Facultad de Ingeniería

Universidad de la República

Proyecto Fenton - Cluster de Computadores de Alto Desempeño con Acceso Remoto (CCADAR)

Informe Final

Julio 2008

Estudiantes:
Santiago Iturriaga, Paulo Maya, Damián Pintos

Supervisor:
Sergio Nesmachnow

Resumen

El informe presenta el trabajo realizado en el marco de la asignatura Proyecto de Grado de la carrera Ingeniería en Computación. Este proyecto fue propuesto por el grupo académico CeCal (Centro de Cálculo) del Instituto de Computación de la Facultad de Ingeniería, Universidad de la República, más específicamente en el área de computación de alto desempeño. El proyecto consistió en la implementación de un cluster de computadoras de alto desempeño con la capacidad de brindar acceso remoto para que sus usuarios interactúen con el y para que sus administradores lo regulen y monitoreen.

Este documento está organizado en siete secciones. La primer sección es una introducción a este proyecto en la cual incluimos una breve introducción a la computación de alto rendimiento, objetivos y motivaciones del proyecto, alcance y actividades desarrolladas en el transcurso del proyecto.

En la segunda sección se profundiza en el estado del arte de la computación de alto rendimiento donde hablamos sobre la importancia de estas tecnologías, organización de computadores, diseño de algoritmos paralelos, arquitecturas de computación paralela y herramientas disponibles. En la tercer sección se presentan las tecnologías relevadas para la implantación de un cluster de alto desempeño. En la cuarta sección se especifica la solución propuesta con las tecnologías elegidas y la interfaz de acceso remoto desarrollada para el cluster de alto desempeño con acceso remoto. También se detalla el proceso de verificación realizado durante el desarrollo. En la quinta sección se describen los ambientes donde el sitema es puesto en producción. Finalmente en la sexta y séptima sección se presentan las conclusiones y los trabajos a futuro.

Palabras clave: Cluster de computadores, Beowulf, DRM, MPI.

Índice

1. Introducción

Durante los primeros años de la historia de las computadoras modernas se consideró que un servidor debía ser necesariamente una única máquina con un procesador muy potente para dar servicio a varias terminales tontas, este servidor es conocido como super computador o mainframe. Según esta doctrina, para aumentar el poder de cálculo del sistema es necesario contar con un procesador más potente en la computadora servidor. De este modo, al aumentar los requerimientos de procesamiento se hace necesario en la mayoría de los casos, cambiar la súper computadora actual por una nueva mejor, pasando a subutilizar o simplemente desechando la súper computadora anterior con un alto costo asociado.

Esta idea encontró oposición en 1994, cuando Donald Becker y Thomas Sterling crearon el proyecto Beowulf, en el que lograron un elevado nivel de procesamiento poniendo a trabajar varias computadoras en paralelo con procesadores 16 DX4, interconectadas en una red Ethernet de 10 Mbit.

A partir de ese momento, y debido al éxito alcanzado en el proyecto, muchos otros proyectos siguieron la investigación del tema, bajo la premisa de convertir hardware de relativo bajo costo en clusters que logren equiparar o superar el desempeño alcanzado por las supercomputadores.

La construcción de los ordenadores del cluster es más fácil y económica debido a su flexibilidad: pueden tener todos la misma configuración de hardware y sistema operativo (cluster homogéneo), diferente rendimiento pero con arquitecturas y sistemas operativos similares (cluster semi-homogéneo), o tener diferente hardware y sistema operativo (cluster heterogéneo).

Para que un cluster funcione como tal, no basta solamente con conectar entre sí los ordenadores, sino que es necesario proveer un sistema de manejo del cluster, el cual se encargue de interactuar con el usuario y los procesos que corren en él para optimizar el funcionamiento.

En general, un cluster necesita de varios componentes de software y hardware para poder funcionar. A saber:

- Nodos (los ordenadores o servidores)
- Sistemas Operativos
- Conexiones de Red
- Middleware (capa de abstracción entre el usuario y los sistemas operativos)
- Protocolos de Comunicación y Servicios.
- Aplicaciones (pueden ser paralelas o no)

Actualmente, el cluster de computadoras se utiliza en diferentes tareas, como data mining, servidores de archivos, servidores de base de datos, servidores web, simuladores de vuelo, renderización de gráficos, modeladores climáticos, entre otros.

El objetivo de un cluster de computadoras es simplemente reunir el poder de cómputo de una serie de nodos para proveer alta escalabilidad, poder de cómputo, o construir redundancia y de esta manera proveer alta disponibilidad. De esta manera en lugar de un simple cliente realizando peticiones a uno o más servidores, los cluster utilizan múltiples máquinas para proveer un ambiente más poderoso de cómputo a través de una sola imagen de sistema. El procesamiento es visible al usuario como una sola unidad, aunque se encuentre compuesto de muchas computadoras trabajando en paralelo para lograr el mismo fin.

El alto poder de procesamiento de varias computadoras trabajando en paralelo (o cluster de computadoras) se perfila como una solución viable a varias problemáticas, que requieren gran poder de cómputo a un bajo costo de implementación. Algunas de las problemáticas actuales más importantes están relacionadas con la ciencia e ingeniería, dinámica de fluidos computacional, simulaciones electromagnéticas, modelado ambiental, dinámica estructural, modelado biológico, dinámica molecular, simulación de redes, modelado financiero y económico, etc.

1.1. Introducción a la computación de alto rendimiento

A medida que la computación avanza los problemas a resolver se tornan cada vez más complicados, con modelos más complejos, con mayores volúmenes de datos y tiempos de respuesta más limitados.

La computación de alto rendimiento o HPC (siglas en inglés de High Performance Computing) nació para satisfacer todos estos requisitos de poder de cómputo; reúne un conjunto de tecnologías, como ser supercomputadores o clusters, y se apoya en el paradigma de programación paralela y distribuida.

La computación distribuida resuelve problemas de computación masiva utilizando un gran número de computadoras con una infraestructura de telecomunicaciones distribuida; y mediante el uso compartido de recursos situados en distintos lugares y pertenecientes a diferentes dominios de administración sobre una red que utiliza estándares abiertos.

La programación paralela es una técnica de programación basada en la ejecución simultánea de procesos o trabajos, bien sea en un mismo ordenador (con uno o varios procesadores) o en un cluster de ordenadores. Existe una diversa gama de arquitecturas de computación paralela, desde arquitecturas de procesadores paralelos con memoria compartida hasta clusters de computadores comunicados por una red como Fast Ethernet.

En la sección 2 se profundizará sobre la computación de alto rendimiento, diferentes tipos de arquitecturas y técnicas, y se analizarán las herramientas y paradigmas de programación paralela y distribuida.

1.2. Motivación del proyecto

Actualmente existe en Facultad de Ingeniería un cluster de computadores utilizado para investigación en proyectos internos. A corto plazo se proyecta alquilar el cluster como servicio a entidades externas. El software utilizado permite

un nivel de abstracción muy bajo, que limita los perfiles de usuarios capaces de utilizar el sistema. Es importante notar que los usuarios finales serán en la mayoría de los casos profesionales en áreas dispares como por ejemplo las ciencias básicas (matemática, física, química, etc.), no necesariamente relacionadas con la informática.

Se desea mantener un control estricto de las actividades de los usuarios de manera de poder garantizar la disponibilidad de recursos en un determinado momento. El grupo responsable del cluster tiene interés en mantener un historial de utilización de recursos por parte de los distintos usuarios y procesos; tanto para poder realizar un seguimiento de su estado actual y verificar su correcto funcionamiento, como para realizar mediciones referidas a la utilización de recursos por parte de los usuarios ya que se lo considera un activo.

El proyecto Fenton intenta resolver estos problemas permitiendo la utilización del sistema por parte de usuarios no especializados, así como también permitir el monitoreo del cluster en tiempo real por parte de los administradores.

1.3. Objetivos y alcance del proyecto

La motivación principal del proyecto Fenton es la construcción e implantación de un sistema de acceso remoto a un cluster de alto desempeño. Dentro del marco de esta motivación se ha definido el alcance del proyecto y varios objetivos y actividades.

Una de las actividades principales englobada dentro del proyecto es el estudio del estado del arte sobre la utilización de equipamiento destinado al procesamiento paralelo-distribuido.

Otro actividad importante es la evaluación y diseño de herramientas automáticas para la administración y utilización de un cluster de computadores. Para esto se planificó realizar el relevamiento de paquetes de software de base (sistemas operativos, bibliotecas, monitores de desempeño, etc.) que permitan el desarrollo y la ejecución de programas paralelos y distribuidos. Durante el relevamiento de las herramientas se deben evaluar tanto características funcionales como características de usabilidad.

Como se mencionó en la sección anterior se debe tener especial consideración en el grado de abstracción de las herramientas de administración y las utilizadas para la ejecución de trabajos en el cluster, para que de esta forma la infraestructura pueda ser utilizada por un conjunto diverso y amplio de usuarios.

A continuación se presentan los objetivos específicos del proyecto:

- El estudio del estado del arte sobre la utilización de equipamiento destinado al procesamiento paralelo-distribuido para la resolución de problemas complejos.
- La evaluación de paquetes de software de base (sistemas operativos, bibliotecas, monitores de desempeño, etc.) que permitan el desarrollo y la ejecución de programas paralelos y distribuidos, instalando una o varias alternativas con tal fin.

- El desarrollo de software que permita la administración semiautomática del clúster, para garantizar su alta disponibilidad y su funcionamiento a niveles razonables de eficiencia computacional.
- El desarrollo de una interfaz amigable que permita la utilización del cluster de forma simple, aún para usuarios sin conocimientos exhaustivos de informática y/o de las técnicas de computación de alto desempeño.

1.4. Trabajos previos

No son pocos los trabajos realizados sobre investigación, desarrollo, construcción y aplicaciones de clusters. Sin embargo es relativamente poca la bibliografía asociada, dado que el tema ha ganado importancia principalmente en la última década. Los estudios se remontan desde los años '70, y puede decirse que IBM fue la primera en establecer una teoría formal con respecto a los clusters a través de un estudio realizado por Gene Myron Amdahl en 1967.

En el CeCal se han llevado a cabo en los últimos años una gran cantidad de trabajos de investigación por parte de docentes y de estudiantes. Dentro del ámbito académico-docente podemos destacar los siguientes trabajos:

Una versión paralela del algoritmo evolutivo para optimización multiobjetivo NSGA-II y su aplicación al diseño de redes de comunicaciones confiables (2003). Proyecto realizado por el Mg. Ing. Sergio Nesmachnow realizado como proyecto final del curso "Introducción a la Optimización Multiobjetivo usando Metaheurísticas", dictado por el Prof. Dr. Carlos Coello en el marco de la VII Escuela Internacional de Informática en el CACIC 2003.

En este trabajo se desarrolló una versión paralela del Algoritmo Evolutivo para Optimización Multiobjetivo NSGA-II. Se introducen los detalles de diseño e implementación de una versión paralela y se analiza la calidad de resultados y la eficiencia computacional, comparando con los resultados y tiempos de ejecución de la versión secuencial del algoritmo NSGA-II sobre un conjunto de problemas de prueba estándar. Adicionalmente, se estudió la aplicación de la versión paralela propuesta a la resolución de un problema de diseño de redes de comunicaciones confiables.

Por parte de los estudiantes podemos mencionar como ejemplo algunos de los proyectos de grado realizados recientemente:

Proyecto MPI.net (2003). Proyecto de grado realizado por Sebastián Baña, Gonzalo Ferres y Nicolás Pepe.

El proyecto propuso la adaptación de la biblioteca MPI para su utilización desde lenguajes a través de la plataforma .NET. Este proyecto formó parte del proceso de desarrollo del área de procesamiento de alta performance en aplicaciones comerciales e industriales utilizando computadoras de bajo costo en curso en el UAS.

Utilizando mecanismos de exportación de clases a través de el ambiente .NET, este proyecto propuso la utilización de MPI desde lenguajes soportados

por .NET de modo que las técnicas de procesamiento paralelo puedan utilizarse desde programas desarrollados en esos lenguajes. Para realizar esto último debieron encapsular en clases las primitivas básicas de comunicación de la biblioteca MPI (envío de mensajes, recepción de mensajes bloqueante y no bloqueante, primitivas de comunicación colectivas) y las primitivas de sincronización de procesos y reducción de resultados de modo que posibilitara su uso desde programas remotos.

Complementariamente, se debió instrumentar un mecanismo que posibilitara la distribución del código paralelo a ejecutar en los diferentes nodos de la máquina virtual, que podían estar en diferentes redes.

Proyecto algoritmos genéticos increméntales (2003). Proyecto de grado realizado por Federico Dominioni y Pablo Musso.

Los Algoritmos Genéticos constituyen uno de los paradigmas de Computación Evolutiva más difundidos, basando su funcionamiento en la simulación del principio de evolución natural en donde a partir de un población (de soluciones potenciales) se establece una especie de ley de la selva virtual en donde los individuos más aptos (mejores soluciones) sobreviven. La evolución de dicha población es a través de interacciones (cruzamiento) y transformaciones únicas (mutaciones), luego de determinadas generaciones la población converge hacia una solución óptima o cercana a la misma.

El auge de la computación paralela y distribuida ha renovado el interés práctico por los algoritmos basados en analogías de procesos naturales para la resolución de problemas de optimización sobre complejos espacios de búsqueda y funciones objetivo.

El proyecto consistió en el estudio, diseño e implementación del modelo de algoritmos genéticos incrementales, un modelo distribuido original de Alba y Troya, dicho modelo deberá ser adaptado para ser ejecutado en un cluster de PC's no dedicadas y por lo tanto para su ejecución se debió tener en cuenta las cargas de dichas PC. Los modelos diseñados fueron aplicados a problemas de optimización combinatoria, en particular en el área de diseño de redes de comunicaciones confiables.

Algoritmos genéticos paralelos y su aplicación al diseño de redes de comunicaciones confiables (2004). Tesis de maestría realizada por Mg. Ing. Sergio Nesmachnow, finalizada en Julio de 2004.

Al tratar con problemas de optimización combinatoria NP difíciles, para los cuales la complejidad de los algoritmos conocidos aumenta de manera superpolinomial con el tamaño del problema, la aplicabilidad de los métodos exactos de resolución se encuentra limitada por el enorme tiempo y consumo de recursos computacionales que demandan. Es aquí cuando las técnicas heurísticas aparecen como la única alternativa viable para abordar problemas NP difíciles.

En este trabajo se realizó un estudio de las técnicas de computación evolutiva y la aplicación de técnicas de procesamiento de alta performance para implementar modelos de algoritmos genéticos capaces de ejecutar en un am-

biente paralelo-distribuido y se presentaron algoritmos evolutivos abocados al caso concreto de diseñar redes de comunicaciones de alta conectividad topológica. También se evaluaron diferentes algoritmos evolutivos puros e híbridos en sus versiones secuenciales y paralelas.

El estudio comparativo realizado en este proyecto reportó resultados satisfactorios tanto desde el punto de vista de la calidad de resultados obtenidos como desde el punto de vista de la mejora de eficiencia computacional alcanzada por las versiones paralelas de los algoritmos con respecto a sus contrapartes secuenciales.

El proyecto Fenton es una muy buena posibilidad para mejorar la eficiencia computacional obtenida con las versiones paralelas desarrolladas en el proyecto de algoritmos paralelos genéticos.

Mejora del desempeño de modelos numéricos del Río de la Plata (2006). Tesis de maestría del Ing. Pablo Ezzatti, finalizada en Junio de 2006.

En las últimas décadas, la simulación computacional de fluidos ha emergido como un área de intenso trabajo. A medida que las técnicas computacionales se fueron perfeccionando, el interés de modelar con precisión el comportamiento de los flujos de fluidos fue en aumento, provocando importantes inversiones en equipamiento informático.

En Uruguay, en el Instituto de Mecánica de Fluidos e Ingeniería Ambiental (IMFIA) de la Facultad de Ingeniería de la Universidad de la República, desde hace algunos años, se está trabajando en el modelado numérico del Río de la Plata.

Con el fin de obtener mejoras significativas en diversas características del modelado, se buscó un nuevo modelo. Finalmente se optó por el modelo RMA-10, ampliamente utilizado en el modelado de estuarios.

Las cualidades de este nuevo modelo permiten la obtención de resultados sumamente alentadores en diversas simulaciones realizadas. En contraposición a la mejora del modelado, se incrementan de forma abrupta los costos computacionales (tiempo de procesamiento), lo que implica un gran obstáculo para su utilización y para aplicar mejoras al modelo.

En esta tesis se estudiaron técnicas de computación de alto desempeño aplicadas al cálculo científico, con el fin de mejorar el desempeño computacional de modelos numéricos que utilizan el paradigma de elementos finitos (MEF). Se abordó la aplicación de las técnicas estudiadas al modelo hidrodinámico RMA-10 aplicado al Río de la Plata.

Se presentó una estrategia secuencial para el cálculo de la matriz de rigidez del modelo RMA-10 utilizando tablas de dispersión y su posterior resolución empleando métodos multifrontales para sistemas lineales dispersos. También se evaluó la aplicación de distintas estrategias de programación paralela a la modificación propuesta y se realizó un estudio comparativo de las diferentes estrategias.

Como en el proyecto anterior de algoritmos genéticos, el proyecto Fenton provee un ambiente que facilitaría las pruebas de las estrategias de programación

paralela estudiadas así como mejoraría la eficiencia computacional de estas.

Algoritmos genéticos paralelos aplicados a la resolución de problemas de asignación de frecuencias en redes celulares (2006). Tesis de licenciatura en informática de la Universidad Nacional de la Patagonia San Juan Bosco (Argentina) realizada por Cristian Perfumo, Gerardo Mora y Lucas Rojas, tutoreada por el Mg. Ing. Sergio Nesmachnow e Ing. José Gallardo.

Con la masificación de las redes inalámbricas, los entes reguladores del espectro de radiofrecuencias se han encontrado con el problema de asignar una frecuencia a cada comunicación sin que esto conlleve a interferencias. Dado que las frecuencias son un recurso finito, el objetivo es maximizar la reutilización de las frecuencias sin disminuir la calidad de la transmisión. En este trabajo se investiga sobre la aplicación de los algoritmos genéticos como herramienta de resolución del problema de asignación de frecuencias en redes inalámbricas.

En este trabajo se llevó a cabo una investigación sobre algoritmos genéticos, considerando sus variantes secuenciales y paralelas, y su aplicación a la resolución del problema de la asignación de frecuencias en las redes de telefonía celular.

Diseño, configuración, administración y evaluación de performance de un HPC cluster(2006). El proyecto propone la construcción de un cluster de alta performance para la solución de problemas con altos requerimientos de computo. Este proyecto se cristalizo en la construcción del cluster Medusa.

El cluster de alto desempeño Medusa es un esfuerzo conjunto del Instituto de Mecánica de los Fluidos e Ingeniería Ambiental (IMFIA) y del Centro de Cálculo (CeCal) (Facultad de Ingeniera, Universidad de la República, Uruguay), con el apoyo financiero del Programa de Desarrollo Tecnológico (PDT), proyecto 48/01 (2006-2008).

En su configuración original (año 2006), Medusa cuenta con seis nodos con procesador Opteron Dual Core y tecnología de interconexión Ethernet.

La aplicación primaria del cluster ha involucrado la ejecución de simulaciones para el desarollo de puentes, barcazas, prevención de efectos ambientales no deseables por buques de gran calado, etc.

El proyecto CCADAR esta íntimamente relacionado con el cluster Medusa pues el sistema de utilización, administración y acceso remoto de buscado en el proyecto CCADAR es para la utilización del cluster Medusa.

1.5. Actividades del proyecto

En el marco del proyecto se realizaron un conjunto de actividades específicas, abarcando actividades de investigación, estudio del arte, análisis, modelación, implementación, ensayo, divulgación de resultados y redacción del presente informe. Las actividades se detallan a continuación.

Investigación y estudio del estado del arte. Unos de los principales objetivos del proyecto es la investigación y estudio del estado del arte a lo que refiere a la computación de alta performance. Esta actividad abarcó auto estudio por parte de los integrantes del proyecto y el curso de la materia Computación de Alta Performance dictada por el grupo CeCal del Instituto de Computación de la Facultad de Ingeniería.

Estudio de tecnologías existentes. En paralelo a la investigación y estudio del estado del arte se llevó a cabo un relevamiento de paquetes de software de base (sistemas operativos, bibliotecas, monitores de desempeño, etc.) que permitieran el desarrollo y la ejecución de programas paralelos y distribuidos.

De esta actividad surgió la propuesta de herramientas que permitieran la implantación y administración del cluster de alto desempeño con acceso remoto.

Relevamiento de requerimientos y determinación del alcance del proyecto. Durante las primeras etapas del proyecto se realizaron reuniones con los tutores para relevar los principales requerimientos, identificar los posibles problemas y determinar el alcance del proyecto que permitiera cumplir con los objetivos del proyecto.

Implementación de la interfaz Web. Dentro de los alcances del proyecto se encuentra la implementación de una interfaz web que permita el acceso remoto al cluster. Esta interfaz debería contener elementos de usabilidad, funcionalidades de administración y reportes del cluster para los administradores del mismo.

Esta actividad se dividió en varias etapas. La primera fase involucró una etapa de relevamiento de requerimientos donde se mantuvieron reuniones con los tutores y especialistas en el área de usabilidad de interfaces web. A continuación se llevó a cabo una etapa de diseño de la interfaz web donde se tuvieron en cuenta conocimientos previos por parte de los integrantes del proyecto así como la curva de aprendizaje de distintas herramientas de desarrollo. Luego de la etapa de diseño se realizó la etapa de desarrollo la cual se hizo en forma iterativa e incremental. Se fijaron pequeñas iteraciones en las cuales se fueron incrementando y mejorando las funcionalidades de la interfaz web.

Verificación de la solución. Una vez implantado el cluster con las tecnologías seleccionadas y luego de terminado el desarrollo de la interfaz web se procedió con la etapa de verificación. En esta etapa se probaron los principales requerimientos funcionales del sistema.

Por último, otras actividades llevadas a cabo fueron la realización de la documentación final como este informe, manuales, y la presentación final.

2. Computación de alto rendimiento

Como se mencionó anteriormente, en el correr de los últimos años ha crecido la importancia de satisfacer los requisitos crecientes de poder de cómputo debido a la necesidad de resolver problemas realistas con modelos más complejos, así como trabajar con grandes volúmenes de datos sin perder de vista la capacidad de respuesta en un tiempo limitado. El procesamiento paralelo (o computación paralela de alto rendimiento) ha ganado terreno y se ha vuelto muy importante en la resolución de problemas, apoyándose en tecnologías como los clusters y mediante el uso de paradigmas de programación paralela y/o distribuida.

La computación distribuida como modelo para resolver problemas de computación consiste en compartir recursos heterogéneos (basados en distintas plataformas, arquitecturas de computadores y programas, lenguajes de programación), situados en distintos lugares y pertenecientes a diferentes dominios de administración sobre una red que utiliza estándares abiertos. La computación distribuida ha sido diseñada para resolver problemas que superan la capacidad de cualquier supercomputadora o mainframe manteniendo la flexibilidad de trabajar en múltiples problemas más pequeños. Existen varios grados de distribución: de hardware y procesamiento, de datos y de control. Algunas de las ventajas de la computación distribuida son la mejora en el desempeño, la robustez, la seguridad no centralizada y el acceso transparente a los datos remotos. Las técnicas de autorización segura son esenciales antes de permitir que los recursos informáticos sean controlados por usuarios remotos dado que el entorno es generalmente multiusuario.

La programación paralela es una técnica de programación basada en la ejecución simultánea, bien sea en un mismo ordenador (con uno o varios procesadores) o en un cluster de ordenadores, en cuyo caso se denomina computación paralela-distribuida. Al contrario que en la programación concurrente, las técnicas de paralelización enfatizan la verdadera simultaneidad de la ejecución de las tareas. El avance en diferentes tecnologías como microprocesadores con mayor poder de procesamiento, comunicación de datos de alta velocidad y el desarrollo de bibliotecas para la comunicación entre procesos han facilitado la utilización de la programación paralela en aplicaciones. Mediante la programación paralela los multicomputadores y los sistemas con múltiples procesadores consiguen un aumento en el rendimiento, en cambio en los sistemas monoprocesador el beneficio en el rendimiento no es tan evidente ya que la CPU es compartida por múltiples procesos en el tiempo. El mayor problema de la computación paralela radica en la complejidad en la sincronización entre tareas, ya sea mediante secciones críticas, semáforos o pasaje de mensajes, para garantizar la exclusión mutua en las zonas del código en las que sea necesario.

La programación paralela proporciona ventajas como mayor capacidad de proceso, menor tiempo de procesamiento y aprovechamiento de la escalabilidad potencial de los recursos. Existen diversos tipos de aplicaciones que pueden verse beneficiadas por la programación paralela. Por un lado están las aplicaciones de cálculo intensivas, que requieren un alto número de ciclos de máquina y han impulsado el desarrollo de los supercomputadores. Algunos ejemplos son aplica-

ciones para dinámica de fluidos computacional, simulaciones electromagnéticas, modelado ambiental, dinámica estructural, modelado biológico, dinámica molecular, simulación de redes y modelado financiero y económico.

Otro tipo de aplicaciones con una gran demanda de recursos son las de almacenamiento masivo, dependen de la capacidad para almacenar y procesar grandes cantidades de información y requieren de un acceso rápido y seguro a una cantidad considerable de datos almacenados. Algunas de ellas son las aplicaciones para análisis de data sísmica, procesamiento de imágenes, minería de datos, análisis estadístico de datos y análisis de mercados.

Finalmente las aplicaciones exigentes comunicacionalmente son relativamente nuevas y pueden ser llamadas servicios por demanda. Requieren de recursos computacionales conectados por redes con un ancho de banda considerable. Por ejemplo procesamiento de transacciones en línea, sistemas colaborativos, texto por demanda, vídeo por demanda, imágenes por demanda o simulación por demanda. Obviamente todas las aplicaciones anteriores dependen en cierto grado de cada uno de los aspectos computacionales mencionados: poder de cómputo, capacidades de almacenamiento y eficientes canales de comunicación, sin embargo las podemos agrupar por su característica dominante.

Los sistemas de sistemas combinan en forma más compleja las características anteriores y dependen, en muchos casos, de sistemas computacionales integrados diseñados primordialmente para ellas. Ejemplos de sistemas de sistemas son las aplicaciones de soporte a decisiones corporativas y gubernamentales, control de sistemas a tiempo real, banca electrónica o compras electrónicas.

Existe una alta correspondencia entre la evolución de tecnologías informáticas y el desarrollo de aplicaciones; en particular el hardware tiene gran influencia en el éxito de ciertas áreas. Las aplicaciones intensivas en cálculo fueron estimuladas principalmente por máquinas vectoriales y procesadores masivamente paralelos. Las aplicaciones de almacenamiento masivo han sido guiadas por dispositivos de almacenamiento como RAID y robots de cintas. Las aplicaciones exigentes comunicacionales como herramientas colaborativas basadas en WWW y servicios por demanda en línea originalmente surgieron con las LAN y están creciendo drásticamente con Internet.

2.1. Organización de computadores

La organización de los procesadores se refiere a como se conectan o enlazan los procesadores o nodos en un computador paralelo. Existen varios criterios para evaluar los distintos diseños de organización. Se puede tener en cuenta el diámetro, el cual viene dado por la mayor distancia entre dos nodos, mientras menor sea el diámetro menor será el tiempo de comunicación entre nodos. Otro factor a tener en cuenta es el ancho de bisección de la red, siendo este el menor número de enlaces que deben ser removidos para dividir la red por la mitad. Un ancho de bisección alto puede reducir el tiempo de comunicación cuando el movimiento de datos es sustancial, y hace al sistema más tolerante a fallas debido a que defectos en un nodo no hacen inoperable a todo el sistema.

Las redes se pueden separar en redes estáticas y dinámicas, en las redes

estáticas la topología de interconexión se define cuando se construye. Si la red es dinámica, la interconexión puede variar durante la ejecución de un programa o entre la ejecución de programas.

Bus y Ethernet (figura 1). En las redes de tipo bus o Ethernet los procesadores comparten un único bus como recurso de comunicación. La arquitectura es fácil y económica de implementar, pero no es escalable ya que solo un procesador puede usar el bus en un momento dado, a medida que se incrementa el número de procesadores, el bus se convierte en un cuello de botella debido a la congestión.

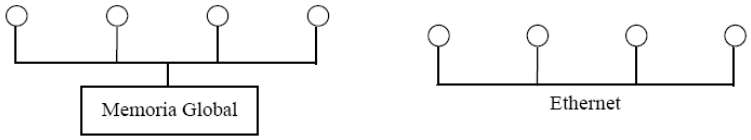


Figura 1: Bus y Ethernet

Mallas (figura 2). Al igual que la interconexión mediante bus (o Ethernet), las mallas son fáciles y económicas de implementar, sin embargo el diámetro se incrementa al añadir nodos. En las mallas de dimensión 1, si los dos nodos extremos también son conectados, entonces se tiene un anillo. Las mallas de 2 dimensiones y de 3 dimensiones son comunes en computación paralela y tienen la ventaja de que pueden ser construidas sin conexiones largas. El diámetro de las mallas puede ser reducido a la mitad si se extiende la malla con conexiones toroidales de forma que los procesadores en los bordes también estén conectados con vecinos. Esto sin embargo presenta dos desventajas: conexiones más largas son requeridas y un subconjunto de un torus no es un torus y los beneficios de esta interconexión se pierden si la máquina es particionada entre varios usuarios.

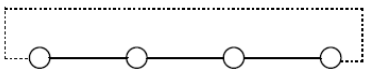


Figura 2: Mallas

Mariposa (figura 3). Las redes de tipo mariposa son similares a las mallas pero presentan menor diámetro.

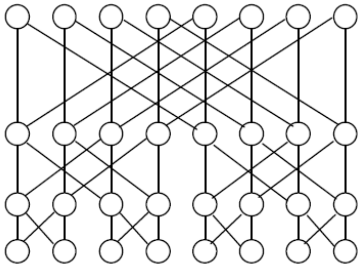


Figura 3: Mariposa

Árboles binarios. Los árboles binarios son particularmente útiles en problemas de ordenamiento, multiplicación de matrices, y algunos problemas en los que los tiempos de solución crecen exponencialmente con el tamaño del problema (NP-difíciles).

Pirámides (figura 4). Las redes de tipo pirámide intentan combinar las ventajas de las mallas y los árboles, se incrementa la tolerancia a fallas y el número de vías de comunicación.

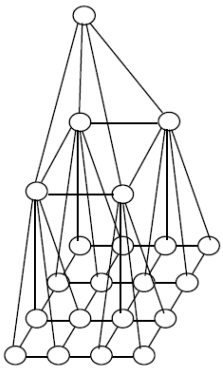


Figura 4: Pirámides

Hipercubo (figura 5). Un hipercubo puede ser considerado como una malla con conexiones largas adicionales, estas conexiones reducen el diámetro e incrementan el ancho de bisección.

Se puede definir recursivamente un hipercubo de la siguiente manera: un hipercubo de dimensión-cero es un único procesador y un hipercubo de dimensión-uno conecta dos hipercubos de dimensión-cero. En general, un hipercubo de dimensión d+1 con 2d+1 nodos se construye conectando los procesadores respectivos de dos hipercubos de dimensión d.

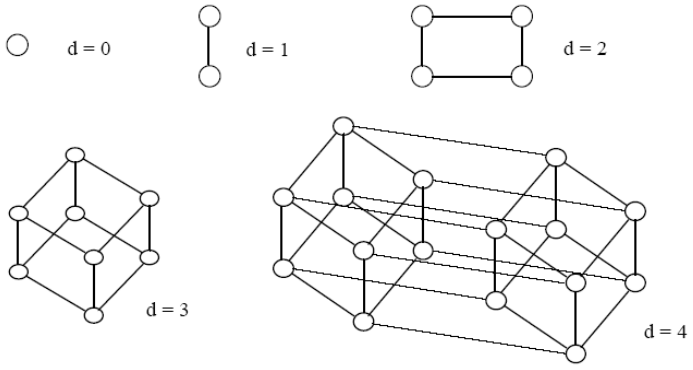


Figura 5: Hipercubo

Omega. Omega es una red formada por crossbar switches 2x2, con cuatro estados posibles: recto, cruzado, broadcast superior y broadcast inferior que son configurados según la conexión que se tiene. Estas topologías se llaman dinámicas o reconfigurables.

Estas redes reducen considerablemente la competencia por ancho de banda, pero son altamente no escalables y costosas. El switch de alto desempeño de la SP2 es una red omega.

La figura 6 muestra una red omega de 3 etapas que conecta 8 procesadores.

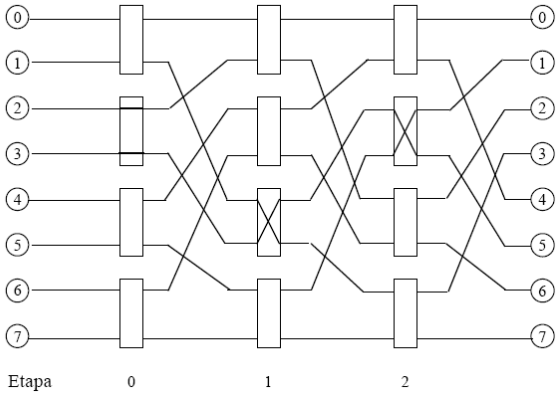


Figura 6: Omega

Fibras de interconexión (figura 7). Las fibras de interconexión son un conjunto de switches, llamados routers, enlazados por distintas configuraciones o topologías.

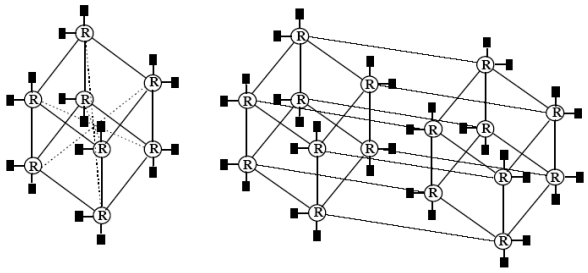


Figura 7: Fibras de interconexión

2.2. Diseño de algoritmos paralelos

El diseño de algoritmos paralelos involucra cuatro etapas, las cuales se presentan como secuenciales pero que en la práctica no lo son.

La primera etapa es la de partición, en esta etapa el cómputo y los datos sobre los cuales se opera se descomponen en tareas. En este punto del diseño se ignoran aspectos como el número de procesadores de la máquina a usar y se concentra la atención en explotar oportunidades de paralelismo.

Finalizada la primera etapa, se inicia la etapa de comunicación donde se determina la comunicación para coordinar las tareas, se definen estructuras y algoritmos de comunicación.

Luego en la etapa de agrupación se evalúa en términos de eficiencia y costos de implementación a las dos etapas anteriores. Se agrupan tareas pequeñas en tareas más grandes.

En la última etapa cada tarea es asignada a un procesador tratando de maximizar la utilización de los procesadores y de reducir el costo de comunicación. La asignación puede ser estática (se establece antes de la ejecución del programa) o en tiempo de ejecución mediante algoritmos de balanceo de carga.

2.2.1. Partición

En la etapa de partición se buscan oportunidades de paralelismo y se trata de subdividir el problema lo más finamente posible, dividiendo tanto los cómputos como los datos.

Existen dos formas de descomposición, descomposición del dominio y descomposición funcional. La descomposición del dominio se centra en los datos, se determina la partición apropiada de los datos y luego se trabaja en los cómputos asociados con los datos. La descomposición funcional es el contrario al enfoque anterior, primero se descomponen los cómputos y luego se ocupa de los datos.

En el proceso de partición existen varios aspectos a tener en cuenta, en particular el orden de tareas debe ser por lo menos superior al número de procesadores para tener flexibilidad en etapas siguientes. Es importante evitar cómputos y almacenamientos redundantes, y las tareas deben ser de tamaños equivalentes para facilitar el balanceo de la carga de los procesadores. Para permitir cierta escalabilidad el número de tareas debe ser proporcional al tamaño del problema.

2.2.2. Comunicación

El proceso de partición de tareas no es independiente del proceso de comunicación. En el proceso de comunicación se especifica como los datos serán transferidos o compartidos entre tareas.

La comunicación puede ser definida en dos fases. Primero se definen los canales que conectan las tareas y luego se especifica la información o mensajes que deben ser enviados y recibidos en estos canales.

La forma de organización de la memoria (distribuida o compartida) determinará la forma de atacar la comunicación entre tareas. En ambientes de memoria distribuida, las tareas tienen una identificación única e interactúan enviando y recibiendo mensajes hacia y desde tareas específicas. Las bibliotecas más conocidas para implementar el pasaje de mensajes en ambientes de memoria distribuida son MPI (Message Passing Interface) y PVM (Parallel Virtual Machine).

En ambientes de memoria compartida no existe la noción de pertenencia y el envío de datos no se da como tal. Todas las tareas comparten la misma memoria. Semáforos, semáforos binarios, barreras y otros mecanismos de sincronización

son usados para controlar el acceso a la memoria compartida y coordinar las tareas.

Para poder dar esta etapa por terminada es necesario tener en cuenta varios aspectos. Cada tarea debe efectuar aproximadamente el mismo número de operaciones de comunicación que cada una de las demás tareas. De otra forma es muy probable que el algoritmo no sea extensible a problemas mayores ya que habrán cuellos de botella. La comunicación entre tareas debe ser tan reducida como sea posible, de esta forma los procesos no se trancan entre sí en la espera de mensajes y se evita que el canal de comunicación sea un cuello de botella. Las operaciones de comunicación y los cómputos de diferentes tareas deben poder proceder concurrentemente.

2.2.3. Agrupación

Durante las tareas de participación y comunicación el algoritmo resultante es aún abstracto en el sentido de que no se tomó en cuenta la arquitectura sobre la que se ejecutará. En este proceso se busca un algoritmo concreto, que se ejecute eficientemente sobre cierta clase de computadores. En particular, se considera la utilidad de agrupar tareas y de replicar datos y/o cómputos.

En el proceso de partición se trata de establecer el mayor número posible de tareas con la intención de maximizar el paralelismo. La maximización de tareas no necesariamente produce un algoritmo eficiente, ya que el costo de comunicación puede ser significativo.

Mediante la agrupación de tareas se puede reducir la cantidad de datos a enviar y así reducir el número de mensajes y el costo de comunicación. Se puede intentar replicar cómputos y/o datos para reducir los requerimientos de comunicación.

En esta etapa es importante chequear si la agrupación redujo los costos de comunicación. A su vez si se han replicado cómputos y/o datos, se debe verificar que los beneficios son superiores a los costos. Se debe verificar que las tareas resultantes tengan costos de cómputo y comunicación similares.

Al igual que en las otras etapas se debe revisar si el número de tareas es extensible con el tamaño del problema. Si el agrupamiento ha reducido las oportunidades de ejecución concurrente, se debe verificar que aun hay suficiente concurrencia y posiblemente considerar diseños alternativos.

Finalmente se deberá analizar si es posible reducir aun más el número de tareas sin desbalancear la carga o reducir la extensibilidad.

2.2.4. Asignación

En este proceso se determina en que procesador se ejecutará cada tarea. En máquinas de memoria compartida tipo UMA no se presenta este problema ya que proveen asignación dinámica de procesos.

Actualmente no hay mecanismos generales de asignación de tareas para máquinas distribuidas por lo que este problema debe ser atacado explícitamente en el diseño de algoritmos paralelos.

La asignación de tareas puede ser estática o dinámica. En la asignación estática, las tareas son asignadas a un procesador al comienzo de la ejecución del algoritmo paralelo y corren ahí hasta el final. La asignación estática en ciertos casos puede resultar en un tiempo de ejecución menor respecto a asignaciones dinámicas y también puede reducir el costo de creación de procesos, sincronización y terminación.

En la asignación dinámica se hacen cambios en la distribución de las tareas entre los procesadores en tiempo de ejecución, con el fin de balancear la carga del sistema y reducir el tiempo de ejecución. Sin embargo, el costo de realizar el balance puede ser significativo y por ende incrementar el tiempo de ejecución.

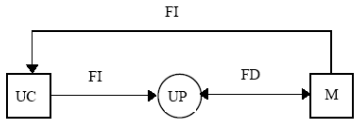
Los algoritmos de balanceo de carga se pueden clasificar como balanceo centralizado, balanceo completamente distribuido y balanceo semi-distribuido. Cuando el balanceo es centralizado un nodo ejecuta el algoritmo y mantiene el estado global del sistema. Como contrapartida cuando el balanceo es completamente distribuido cada procesador mantiene su propia visión del sistema intercambiando información con sus vecinos y así hacer cambios locales.

Como posible punto intermedio entre ambos tipos de algoritmo está el balanceo semi-distribuido, donde los procesadores se dividen en regiones, cada una con un algoritmo centralizado local mientras que otro algoritmo balancea la carga entre las regiones.

2.3. Taxonomía de Flynn

Una forma de clasificar las diferentes arquitecturas de computación secuencial y paralela es mediante la taxonomía de Flynn, que está basada en la multiplicidad del flujo de instrucciones y del flujo de datos en un computador. Un flujo de instrucciones es una secuencia de instrucciones ejecutadas por el computador y un flujo de datos es la secuencia de datos sobre los cuales operan las instrucciones.

Dentro de la clasificación de Flynn hay varias categorías. La primera de ellas es SISD, Single Instruction stream, Single Data stream (ver figura 8). La mayor parte de computadores seriales son SISD, teniendo en general un CPU que ejecuta una instrucción y busca o guarda datos en un momento dado.



UP: Unidad de Procesamiento

UC: Unidad de Control

M: Memoria

FI: Flujo de Instrucciones

FD: Flujo de Datos

Figura 8: Single Instruction stream, Single Data stream

La segunda categoría es SIMD, Single Instruction stream, Multiple Data stream (ver figura 9). Esta categoría comprende los arreglos de procesadores. Tiene un conjunto de unidades de procesamiento cada una ejecutando la misma operación sobre datos distintos.

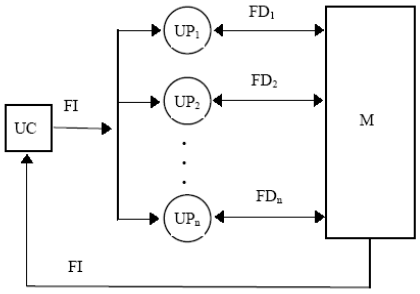


Figura 9: Single Instruction stream, Multiple Data stream

En la categoría MISD, Multiple Instruction stream, Single Data stream (ver figura 10) existen n procesadores, cada uno recibiendo una instrucción diferente y operando sobre el mismo flujo de datos. Actualmente no hay máquinas de este tipo por ser poco prácticas, a pesar de que ciertas MIMD puedan ser usadas de esta forma.

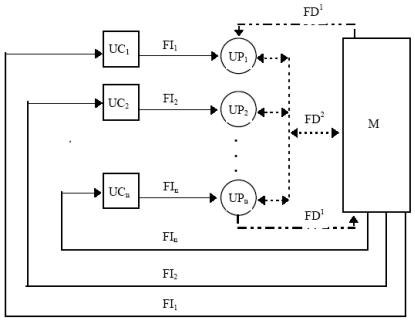


Figura 10: Multiple Instruction stream, Single Data stream

La categoría de MIMD, Multiple Instruction stream, Multiple Data stream (ver figura 11) incluye a la mayoría de los multiprocesadores y multicomputadores, que en general tienen más de un procesador independiente, y cada uno puede ejecutar un programa diferente sobre sus propios datos. Es posible hacer una segunda categorización según su forma de organización de memoria, ya sea memoria compartida o memoria distribuida.

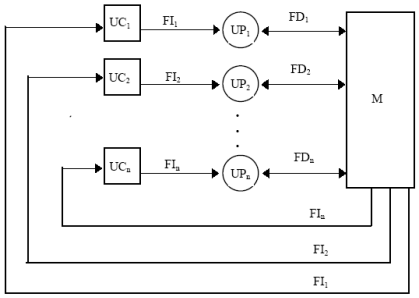


Figura 11: Multiple Instruction stream, Multiple Data stream

Por último la categoría SPMD (Single Program, Multiple Data), donde cada procesador ejecuta una copia exacta del mismo programa, pero opera sobre datos diferentes. Pese a no ser una categoría definida por Flynn es muy usada y puede ser considerada como un caso particular de MIMD.

2.4. Arquitecturas de computación paralela

En esta sección se enumeran las diferentes arquitecturas que se pueden encontrar relacionadas a la computación paralela.

2.4.1. Arreglos de procesadores

En un arreglo de procesadores tenemos un computador cuyo conjunto de instrucciones permite operaciones tanto sobre vectores como escalares al que se denomina computador vectorial.

Procesador Vectorial Pipelined. En estas máquinas los vectores fluyen a través de las unidades aritméticas pipelined. Las unidades consisten de una cascada de etapas de procesamiento compuestas de circuitos que efectúan operaciones aritméticas o lógicas sobre el flujo de datos que pasa a través de ellas.

Las etapas están separadas por registros de alta velocidad usados para guardar resultados intermedios. La información que fluye entre las etapas adyacentes está bajo el control de un reloj que se aplica a todos los registros simultáneamente. Ejemplos de esta categoría son máquinas como la Cray-1 y la Cyber-205.

Arreglos de Procesadores. Son máquinas que constan de un computador secuencial conectado a un arreglo de elementos idénticos de procesamiento sincronizado capaces de ejecutar las mismas operaciones sobre datos diferentes. El computador secuencial generalmente es un CPU de propósito general que almacena el programa y los datos que serán operados en paralelo, además de ejecutar la porción secuencial del programa. Los elementos de procesamiento se asemejan a CPUs pero no tienen unidades de control propias; el computador secuencial genera todas las señales de control para las unidades de procesamiento en el computador.

Los arreglos de procesamiento difieren fundamentalmente en la complejidad y la topología de interconexión de sus elementos de procesamiento. Ejemplos de estas máquinas son: IILIAC IV, Goodyear MPP y Connection Machine CM-200.

2.4.2. Multiprocesadores

Son equipos formados por un número de procesadores completamente programables capaces de ejecutar su propio programa.

La principal debilidad que presentan los multiprocesadores (máquinas de memoria compartida) radica en que no se pueden agregar procesadores indefinidamente ya que a partir de cierto número y dependiendo de la aplicación, el mecanismo de switches o enrutamiento se satura, generando un problema importante a la hora de escalar la solución.

UMA: Multiprocesadores de Acceso Uniforme a Memoria. Estos computadores tienen sus procesadores interconectados a través de un mecanismo de switches a una memoria compartida centralizada. Entre estos mecanismos están: un bus común, crossbar switches o packet-switched networks.

Encore Multimax y Sequent Symetry S81 son ejemplos comerciales de este tipo de multiprocesadores.

NUMA: Multiprocesadores de Acceso No-Uniforme a Memoria. Estos multiprocesadores tienen el espacio de direccionamiento compartido y la memoria distribuida. La memoria compartida está formada por la memoria local de los procesadores. El tiempo de acceso a memoria depende de si el acceso es local al procesador o no. La BBN TC2000 y la SGI Origin 2000 son ejemplos de este modelo de computación paralela.

2.4.3. Multicomputadores

Una de las principales características de los multicomputadores es que los procesadores que los conforman no comparten memoria, cada procesador tiene su propia memoria privada (máquinas de memoria distribuida) y la interacción entre ellos es a través de pasaje de mensajes. Ejemplos son: Intel ParagonXP/S, Meikos Computing Surface, nCUBE 2, Parsytec SuperCluster, Thinking Machine CM-5 y la IBM SP2.

2.4.4. Máquinas de memoria compartida distribuida

Actualmente las máquinas paralelas tienden a aprovechar las facilidades de programación que ofrecen los ambientes de memoria compartida y la escalabilidad de los ambientes de memoria distribuida. Este modelo conecta entre sí módulos de multiprocesadores manteniendo la visión global de la memoria.

Las máquinas de memoria compartida distribuida entra dentro de la categoría de NUMA y un ejemplo es la SGI Origin 2000.

2.4.5. Multiprocesadores multi-hebrados

En los multiprocesadores multi-hebrados, cada procesador tiene cierto número de flujos de instrucciones implementados en hardware, incluyendo el contador de programa y registros, cada uno destinado a ejecutar una hebra. En cada ciclo el procesador ejecuta instrucciones de una de las hebras. En el ciclo siguiente, el procesador hace un cambio de contexto y ejecuta instrucciones de otra hebra.

La Tera MTA (multithreaded architecture) es el primer ejemplo con esta arquitectura, donde cada procesador tiene 128 flujos de instrucciones. Un acceso a memoria dura aproximadamente 100 ciclos por lo que en la próxima ejecución de la hebra se tendrán los datos requeridos. Este mecanismo permite a la MTA tolerar la latencia a memoria y por lo tanto no requiere de memorias cache.

Cada instrucción de 64-bits codifica 3 operaciones (una de memoria y dos que pueden ser aritméticas o de control). Cuenta con un sistema operativo de versión distribuida completamente simétrica de UNIX.

El sistema cuenta con entre 1 y 256 procesadores que comparten una enorme memoria. A su vez cada procesador tiene 1 o 2 GB de memoria, un mapeo aleatorio de la memoria y una red altamente interconectada proveen acceso casi uniforme de cualquier procesador a cualquier memoria.

Los estudios realizados muestran que el costo del multi-hebrado es pequeño, con un rendimiento comparable con el de la T90 y los códigos de la MTA

son significativamente más fáciles de optimizar que en máquinas masivamente paralelas o estaciones de trabajo de alto rendimiento.

2.4.6. Cluster de PC's

El término cluster se aplica a los conjuntos de computadoras unidos mediante una red de alta velocidad que se comportan como si fuesen un único recurso computacional con mayor poder de cómputo. Hoy en día los clusters de PC's tienen un papel importante en aplicaciones científicas, de ingeniería, comerciales, simulaciones, etc.

La tecnología de clusters ha evolucionado apoyándose en actividades que van desde aplicaciones de supercómputo y software de misiones críticas, servidores Web y comercio electrónico, hasta bases de datos de alto rendimiento.

El uso de clusters surge gracias a la convergencia de varias tendencias actuales como la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, la existencia de herramientas para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran.

Se espera de un cluster que presente combinaciones de las siguientes características: alto rendimiento, alta disponibilidad, equilibrio de carga y escalabilidad.

La construcción de un cluster es económica debido a su flexibilidad, permitiendo tener varios computadores con la misma configuración de hardware y sistema operativo (cluster homogéneo), diferente rendimiento pero con arquitecturas y sistemas operativos similares (cluster semi-homogéneo), o tener diferente hardware y sistema operativo (cluster heterogéneo).

Se pueden construir clusters con ordenadores personales desechados por anticuados que consiguen competir en capacidad de cálculo con superordenadores de precios elevados.

Cluster de alto rendimiento. Un cluster de alto rendimiento está diseñado para dar altas prestaciones de capacidad de cálculo.

Por medio de un cluster se pueden conseguir capacidades de cálculo superiores a las de un ordenador más caro. Para garantizar esta capacidad de cálculo, los problemas necesitan ser paralelizables, ya que el método con el que los clusters agilizan el procesamiento es dividir el problema en subproblemas más pequeños y resolverlos en los nodos.

Cluster de alta disponibilidad. Un cluster de alta disponibilidad se caracteriza por compartir los discos de almacenamiento de datos, los distintos computadores se monitorean constantemente entre sí para responder en forma instantánea frente a la caída de uno de ellos.

Los clusters de alta disponibilidad pueden dividirse entre clusters de alta disponibilidad de infraestructura y clusters de alta disponibilidad de aplicación.

Cluster de balanceo de carga. Un cluster de equilibrio de carga o de cómputo adaptativo está compuesto por uno o más ordenadores que actúan como front-end del cluster, y que se ocupan de repartir las peticiones de servicio que reciba el cluster a otros ordenadores del cluster que forman el back-end de éste.

Escalabilidad. La escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para crecer en tamaño sin perder calidad en los servicios ofrecidos. La capacidad de este tipo de clusters se puede ampliar fácilmente añadiendo más ordenadores al cluster.

La robustez es una de las características más importantes ya que ante la caída de alguno de los ordenadores del cluster el servicio se puede ver mermado, pero mientras haya ordenadores en funcionamiento, éstos seguirán dando servicio.

2.5. Herramientas para la programación paralela

En programación paralela existen diferentes lenguajes y herramientas de programación con características específicas para diferentes clases de problema.

C++ Composicional es una extensión de C++ que provee al programador facilidades para controlar localidad, concurrencia, comunicaciones, y asignación de tareas. Puede ser usado para construir librerías que implementen tareas, canales, y otras abstracciones básicas de la programación paralela.

High Performance Fortran (HPF) es un ejemplo de lenguajes datos-paralelos y se ha convertido en un estándar para aplicaciones científicas e ingenieriles. El paralelismo es expresado en términos de operaciones sobre matrices y la comunicación es inferida por el compilador.

Parallel Virtual Machine (PVM) es una biblioteca de subrutinas para enviar y recibir mensajes.

Message Passing Interface (MPI) es una biblioteca similar a PVM pero, tal como HPF, MPI se ha convertido un estándar.

2.5.1. MPI: Message Passing Interface

MPI es un estándar que define la sintaxis y la semántica de las funciones contenidas en una librería de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores. Fue creado por científicos desarrolladores de software y aplicaciones bajo el objetivo de desarrollar un estándar portable y eficiente para programación paralela. Implementaciones del estándar en forma de biblioteca permiten trabajar con memoria distribuida, tener paralelismo explícito y contener un único mecanismo de comunicación que puede ser punto a punto.

El número de tareas es fijado en tiempo de pre-ejecución (aunque esto ha cambiado en la versión 2 del estándar) y permite agrupamiento de procesos e incluye contextos de comunicación entre grupos de procesos.

La característica que hace que las implementaciones de este estándar sean de especial interés para sistemas distribuidos es que no utiliza memoria compartida.

Los elementos principales que intervienen en el paso de mensajes son el proceso que envía, el que recibe y el mensaje. Dependiendo de si el proceso que envía el mensaje espera a que el mensaje sea recibido, se puede hablar de paso de mensajes síncrono o asíncrono. En el paso de mensajes asíncrono, el proceso que envía, no espera a que el mensaje sea recibido, y continúa su ejecución, siendo posible que vuelva a generar un nuevo mensaje y a enviarlo antes de que se haya recibido el anterior. Por este motivo se suelen emplear buzones, en los que se almacenan los mensajes a espera de que un proceso los reciba. Generalmente empleando este sistema, el proceso que envía mensajes solo se bloquea cuando finaliza su ejecución, o cuando el buzón está lleno. En el paso de mensajes síncrono, el proceso que envía el mensaje espera a que un proceso lo reciba para continuar su ejecución. Por esto se suele llamar a esta técnica encuentro, o rendezvous. Dentro del paso de mensajes síncrono se engloba a la llamada a procedimiento remoto, muy popular en las arquitecturas cliente/servidor. La Interfaz de Paso de Mensajes es un protocolo de comunicación entre computadoras, es el estándar para la comunicación entre los nodos que ejecutan un programa en un sistema de memoria distribuida.

Las implementaciones en MPI consisten en un conjunto de bibliotecas de rutinas que pueden ser utilizadas en programas escritos en los lenguajes de programación C, C++, Fortran y Ada. La ventaja de MPI sobre otras bibliotecas de paso de mensajes, es que los programas que utilizan la biblioteca son portables (dado que MPI ha sido implementado para casi toda arquitectura de memoria distribuida), y rápidos, (porque cada implementación de la librería ha sido optimizada para el hardware en la cual se ejecuta).

2.5.2. PVM: Parallel Virtual Machine

PVM permite utilizar una colección de computadores heterogéneos como una única máquina virtual paralela. El sistema administra en forma transparente el enrutamiento de mensajes, la conversión de datos y la planificación de tareas en la máquina virtual. Los programas están compuestos por una serie de tareas que cooperan entre sí. Estas tareas acceden a recursos de PVM a través de una biblioteca estándar, que contiene funciones para iniciación y finalización de tareas así como para comunicación y sincronización entre tareas.

El sistema PVM se compone de dos partes. La primera parte es un daemon, llamado pvmd3, el cual reside en todas las máquinas que forman la máquina virtual. Cuando un usuario desea ejecutar una aplicación, lo primero que debe hacer es crear la máquina virtual, iniciando PVM. Luego, la aplicación se puede ejecutar desde cualquiera de los computadores en el sistema. Múltiples usuarios pueden configurar máquinas virtuales y cada uno de ellos puede ejecutar múltiples aplicaciones simultáneamente en su configuración.

La segunda parte de PVM es una biblioteca de funciones que ofrece un completo repertorio de primitivas. Existen funciones para enviar y recibir mensajes, iniciar procesos, coordinar tareas y modificar la máquina virtual. El modelo de computación de PVM se basa en la noción de que una aplicación consiste de múltiples tareas y cada tarea es responsable de una parte de la carga de trabajo

computacional de la aplicación. Las tareas se identifican mediante un entero llamado el identificador de tarea y abreviado TID. Los mensajes se envían y reciben utilizando estos TID, los cuales son asignados por PVM.

Algunas de sus cualidades de PVM son potencia, simplicidad, portabilidad, practicidad. Permite la administración dinámica de un conjunto de equipos que conforman la máquina virtual. Contiene mecanismos de creación e identificación de procesos, un modelo de comunicación entre procesos basado en pasaje de mensajes y brinda mecanismos de sincronización de procesos. PVM brinda soporte para manejar heterogeneidad y aprovecha las arquitecturas multiprocesador.

2.5.3. Middleware

El middleware es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El middleware nos abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas, dando a su vez la sensación al usuario de que utiliza un único computador muy potente. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipo de servicios de middleware.

Por lo general el middleware del lado cliente está implementado por el Sistema Operativo subyacente, el cual posee las librerías que implementan todas las funcionalidades para la comunicación a través de la red.

Además ofrece herramientas para la optimización y mantenimiento del sistema: migración de procesos, checkpoint-restart (congelar uno o varios procesos, mudarlos de servidor y continuar su funcionamiento en el nuevo host), balanceo de carga, tolerancia a fallos, etc. Es importante que la solución sea escalable, en este caso debe poder detectar automáticamente nuevos servidores conectados al cluster para proceder a su utilización.

El middleware recibe los trabajos entrantes al cluster y los redistribuye de manera que el proceso se ejecute más rápido y el sistema no sufra sobrecargas en un servidor. Esto se realiza mediante políticas definidas en el sistema (automáticamente o por un administrador) que le indican dónde y cómo debe distribuir los procesos, por un sistema de monitorización, el cual controla la carga de cada CPU y la cantidad de procesos en él.

El middleware también debe poder migrar procesos entre servidores con distintas finalidades. Debe ser capaz de balancear la carga, si un servidor está muy cargado de procesos y otro está ocioso, pueden transferirse procesos a este último para liberar de carga al primero y optimizar el funcionamiento. Además debe permitir al administrador realizar un mantenimiento de los servidores, si hay procesos corriendo en un servidor que necesita mantenimiento o una actualización, deberá ser posible migrar los procesos a otro y proceder a desconectar

del cluster al primero. A su vez es el encargado de priorizar los trabajos. En caso de tener varios procesos corriendo en el cluster, pero uno de ellos de mayor importancia que los demás, puede migrarse este proceso a los servidores que posean más o mejores recursos para acelerar su procesamiento.

Las aplicaciones middleware han aparecido de manera relativamente reciente en el mundo de la informática ganando popularidad en la década de los 80 ya que eran la solución innovadora para integrar las nuevas aplicaciones con los sistemas antiguos (legacy systems), en todo caso, el término ha sido usado desde 1968. El uso de middleware también facilita la computación distribuida mediante la conexión de múltiples aplicaciones para crear una aplicación mucho mayor sobre una red.

2.6. Conclusiones sobre la computación de alto rendimiento

El modelo de computación de alto rendimiento es una realidad cada vez más utilizada para la resolución de problemas complejos. Los clusters de computadores y el uso de paradigmas de programación paralela y distribuida han ganado popularidad en los últimos tiempos. La popularidad de los clusters se ve estimulada por varios factores; como los bajos costos de las PC y el incremento en la velocidad y capacidad de tecnologías de comunicación. La escalabilidad es otro factor importante que estimula la utilización de los clusters, ya que si se desea incrementar el poder de cómputo se pueden agregar más nodos sin perder la inversión hecha previamente.

Dentro del ámbito de la Facultad de Ingeniería se han realizado inversiones en la investigación e implementación de clusters y herramientas de cómputo de alto rendimiento. Un ejemplo de estos trabajos es el cluster Medusa que fue llevado a cabo en conjunto entre el Instituto de Mecánica de los Fluidos e Ingeniería Ambiental (IMFIA) y del Centro de Cálculo (CeCal) con el apoyo financiero del Programa de Desarrollo Tecnológico (PDT) en el año 2006.

La programación paralela y distribuida es fundamental para la completa utilización de los recursos que brinda un cluster de computadores. Existen muchas herramientas que facilitan el desarrollo de programas paralelos y distribuidos y dos de las herramientas más utilizadas son las bibliotecas MPI y PVM. Las bibliotecas MPI y PVM contienen subrutinas para enviar y recibir mensajes y manejar la sincronización entre procesos que facilitan el desarrollo de aplicaciones paralelas y distribuidas.

En el proyecto de grado CCADAR se busca la implementación de una herramienta de fácil acceso para los usuarios, que permita la ejecución de trabajos paralelos y distribuidos en un cluster de computadores de forma de aprovechar las ventajas de la programación paralela y distribuida. Para facilitar la ejecución de trabajos paralelos en el cluster el proyecto de grado tiene como objetivo brindar soporte para las bibliotecas MPI y PVM.

Para el manejo de la asignación de recursos y control de la ejecución de trabajos paralelos y distribuidos en el cluster fue necesario contar con un Middleware

especializado en estas funciones. Un Middleware de este tipo se encuentra encargado de recibir las peticiones de ejecución de trabajos en el cluster, realiza la asignación de recursos a los trabajos para su ejecución y prioriza la ejecución de trabajos según reglas predefinidas.

En el siguiente capítulo se investigarán en profundidad las tecnologías necesarias para el proyecto CCADAR como middlewares, herramientas de monitoreo del sistema, etc.

3. Relevamiento de tecnologías

El relevamiento de paquetes de software fue una actividad muy importante dentro del proyecto. Encontrar un conjunto de aplicaciones que permitan la ejecución de programas paralelos y distribuidos fue vital para cubrir el alcance del proyecto. El relevamiento se enfocó en sistemas de administración de cluster, sistemas de despacho de trabajos y sistemas de monitoreo de clusters.

Las características en las cuales se centró el relevamiento de aplicaciones del cluster fueron: su desempeño, la interfaz disponible para el usuario, la interfaz de administración del cluster, la información disponible del cluster y sus nodos individuales, los mecanismos de despacho e inicialización de trabajos y finalmente que se trate de una herramienta de código abierto.

Previamente al relevamiento se estableció GNU/Linux como sistema operativo del cluster, pudiendo tratarse de una distribución como Fedora u openSUSE, y la biblioteca MPI como principal herramienta para el desarrollo de programas paralelos y distribuidos.

En esta sección se presentarán las aplicaciones que fueron relevadas para el despacho de trabajos y administración del cluster, las aplicaciones de monitoreo del cluster, los lenguajes de desarrollo, los manejadores de base de datos relacionales y otras aplicaciones que fueron necesarias para la implementación del proyecto.

3.1. SSI:OpenMosix[1], OpenSSI y Kerrighed

A diferencia de un cluster tradicional en un cluster SSI todas las computadoras vinculadas dependen de un sistema operativo idéntico en común. Un middleware SSI oculta la naturaleza heterogénea y distribuida de los recursos, y los presenta a los usuarios y a las aplicaciones como un recurso computacional unificado y sencillo. Una de las metas que más diferencia un SSI de un cluster o un grid tradicional es su completa transparencia en la gestión de recursos. Es debido a esta completa transparencia que es posible migrar procesos de un nodo a otro. Ni siquiera es necesario programación adicional para beneficiarse del paralelismo, de esta manera no es necesario re-escribir programas utilizando bibliotecas como PVM (Parallel Virtual Machine) o MPI (Message Passing Interface).

A pesar de las ventajas presentadas por los SSI a nivel de manejo de recursos no se cree que se ajusten a los requerimientos planteados en este proyecto principalmente debido a que el cluster ya se encuentra en funcionamiento siguiendo un paradigma estilo Beowulf más clásico. Quizás sea interesante construir un pequeño cluster SSI e investigar más a fondo este tipo de tecnologías de manera de poder comparar ambas soluciones.

[1] Moshe Bar, fundador y director de proyecto de openMosix (uno de los proyectos SSI más influyentes al momento del relevamiento de tecnologías), anunció el 15 de julio del 2007 el cierre del proyecto y que se congelaría tanto su desarrollo como el soporte brindado a partir del 1 de marzo del 2008.

3.2. Condor

3.2.1. Características generales

Condor es un proyecto open-source de la universidad de Wisconsin específicamente diseñado para High-Throughput Computing (HTC). Si bien Condor puede utilizarse con fines de High-Performance Computing (HPC) existen importantes diferencias entre HPC y HTC. Fundamentalmente los trabajos ejecutados en un entorno HPC se caracterizan por necesitar resolver un problema en el menor tiempo posible; en cambio en un entorno HTC los problemas se caracterizan en general por requerir de meses o años de procesamiento en lugar de horas o días. Como consecuencia en HPC el poder computacional de un cluster normalmente es medido en operaciones de punto flotante por segundo (FLOPS), en cambio la comunidad HTC se interesa más en la cantidad de trabajos que es posible completar en un largo periodo de tiempo, sin interesarse especialmente en que tan rápido puede terminarse un trabajo individual. Debido a este perfil de tipo HTC, Condor se enfoca principalmente en la robustez y la confiabilidad brindando funcionalidades como CPU harvesting para la utilización de ordenadores no dedicados en momentos de ocio de los procesadores, migración de procesos, tolerancia a fallas y puntos de recuperación.

Paradójicamente si bien se trata de un proyecto open-source su código fuente no se distribuye libremente. Para obtener el código fuente de Condor es necesario enviar un correo electrónico explicando la razón por la que se quiere obtener y cada petición es analizada puntualmente.

3.2.2. Bibliotecas para programación paralela

Condor se encuentra diseñado principalmente para la ejecución de trabajos seriales. Si bien provee soporte limitado para las bibliotecas PVM y MPI para procesamiento paralelo su desempeño en esta área se encuentra por debajo de otros manejadores de recursos distribuidos como TORQUE o SGE.

3.2.3. Interfaz de programación

Condor brinda una interfaz de programación propietaria, y no soporta el estándar Distributed Resource Management Application API (DRMAA).

3.2.4. Resumen

Condor no parece ser el software ideal para el proyecto. Si bien puede adaptarse a los requerimientos planteados y brinda funcionalidades interesantes (e.g. migración de procesos, cpu harvesting, etc.), existen áreas críticas en las que se encuentra claramente en desventaja con respecto a otros productos. El limitado soporte que brinda Condor a las bibliotecas PVM y MPI resulta ser un problema crítico y determinante para que fuera descartado en el sistema planteado en el proyecto.

3.3. Sun Grid Engine

Sun Grid Engine es un software para administración de recursos distribuidos que dinámicamente asocia requerimientos de hardware y software de los usuarios con los recursos disponibles en la red (generalmente heterogéneos) de acuerdo a políticas predefinidas.

Sun Grid Engine actúa como el sistema nervioso central de un cluster de computadoras conectadas por medio de una red. A partir de algunos procesos en ejecución constante (demonios), el Sun Grid Master supervisa todos los recursos de la red para permitir control completo y alcanzar la utilización óptima de los mismos.

Sun Grid Engine fue desarrollado como realce de Codine de Genias GmbH y Grisdware Inc., de acuerdo a requerimientos de varios clientes tales como el laboratorio de investigación del ejército de Aberdeen y BMW. Con Sun Grid Engine el uso medio de los recursos aumentó de menos del 50 % a más del 90 % en ambos ambientes.

Sun Grid Engine reúne el poder de cálculo disponible en granjas de computadoras dedicadas, servidores conectados y computadoras de escritorio, y las presenta desde un único punto de acceso para el usuario que necesita ciclos de cómputo. Esto se logra distribuyendo la carga de trabajo entre los sistemas disponibles, aumentando la productividad de máquinas y del uso de licencias, mientras se maximiza el número de trabajos que pueden ser completados.

Los requerimientos de hardware para Sun Grid Engine son mínimos (100 MB de memoria disponible y 500MB de disco) y soporta la mayoría de los sistemas operativos populares. Sun Grid Engine también soporta las plataformas SPARC Ultra III, SPARC Ultra IV, AMD64, x86 y Mac.

Sun Grid Engine permite control de usuarios, limitando tanto el número máximo de trabajos por usuario, grupo y proyecto, como recursos tales como colas, hosts, memoria y licencias de software. Los recursos requeridos por cada trabajo se pueden indicar mediante expresiones lógicas (por ejemplo, un usuario puede requerir que un trabajo corra en un host que cumpla la condición "Solaris o Linux pero no Linux en IA64").

La DRMAA es un conjunto de APIs estándar desarrollado por Global Grid Forum para application builders, portal builders e ISV's. La versión 6.1 soporta los últimos C y Java bindings de DRMAA 1.0. Adicionalmente, es provisto de compatibilidad hacia atrás para: DRMAA 0.5 Java binding y DRMAA 0.95 C binding.

Sun Grid Engine guarda la información referida a la cuenta de cada trabajo en una base de datos relacional (soportando Oracle, MySQL y PostgreSQL). Maneja aplicaciones paralelas (MPI o PVM habilitados) a través de una interfase dedicada y permite la distribución de recursos a equipos o departamentos, por ejemplo proporcionalmente a su contribución económica.

El código muestra una complejidad alta, además el sistema es integrado, no muestra facilidades a la hora de dividirlo en subsistemas. Estas características hacen que en algunos aspectos se vea como un sistema cerrado, que no permite adaptaciones en caso de realidades no abarcadas por el mismo.

3.3.1. Resumen

Sun Grid Engine provee la mayoría de las funcionalidades requeridas para el proyecto. Por otro lado, tiene como desventaja el ser un único gran paquete indivisible, lo cual hace muy difícil cualquier adaptación o modificación de su funcionamiento.

Observando la historia reciente del producto, en una versión anterior se encontró un problema de seguridad (un salto en las restricciones) y quienes lo estaban utilizando en ese momento descartaron la idea de buscar el problema y solucionarlo, limitándose solamente a esperar que los creadores de la herramienta lo corrigieran. La decisión de obviar el problema se debe a que la herramienta cuenta con un código muy complejo y difícil de modificar, lo que lo coloca como una opción poco viable para ser utilizada dentro de este proyecto.

3.4. TORQUE

El sistema PBS (Portable Batch System) fue desarrollado a principios de los '90 por MRJ para la National Aeronautics and Space Administration (NASA). PBS es un software que provee controles sobre la iniciación y despacho de trabajos batch; permite la administración y asignación de recursos a los diferentes trabajos permitiendo asegurar a los usuarios que los trabajos obtendrán los recursos necesarios para terminar su ejecución. Del sistema PBS surgen dos aplicaciones: OpenPBS y TORQUE. Ambas fueron relevadas en el proyecto Fenton.

OpenPBS es la versión de código abierto de la versión original de PBS desarrollada para la NASA. Actualmente es ofrecida por Altair Grid Technologies, aunque no se encuentra en desarrollo y la empresa no brinda ningún tipo de soporte sobre el software. Altair ha apostado a una versión comercial del producto llamada PBS Pro, y en lo que a Altair respecta OpenPBS es simplemente una forma de comenzar a utilizar PBS con el objetivo de comprar PBS Pro.

TORQUE (Terascale Open-source Resource and QUEue manager) es una bifurcación open-source de la versión 2.3.12 de OpenPBS mantenido y desarrollado por Cluster Resources. Incorpora numerosas mejoras con respecto al proyecto PBS original provistas por NCSA (National Center of Supercomputing Applications), OSC (Ohio Supercomputer Center), Sandia, PNNL (Pacific Northwest National Laboratory), y otros centros HPC junto con las mejoras desarrolladas por Cluster Resources.

A continuación se verán las características generales de TORQUE. OpenPBS no fue investigado a fondo pues como se mencionó anteriormente no está en desarrollo actualmente y no existe soporte del mismo.

3.4.1. Características generales

TORQUE ofrece apoyo para programar y ejecutar trabajos que requieren el acceso interactivo de los usuarios a la entrada y salida del trabajo durante el tiempo de ejecución. Esto es a menudo necesario para ejecutar debuggers u

otros programas que requieren información como parte de un trabajo que debe tener previsto el acceso a hardware.

TORQUE ejecuta dos scripts administrativos con la ejecución de cada trabajo llamados prólogo y epílogo. El prólogo se ejecuta inmediatamente antes de que se ejecute el trabajo y el epílogo se ejecuta de inmediatamente después. Ambos scripts se ejecutan con privilegios de root y pueden utilizarse para colocar una bandera o establecer parte del entorno de trabajo como la creación de directorios temporales, realizar una limpieza de datos temporales o ejecuciones previas.

En TORQUE los trabajos son manejados de la misma manera que lo hace PBS. Se cuenta con un conjunto de colas de trabajos que definen propiedades generales para los trabajos que contienen (e.g., recursos disponibles, etc.). Al ingresar un trabajo al sistema, éste debe especificar características específicas de sí mismo que ayuden al sistema a planificar su ejecución (e.g., nodos a utilizar, memoria máxima a utilizar, etc.). Con la información de la cola a la que pertenece el trabajo y los datos específicos del trabajo en cuestión el despachador de trabajos decide el momento en que un trabajo comienza a ejecutarse y con que recursos cuenta en cada instante.

TORQUE cuenta con una interfaz de usuario de línea de comando (CLI) como principal manera de interactuar con el sistema. También cuenta con una interfaz gráfica para X-Windows y una biblioteca para desarrollo en C aunque es posible utilizar bibliotecas desarrolladas por terceros para otros lenguajes (e.g., Perl o Python).

Existen tres tipos de nodos en un sistema TORQUE:

Nodo Maestro. Es necesario que exista un nodo maestro en el sistema. El nodo maestro es quien ejecuta el servidor de TORQUE que a su vez administra los nodos de cómputo del cluster. Dependiendo del sistema este nodo maestro puede encontrarse dedicado únicamente a este rol o compartir otros roles.

Nodos Interactivos. Los nodos interactivos proveen un punto de entrada al sistema para los usuarios. En estos nodos los usuarios pueden ingresar tareas al sistema y monitorear su progreso. Los nodos interactivos deben tener comandos disponibles para ingresar trabajos al sistema.

Nodos de cómputo. Los nodos de cómputo son los responsables de la ejecución de los trabajos encolados en el sistema. Son los encargados de iniciar, detener y manejar los procesos de cómputo en el sistema operativo del nodo en cuestión.

3.4.2. Bibliotecas para programación paralela

MPI (Message Passing Interface). El soporte para bibliotecas tipo MPI se encuentra integrado en TORQUE. El sistema puede ejecutar varias implementaciones de MPI: MPICH, MPICH2, OpenMPI, etc. Los recursos utilizados

por procesos paralelos son correctamente registrados y se reporta su uso en el log del PBS.

PVM (Parallel Virtual Machine) Una de las principales desventajas de TORQUE es la dificultad que plantea en el uso de la biblioteca PVM. El soporte para PVM no se encuentra integrado al sistema por lo que debe manejarse cuidadosamente, de lo contrario es posible que la máquina virtual de PVM no libere adecuadamente los recursos asignados a un trabajo luego de finalizada su ejecución.

3.4.3. Interfaz de programación

TORQUE soporta el estándar Distributed Resource Management Application API (DRMAA) al igual que la mayoría de los manejadores de recursos.

3.4.4. Maui

TORQUE contiene la lógica necesaria para llevar a cabo la planificación de trabajos, pero se trata de una lógica muy simple que no resulta adecuada para un ambiente de producción. Básicamente el planificador que se encuentra incorporado a TORQUE maneja los trabajos como una cola FIFO (First-In First-Out). Para mejorar el despacho de trabajos se utiliza Maui.

Maui se enfoca en la planificación de trabajos y deja la problemática de iniciar los trabajos y la interacción con los usuarios a los manejadores de recursos (Distributed Resource Management, DRM por sus siglas en inglés) como OpenPBS, TORQUE, SGE, etc.

Actualmente Cluster Resources ofrece el producto Moab que fue desarrollado a partir de Maui. Moab brinda mejoras en las funcionalidades con respecto a Maui pero se trata de un producto comercial y no es de código abierto.

Requerimientos de hardware. Maui requiere de muy pocos recursos de hardware para su funcionamiento: en clusters de hasta 10 teraflops basta con entre 20 a 50 MB de RAM.

Plataformas soportadas. Maui soporta los sistemas operativos Solaris de SUN, cualquier distribución de GNU/Linux, AIX (Advanced Interactive eXecutive) el sistema UNIX propiedad de IBM, OSF/Tru-64, HP-UX, IRIX, FreeBSD, y otras plataformas de tipo UNIX.

También es posible integrar Maui con los manejadores de recursos distribuidos TORQUE, OpenPBS, PBSPro, Sun Grid Engine (SGE), LoadLeveler, LSF, BProc/Scyld, Scalable System Software (SSSRM) y SLURM.

Características generales. El algoritmo de planificación de Maui presenta características muy interesantes algunas de las cuales se presentarán a continuación.

Fairness. Fairness implica dar igual acceso a los recursos a todos los usuarios del cluster.

Preemption. Otra característica del algoritmo de planificación es que permite suspender trabajos en ejecución para poder ejecutar trabajos con mayor prioridad.

Allocation manager. Maui tiene una interfaz para la interacción con un allocation management externo. Un allocation manager (también conocido como allocation bank o cpu bank) funciona como un banco en el cual la moneda son los recursos del sistema (e.g., procesadores, memoria, etc.) autorizando a los trabajos cierta cantidad de recursos.

Backfill. Backfill es una optimización en el algoritmo de planificación que permite asignar de mejor forma los recursos permitiendo ejecutar algunos trabajos fuera de orden siempre y cuando éstos no retrasen los trabajos de prioridad superior de la cola. Para determinar si un trabajo será retrasado, cada trabajo debe proveer una estimación de cuánto tiempo necesitará para su ejecución. Esta estimación, conocida como límite wallclock, es una valoración del tiempo desde el comienzo del trabajo hasta su final. Es útil sobrestimar levemente el límite de wallclock porque el planificador se puede configurar para eliminar a los trabajos que exceden sus límites del wallclock. Sin embargo, la sobrestimación demasiado grande del tiempo del wallclock de un trabajo evitará que el planificador pueda optimizar correctamente la cola de trabajo. Cuanto más exacto el límite del wallclock, mayor será la posibilidad de que Maui encuentre agujeros en la planificación para comenzar a ejecutar su trabajo con mayor anticipación.

Reservas anticipadas. Las reservas anticipadas permiten que un sitio disponga ciertos recursos para el uso específico de ciertas aplicaciones durante cierto tiempo. El acceso a una reserva dada es controlado por un Access Control List (ACL) que determina quién puede utilizar los recursos reservados. Es importante observar que mientras que un ACL permite qué trabajos particulares utilicen recursos reservados, no fuerzan al trabajo a utilizar estos recursos. Maui procurará utilizar la mejor combinación posible de recursos disponibles, sean éstos reservados o no. Maui puede ser configurado para que ciertos trabajos sean restringidos y que funcionen utilizando solamente recursos reservados, aplicando restricciones a nivel de trabajo o especificando ciertas restricciones especiales de QoS.

Calidad de servicio (QoS). Las funciones de QoS permiten otorgar ciertos privilegios especiales a usuarios. Estos beneficios pueden incluir acceso a recursos adicionales, exclusiones de determinadas políticas, acceso a capacidades especiales y mejoras en la priorización de trabajos.

Fairshare. Este componente permite favorecer trabajos en base al uso histórico a corto plazo. Es posible así ajustar la prioridad de un trabajo dependiendo de la utilización porcentual del sistema de usuarios, grupos, o QoS. Dada una determinada ventana de tiempo sobre la cual se evalúa la utilización de recursos del sistema se determina si está siendo mantenido un cierto balanceo de la asignación de recursos en el cluster.

Interfaz de programación. Maui brinda dos interfases de programación para extender o personalizar el despacho de trabajos:

***Interfaz de extensión* (Extension Interface).** Esta interfaz permite que bibliotecas externas sean conectadas con el servidor de Maui brindando acceso a todos los datos y objetos utilizados por el planificador. Además, permite que estas bibliotecas sobrescriban las principales funciones de Maui.

Interfaz local. Se trata de una interfaz en C enfocada exclusivamente al desarrollo de nuevos algoritmos de despacho.

Estadísticas. Maui almacena tres diferentes clases de estadísticas:

Estadísticas de tiempo real. Las estadísticas de tiempo real son mantenidas en memoria y pueden ser consultadas mediante comandos. El comando 'showstats' provee información detallada por usuario, por grupo, por cuenta o por nodo. Además en cualquier momento estas estadísticas pueden limpiarse utilizando el comando 'resetstats'.

Histórico. Las estadísticas de histórico pueden ser obtenidas para un lapso de tiempo, un tipo de trabajo y/o una porción de recursos utilizando el comando 'profiler'. Este comando utiliza la traza de información detallada de un trabajo, que es guardada al dar por finalizado un trabajo.

Fairshare. Este tipo de estadísticas son mantenidas sin importar si fairshare se encuentra habilitado o no. Al igual que las trazas de los trabajos, las estadísticas son almacenadas en archivos utilizando texto plano por cada ventana de fairshare.

3.4.5. Gold

Gold es un sistema de contaduría open-source desarrollado en Pacific Northwest National Laboratory (PNNL) bajo el proyecto Scalable Systems Software (SSS), que lleva registro y maneja el uso de recursos en clusters de alto desempeño. Actúa de la misma forma que un banco en el que son depositados créditos en cuentas, estos créditos representan recursos en el sistema. A medida que los usuarios consumen recursos en el sistema con la ejecución de trabajos se debitan créditos de sus respectivas cuentas.

En Gold es posible realizar operaciones como depósitos, retiros, transferencias o reembolsos sobre las cuentas del sistema, además, provee listados de balances a usuarios y administradores.

Características generales.

Reservas. Previo al inicio de un trabajo se realiza una estimación del total de recursos que consumirá. En base a la estimación se reservan créditos en la cuenta correspondiente. Los créditos 'reservados' se debitan una vez que el trabajo es terminado, evitando que un proyecto consuma más recursos de los que tiene asignados.

Vencimiento de créditos. Puede especificarse un lapso de validez a los créditos en el sistema, permitiendo que se implemente una política de úsalo-opiérdelo previniendo el uso exhaustivo de créditos acumulados. De esta manera se establecen ciclos de utilización de recursos evitando el uso de recursos excedentes de un ciclo a otro.

Interfaz web. La interfaz web permite el acceso remoto a usuarios y administradores.

Interfaz de programación. Existen diferentes formas de integrar Gold al sistema: Perl API, Java API o directamente utilizando el protocolo SSSRMAP (basado en XML).

3.4.6. Casos de estudio

Varias universidades utilizan con éxito TORQUE y Maui para manejar los recursos de sus clusters. Se presentarán algunos casos.

Oregon State University, Laboratorio de física. La universidad Oregon State University cuenta actualmente con un cluster de 34 computadoras Dell Optiplex GX620's con procesadores Intel Pentium D 830 (3.0 GHz) y 1 GB of RAM con sistema operativo SUSE GNU/Linux 10.1 64-bit.

Las computadoras del cluster cuentan con compiladores Intel para C, C++ y Fortran y la biblioteca Math Kernel Library (cluster edition). Para el procesamiento paralelo se utilizan las bibliotecas MPI. El cluster utiliza TORQUE como manejador de recursos, y desde el 4 de febrero del 2007 se utiliza Maui para la planificación de tareas.

University of Glasgow. La universidad de Glasgow dispone de un cluster que hoy en día cuenta con 60 nodos de procesamiento disponibles, cada uno con procesadores duales Opteron 248 y 2 GB RAM. El cluster utiliza TORQUE y Maui sobre Redhat Enterprise GNU/Linux 3. Además de los componentes

estándares incluidos en la distribución (e.g. compilador gcc, g++, g77, etc.) se han instalado las bibliotecas MPICH 1.2.6 (MPI) para procesamiento paralelo.

University of Heidelberg. Dispone de un cluster instalado a principios de 2002 que consta de 512 procesadores AMD Athlon MP, instalados en 256 nodos de procesamiento SMP con 2 GB de memoria RAM cada uno. Los procesadores funcionan cada uno a 1.4GHz y alcanzan un máximo teórico de 2.4 billones de operaciones de punto flotante por segundo (Gflops). El sistema total indica un máximo teórico de desempeño de mas de 1.4 Teraflops. Las primeras mediciones de desempeño utilizando Linpack Benchmark mostraron un rendimiento de 825 Gflops, ubicando el cluster en la posición 35va del top 500 de supercomputadoras del mundo en Junio del 2002.

Como base del cluster se utiliza un sistema Debian GNU/Linux. Para procesamiento paralelo se utiliza la biblioteca MPICH (MPI) y como manejador de recursos se utiliza TORQUE. Por sobre TORQUE se encuentra instalado Moab, el sucesor comercial de Maui.

Stony Brook University. El cluster tiene 235 nodos de procesamiento dual (470 procesadores individuales). Cada procesador es un Dell Pentium IV Xeon de 3.4GHz con 2 GB de memoria RAM y 40 GB de disco duro. Los computadores operan con Debian GNU/Linux, utilizan TORQUE y Maui para manejar los trabajos y la versión 1.4 de MPI para el procesamiento paralelo.

Dansk Center for Scientific Computing. El cluster consta de 200 nodos Dell PowerEdge 1950 1U con dos procesadores Intel Woodcrest de 2,66 GHz. De estos 200 nodos 160 cuentan con 4 GB de RAM y 40 cuentan con 8 GB de RAM. Como sistema se utiliza openSUSE GNU/Linux 10.1 con kernel 2.6 y para la planificación y manejo de trabajos se utiliza TORQUE 2.1.2 y Maui 3.2.6. Para el procesamiento paralelo se disponen de bibliotecas tanto de tipo MPI como de tipo PVM.

3.4.7. Resumen

TORQUE, junto con Maui y Gold, satisfacen los requerimientos planteados para el proyecto. Su consumo de recursos del cluster es relativamente bajo, se adapta muy bien al manejo de clusters pequeños y posee un manejo eficiente de trabajos paralelos (sobre todo si estos son homogéneos).

Existen ciertos aspectos en los que TORQUE no se desempeña de la mejor manera. Si bien se encuentra integrado el soporte para bibliotecas tipo MPI, debe tenerse mucho cuidado al momento de ejecutar trabajos utilizando PVM. TORQUE presenta carencias también en sus funcionalidades de gerenciamiento de recursos; e.g. no es capaz de realizar CPU Harvesting o migración de procesos.

El soporte por parte de la comunidad de usuarios de TORQUE y Maui por medio de listas de correo es muy bueno y también se encuentra mucha documentación disponible en el sitio web por medio de una Wiki (aunque aún se encuentra en construcción).

3.5. Ganglia

Ganglia es un sistema de monitoreo, escalable y distribuido, para sistemas de alto desempeño como clusters de computadores o grids. Es un proyecto de código abierto y se encuentra disponible para una amplia cantidad de sistemas operativos: Linux, BSD, Solaris, Windows, Darwin, etc.

Ganglia utiliza tecnologías ampliamente difundidas como XML para la representación de datos, XDR para la comunicación y RRDtool para el almacenamiento y visualización de datos. El proyecto posee un amplio énfasis en consumir el mínimo de recursos posibles por nodo y brindar un máximo de concurrencia. Posee una implementación robusta y puede manejar clusters con hasta 2000 nodos.

Ganglia se encuentra integrado por tres componentes:

GMonD GMonD (Ganglia Monitoring Daemon) es un servicio encargado del monitoreo de un nodo en particular y debe estar instalado en cada una de las máquinas que se desee monitorear. Este servicio recolecta métricas del sistema en base a CPU, memoria, disco duro, tráfico de red y utiliza un protocolo simple vía XDR para compartir esta información mediante XML sobre TCP.

GMetaD El metaservicio de Ganglia (Ganglia Meta Daemon) es un servicio que se encarga de recolectar y almacenar información de otros servicios (ya sean servicios GMonD u otros servicios GMetaD). GMetaD provee un mecanismo de consultas muy simple para obtener información histórica acerca de grupos de máquinas.

Interfaz web Ganglia brinda además un front-end web implementado en PHP para la presentación al usuario final de la información almacenada por el sevicio GMetaD.

3.6. Apache

Apache es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, etc. Desde 1996, Apache es el servidor HTTP más utilizado en la Internet. Alcanzó su máxima cuota de mercado en 2005 cuando llegó a ser utilizado por el 70 % de los sitios web en el mundo. Sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años y hoy en día brinda servicio al 49 % de los sitios de Internet[2].

Entre las ventajas que brinda Apache se puede destacar su enfoque altamente modular y un excelente soporte gracias a su gran cantidad de usuarios. Además, gracias a su enfoque altamente modular, Apache soporta una gran cantidad de lenguajes server-side como: PHP, Python, Ruby, Perl, etc.

[2]Según los datos de la encuesta realizada en Junio del 2008 por Netcraft.com. Netcraft fue fundada en 1995 y se especializa en el análisis de mercado de servidores y alojamiento web incluyendo detección de servidor web y sistema operativo.

3.7. PHP

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (aunque inicialmente significaba Personal Home Page). Fue creado por Rasmus Lerdof en 1994; sin embargo la implementación principal de PHP es producida actualmente por The PHP Group y sirve como el estándar de facto para PHP al no existir una especificación formal del lenguaje. PHP es publicado bajo la licencia PHP, la cual es considerada por la Free Software Fundation (FSF) como software libre.

PHP es un lenguaje de propósito general e interpretado, muy popular debido a la rapidez y facilidad con la que permite desarrollar aplicaciones. Fue diseñado originalmente para la creación de páginas web dinámicas pero actualmente puede ser utilizado desde una interfaz de línea de comandos o desde una interfaz gráfica usando las bibliotecas Qt o GTK+. Cuenta con una comunidad de usuarios muy amplia y se encuentra disponible para su instalación en la mayoría de los servidores web en casi cualquier plataforma. Se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, aunque el número de sitios implementados en PHP ha declinado desde agosto de 2005.

Algunos de los puntos a destacar de PHP son: su facilidad de uso, su amplia documentación que incluye muchos ejemplos prácticos, que brinda soporte al paradigma de programación de orientación a objetos y que posee manejo de excepciones (aunque algunas de estas ventajas han aparecido recientemente en las últimas versiones de PHP).

Como mayor desventaja se presenta generalmente su característica de favorecer la creación de código desordenado y complejo de mantener. Esta desventaja tiene dos orígenes, por un lado se debe a que PHP brinda tantas facilidades en su uso que es muy popular entre usuarios programadores con poca formación técnica, y por otro lado la falta de una especificación formal resulta en algunos casos en una definición poco homogénea de las bibliotecas proporcionadas por el lenguaje.

3.8. PostgreSQL

PostgreSQL es un servidor de base de datos relacional con una trayectoria de más de 15 años de desarrollo activo. Se trata de un proyecto de código abierto liberado bajo la licencia BSD y desarrollado por una comunidad de desarrolladores y organizaciones comerciales denominada PGDG (PostgreSQL Global Development Group).

PostgreSQL comenzó con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos de implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en Ingres y de haber tenido amplia experiencia comercial, Michael decidió volver a la Universidad para trabajar en un nuevo proyecto sobre la experiencia de Ingres. Dicho proyecto fue llamado Post-Ingres o simplemente Postgres.

PostgreSQL se encuentra disponible para una gran cantidad de sistemas

operativos incluyendo GNU/Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Una característica interesante que presenta PostgreSQL es su capacidad de acceso concurrente a la información, MVCC por sus siglas en inglés (Multi-Version Concurrency Control). Mediante MVCC PostgreSQL permite el acceso de lectura a una tabla mientras que otro proceso realiza una operación de escritura sobre esa misma tabla. Este comportamiento se debe a que cada usuario obtiene una visión consistente de la última versión actualizada. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases y elimina la necesidad del uso de bloqueos explícitos.

PostgreSQL realiza un gran énfasis en el seguimiento de los estándares ANSI/ISO SQL y su implementación de SQL se respeta muy fuertemente los estándares ANSI/ISO SQL-92 y 99.

3.9. MySQL

MySQL es un sistema de gestión de base de datos relacional. MySQL AB[3] desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la licencia pública GNU para software de código abierto, pero las empresas que quieran incorporarlo a productos que no poseen código abierto deben comprar a la empresa una licencia específica para uso comercial.

El manejador de base de datos MySQL se ha convertido en el sistema de gestión de base de datos relacional más popular, principalmente debido a su gran difusión en Internet en el desarrollo de sitios web dinámicos.

MySQL no posee un enfoque purista en su implementación como lo hace PostgreSQL, si bien MySQL respeta en su mayoría los estándares '92 y '99 no es una meta principal del producto.

[3]Desde enero de 2008 una subsidiaria de Sun Microsystems.

4. Cluster con acceso remoto

En esta sección se describirá el sistema Fenton, se profundizará en las tecnologías elegidas en la etapa de relevamiento e investigación para la implementación del gerenciamiento de recursos (DRM) y el despacho de trabajos. Se detallará la construcción de la interfaz para acceso remoto al cluster y además se mencionarán las herramientas utilizadas para el desarrollo y la documentación a lo largo del proyecto. Cabe destacar que durante todo el proceso de desarrollo del sistema se utilizaron exclusivamente herramientas de código libre (OSS).

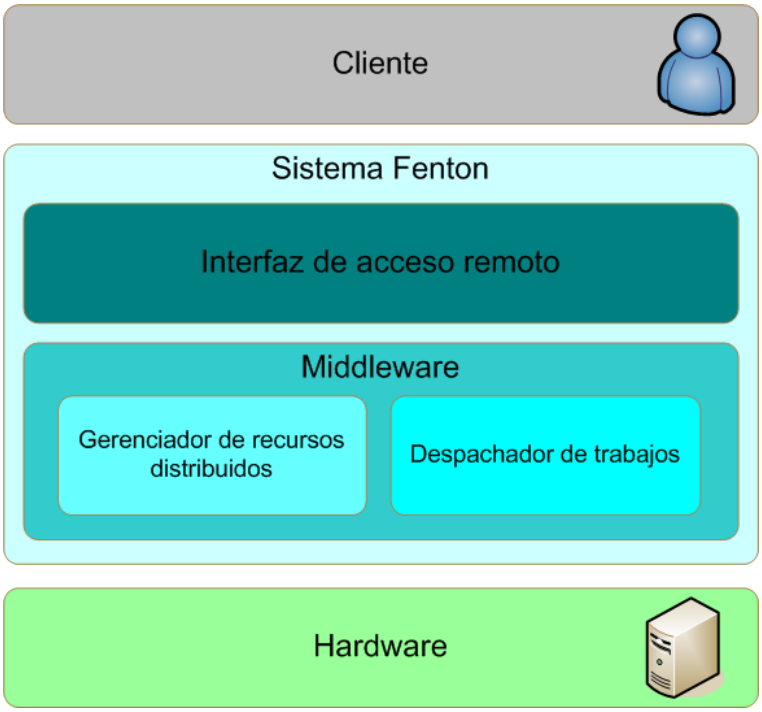


Figura 12: Arquitectura del sistema.

4.1. Gerenciador de recursos y despachador de trabajos

A continuación se presentarán y justificarán las tecnologías seleccionadas para la implementación del gerenciador de recursos (DRM) y despachador de trabajos del sistema Fenton. La etapa de relevamiento de tecnologías fue una de las más importantes del proyecto, y en la implementación del gerenciador de recursos y el planificador de trabajos radica uno de los mayores riesgos para el proyecto.

Luego de evaluar diversas tecnologías para la implementación de clusters se optó por utilizar TORQUE como DRM y Maui como despachador, dejando

abierta la posibilidad de anexar Gold para la contabilización de recursos. Existen varios casos de estudio donde es posible comprobar que TORQUE resulta muy adecuado para la gestión de clusters de pequeño a mediano porte. Tanto TORQUE como Maui requieren pocos recursos de hardware para su ejecución, ventaja que resulta muy importante en un cluster de reducidas dimensiones como el que se desea utilizar.

TORQUE es un derivado de PBS [4] por lo que cuenta con un código base muy estable y sólido. Estas características ayudan a mitigar los riesgos presentados por tecnologías emergentes que ofrecen funcionalidades innovadoras pero que también resultan problemáticas en un proyecto de corta duración. TORQUE y Maui son productos de relativamente pequeño tamaño, esto facilita su comprensión y por lo tanto su modificación y adaptación a necesidades particulares de investigación.

Si bien TORQUE brinda todas las funcionalidades necesarias para satisfacer los requerimientos del proyecto, también presenta ciertas carencias dentro de las cuales las más importantes son: el uso de ciclos de procesador no aprovechados (CPU harvesting) y la migración de procesos. Estas carencias no son un problema importante ya que el cluster cuenta con nodos homogéneos y dedicados, con capacidades de procesamiento similares y que no serán compartidos con otras aplicaciones externas al cluster (e.g., aplicaciones de escritorio).

Como se mencionó en la sección de relevamiento de tecnologías, si bien TORQUE contiene la lógica necesaria para llevar a cabo la planificación de trabajos se trata de una lógica muy básica. Por esta razón se optó por incorporar a Maui para este propósito. Maui presenta un algoritmo de planificación con diversas características y configuraciones que permiten mejorar la performance del cluster y además brinda la posibilidad de implementar un algoritmo de planificación desarrollado a medida.

Para el monitoreo del sistema se decidió utilizar la herramienta Ganglia. Ganglia ya era utilizado como herramienta de monitoreo por el grupo CeCal y luego de evaluar sus funcionalidades se optó por continuar con su utilización debido su excelente desempeño, su bajo consumo de recursos y sus reportes en tiempo real e históricos del uso del cluster.

4.2. Desarrollo de la interfaz de acceso remoto

A continuación se detallará el componente de acceso remoto del sistema Fenton. Un requerimiento importante del sistema es brindar una interfaz de usuario simple y de fácil acceso para usuarios no especializados. Por esta razón, y para evitar problemas con la posible heterogeneidad de los sistemas clientes, se decidió brindar una interfaz de usuario de tipo web para el acceso remoto al cluster.

El desarrollo del componente de acceso web requirió la definición de varias herramientas: un servidor web para alojar la aplicación, un lenguaje server-side para implementar la lógica y un manejador de bases de datos relacionales

[4] PBS es un DRM que fue desarrollado a mediados los años '90 para manejar recursos de computo aeroespacial en la NASA.

para almacenar usuarios, trabajos y resultados de ejecuciones. La utilización de TORQUE y Maui para el manejo de recursos y la planificación de trabajos del cluster nos condiciona a trabajar sobre un sistema operativo GNU/Linux o Unix-BSD. Por esta razón se decidió utilizar Apache como servidor web de la interfaz de usuario.

Teniendo en cuenta los conocimientos previos de los integrantes del grupo y la curva de aprendizaje de los diferentes lenguajes para desarrollar aplicaciones web, se decidió utilizar PHP como lenguaje de implementación. El servidor web Apache brinda un excelente soporte para la ejecución de código PHP por lo que resultan una pareja ideal. Como entorno de desarrollo para la implementación de la lógica en código PHP necesaria se utilizó Eclipse con el plugin PHPEclipse.

Para el almacenamiento de datos se consideraron dos motores de base de datos relacional: PostgreSQL y MySQL. Finalmente se seleccionó trabajar con PostgreSQL, pero igualmente ambos cumplen los requerimientos del proyecto. PostgreSQL es un motor de base de datos relacional de tipo empresarial, maduro, con buena integración con PHP y un muy buen soporte por parte de la comunidad. Si bien PostgreSQL no es tan amigable ni tan popular como MySQL, cuenta con una licencia de tipo BSD mucho más flexible y permisiva que la licencia utilizada por MySQL.

A la hora de la implementación se realizó una clara separación entre la capa de presentación y la capa con la lógica de negocios desacoplando la presentación de la información con el comportamiento de la aplicación. El acceso a datos se encapsuló en un módulo específico, esto redundó en una ventaja para el producto: para utilizar otro manejador de base de datos relacional de condiciones similares solamente es necesario reimplementar el módulo y repetir la etapa de verificación. La apariencia del sitio también podía variar durante la implementación, por lo tanto además de separar la lógica de la aplicación de la capa de presentación, esta última se dividió por un lado en scripts PHP que reciben los pedidos del usuario y actúan en consecuencia, y por otro lado en archivos HTML y CSS encargados de definir los componentes visuales de la interfaz. Para esto se utilizaron plantillas que son cargadas en cada script PHP para luego ser desplegadas en pantalla. En definitiva se obtuvo una versión moderada del patrón Model-View-Controller (MVC). Si en algún momento se desea rediseñar el sitio manteniendo el mismo funcionamiento solamente es necesario modificar las hojas de estilos (CSS) y las plantillas en formato HTML[5].

[5]Realmente estas plantillas no son HTML puro ya que cuentan con etiquetas especiales que determinan la posición del contenido dinámico en la página.

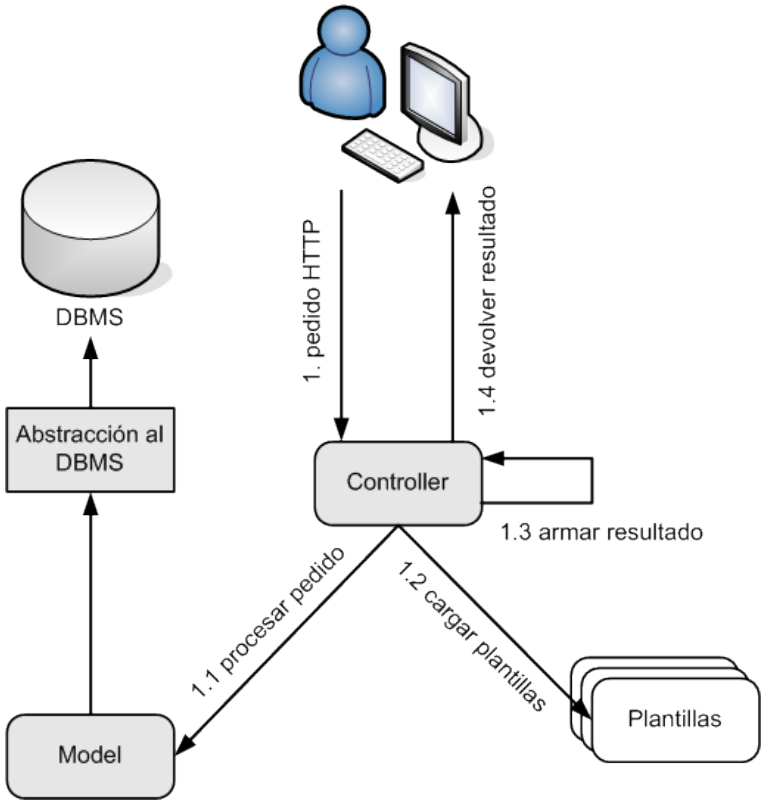


Figura 13: Diagrama de interacción.

Se separaron en módulos distintos las interfaces con TORQUE, Maui y el sistema operativo. Las interfaces con TORQUE y Maui brindan al usuario administrador información sobre el estado del sistema y permiten que este realice acciones sobre los trabajos (e.g., iniciar, detener, etc.). La interfaz con el sistema operativo es utilizada para almacenar los trabajos con el código fuente, archivos binarios, ejecutables y los archivos conteniendo los resultados de los trabajos ejecutados por los usuarios.

La aplicación permite un manejo de usuarios con perfiles diferenciados. Se realizó una clara separación entre usuarios administradores y usuarios normales, pero es posible ejercer diferencias más sutiles entre ellos. Se diseñó un sistema de grupos de usuarios con funcionalidades asociadas, cada usuario puede pertenecer a uno o más grupos, y estos a su vez pueden tener funcionalidades y trabajos asociados. Un usuario determinado puede hacer uso de una funcionalidad dada si pertenece a algún grupo que se lo permita, mientras que puede acceder a un trabajo determinado si pertenece a algún grupo con acceso al mismo.

La interfaz de usuario varía notablemente dependiendo si el usuario es un usuario administrador o un usuario normal. Si bien ambos usuarios tienen cierto conocimiento sobre el uso de aplicaciones, el usuario normal no tiene por qué

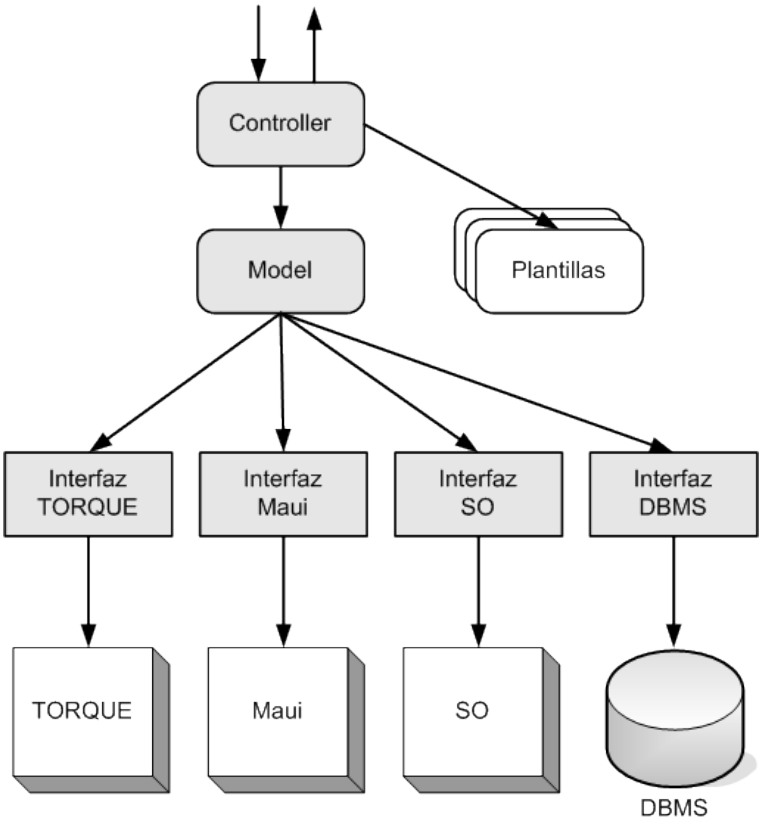


Figura 14: Diagrama de dependencias.

estar interiorizado con el funcionamiento del cluster, por lo tanto se le presenta el sistema como un todo. Por otro lado el usuario administrador podrá desde su propia interfaz administrar los recursos del cluster, conocer el estado de los nodos del cluster e interactuar con ellos para solucionar posibles problemas o mejorar su performance. El usuario administrador tendrá a su cargo la tarea de mantener los usuarios y grupos del sistema así como el acceso de los mismos a los distintos trabajos.

Dado que se trata de un ambiente multiusuario, la aplicación permite administrar las reservas del cluster para organizar la utilización de recursos por parte de varios usuarios de forma simultánea. Cada trabajo creado tendrá asociado un conjunto de recursos disponibles que determinarán el posterior funcionamiento de las ejecuciones de los trabajos por parte de los usuarios. Para esta implementación se restringió el uso de espacio de almacenamiento en el disco duro, tiempo máximo de ejecución y cantidad de nodos disponibles simultáneamente, pero es posible agregar otros recursos a restringir (e.g, memoria, ciclos de procesador, etc.).

El usuario tendrá la posibilidad de subir un proyecto al cluster, compilarlo, ejecutarlo y consultar la salida a pantalla en tiempo real. Dados los extensos tiempos de ejecución de los programas que se corren en clusters de estas características, el usuario puede abandonar el sitio para luego volver en otro momento y verificar el estado de la ejecución. Si el programa continúa en ejecución podrá ver la salida a pantalla (nuevamente en tiempo real) junto con las salidas anteriores de la misma ejecución. Para los trabajos terminados existe la posibilidad de ver la salida en pantalla o descargarla en un archivo.

Los usuarios administradores cuentan con un conjunto de reportes estadísticos sobre los recursos del cluster. Es posible consultar el porcentaje total de carga de CPU y memoria, los tiempos de procesamiento por usuario y por proceso, etc. La generación de los reportes estadísticos se realiza con información brindada por TORQUE, Maui y Ganglia; es posible detallar la información por nodo del cluster, usuario o proceso.

Por último, se definió un mecanismo de alertas a los usuarios que permite enviar alertas a los diferentes usuarios del sistema, como ser inicio y finalización de trabajos, excesos de recursos del cluster, etc. Este mecanismo es totalmente extensible en cuanto a las definiciones de alertas y es soportado por funcionalidades de TORQUE y la base de datos. Un usuario puede observar las alertas que le fueron enviadas por Fenton en la interfaz web y se encuentra disponible la posibilidad de enviar las alertas por correo electrónico mediante un servidor SMTP. Este sistema de alertas debe ser disparado periódicamente por el sistema operativo (e.g., en GNU/Linux mediante el utilitario cron o anacron).

4.3. Otras herramientas

En el transcurso del proyecto fueron necesarias otras herramientas para algunas tareas específicas. Fue necesario contar con una herramienta colaborativa que permitiera compartir y versionar tanto el código como los documentos. Para esto se utilizó Google Code, un servicio de Google que brinda alojamiento a

proyectos de software. Algunas de las funcionalidades que ofrece Google Code son: control de versiones utilizando Subversion, un manejador de incidencias, una Wiki, etcétera.

Subversion (también conocido como SVN) es un software de control de versiones diseñado específicamente para reemplazar al popular Concurrent Versioning System (CVS). Los sistemas de control de versiones mantienen un seguimiento de los cambios realizados al código fuente de un proyecto y permiten la colaboración de varios desarrolladores. Dick Grune desarrolló CVS en 1984 y fue liberado bajo una licencia pública general GNU ganando mucha popularidad entre los proyectos open source. Lamentablemente CVS posee muchas deficiencias, por esta razón CollabNet Inc. comenzó en el año 2000 con el desarrollo de Subversion. Subversion fue liberado bajo una licencia Apache por lo que es considerado software libre y desde entonces se ha vuelto muy popular en la comunidad open source y en ambientes corporativos.

Para la documentación del proyecto se utilizó LyX, un programa gráfico que permite la edición de texto usando LaTeX. LyX hereda todas las capacidades de LaTeX (notación científica, edición de ecuaciones, creación de índices, etcétera) y por supuesto mantiene el enfoque WYSIWYM de LaTeX.

4.4. Verificación de la solución

Una de las actividades del proyecto fue la verificación de las tecnologías propuestas y de la interfaz web desarrollada. Los objetivos de esta etapa son la verificación y validación de los requerimientos funcionales establecidos en etapas más tempranas y que forman parte de los objetivos del proyecto.

Durante el proceso de verificación se aplicaron dos tipos de test, se realizó un test funcional complementado por un test exploratorio. El test funcional verifica el cumplimiento de los requerimientos funcionales contra los casos de test, en un entorno que emula un ambiente de producción del sistema. El objetivo de este test es verificar si el comportamiento observado del software a prueba coincide o no con sus especificaciones.

El test exploratorio es un proceso simultáneo de exploración y aprendizaje del producto, diseño y ejecución de pruebas. La estrategia en este caso fue recorrer las funcionalidades de la interfaz en busca de defectos de diseño, de programación, de funcionamiento, etc. Durante el test exploratorio se generaron una serie de casos de test que apuntan a cubrir los requerimientos funcionales principales del sistema. Estos casos de test se incluyeron en la Guía de Test (ver página 59), pretendiendo guiar la verificación en posteriores ocasiones. El mismo incluye casos de test y escenarios de pruebas que integran los casos de test, así como una descripción del proceso ejecutado en cada caso.

Las herramientas utilizadas en la verificación fueron la guía de test y las funcionalidades provistas por el Google Code como son registro de incidencias y Wiki.

5. Puesta en producción

Finalizada la descripción del trabajo realizado, en esta sección se presentarán las instalaciones y puestas en producción que se han realizado del sistema Fenton. Se describirán las características de hardware y software de los clusters de computadores de alto desempeño en los que fue instalado Fenton. Por último se especificará el proceso que se siguió durante la instalación del sistema.

5.1. Cluster de computadores del CeCal

Como primer caso se presentará el cluster de computadores de alto desempeño con el que cuenta el CeCal dentro del InCo. Este cluster es utilizado internamente por los integrantes del grupo de Computación de Alta Performance del CeCal como ambiente de investigación y desarrollo.

Se trata de un cluster homogéneo que cuenta con cuatro nodos: `lennon.fing.edu.uy`, `marley.fing.edu.uy`, `hendrix.fing.edu.uy` y `joplin.fing.edu.uy`. Cada nodo cuenta con un procesador AMD Athlon 64 3000+, 1 GB de memoria RAM, sistema operativo openSUSE 10.1 y se encuentran conectados entre si mediante una red Ethernet de 100 Mbps.

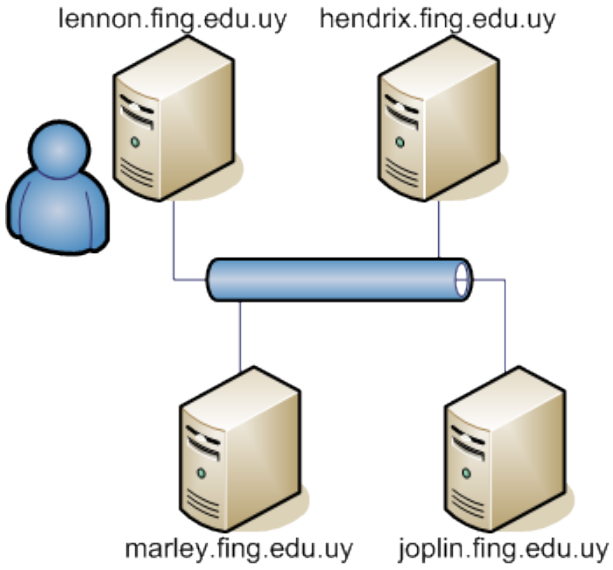


Figura 15: Cluster de computadores del CeCal.

En el marco del proyecto se realizó una primera instalación de Fenton en este cluster. Inicialmente se utilizó para realizar una validación de las tecnologías elegidas en un ambiente controlado pero a medida que el proyecto fue madurando y se fueron cumpliendo etapas de su desarrollo se realizaron actualizaciones del software instalado en el cluster.

A continuación describiremos con mayor detalle la evolución del cluster del CeCal.

Primer hito. En esta primera etapa se instalaron solamente TORQUE y Maui ya que el cluster contaba previamente con algunas bibliotecas para el desarrollo de aplicaciones paralelas como MPICH y PVM. Como nodo maestro se utilizó a `lennon.fing.edu.uy`, en este nodo se instaló Maui y se instalaron el servidor y el demonio de ejecución de TORQUE. De esta manera el nodo `lennon.fing.edu.uy` podrá ser utilizado como nodo maestro y como nodo de cómputo, en cambio en el resto de los nodos se instaló solamente el demonio de ejecución de TORQUE ya que solo se utilizarán como nodo de cómputo.

Segundo hito. A medida que el sistema fue evolucionando se instalaron otras aplicaciones y para cuando el sistema llegó al siguiente hito se realizó una actualización del cluster y se instalaron PHP y PostgreSQL. Nuevamente PostgreSQL se instaló solamente en `lennon.fing.edu.uy` y si bien fue necesario instalar PHP en todos los nodos del cluster, el nodo `lennon.fing.edu.uy` es quien brinda el servicio de interfaz web. El resto de los nodos solamente requieren PHP para ejecutar scripts de notificación de inicio y finalización de trabajos que son manejados internamente por el sistema.

Tercer hito. Durante la iteración final de desarrollo se actualizó una vez más el cluster con la última versión del sistema y se instaló además Ganglia. Ganglia cuenta con dos componentes: un demonio encargado de recolectar información del estado de los nodos del cluster y un demonio encargado de monitorear un nodo individual y enviar información al demonio de recolección. Además Ganglia cuenta con una interfaz web en donde se puede consultar la información recogida por el servicio de recolección. En el nodo `lennon.fing.edu.uy` del cluster se instalaron: la interfaz web, el servicio de recolección y el servicio de monitoreo. En el resto de los nodos se instalaron solamente demonios de monitoreo. Finalmente como último paso del tercer hito, se instaló la biblioteca OpenMPI para brindar mayor flexibilidad a la hora del desarrollo de aplicaciones paralelas.

Con esta distribución de los servicios en el cluster el nodo maestro actúa como front-end permitiendo que el resto de los nodos puedan ser privados e invisibles al usuario del cluster. Además debido a los reducidos requerimientos de software de los nodos de cómputo resulta relativamente simple agregar o quitar nodos del cluster.

Actualmente el cluster cuenta con la versión final de Fenton y se encuentra disponible para su utilización en producción. La interfaz web del sistema Fenton se encuentra disponible desde dentro del InCo en `http://lennon.fing.edu.uy/pgccadar`.

Figura 16: Acceso al sistema Fenton.

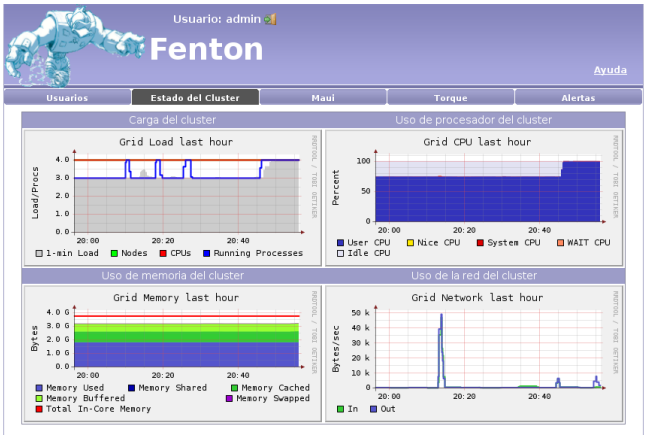


Figura 17: Estado del sistema Fenton.

5.2. Cluster de computadores Medusa

El cluster de computadores de alto desempeño Medusa fue la motivación inicial del proyecto. Este cluster es mantenido en conjunto por el IMFIA y el CeCal, con el apoyo financiero del Programa de Desarrollo Tecnológico. El cluster fue construido en el año 2006 y está constituido por seis servidores modelo Sun Fire X2100 x64 interconectados entre si mediante una red Ethernet.

Cada nodo Sun Fire X2100 x64 posee un procesador AMD Opteron doble núcleo modelo 175, cada uno de estos núcleos trabaja a 2.2 GHz y tiene 1 MB de cache nivel 2. Además cada nodo cuenta con una memoria de 2 GB ECC PC3200 DDR-400, un disco de 80 GB de capacidad y 7.200 RPM SATA y dos interfaces de red Ethernet 10/100/1000. El nodo utilizado como maestro cuenta

además con un disco duro SATA 2 adicional con una capacidad de 250 GB. Todos los nodos utilizan GNU/Linux Fedora Core 5 x86 SMP como sistema operativo.

Medusa fue implementado en el marco del proyecto "Laboratorio de simulación numérica de flujos a superficie libre" y algunas de las aplicaciones para las cuales se ha utilizado el cluster son: la ejecución de simulaciones para el desarrollo de puentes, barcazas, prevención de efectos ambientales no deseables por buques de gran calado, etc.

Se realizó la instalación del sistema Fenton en el cluster Medusa. Para esto se incorporó un nuevo nodo (medusa10) al cluster que desempeñará el papel de nodo maestro. En este nodo se instaló el sistema Fenton en su totalidad, pero aún se encuentra pendiente la instalación del sistema en los restantes nodos del cluster.

6. Conclusiones

La computación de alto rendimiento es fundamental para la resolución de problemas con un alto grado de complejidad computacional, estos problemas provienen de disciplinas como matemática, física o química y no se encuentran necesariamente relacionados con la informática. Muchos de los usuarios de clusters de computadores de alto rendimiento no poseen una formación en informática y su uso les presenta una importante dificultad técnica. En el primer capítulo de éste documento se describen como objetivos principales del proyecto de grado la construcción de un cluster de computadores de alto rendimiento y la necesidad de facilitar el acceso al mismo a usuarios no especializados en informática.

Durante su desarrollo, éste proyecto de grado permitió explorar a fondo la problemática de la computación de alto rendimiento mediante etapas de aprendizaje e investigación de tecnologías. Como parte del aprendizaje los estudiantes cursaron la materia 'Computación de Alta Performance' dictada por el grupo CeCal del Instituto de Computación de la Facultad de Ingeniería. Esta materia introdujo a los estudiantes a los conceptos principales de la computación de alto rendimiento y a la programación paralela y distribuida. Por otro lado se realizó en el proyecto de grado una etapa de investigación de tecnologías que resultó muy importante y positiva para el desarrollo del proyecto. En esta etapa se investigaron las herramientas de código abierto para la implementación de clusters de computadores; sus características, sus funcionalidades y sus limitaciones.

Una de las primeras aplicaciones investigadas fue Condor, un sistema diseñado fundamentalmente para High-Throughput Computing (HTC) pero que puede adaptarse a los requerimientos planteados en el proyecto de grado. Condor brinda funcionalidades interesantes como migración de procesos entre nodos y la capacidad de aprovechar ciclos ociosos de procesador (CPU Harvesting). Sin embargo Condor ofrece un soporte muy básico a las bibliotecas PVM y MPI para programación distribuida. Otra aplicación investigada fue Sun Grid Engine, un software para administración de recursos distribuidos actualmente desarrollado por Sun Microsystems. Sun Grid Engine provee las funcionalidades requeridas para el proyecto de grado, pero presenta como desventaja ser un único gran paquete indivisible y muy complejo, lo cual hace muy difícil cualquier tipo de adaptación o modificación a su funcionamiento. Por último se investigó TORQUE, un software para administración de recursos distribuidos con funcionalidades muy similares a las presentadas por Sun Grid Engine desarrollado por Cluster Resources Inc. A diferencia de Sun Grid Engine, TORQUE es un software de reducidas dimensiones enfocado específicamente en la administración de recursos que no contiene la lógica necesaria para la planificación de la ejecución de trabajos en un cluster de computadores. Cluster Resources Inc. desarrolla también Maui, un software especializado en la planificación y despacho de trabajos en clusters de computadores que resulta la pareja ideal para TORQUE. TORQUE y Maui brindan además un excelente soporte de forma integrada a la biblioteca MPI para programación paralela y distribuida. TORQUE junto con Maui proveen las funcionalidades requeridas para el proyecto de grado y

además presentan un bajo consumo de memoria, característica que resulta muy importante en clusters pequeños.

Como se detalló en el cuarto capítulo de este documento, TORQUE y Maui fueron las herramientas seleccionadas para la administración de recursos y despacho de trabajos en el cluster. TORQUE y Maui permitieron construir una sólida plataforma de ejecución de trabajos, y fue a partir de esta plataforma que se desarrolló una interfaz web para el acceso remoto al sistema. Ésta interfaz web de acceso al sistema permite una utilización amigable del cluster por parte de usuarios no especializados, además permite a usuarios técnicos la administración de los recursos del cluster, la administración de la seguridad del cluster y ofrece un conjunto de reportes históricos y en tiempo real sobre el uso de los recursos en el cluster. Para la generación de estos reportes se integró la interfaz web del sistema con Ganglia, un sistema de monitoreo escalable y distribuido para sistemas de alto rendimiento.

Podemos decir que el resultado de éste proyecto de grado fue muy positivo y los objetivos mencionados en el primer capítulo del documento se alcanzaron en buena forma. Esto permite que hoy en día el sistema Fenton se encuentre en producción y listo para su utilización.

7. Trabajo futuro

Como se mencionó en la sección anterior este proyecto deja planteados algunos trabajos a futuro. A continuación se enumerarán los trabajos más relevantes a implementar y/o investigar pero que dados los tiempos del proyecto no fueron incorporados a la solución desarrollada.

- Integración de Gold con el sistema para permitir contabilizar los recursos disponibles en el cluster y limitar los recursos utilizados cada usuario.
- Aumentar la características y opciones de monitoreo para chequeo preventivo (hoy en día solo se chequea espacio en disco).
- Trabajar en la integración de biblioteca PVM al sistema.
- Profundizar en el fine-tunning de TORQUE y Maui para obtener un desempeño óptimo del cluster.
- Mejorar la integración del sistema con Ganglia para utilizar más de las funciones que provee.
- Mejorar los reportes de los usuarios administradores para el análisis del uso del sistema y de los datos históricos que son provistos por TORQUE y Maui.
- Continuar con la instalación del sistema Fenton en el cluster Medusa (ver página 54).
- Incorporar el uso de un addon de Ganglia llamado Job Monach. Este addon permite a Ganglia comunicarse con TORQUE y recopilar información sobre nodos, trabajos y colas de ejecución disponibles. Ganglia genera gráficas y mantiene un historial con la información recopilada de TORQUE al igual que con el resto de las métricas que maneja.
- Modificar la comunicación entre el sistema Fenton y TORQUE para utilizar la API DRMAA. TORQUE brinda una implementación en lenguaje C del estándar DRMAA disponible para el usuario programador. Si bien esta es una posibilidad, otra opción más accesible podría ser la utilización de la biblioteca pbs_python. Esta biblioteca encapsula la biblioteca implementada en C de TORQUE y permite una comunicación más simple con TORQUE utilizando Python.

A. Documentos anexos

Glosario

Este documento contiene un conjunto de definiciones y abreviaturas pertinentes a las áreas de estudio relacionadas con el proyecto. Con este documento se busca proporcionar el entendimiento adecuado de estas terminologías. Ver documento `PGCCADAR-Glosario`.

Manual de instalación

Este documento sirve como guía de instalación de las tecnologías propuestas en el proyecto, instalación de la interfaz de acceso remoto desarrollada y creación del ambiente de trabajo inicial del sistema. Ver documento `PGCCADAR-GuiaDeTest`.

Manual de usuario y administrador

Estos documentos presentan y describen la utilización del sitio Web desarrollado. El primer manual se encuentra dirigido a las actividades de los usuarios-administradores y se encuentra enfocado a las funcionalidades de monitoreo y control. El segundo manual se encuentra dirigido a los usuarios-clientes y cubre funcionalidades como la ejecución de trabajos en el sistema, la obtención de resultados de una ejecución, etc. Ver documentos `PGCCADAR-ManualDeUsuario` y `PGCCADAR-ManualDeAdministrador`.

Guía de testeo

Este documento es una guía para el testeo del cluster. La misma contiene casos de test, escenarios de integración de los mismos y una descripción de como realizar este proceso. Ver documento `PGCCADAR-ManualDeInstalacion`.

Referencias

[1] B. Radic y E. Imamagic. Benchmarking the Performance of JMS on Computer Clusters. Presentado en CARNet Users' Conference, 2004. Disponible en línea en `http://www.srce.hr/dokumenti/crogrid/radovi/Benchmarking_the_Performance_of_JMS_on_Computer_Cluster.pdf`.

[2] E. