir los elementos básicos que se emplearán para implementar el cluster: capacidad de procesamiento, memoria, espacio del disco de cada nodo, así como también el ancho de banda de la comunicación entre los nodos. Se deberá decidir cuáles son importantes basándose en la variedad de aplicaciones que se piensen ejecutar en el cluster, y de la cantidad de dinero que se disponga para construirlo.

Bajo estos mismos parámetros, se debe escoger también el tipo de red que interconectará los nodos [24].

Luego se debe escoger el sistema operativo a utilizar. Esta elección dependerá mucho de la máquina que se escoja. Linux es siempre una opción en cualquier máquina, y es la elección más común. Muchas de las herramientas necesarias para la implementación de un cluster han sido desarrolladas bajo Linux. Pero existen clusters corriendo bajo Windows NT, UNIX, Solaris y AIX.

La primera elección para este trabajo fue implementar un Cluster tipo OpenMosix. Se eligió éste porque al estudiar los diversos tipos se encontró que era el más sencillo de trabajar ofreciendo un mayor rango de las aplicaciones que se pueden correr, ya que no exige que estén escritas en un lenguaje explicito, como por ejemplo el cluster tipo Beowulf, en el cual las aplicaciones deben estar escritas utilizando los elementos de las librerías PVM o MPI.

Por otro lado, OpenMosix incluye unas herramientas de usuario que ayudan con la administración del Cluster haciendo que la labor de mantenimiento sea menos molesta.

En esencia OpenMosix es un modelo de Cluster de balanceo de carga, lo que significa que la distribución de los procesos se

hace automáticamente, a diferencia de otros modelos, por lo cual es idóneo para el propósito de éste trabajo, que es demostrar que un cluster efectivamente aumenta la eficiencia a la hora de resolver problemas robustos de ingeniería.

Para la elección de los elementos básicos, se tomaron aquellos elementos que se tenían a la mano. Se dispuso de 6 computadores, todos con las mismas características de hardware: Procesador Intel Pentium 4, 1.50 GHz AT/AT Compatible, 512 Mb de memoria RAM, Controlador Fast Ethernet integrado 3Com 3C920 (compatible 3C905C-TX) y un Switch ENH908-NWY 10/100Mbps NWay Switch.

Como se escogió trabajar con OpenMosix, el sistema operativo a utilizar debía ser Linux, pues OpenMosix está desarrollado específicamente para este sistema. Se trabajó con la versión 2.4.26 de Linux.

Si se realiza apropiadamente, un cluster puede ser relativamente fácil de mantener. Esta labor es aún menos complicada en un Cluster OpenMosix, pues no sólo incluye herramientas para tal fin, sino que también la adición de nodos al Cluster es una tarea mas sencilla que en otros tipos de Cluster, y además si algún nodo llega a fallar su trabajo será repartido automáticamente a los otros nodos que se encuentran en funcionamiento, sin perder la efectividad de éste.

VI. ANÁLISIS DE RESULTADOS

Para realizar la validación del prototipo se efectuaron diversas pruebas las cuales debían estar en forma paralela o distribuida para que el cluster pudiera ser lo suficientemente útil y poder concluir que el manejo de éste resulta ser efectivo; el algoritmo que fue utilizado para realizar las pruebas del Cluster OpenMosix fue el Algoritmo de PI de Plouffe y Bellard llamado Mospi, el cual se encarga del cálculo de dígitos decimales del número PI. Este algoritmo fue implementado por la facultad de Informática y Comunicaciones de la Universidad Central Queensland. La implementación del algoritmo permite seleccionar el número de dígitos decimales a calcular y la cantidad de procesos requeridos para el cálculo.

Para las pruebas se decidió calcular los 500, 750, 1000 y 1500 dígitos decimales, en 1, 5, 10, 15, 20, 25, 30, 35, 40, 45 y 50 procesos, midiendo cuanto tiempo requería ejecutar este cálculo utilizando de 1 a 6 nodos del Cluster OpenMosix. Por cada proceso se hicieron 10 corridas, con lo cual se obtuvo una muestra considerable de datos que explican el comportamiento del algoritmo en el Cluster.

