Artículo **Revista de Sistemas Computacionales y TIC's**
Junio 2016 Vol.2 No.4 19-25

Construcción de clusters de computadoras de bajo costo utilizando software libre

VARGAS-MARTÍNEZ, Manuel*†, GÓMEZ-CARPIZO Santiago, SANDOVAL-SÁNCHEZ, Juan y CASTILLO-VALDEZ, Georgina.

Universidad Politécnica de Altamira, Nuevo Libramiento Altamira KM 3 Santa Amalia, Altamira, Tamps. C.P. 89602

Recibido Abril 12, 2016; Aceptado Junio 2, 2016

Resumen

En este documento se aborda la implementación de una plataforma de computación de alto rendimiento (High Performance Computing, HPC) en la Universidad Politécnica de Altamira. Este es un trabajo exploratorio para incursionar en el ámbito del cómputo paralelo, la idea surge con la necesidad de adquirir servidores dedicados para realizar tareas de procesamiento con fines de investigación. Sin embargo, el costo elevado de éstos es un problema para algunas instituciones; donde los recursos económicos son de difícil acceso. La plataforma implementada es clasificada como un Beowulf de alto rendimiento, el cual utiliza de forma completa software libre para su funcionamiento. Las pruebas experimentales muestran un alto porcentaje de reducción del tiempo de procesamiento, aproximadamente en un 90%, al emplear una de las técnicas de cómputo paralelo.

HPC, Cluster, Software libre, Rocks, MPI

Abstract

In this paper, we present the implementation of a High Performance Computing (HPC) Platform at the Polytechnic University of Altamira. This document is an exploratory research in order to break into the field of Parallel Computing, from the need to acquire dedicated servers for processing tasks with research purposes. However, the high-cost of HPC platforms is a problem for institutions where funding is hard to secure. The built platform is classified as a Beowulf of High Performance, which uses free software for operate. The experimental tests show a high reduction in the processing time, about 90% by using one of techniques of Parallel Computing.

HPC, Cluster, Free Software, Rocks, MPI

Citación: VARGAS-MARTÍNEZ, Manuel, GÓMEZ-CARPIZO Santiago, SANDOVAL-SÁNCHEZ, Juan y CASTILLO-VALDEZ, Georgina. Construcción de clusters de computadoras de bajo costo utilizando software libre**.** Revista de Sistemas Computacionales y TIC'S 2016, 2-4: 19-25

* Correspondencia al Autor (Correo Electrónico: manuel.vargas@upalt.edu.mx)
† Investigador contribuyendo como primer autor.

Introducción

Existen en el ámbito de nuestra realidad problemas difíciles de resolver computacionalmente, por lo cual áreas como la optimización inteligente hacen uso de programación heurística para resolver problemas de este tipo, sin embargo se deja de lado una parte importante: el hardware. Actualmente se manufacturan computadoras personales cada vez más rápidas, pero comparadas con muchos de los problemas computacionales existentes distan de ser una solución satisfactoria.

La programación paralela sin embargo divide un problema en partes que se resuelven simultáneamente y que permite manejar problemas difíciles. La programación paralela se ha aplicado a las siguientes áreas: Aplicaciones Científicas, Aplicaciones de Ingeniería y Aplicaciones de Procesamiento de Datos (Ver Tabla 1). Se pueden citar como ejemplos la simulación de la explosión de una supernova, la secuenciación del ADN, el análisis de datos en la industria del gas y petróleo, paralelización en FPGAs, pruebas de simulación para vehículos, etc. (N. Sun, 2010).

Aplicaciones Científicas	Aplicaciones de Ingeniería	Aplicaciones Típicas de Datos
Estudio del clima. Computación en química. Bioinformática. Computación en física. Matemáticas aplicadas.	Diseño virtual de productos. Computación de dinámica de fluidos. Análisis de elemento finito. Análisis térmico. Análisis del campo electromagnético. Análisis multifísico. Simulación de sistemas. Planeación. Diagnóstico de fallos en sectores industriales como: aeroespacial, automóviles, aviación, electrónica, ingeniería civil, construcción de barcos etc.	Física experimental de altas energías. Observación astronómica. Recuperación de información y minería de datos. Censado remoto. Centros de datos empresariales. Respaldo de sistemas de negocio. Recuperación de desastre de datos. Inteligencia de negocios. Educación online.

