

HPC

High Performance Computing



Alfonso Ros Dos Santos

3 de diciembre de 2010

Índice general

Introducción	2
HPC: High Performance Computing	3
Supercomputadoras y Clusters	4
Arquitectura de un Cluster	8
Cluster Beowulf en Debian	13
Utilizando MPICH	18

Introducción



La computación de alto rendimiento (HPC, según sus siglas en inglés), que es a veces llamada computación de alta productividad, ha sido usada por décadas por investigadores académicos y del gobierno como instrumento de apoyo para los problemas de ingeniería más pesados. Estos sistemas han sido tradicionalmente propietarios, fuertemente integrados y costosos. Como lo detallan la serie de estudios por el departamento de consejo a comerciantes en competitividad de los Estados Unidos, HPC tiene un tremendo potencial de ayudar empresas comerciales a diseñar mejores productos a bajos precios para ganar ventajas competitivas. Sin embargo, como estos sistemas cuentan típicamente millones de dolares, raramente pocas empresas han sido capaces de producir sus propios centros de recursos compartidos HPC, dejando HPC fuera del alcance para muchos miembros de la comunidad industrial.

En los últimos 5 años, la comunidad HPC ha sufrido un rápido y dramático cambio implementaciones de HPC de menor costo usando múltiples elementos de computación (nodos) que operan en una tarea en común en paralelo: referido formalmente como “*cluster*”. *Clusters* son computadoras individuales en red en las que cada una corre una instancia local del mismo sistema operativo y son dirigidas juntas. El costo salvado por utilizar un gran número de computadoras basadas en micropocesadores permite tratar problemas que se consideraban intratables o no prácticos en un simple sistema más grande. De hecho, los *clusters* basados en micropocesadores son ahora la arquitectura HPC dominante, subiendo desde 6.6 % hasta 87.4 % en el “*Top 500*” desde 11/2000 hasta 11/2008.

Cuando se considera el cambio de *mainframes* y minicomputadoras a computadoras personales y el impacto que esta transición tuvo en la forma que el trabajo se realizaba, se cree que el cambio a *clusters* económicos y de alto rendimiento, presenta una oportunidad enorme de tomar una HPC de recursos limitados y centralizados a una que es abiertamente disponible. Los desafíos que enfrenta la comunidad HPC son similares a aquellos de la industria de la computadora personal. En este texto hablaremos sobre estos desafíos y el posible acercamiento a HPC con el manejo de pocos recursos.

HPC: High Performance Computing

El termino HPC es mayormente asociado con la computación usada para investigación científica o ciencia computacional. Un termino relacionado, *high performance technical computing* (HPTC), se refiere generalmente a las aplicaciones ingenieriles de sistemas basados en *cluster* para problemas como física cuántica, predicción climática, investigación del clima, modelamiento molecular, simulaciones físicas de aviones en tuneles de viento, simulación de detonación de armas nucleares e investigación de la fusión nuclear.

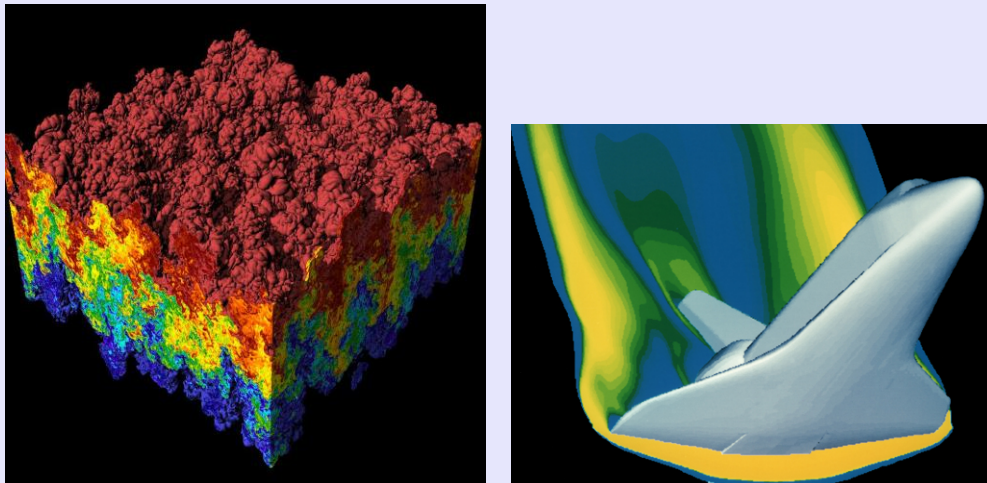


Figura 1: Simulación de fluidos

Recientemente, HPC se ha estado aplicando a los negocios con el uso de *clusters* de supercomputadoras, como *data warehouses*, aplicaciones *line-of-business* (LOB) y procesamiento de transacciones.

Supercomputadoras y Clusters

Una Supercomputadora es una computadora con capacidades de cálculo muy superiores a las comúnmente disponibles. Las supercomputadoras están caracterizadas por una inmensa capacidad de realizar operaciones computacionales a velocidades no comparables a computadores de naturaleza personales. Estas capacidades de cálculo o procesamiento intensivo están justificadas en un arreglo de hardware extremadamente caro especializado en operaciones paralelas, la jerarquía de memoria tiende a ser cuidadosamente diseñada para que el CPU este constantemente trabajando. En los PC regulares la inmensa mayoría del tiempo las operaciones que ocurren son de entrada / salida y no de uso del procesador. Estas computadoras tienden a ser especializadas para ciertos tipos de operaciones, usualmente cálculos numéricos y tienden a tener rendimientos inferiores en otras áreas de operación. Los sistemas operativos de este tipo de computadora tienden a ser un énfasis en el uso más apropiado de los recursos de hardware y en las tareas administrativas que en proveer una interfase gráfica de alta calidad.



Hay disciplinas científicas que son impensables sin la herramienta que significa una supercomputadora tales como los estudios sobre cambios climáticos, biología computacional, y simulaciones que envuelven gran número de variables.

Hoy, las supercomputadoras son típicamente del tipo producido por las compañías “tradicionales” como Cray, IBM y Hewlett-Packard, que han comprado muchas de las compañías de los ochenta para ganar su experiencia.

Las supercomputadoras son extremadamente caras, como consecuencia de esto en muchos ambientes científicos se usan otras alternativas. El uso de clusters de computadoras es una opción viable y relativamente simple para enfrentar tareas que requieren volúmenes intensivos de operaciones computacionales y no se dispone de acceso a una supercomputadora. Simplemente, un cluster es un grupo de múltiples computadoras unidas mediante una red de alta velocidad, de tal forma que este grupo computacional es visto como un único ordenador, más potente que las computadoras personales por separado. El término cluster se aplica a los conjuntos o grupos de computadoras construidos mediante la utilización de componentes de hardware comunes y que se conducen como si fuesen una única computadora. El uso creciente de clusters surge como resultado de la aparición de varias tendencias actuales que incluyen la disponibilidad de computadoras personales de alto rendimiento a precios muy económicos, el

advenimiento de redes de computadoras con una alta velocidad de transferencia de datos, el desarrollo de software para distribución de calculo de rendimiento intensivo , sistemas operativos altamente efectivos y baratos, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran. Los clusters son usados mas para propósitos computacionales que para operaciones entrada / salida. Un uso común de clusters es para “balancear carga” en sitios web. Una página web es pedida a un servidor administrador que decide cual de los servidores idénticos entrega la página a la computadora cliente. Este uso de cluster es muy común y es llamado granja de servidores. Esta configuración permite estabilidad y añade que el tráfico sea manejado más eficientemente.



La mayoría de las supercomputadoras modernas son ahora clusters de computadoras altamente optimizados. A medida que la popularidad de los clusters va en aumento, el término “supercomputadora” esta siendo usado también para este tipo de sistema.

Es importante distinguir la diferencia entre los conceptos **computación competente** y **computación capacitada**.

- **computación competente:** Se refiere al uso del máximo poder de computación para resolver un problema grande en el menor período de tiempo. También se refiere a la competencia del sistema para resolver un problema de un tamaño o complejidad que ningún otro puede.
- **computación capacitada:** Se refiere a usar poder de coputación de bajo costo en relación a la efectividad para resolver problemas relativamente grandes o muchos problemas pequeños.

Midiendo la velocidad de las supercomputadoras

En general, la velocidad de una supercomputadora es medida en “FLOPS” (FLoating Point Operations Per Second), especialmente en campos de científicos que hacen fuerte uso de cálculos de punto flotante, similar al antiguo, simple, instrucciones por segundo.

Para comparar, una calculadora de mano debe realizar relativamente pocos FLOPS. Cada cálculo, como la adición o substracción de dos números, requiere

una simple operación, entonces es muy rara la necesidad de que el tiempo de respuesta sea menor que lo que pueda hacer la persona que utiliza la calculadora físicamente. Un tiempo de respuesta menor a 0.1 segundos en el contexto de un cálculo es percibido como instantáneo por un humano, entonces una simple calculadora necesita solamente 10 FLOPS para ser considerada funcional.