Las siguientes tablas y gráficas presentan los tiempos de ejecución del algoritmo para el cálculo de los dígitos de PI con 30 procesos, el cual, de acuerdo a lo observado en los resultados obtenidos, es el valor que minimiza el tiempo de ejecución; para un número de procesos superior a 30, se presenta un aumento del tiempo de ejecución, debido al incremento del tráfico de red generado por el aumento de la carga de entrada/salida de los procesos.

Tabla 1. Resultados de "MOSPI" para calcular 500 dígitos en 30 procesos

Número de nodos	Promedio	Porcentaje de mejora relativa	Porcentaje de mejora total
1	6,16		
2	6,35	-3,08 %	-3,08 %
3	6,16	2,99 %	0 %
4	5,89	4,38 %	4,38 %
5	5,96	-1,19 %	3,25 %
6	5,99	-0,50 %	2,76 %

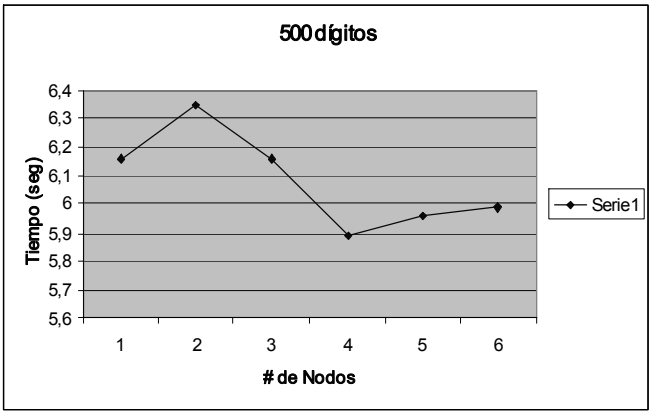

Figura 1. Resultados de la ejecución de "MOSPI" para calcular 500 dígitos en 30 procesos.

Tabla 2. Resultados de "MOSPI" para calcular 750 dígitos en 30 procesos.

Número de nodos	Promedio	Porcentaje de mejora relativa	Porcentaje de mejora total
1	19,51		
2	18,12	7,12 %	7,12 %
3	13,64	24,72 %	30,09 %
4	14,14	-3,67 %	27,52 %
5	12,96	8,35 %	33,57 %
6	12,2	5,86 %	37,47 %

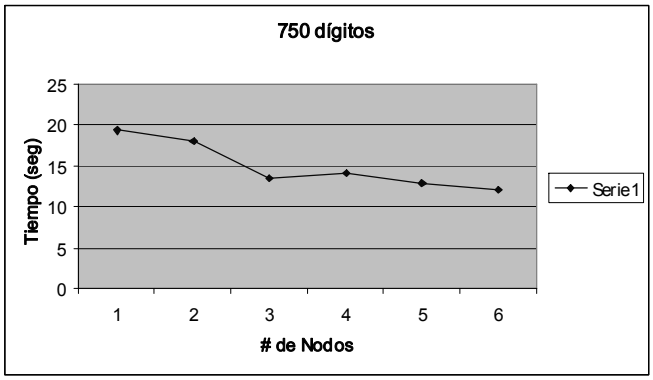

Figura 2. Resultados de la ejecución de "MOSPI" para calcular 750 dígitos en 30 procesos.

Tabla 3. Resultados de "MOSPI" para calcular 1000 dígitos en 30 procesos

Número de nodos	Promedio	Porcentaje de mejora relativa	Porcentaje de mejora total
1	44,65		
2	38,82	13,06 %	13,06 %
3	26,3	32,25 %	41,10 %
4	24,89	5,36 %	44,26 %
5	23,73	4,66 %	46,85 %
6	21,97	7,42 %	50,80 %