Tabla 1 Aplicaciones de la programación paralela.

Para mejorar las soluciones de problemas difíciles en el ámbito científico es conveniente tener una plataforma de computación de alto rendimiento (HPC Platform, High-Performance Computing Platform) y sobre toda una que no sea tan costosa.

La optimización inteligente es un área que busca encontrar mediante el uso de algoritmos heurísticos las mejores soluciones a problemas difíciles de crecimiento no lineal. Su aplicación en la actualidad está muy extendida tanto en la investigación como en la industria. Sin embargo estos algoritmos son usados en máquinas comunes o en supercomputadoras que pocos tienen la posibilidad de comprar.

Dados los altos costos de una HPC la programación paralela ha estado restringida a las grandes universidades, a centros de investigación muy especializados públicos y privados, pero los volúmenes de datos que genera internet se elevan cada día y hacen necesario que se cuente con personal capacitado en el manejo de este tipo de plataformas.

Es en este tipo de casos donde el software libre se erige como opción para la construcción de HPCs o clusters de computadoras de bajo costo y usando hardware común u obsoleto, material que es posible conseguir sin erogar grandes cantidades de dinero, sobre todo si se pertenece a una universidad de nueva creación y en países en vías de desarrollo.

ISSN-2444-5002

Antecedentes

A continuación de presentan conceptos relacionados con la implementación de la plataforma de computación de alto rendimiento.

High Performance Computing (HPC)

La Computación de Alto Rendimiento (HPC, High-Performance Computing) ha sido recientemente requerida por aplicaciones cómputo intensivo, la mayoría en campo científico e industrial. Los sistemas paralelos modernos tales como clúster y Grid Computing consisten en diferentes nodos de computadoras que ayudan en la velocidad de cómputo de tales aplicaciones (P. Sajjipanon, 2008). En una plataforma HPC o Clúster se envía una tarea, la cual se desglosa en varios procesos ligados, que comparten estructura y comunicaciones, esto permite abordar problemas que ninguna de las computadoras que integran el clúster podría resolver por separado (Martinez, 2009).

La solución a este problema es el aprovechamiento de equipo de bajo costo o equipo con el que ya se cuenta para crear una plataforma HPC que proporcionará un procesamiento similar o superior al de una costosa supercomputadora.

Cluster Beowulf

La saga de Beowul narra la historia de la exitosa lucha de un héroe sobre un gran y temible enemigo (Heany, 2001). Beowulf tenía el poder de 30 hombres en cada una de sus manos. El poder de muchos en un solo hombre, con esa premisa, el héroe puede vencer al demonio Grendel con sus propias manos como única arma. Un clúster Beowulf consiste en una red de computadoras estrechamente conectadas dedicadas a la solución de un solo problema (M. Meredith, 2003). Un sistema Beowulf usualmente consiste de un nodo servidor (maestro) y uno o más nodos clientes (esclavos), interconectados a través de una red Ethernet u otro tipo de red.

La arquitectura de tipo Beowulf es quizás una de las más atractivas porque utiliza equipo de cómputo tradicional u obsoleto para construir una plataforma HPC. Hace más de una década el poder de la programación paralela estaba económicamente prohibido, solo los investigadores y universidades con más recursos accedían a ella. La llegada de los clústeres tipo Beowulf colocaron el poder del paralelismo en las manos del resto de los investigadores.

Esta transición solo pudo ser lograda mediante el uso y aplicación del "Software Libre", este tipo de software se asegura que nadie se apropie del mismo y en contra partida el software pertenece a la comunidad, la cual tiene el derecho de compartirlo, modificarlo y compartir dichas modificaciones, entre otras cosas. De esta forma surgieron soluciones para muchas áreas de investigación y la programación paralela no fue la excepción.