Registros y marcas

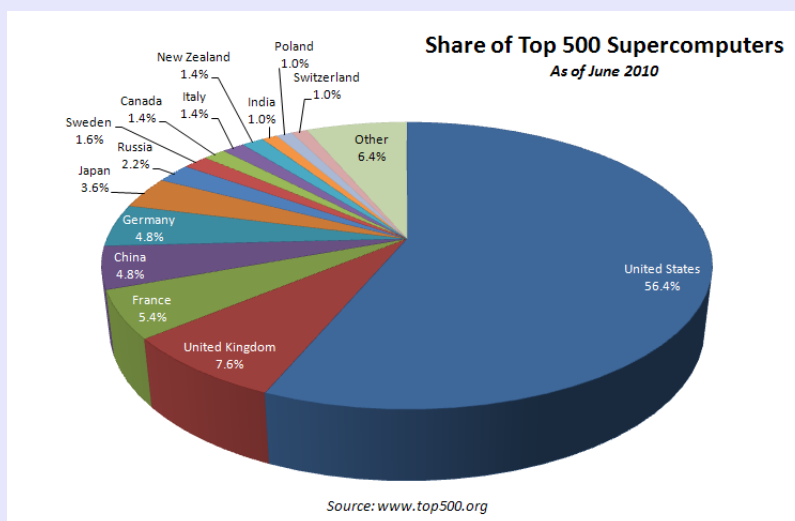
- **Junio 2006:** El centro de investigación japonés RIKEN produce la supercomputadora MDGRAPE-3 capaz de ejecutar un petaFLOPS. Diseñada para simular dinámica molecular.
- **2007:** Intel revela el chip experimental multi-core POLARIS. Capaz de producir 1 teraFLOPS a 3.13 GHz.
- **junio 26, 2007:** IBM anuncia la segunda generación de dubbed Blue Gene/P, que pueden llegar a velocidades de 3 petaFLOPS.
- **junio 2007:** La computadora más rápida del mundo es la IBM Blue Gene/L. Llegando a los 596 teraFLOPS.
- **Octubre 25, 2007:** La corporación japonesa NEC anuncia el modelo SX-9 como la supercomputadora más rápida del mundo. Esta computadora es capaz de realizar 102.4 gigaFLOPS por núcleo.
- **Febrero 4, 2008:** La supercomputadora Sun llamada Ranger, la más poderosa en el mundo para investigación científica abierta, opera a velocidades de hasta medio petaFLOP.
- **Mayo 25, 2008:** Una supercomputadora militar construida por IBM llamada “*Roadrunner*”, alcanza el petaFLOPS por procesar más de 1,026 cuatrillones de cálculos por segundo.
- **Junio 2008:** AMD lanza la serie ATI Radeon HD4800, que son las primeras GPUs en alcanzar un teraFLOP.
- **November 2008:** Una mejora a la supercomputadora Cray XT Jaguar llevó al sistema a un poder de 1.64 petaFLOPS o un cuatrillon de operaciones matemáticas por segundo, haciéndola la computadora más poderosa dedicada a la investigación.
- **2009:** Cray Jaguar logró 1.75 petaFLOPS ganándose a Roadrunner en el Top 500.
- **2010:** China construye el Tianhe-I. La supercomputadora más rápida del mundo hasta ahora, con capacidad de hasta 2.5 petaFLOPS.

También en el 2010, Intel lanzó el procesador de PC más rápido de seis núcleos con una velocidad teórica de 107.55 gigaFLOPS en cálculos de doble precisión, el Core i7 980 XE. Las GPUs son considerablemente más poderosas. Por ejemplo, NVIDIA Tesla C2050 GPU llega hasta los 515 gigaFLOPS en cálculos de doble precisión. En simple precisión, NVIDIA Tesla C2050 llega a 1.03 teraFLOPS.

Top 500

Top 500 es un proyecto para clasificar y detallar las 500 computadoras más poderosas no distribuidas en el mundo. El proyecto utiliza el benchmark de LINPACK escrito en Fortran para computadoras de memoria distribuida, para medir en desempeño.

Actualmente la distribución de las 500 computadoras más poderosas entre las naciones se encuentra dada por el siguiente gráfico.



Las mejores 10 computadoras actualmente son las siguientes

Posición	petaFLOPS	Nombre	País
1	2.566	Tianhe-1A	china
2	1.759	Jaguar	USA
3	1.271	Nebulae	china
4	1.192	TSUBAME 2.0	Japon
5	1.054	Hopper	USA
6	1.050	Tera 100	Francia
7	1.042	Roadrunner	USA
8	0.831	Kraken	USA
9	0.825	JUGENE	Alemania
10	0.816	cielo	USA

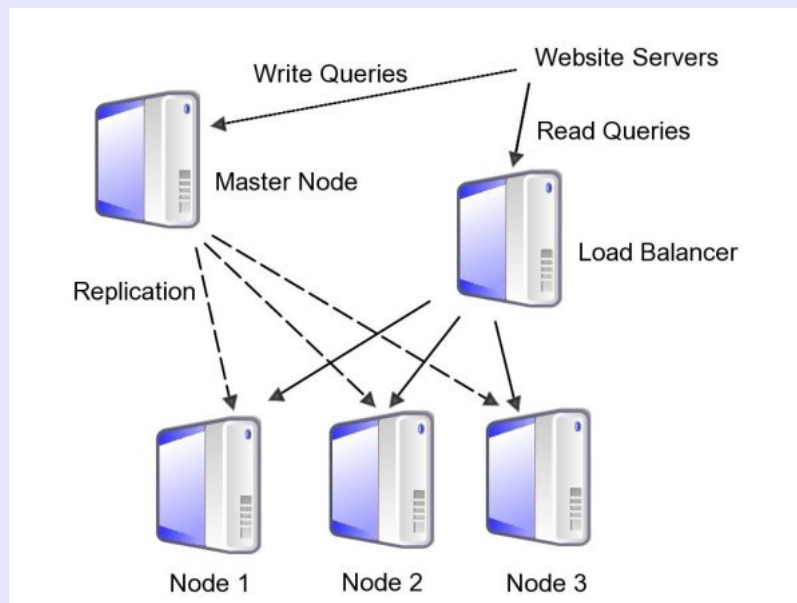
Arquitectura de un Cluster

Los clusters siguen lineamientos generales para diferentes aplicaciones. Cada aspecto de diseño debe ser tomado en cuenta cuando se toman decisiones sobre la arquitectura del cluster.