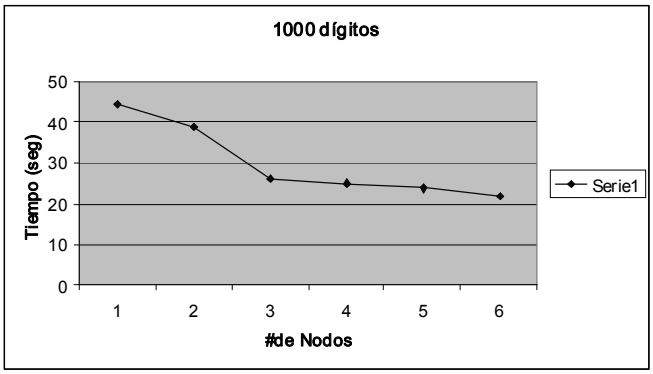

Figura 3. Resultados de la ejecución de "MOSPI" para calcular 1000 dígitos en 30 procesos

Tabla 4. Resultados de "MOSPI" para calcular 1500 dígitos en 30 procesos

Número de nodos	Promedio	Porcentaje de mejora relativa	Porcentaje de mejora total
1	141,81		
2	86,93	38,70 %	38,70 %
3	63,88	26,52 %	54,95 %
4	58,51	8,41 %	58,74 %
5	50,76	13,25 %	64,21 %
6	48,34	4,77 %	65,91 %

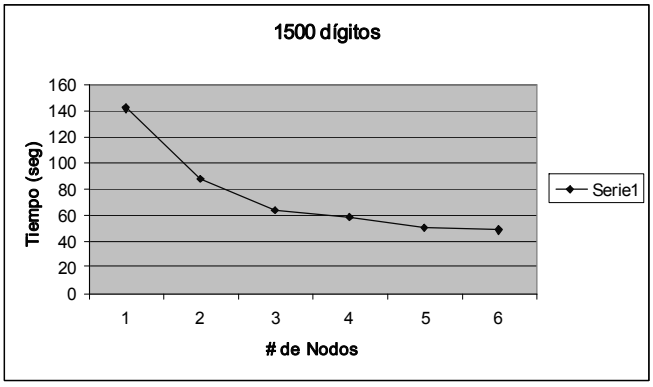

Figura 4. Resultados de la ejecución de "MOSPI" para calcular 1500 dígitos en 30 procesos.

Se puede observar en la Figura 1, que para el cálculo de los 500 primeros dígitos de Pi, el cluster no hace una verdadera diferencia puesto que los tiempos de ejecución son similares entre 1 y 6 nodos. El mayor rendimiento se da para 4 nodos aunque la diferencia con los demás nodos no es muy considerable.

Esta situación comienza a variar en el cálculo de los 750 dígitos de Pi, en donde se observa que se obtiene una disminución del tiempo de ejecución de hasta un 37% aproximadamente, lo cual es algo que nos comienza a demostrar la real eficiencia de un cluster. Sin embargo los porcentajes de mejora relativa indican que añadir más nodos no produciría mayor diferencia, pudiendo sí provocar el aumento de tráfico en la red, lo que causaría una disminución del rendimiento.

Algo similar ocurre cuando se calculan los 1000 y 1500 dígitos, para los cuales se tuvo una mejora de hasta el 50 % y 66 % respectivamente. Se puede observar la misma tendencia a no mejorar mucho a partir de 4 nodos.

En todos los casos el tiempo de ejecución no presenta una relación lineal con respecto al número de nodos. Este resultado se debe a que el tiempo de ejecución depende también de la latencia de red, carga del IPC y de las entradas/salidas requeridas por proceso.

En conclusión, se observa que a mayor cantidad de nodos en un cluster, la eficiencia aumenta, aunque existe un punto, para nosotros sólo deducible a partir de la experiencia, en el que añadir más nodos perjudicaría dicha eficiencia.

Como factor de error importante se tiene que en los resultados obtenidos, no se sabe que tanto pueda influir la forma en la que está realizado el algoritmo.

Es un algoritmo que es paralelizable por naturaleza, pero no se contó con resultados de otros estudios y pruebas realizados sobre el mismo, por lo cual no se puede concluir si algunos de los efectos indeseables observados sean debidos al código o a alguna falla en el cluster.

VII. CONCLUSIONES Y TRABAJO FUTURO

La evolución de los computadores paralelos ha sido clara en varias direcciones, una de las cuales es la utilización de hardware de cómputo estándar. En este sentido, los microprocesadores de uso masivo en los computadores de bajo costo como las estaciones de trabajo y PCs son utilizados en los computadores paralelos que se reportan entre las de mayor capacidad de cálculo absoluta.

Las capacidades tecnológicas de OpenMosix son bastante grandes ya que permiten aprovecharse en proyectos de pequeña, mediana o gran dimensión gracias a su escalabilidad y flexibilidad. Por otro lado se ve que las aplicaciones de

programación libre se muestran como mejor solución pues ponen en manos del usuario las mejores herramientas que posibilitarán un futuro enriquecedor tanto tecnológico como social.

De la experiencia se pudo comprobar que en efecto un Cluster puede reducir el tiempo de ejecución de cualquier aplicación, aumentando el rendimiento de las máquinas maximizando la utilización de los recursos disponibles.

En el escalafón más bajo en cuanto a costo de cómputo paralelo se pueden ubicar a las redes locales de computadores, que tienen el mismo tipo de procesadores de base pero que en principio no han sido orientadas a cómputo paralelo. De hecho, la mejor relación costo/rendimiento de hardware paralelo es el de las redes locales ya instaladas porque no tienen ningún tipo de costo de instalación ni de mantenimiento, dado que su existencia es independiente del cómputo paralelo. Sin embargo, no se pueden dejar de lado otros costos adicionales que tienen estas redes, tales como el de instalación y mantenimiento del software específico para cómputo paralelo y el de la baja disponibilidad de los computadores que, como se ha mencionado, no tienen como prioridad la ejecución de programas paralelos [25].

La amplia variedad del hardware de red disponible y el soporte de software hace de la selección de la tecnología de interconexión del cluster una tarea difícil.

En esta, la era de las redes, el nivel de servicio de un sistema constituye uno de sus principales activos. Los clientes (usuarios) están a pocos movimientos de la competencia, por lo que siempre deben estar disponibles.

Si bien el procesamiento paralelo ofrece una ventaja definitiva en cuanto a costos, su principal beneficio, la escalabilidad, es decir, su capacidad de crecimiento, puede ser difícil de alcanzar. Esto se debe a que conforme se añaden procesadores, las disputas por los recursos compartidos se intensifican [26].

La evolución de los clusters se puede enfocar sobre los tres componentes fundamentales de ellos: hardware (nodos), software y redes (interconexión).

La tendencia es desarrollar mejores herramientas de programación y administración, mejorar los precios y aumentar la capacidad de diagnóstico y recuperación de errores para lograr un mantenimiento más simple [27].

Hasta el momento se han analizado casos en los que para el problema de cálculos robustos, se usan arquitecturas de cluster homogéneas, que aunque en ocasiones son bastante prácticas, en otras por el contrario no lo son, como es el caso de OpenMosix.

Hoy en día se requiere que se resuelvan aplicaciones sobre plataformas heterogéneas, ya que en ocasiones no se cuenta con el mismo hardware en los diferentes equipos que conformarán el cluster. Por otro lado, se hace necesario trabajar con variables que afectan de forma circunstancial el rendimiento general del cluster, tales como: balanceo de carga, latencia de red, gestión de memoria, entre otras.

En el mundo de la computación se observa la necesidad de migrar procesos y/o tareas de un equipo que se encuentre bastante cargado a otro que presente mayor disponibilidad, para así optimizar diversos recursos. Este procedimiento de migrar procesos es lo que se conoce como balanceo de carga, la cual es una de las variables que más ayuda a una organización a mejorar la velocidad de procesamiento o el tiempo de ejecución en sus procesos.

Por lo tanto, como trabajo futuro se propone explorar nuevas propuestas para la solución del problema del balanceo de carga en cluster heterogéneos con el fin de solucionar problemas de alto poder computacional. Para esto se hace necesario tomar la mejor decisión a la hora de implementar algún método, seleccionando la arquitectura de red y modelo computacional que mejor se adapte a la variable a estudiar: balanceo de carga.

AGRADECIMIENTOS

A todas aquellas personas interesadas en el tema y que por sus comentarios y sugerencias aportaron para que la culminación de este artículo fuera un hecho.