Rocks Cluster Distribution

Rocks es una distribución de software libre que centra sus esfuerzos y filosofía en crear un sistema escalable, heterogéneo, sencillo de usar, y que trata de evitar la dependencia de un administrador. Rocks es un cluster distribuido en un conjunto de discos, de instalación simple, está basado Centos 6.6 (Versión libre de Red Hat Enterprise). Este tiene una gran variedad de paquetes llamados "rolls" entre los cuales se cuenta: PBS, SGE, Ganglia, MPI, GT4 y en forma adicional Java, Condor, Compiladores Intel, etc. (Regents, 2008). Rocks fue desarrollado en el San Diego Supercomputer Center (Martinez, 2009).

Message Passing Interface (MPI)

La construcción de clústeres provee un método de bajo costo, efectivo y fácil para construir supercomputadoras (D. C. Bergen, 2002).

ISSN-2444-5002

El Message Passing Interface (MPI) es uno de los modelos de programación paralela más portable de HPC, con implementaciones de plataforma optimizada y es la más utilizada con nuevos sistemas HPC (J. A. Zoumevo, 2013). La mayor dificultad en programación paralela es subdividir problemas en diferentes partes que puedan ser ejecutas simultáneamente en diferentes máquinas y que inherentemente llevan un coste residual (secuencial) como se menciona en la ley de Amdahl. MPI es una librería de rutinas que provee la funcionalidad para permitir que estas partes se comuniquen (Sloan, 2004).

Problema

En la Universidad Politécnica de Altamira existe la necesidad de contar con una plataforma de computación de alto rendimiento de bajo costo, pero con prestaciones altas y bajo condiciones similares a los grandes clusters de computadoras para la investigación y enseñanza de la programación paralela.

Metodología

En esta sección se muestra la metodología llevada a cabo para la construcción del clúster Prometeo de arquitectura Beowulf llevando a cabo los siguientes pasos:

- Elección del software necesario para construir el clúster
- Elección del hardware disponible
- Instalación y configuración

Elección del software necesario para construir el clúster

Se realizó una búsqueda de software utilizado para la construcción de clústeres basados principalmente en su bajo costo, por lo cual se optó por utilizar software libre. Las opciones encontradas se muestran en la Tabla 2:

Software	Ventajas	Desventajas
Rocks (Basado en Centos)	Instalación y configuración entendible Documentación disponible Actualizaciones Control del desempeño	Se necesita reprogramar las aplicaciones.
Openmosix	Instalación y configuración entendible Documentación disponible No se necesita reprogramar las aplicaciones.	Proyecto detenido, sólo disponible para kernel 2.4
Conga (Luci y Ricci)	Se pueden instalar sus componentes en forma separada de Conga	Poca documentación

Tabla 2 Tabla comparativa de software revisado

Debido a la documentación disponible en la red, accesibilidad del software, robustez de su sistema base, actualizaciones y una interfaz amigable para su instalación se eligió la Distribución Rocks 6.1.1 basado en Centos.

Elección del hardware disponible

La elección del hardware se basó en la disponibilidad del equipo con el que se contaba en el laboratorio de optimización y redes, básicamente hardware de red, equipo usado para la demostración de la arquitectura de computadoras y de mantenimiento preventivo, el hardware elegido se muestra en la Tabla 3.

Software	Cantidad	Características
Computadora tipo Torre	5	Core i7, 8 núcleos 500GB de disco duro, 8Gb de RAM, tarjeta Ethernet integrada.
Switch	1	10/100 mbps de 24 puertos
Tarjeta Ethernet	1	10/100 mbps
Gabinete	1	42U, con ventilación

Tabla 3 Hardware seleccionado

Instalación y configuración

Uno de los equipos seleccionados anteriormente debe de contar con una tarjeta adicional Ethernet para conexión externa a la red local del clúster (usada para la comunicación remota del clúster). Este equipo es el nodo maestro en el cual se instaló el FrontEnd de Rocks. Una vez instalado el FrontEnd en el servidor maestro, fue necesario que éste se conectara a la red local del clúster para poder agregar los nodos, cada nodo se conecta al switch y posteriormente el servidor debe detectarlo de manera automática e instalar el BackEnd de Rocks, este proceso se realizó por cada uno de los nodos, lo cual muestra la facilidad en la expansión del clúster (Véase Figura 1). La arquitectura del clúster tipo Beowulf se muestra en la Figura 2.