- **Consultas en Bases de Datos:** Cluster de consultas a una base de datos para un sitio web.
- **Procesamiento por lotes:** Cuando muchas tareas necesitan ejecutarse a la vez.
- **Granja de renderizado:** Configuración de renderizado distribuido en un cluster.
- **Desarrollo de software:** Granja de compilación - clusters para desarrollo de software.
- **Message Passing Architectures:** Esencialmente supercomputadoras para alto rendimiento en aplicaciones.

Clusters para consultas en bases de datos

La consulta a bases de datos es una necesaria y util aplicación de clusters. Muchas bases de datos son leídas intensivamente con muchas más solicitudes de lectura que de escritura. Con consultas sobre una comunidad de nodos es posible escalar el número de lecturas que pueden ser satisfechas por segundo en una forma lineal.



Un sitio web tiene grandes cantidades de contenido guardados en una base de datos. El servidor web (Que tambien puede estar probablemente en un cluster) , hace consultas tipo lectura sobre los nodos de consulta a través de un distribuidor de carga. Las solicitudes de escritura sobre la base de datos son enviadas al nodo maestro.

Hay muchas configuraciones comunes para el nodo maestro. In situaciones con gran cantidad de escritura es necesario ser creativos con la arquitectura de la base de datos para permitir consultas entre nodos maestros o particionar la base de datos de tal forma que haya esencialmente dos bases de datos separadas para diferentes consultas. Por ejemplo una base de datos para consulta y otra para datos del usuario.

Procesamiento por lotes

El procesamiento por lotes es la clave para la industria bancaria. Buena programación y respuestas rápidas son importantes si por ejemplo no vamos a mantenernos esperando por el dinero en un cajero mientras nuestro banco verifica que tenemos el dinero que pedimos en nuestra cuenta.

A veces referido como una granja de computadoras, la parte clave de los sistemas de procesamiento por lotes es maximisar el tiempo y rendimiento de mantenimiento al tomar cargar mientras se minimiza los niveles de costo.

En esta clase de situaciones, puede ser sabio salvar dinero al reducir la capacidad mientras hay poca demanda al apagar algunos nodos y volviendolos a encender cuando la demanda crezca de nuevo. Para maximizar la efectividad,

un sistema de manejo de carga inteligente (WMS) debe ser implementado.

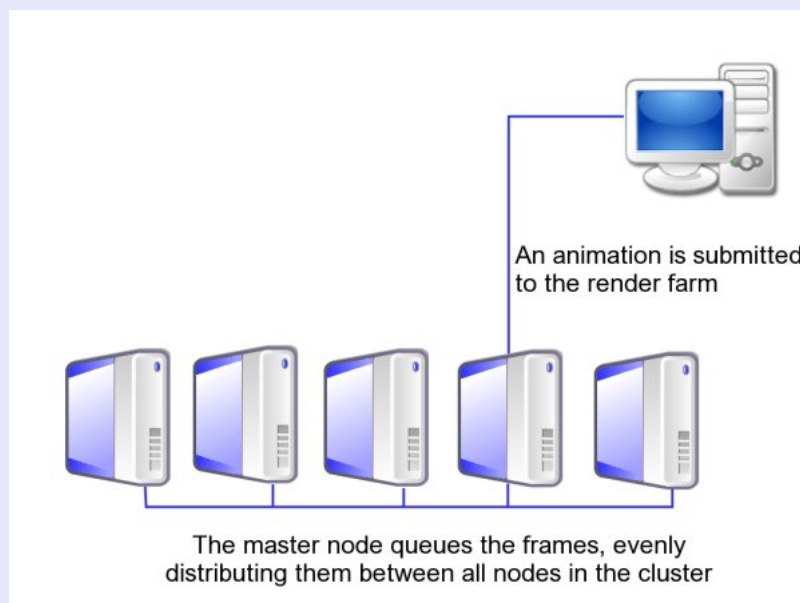
Granjas de renderizado

Las granjas de renderizado son un caso especial del procesamiento por lotes, donde se hace menos énfasis en el tiempo de respuesta. La mayoría del procesamiento tomará mas de un minuto. Bajo costo de hardware y calidad del poder de procesamiento disponible es más importante. El renderizado es usado en los efectos visuales, modelaje por computadora y en la industria de la computación gráfica y se refiere al proceso de crear una imagen de lo que es esencialmente fórmulas matemáticas. Los motores de renderizado proveen numerosas y diferentes opciones, que combinadas pueden producir una escena con los efectos deseados.

Donde se consigue el renderizado.

- Diseño asistido por computadora (CAD), en ingeniería y diseño.
- Efectos visuales (VFX), para películas, televisión y publicidad.
- Renderizado arquitectónico, para visualizar edificios, cuartos y espacios abiertos.