REFERENCIAS

[1] Autor Desconocido, s.a. ENIAC -Electronica Numeral Integrator and Computer. Departamento de matemática aplicada. Universidad Politécnica de Madrid. http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/ENIAC.htm

[2] Autor Desconocido, 1999. Auge y ocaso de la supercomputacion: la figura de Seymour Cray. En: Lo que cuentan de ti. LQCTI N-9. Magazine independiente sobre nueva economía, tecnología e innovación. pp. 18-20.

[3] http://www.lawebdelprogramador.com/diccionario/mostrar.php?letra=U&pagina=2

[4] García, F., s.a. La supercomputación. Autores científico-técnicos y académicos.

[5] Autor Desconocido., s.a. Superordenadores. Salamanca (España). http://html.rincondelvago.com/superordenadores.html

[6] http://etsiit.ugr.es/alumnos/mlii/Cray.htm

[7] Lizárraga, C., 2002. Cluster de Linux. Departamento de Física. Universidad de Sonora. México. http://clusters.fisica.uson.mx/

[8] Dormido, S., Hernández, R., Ros, S. y Sánchez, J., 2003. Procesamiento Paralelo, Teoría y Programación. Madrid.

[9] http://es.wikipedia.org/wiki/Computación_distribuida

[10] http://fbio.uh.cu/bioinfo/glosario.html

[11] Arroyo, R.F., Nievas, F. J., y Pino, O., s.a. Los Clusters como Plataforma de Procesamiento Paralelo. http://usuarios.lycos.es/lacaraoculta/descargas/Clusters_definitivo.pdf

[12] Chirinov R., 2003. Proyecto Cluster openMosix (Linux). http://www.noticias3d.com/articulo.asp?idarticulo=248&pag=4

[13] Cueto, L.F., s.a. Estudio e Implementación de un Cluster Clase Beowulf.

[14] Plaza, E. J., 2003. Cluster Heterogéneo de Computadoras.

[15] Azar, A.A., Milone, D.H. y Rufiner, L.H., 2001. Desarrollo de una supercomputadora basada en un cluster de PCs. Laboratorio de Czibernética de la Facultad de Ingeniería de la Universidad Nacional de Entre Ríos. Argentina.

[16] Engler V., 2004. Novedosa red informática para tratar problemas científicos complejos. La tecnología Grid desembarca en la Argentina. Centro de Divulgación Científica - FCEyN. http://www.fcen.uba.ar/prensa/noticias/2004/noticias_06sep_2004.html
[17] http://www.catedravodafone.etsit.upm.es/formacion/seminarios/grid.html
[18] Cueto, L. F., s.a. Estudio e Implementación de un Cluster Clase Beowulf.
[19] Llorens E.P. y Peña M.A., 2002. Computación en Cluster Beowulf. http://personals.ac.upc.edu/enric/PFC/Beowulf/beowulf.html
[20] Robbins, D., s.a., OpenMosix. http://www.intel.com/cd/ids/developer/asmona/eng/20449.htm
[21] Pérez, M.A., 2001. Arquitecturas Paralelas. http://www.dragones.org/Biblioteca/Articulos/ArquitecturaParalela2.pdf
[22] Correa, M., et al., 1999. Instalación y Uso del Cluster Beowulf en CeCalCULA. http://www.cecalc.ula.ve/documentacion/tutoriales/beowulf/node1.html
[23] Catalán i Coït M., 2004. El manual para el clustering con OpenMosix. Versión 1.0. http://alumnes.eps.udl.es/~b4767512/07.openMosix/oM_como.html
[24] Turner, D., 2003. Introduction to Parrallel Computing and Clusters Computers. http://www.scl.ameslab.gov/Projects/parrallel_computing/para_into.html
[25] Tinetti, F.G., 2003. Cómputo Paralelo en Redes Locales de Computadoras. Universidad Autónoma de Barcelona. https://www-lidi.info.unlp.edu.ar/~fernando/publis/publi.html#PhD
[26] Romo, M., 2003. Procesamiento Paralelo. Universidad Internacional del Ecuador. Boletín 5. http://www.internacional.edu.ec/academica/informatica/creatividad/uide -bits/uide-bits-05-2003.pdf
[27] Palafox, R., 2000. Clusterings: La disponibilidad de información al cien por ciento. http://www.red.com.mx/scripts/redArticulo.php3?idNumero=25&articuloID=4437