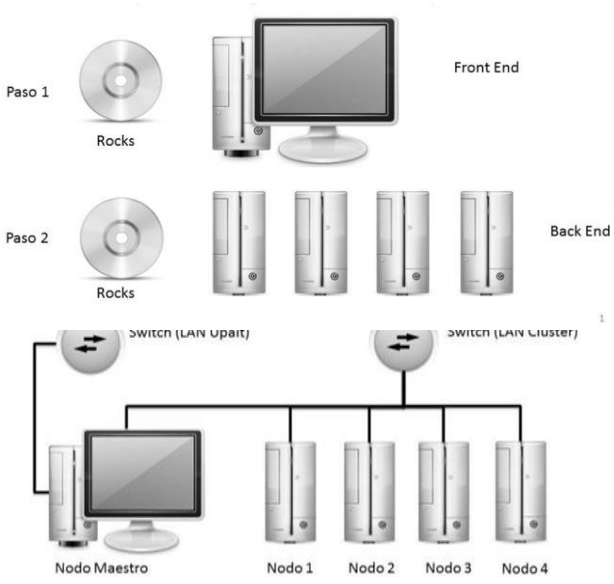

Figura 2 Arquitectura tipo Beowulf del clúster Prometeo del Laboratorio de Investigacion ITI de la Universidad Politécnica de Altamira.

Para asegurarse que se haya agregado el equipo se hace uso del proceso Ganglia en el servidor maestro por medio de un explorador web, Ganglia permite observar el número de núcleos con los que se cuenta en el clúster, además de la cantidad de memoria y el rendimiento del clúster en general o de manera individual (cada nodo), una imagen del reporte se observa en la Figura 3.

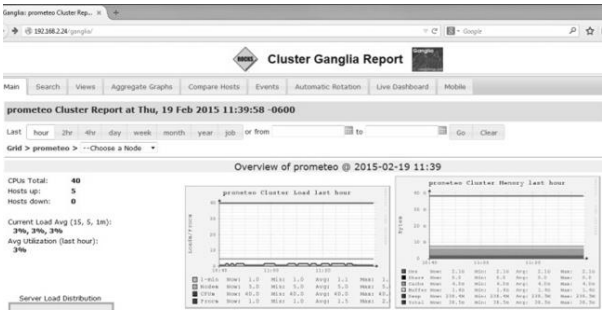

Figura 3 Reporte Ganglia.

En este reporte se puede apreciar el número de nodos que tiene el clúster (incluye al nodo maestro) y el número de núcleos disponibles, en este caso 5 nodos y 40 núcleos.

Pruebas y Resultados

Básicamente se realizaron dos pruebas con el clúster una vez que fue posible utilizarlo. La primera es una prueba de rendimiento llamada HPLinpack y la segunda la ejecución de un programa de suma de matrices usando MPI para observar el desempeño de un nodo contra cuatro nodos en paralelo.

Prueba del Clúster mediante HPLinpack

Se aplicó HPLinpack al clúster para observar su funcionamiento. Linpack realizó 18 pruebas exitosas con duración de 91 segundos en promedio, en dichas pruebas el clúster registró un flujo de 7.2 GFlops (Operaciones de punto flotante por segundo) en promedio. La Figura 4 muestra la última prueba y un resumen.

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00R2R4       10000  1280     2     2              91.62              7.278e+00
HPL_pdgesv() start time Mon Jul 20 11:29:53 2015

HPL_pdgesv() end time   Mon Jul 20 11:31:25 2015

--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0048696 ...... PASSED
================================================================================

Finished     18 tests with the following results:
             18 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```

Figura 4 Reporte de la prueba de rendimiento de HPLinpack.

Suma de matrices

Se tomó además uno de los ejemplos clásicos de suma de matrices para MPI. Con tamaños de 1,400,000,000 y 2,000,000,000 de elementos, se utilizaron 4 hilos de ejecución usando 1 y 4 nodos físicos. Los resultados pueden observarse en la Tabla 3.