Uno de los más intensivos aspectos de este proceso esta en producir escenas foto-realísticas y animaciones. Es aquí donde entran los clusters.



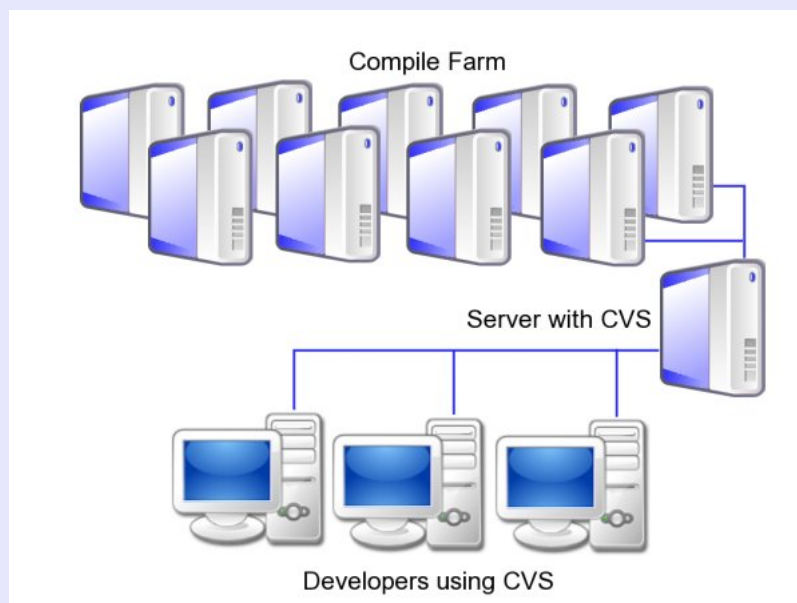
Alternativamente, un cuadro puede ser enviado y distribuido entre todos los nodos. Esto puede hacer uso de MPI o picar el cuadro en trozos más pequeños y

pasar cada uno a un nodo diferente.

Granjas de compilación

Una granja de compilación toma un acercamiento muy similar a las granjas de renderizado. La diferencia es en como el código siendo desarrollado es manejado.

Las granjas de compilación proporcionan la oportunidad de incrementar la velocidad de la compilación de un programa completo. También, los archivos individuales en los que trabaja un desarrollador pueden tomar cuestión de segundos en ser compilados cuando el programa entero es reunido puede tomar una hora o más. Adicionalmente es importante proporcionar los medios para desarrollar aplicaciones internas, donde la compilación en el host puede ser muy lenta.



Supercomputadoras MPI

La más impresionante aplicación masivamente paralela hace uso de todos los CPU a través de los nodos de un sistema gigante, probando significativas cantidades de poder de procesamiento para simulaciones intensivas y aplicaciones de modelaje.

¿Dónde se utiliza MPI?

MPI viene del inglés Message Passing Interface y existen numerosas implementaciones, todas con sus propias ventajas particulares.

La arquitectura de un cluster MPI depende de aplicaciones específicas y muchos clusters de supercomputadoras son diseñados específicamente con un par de aplicaciones en mente. Sin embargo, hay algunos puntos generales para anotar.

Una característica clave de sistemas MPI es la baja congestión de la red para la intercomunicación entre nodos. Por lo tanto, la infraestructura de la red es importante para determinar el eventual desempeño del sistema. Adicionalmente la aplicación debe ser diseñada para tomar ventaja del sistema y debe también tomar ventaja de la arquitectura de los procesadores en uso.



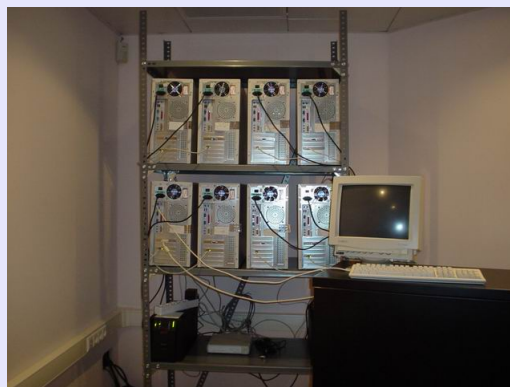
Cluster Beowulf en Debian



Cluster Beowulf

En 1994 la NASA construyó un cluster bajo Linux con hardware barato, con 16 procesadores 486 conectados mediante una red local Ethernet, con el objetivo de conseguir alto rendimiento. Para ello se utilizó computación paralela por lo que los programas (escritos en C y Fortran) estaban paralelizados, es decir, utilizaban librerías de Message Passing (PVM y MPI) para que los procesos se ejecutasen en múltiples procesadores siguiendo el paradigma master/esclavo. El proyecto se llamó Beowulf y fue un gran éxito, por lo que actualmente cualquier sistema similar se denomina cluster tipo Beowulf.

¿Cómo empezar?



Primero, hay que considerar el propósito del cluster. ¿Qué tipo de aplicacio-

nes se van a correr en el? Aquí hay algunas aplicaciones típicas para clusters beowulf:

- Fenómenos de transporte, dinámica de fluidos, calor y transferencia de masa, flujos multifase, etc.
- Dinámicas de moléculas de múltiples millones de átomos y formación de proteínas.
- Automata celular para modelar fenómenos epidemiológicos para toma de decisiones.
- Renderizado distribuido y raytracing.
- Problemas NP completo como alineación de secuencias de ADN.

Note que algunas de las aplicaciones no escalan a un gran número de procesadores como otras. También, algunos problemas se benefician de tener memoria compartida. Para estas aplicaciones, la escalabilidad lineal en procesadores requiere una gran máquina de memoria compartida como la línea Altix de SGI, que es la arquitectura más grande de memoria compartida con Linux nativo en el mundo Intel. IBM ofrece máquinas similares en su arquitectura PowerPC también como SUN con Sparc.

Herramientas para paralelizar

La computación paralela puede hacerse de varias maneras. Todo se centra en el cual se adapta mejor a la aplicación que vamos a paralelizar. Desde agosto del 2010, hay cuatro mejores formas de crear paralelización.

- Symetric Multiprocessing (SMP) y múltiples hilos (threads):

En computación, SMP involucra una computadora con múltiples procesadores donde dos o más procesadores idénticos son conectados a un área de memoria compartida y son controlados por una sola instancia del sistema operativo. Los sistemas más comunes de múltiples procesadores hoy en día usan la arquitectura SMP. En el caso de procesadores de múltiples núcleos, la arquitectura SMP aplica a los núcleos, hilos como procesadores separados. Los procesadores deben ser interconectados usando buses, crossbar switches o redes de chips.

La paralelización viene de escribir las aplicaciones utilizando hilos y procesos, que pueden ser ejecutar múltiples tareas y concurrentes.

- Message Passing Interface (MPI):

MPI es un protocolo independiente del lenguaje usado para programar computadoras en paralelo. Ambas, punto a punto y comunicaciones colectivas son soportadas. MPI es una interfaz de programación en aplicaciones de paso de mensajes, junto con protocolo y especificación de semánticas para la forma en que sus componentes deben comportarse en cualquier implementación. El objetivo de MPI es el alto desempeño, escalabilidad y portabilidad.

En pocas palabras, las aplicaciones usan librerías MPI, que a su vez pasan comandos y datos a través de diferentes nodos en un cluster y ejecutan las operaciones desde un mismo filesystem, común a través de todos los nodos en el cluster.

- Parallel Virtual Machine (PVM):

PVM es un software que permite a una colección de computadoras diferentes a ser usadas como un recurso computacional coherente y flexible, o una máquina virtual paralela.

Las computadoras individuales pueden ser de memoria compartida o memoria local con múltiples procesadores, supercomputadores con vectores de procesadores, motores gráficos especializados, o estaciones de trabajo personales escalables, que deben estar interconectadas por una variedad de redes, como Ethernet o FDDI.

PVM es una alternativa a MPI.

- Paralelización de trabajos:

Para la paralelización de trabajos se utilizan sistemas de cola de trabajo, como OpenPBS, LoadLeveler, el motor Grid de Sun/Oracle o la plataforma LSF. En los sistemas de múltiples núcleos, sistemas de múltiples hilos, encolar hilos de programas para ejecutar las tareas, puede mejorar el tiempo necesario para completar un trabajo en una forma logarítmica inversa.

Symmetric Multiprocessing (SMP)

Hay dos formas de hacer SMP. En ambas, el sistema operativo ve todo el hardware como una única imagen del sistema.

Una de las formas es con hardware SMP. Ejemplos son los sistemas con múltiples núcleos, como la arquitectura Sparc o el Intel i7, múltiples sistemas

CPU, como el Origin2 SG1, o múltiples CPU, sistemas de múltiples núcleos, como el IBM Power6.

La otra forma, es por emulación de software. Tomando ventaja de redes 100mbit/1Gbit/10Gbit, software SMP automáticamente direcciona todas las comunicaciones internas de los procesos localmente si los procesos estan en la misma máquina, o a través de la red si no lo estan. También permite balanceo de carga dinámica a través de la migración de procesos de una máquina a otra.

Los mejores ejemplos de software SMP son Mosix y Kerrighed.

Mosix es un parche para el kernel que permite una serie de utilidades que permiten que el cluster de varias maquinas se vea como una sola máquina de múltiples procesadores con memoria compartida. Desafortunadamente, Mosix es un poco difícil de configurar que los demás, porque se debe aplicar un parche especial para las fuentes del kernel y después compilarlo. La licencia de Mosix cambió en 2002 y se empezó un proyecto en paralelo llamado OpenMosix.

Hasta Debian Sarge, había un parche para el kernel OpenMosix que se podía instalar. El paquete fue removido a comienzos del 2006 y el proyecto OpenMosix fue descontinuado el primero de marzo del 2008. Hasta agosto del 2010, el código fuente esta aún disponible en SourceForge. Combinado con el kernel de Debian, se puede parchar el kernel relativamente fácil y construir la propia versión del kernel corriendo OpenMosix. Si se desea hacer esto, se debe instalar el paquete kernel-package y leer la documentación relevante.

Si se quiere probar OpenMosix usando un live cd y por lo tanto sin cambiar el sistema existente, se puede buscar en clusterknoppix o quantian. Esos son live cds que también incluyen un servidor terminal, así que se puede obtener un cluster utilizando un solo cdrom y cero discos duros, y clientes sin disco pueden ser usados con kde. ClusterKnoppix no ha sido actualizado desde el 28 de agosto del 2004. El proyecto Quantian no ha sido actualizado desde el 2 de febrero del 2006.

Otro ejemplo es Kerrighed.

Kerrighed es un proyecto mantenido activamente, que proporciona la misma funcionalidad de OpenMosix. La licencia esta bajo GPLv2. La mala noticia es que el código no es completamente portable para todas las versiones de kernel, por lo tanto debe ser compilado con una versión en específico del kernel. Se puede descargar el código y compilarlo con el kernel 2.6.20.

Diseño del cluster

Lo próximo es el diseño del cluster, para el que se debe escoger el hardware y la infraestructura de la red. Si se van a utilizar máquinas con múltiples procesadores, el costo de CPU y RAM se va a mantener generalmente igual, asumiendo que la tarjeta SMP tiene espacio para la RAM que se necesita. El poder de

CPU crecerá según la ley de Mooring, y para la relación disco/CPU, el costo va a disminuir también. También los sistemas SMP también necesitan menos espacio y menos conexiones en red por CPU, y por lo tanto menos switches, y son más baratos y fáciles de instalar y administrar.

Al montar la red, la primera consideración es lo que limita el desempeño de la aplicación. Si la aplicación va a realizar una gran cantidad de álgebra lineal, entonces el CPU puede trabajar extremadamente rápido, entonces la disponibilidad de memoria limita el desempeño en un sistema de un solo CPU, y la red en un cluster. Si se necesitan grandes cantidades de transferencia de datos, la red es el factor limitante.

Si el factor limitante es la red, entonces probablemente sea mejor invertir en hardware SMP (dos o cuatro procesadores), y puede justificar el gasto adicional en artículos para la red más costosos como ethernet gigabit.

Diskless

Por razones administrativas, es también más fácil hacer un cluster que corra sin disco, esto es, con el sistema raíz y las aplicaciones en una máquina. La ventaja es que solo se necesita instalar y configurar el software en un lugar.

Software instalado en un cluster beowulf

- OpenMPI/LAM
- NFS/CIFS/AFS o otro sistema de archivos distribuido para compartir las herramientas y los datos que se van a utilizar.
- DHCPd Para asignar direcciones IP al cluster.
- tFTP/FAI para instalar los nodos del cluster.
- ntp para no tener problemas con el NFS.
- OpenSSH

Utilizando MPICH

MPICH (MPI Chameleon, mcs.anl.gov/mpi/mpich/) funciona sobre una capa de abstracción del hardware que le permite ser independiente de la arquitectura y fácilmente portable. Esta capa, llamada ADI (Abstract Device Interface, Interfaz de Dispositivo Abstracto) se encarga de facilitar el acceso al hardware mientras que el resto de MPICH por encima de la ADI se encarga de la sintaxis y la semántica MPI.

Algunos ADIs (también llamados dispositivos) disponibles en MPICH son:

- Clusters Beowulf y workstations:
 - dispositivo `ch_p4`: es el más general y soporta nodos SMP y sistemas heterogéneos.
 - dispositivo `ch_p4mpd`: más rápido pero sólo soporta clusters homogéneos monoprocesadores.
- Clusters tipo Grid:
 - dispositivo `globus2`: soporta sistemas en los que está instalado Globus.
- Multiprocesadores Simétricos:
 - dispositivo `ch_shmem`: apropiado para sistemas basados en memoria compartida.
 - dispositivo `ch_lfshmem`: versión de `ch_shmem` que no utiliza bloqueos.
- Procesadores Paralelos Masivos (MPPs):
 - existen dispositivos para la mayoría de los MPPs existentes, como `ch_meiko`, `ch_nx` y `ch_mpl`.

Instalar MPICH

En Debian disponemos de varias versiones de MPICH:

- `mpich-bin`: versión con soporte para `ch_p4`, sólo requiere instalarlo en el master.
- `mpich-mpd-bin`: versión para `ch_p4mpd`, requiere el daemon `mpd` activo en todos los nodos.

- `mpich-shmem-bin`: versión con soporte para `ch_shmem` (sistemas con memoria compartida).

Nosotros instalaremos `mpich-bin` (paquete `mpich-bin`), la versión para `ch_p4`.

Configurar MPICH

1. Configurar los nodos: para indicarle a `mpirun` cuáles son las máquinas disponibles para ejecutar programas MPICH usaremos el archivo `/etc/mpich/machines.LINUX`, llamado `chero` de máquinas (en los nodos con múltiples procesadores indicaremos el número, por ejemplo el nodo `slave2` tiene dos procesadores):

```
master
slave1
slave2:2
slave3
```

Los nombres utilizados en `/etc/mpich/machines.LINUX` deben estar en `/etc/hosts`:

```
127.0.0.1 localhost
192.168.0.200 master
192.168.0.201 slave1
192.168.0.202 slave2
192.168.0.203 slave3
```

2. Habilitar SSH sin password: para que el master pueda ejecutar comandos remotos en los esclavos, MPICH utiliza `rsh` (Remote Shell) o `ssh` (Secure Shell). En Debian el comando `rsh` es un enlace del sistema de alternativas que apunta a `/usr/bin/ssh`, por lo que MPICH utiliza por defecto conexiones seguras.

Para disponer de SSH instalaremos en los esclavos el servidor SSH (paquete `openssh-server`) y los configuraremos para que acepten conexiones desde el master sin pedir password ni pass-phrase (utilizaremos el mismo usuario en todas las máquinas). Para ello:

- crearemos una clave RSA en el master:

```
pepe@master:~$ ssh-keygen -t rsa -f ~/.ssh/id_rsa
```

- activaremos ssh-agent en el master para que no nos pida la passphrase:

```
pepe@master:~$ eval 'ssh-agent -s'
Agent pid 12297
pepe@master:~$ ssh-add
Enter passphrase for /home/francis/.ssh/id_rsa:
Identity added: /home/francis/.ssh/id_rsa (/home/francis/.ssh/id_rsa)
```

- copiaremos la clave pública (~/.ssh/id_rsa.pub) en todos los esclavos:

```
pepe@master:~$ scp ~/.ssh/id_rsa.pub francis@slave1:~/.ssh/id_rsa.pub
```

- la añadiremos al final del fichero ~/.ssh/authorized_keys en cada esclavo:

```
pepe@slave1:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- y comprobaremos que podemos iniciar sesión SSH en los esclavos sin que nos pida password:

```
pepe@master:~$ ssh slave1
pepe@slave1:~$
```

3. Ajustar las variables de entorno: para decirle a MPICH que use SSH debemos establecer la variable de entorno P4_RSHCOMMAND, editando .bashrc y añadiendo la línea:

```
export P4_RSHCOMMAND=ssh
```

Cargamos .bashrc de nuevo y comprobamos:

```
$ source ~/.bashrc
$ echo $P4_RSHCOMMAND
ssh
```

4. Directorio compartido ~/: MPICH, para ejecutar mpirun, requiere que todas las máquinas listadas en /etc/mpich/machines.LINUX tengan un filesystem compartido, para lo cual el master exportará vía NFS el directorio ~/ y los esclavos lo montarán en ~/.

Probar MPICH

MPICH proporciona el comando `tstmachines`, que comprueba la disponibilidad de las máquinas listadas en `/etc/mpich/machines.LINUX` para ejecutar programas MPI.

Ejecutar programas MPI

MPICH proporciona cuatro comandos para compilar programas MPI:

- `mpicc`: programas escritos en C.
- `mpiCC`: programas escritos en C++.
- `mpif77`: programas escritos en Fortran 77.
- `mpif90`: programas escritos en Fortran 90.

Por ejemplo, para compilar el programa `cpi.c` (que calcula PI usando MPI) haremos:

```
$ mpicc -o cpi cpi.c
```

Para ejecutar programas MPI utilizaremos el comando `mpirun`. Por ejemplo, para ejecutar el programa `cpi` en 2 procesadores haremos:

```
$ mpirun -np 3 cpi
Process 0 on asterix
Process 2 on slave1
Process 1 on slave2
pi is approximately 3.1416009869231245, Error is 0.00000833333333314
wall clock time = 0.001762
```

OpenMPI

El proyecto OpenMPI (open-mpi.org) es el resultado de la unión de varias implementaciones de MPI, como LAM/MPI, FT-MPI y LA-MPI.

Es muy similar en su manejo a MPICH, algunas diferencias son:

- instalar paquete `openmpi-bin`.
- el chero de máquinas es `/etc/openmpi/openmpi-default-hostfile` y su sintaxis cambia para nodos con múltiples procesadores:

```
master
slave1
slave2 slots=2
slave3
```

LAM/MPI

LAM/MPI (lam-mpi.org) es una implementación de MPI que utiliza un demonio lamd activo en cada nodo.

- Instalar paquete lam-runtime.
- El chero de máquinas es `/etc/lam/bhost.conf` y contiene los nombres completos de las máquinas:

```
master.ldc.usb.ve
slave1.ldc.usb.ve
slave2.ldc.usb.ve
slave3.ldc.usb.ve
```

Para comprobar si el cluster LAM/MPI puede activarse:

```
$ recon
```

Para activar el cluster (lamboot lanza el demonio lamd en cada máquina listada en `/etc/lam/bhost.conf`):

```
$ lamboot
```

Para ejecutar programas MPI haremos:

```
$ mpirun C hello
```

Una vez activado el cluster, para comprobar el estado de los nodos:

```
$ tping N
```

Para apagar el cluster:

```
$ lamhalt
```

LAM/MPI proporciona la herramienta XMPI (paquete xmpi), que permite ejecutar programas paralelos MPI y monitorizarlos. XMPI no puede activar el cluster, éste tiene que haber sido activado con lamboot antes de lanzar XMPI. Para arrancar XMPI ejecutaremos el comando:

```
$ xmpi
```