Tamaño	#Hilos	#Máquinas	Tiempo en segundos	Ahorro %
1,400,000,000	4	1	712.08416	0
1,400,000,000	4	4	79	88.9058
2,000,000,000	4	1	1411.19641	0
2,000,000,000	4	4	133.306971	90.5536

Tabla 3 Resultados de la suma de matrices

Cabe destacar que para la prueba de 1,400,000,000 de elementos el usar cuatro nodos implica un ahorro del 88.9% con respecto a la misma prueba usando sólo un nodo físico. De igual forma la prueba de 2,000,000,000 de elementos al usar cuatro nodos representó un ahorro del 90.5% con respecto a la misma prueba con un sólo nodo físico. Mediante el programa de reporte Ganglia (Figura 5) se puede observar la carga de trabajo de los nodos al momento de hacer las pruebas usando el algoritmo paralelizado de la suma de matrices.

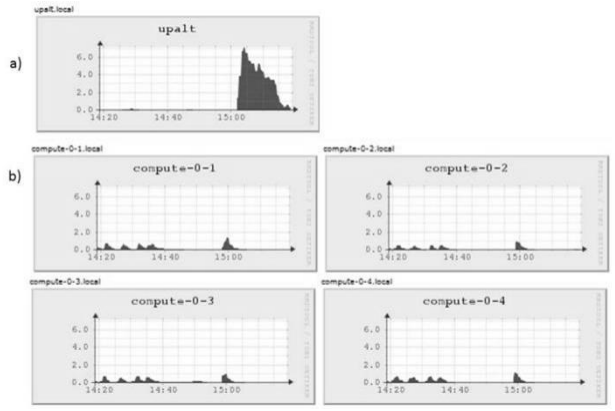

Figura 5 Reporte Ganglia: a) Carga de un sólo nodo b) Carga de cuatro nodos para la misma tarea.

Conclusiones

Al término de este trabajo, con base en los resultados y experiencia adquirida en este corto tiempo se llega a las siguientes conclusiones:

-La curva de aprendizaje apenas comienza para el cuerpo académico. La premisa es dominar el lenguaje de paso de mensajes MPI sobre el lenguaje de programación C/C++.

-Se ha construido un HPC de bajo costo. El costo real fue el tiempo invertido en su construcción pues se realizó con equipo con el que ya se contaba de otros proyectos.

-Se provee una plataforma para el desarrollo de cómputo paralelo en la universidad.

-Es importante explotar el área de cómputo paralelo tanto en las aulas como en el laboratorio y contar con recursos humanos capacitados en el tema.

Trabajos futuros

Como trabajos a futuro se pretenden realizar los siguientes puntos:

-Reprogramar los algoritmos heurísticos con MPI.

-

-Evaluar el aumento de nodos o piezas de hardware para mejorar el rendimiento.

Referencias

D. C. Bergen, B. P. (2002). Building an MPI Cluster. *Crossroads*.

Heany, S. (2001). *Beowulf*. New York: W. W. Norton & Company.

J. A. Zoumevo, D. K. (2013). Using MPI in High-Performance Computing Services. *EuroMPI'13*, 43-48.

M. Hafeez, S. A. (2011). Survey of MPI Implementatios. *DICTAP 2011*, 2016-220.
M. Meredith, T. C. (2003). Exploring Beowulf Clusters. *Journal Computing*, 268-284.

Martinez, I. F. (2009). *Creacion y Validación de un Cluster de Computación Cientiffica Basado en Rock.* Madrid: Universidad Carlos III de Madrid.

N. Sun, D. K. (2010). High-performance in China: Research and Applications. *International Journal High Performance Computing Applications*, 363-409.

P. Sajjipanon, S. N. (2008). Web Services for MPI-based Parallel Applications on a Rocks Cluster. *IEEE Computer Society: Asian-Pacific Services Computing Services*, 265-270.

R. Sposito, P. M. (2003). OpenMosix aproach to build scalable HPC farms with an easy management infrastructure. *CHEP 2003*, 24-28.

Regents, U. (2008). Recuperado el 1 de 9 de 2016, de http://www.rocksclusters.org/presentations/tutorial/tutorial-1.pdf

Sloan, J. D. (2004). *High Performance Clusters with Oscar, Rocks, OpenMosix, and MPI.* O'Reilly Media.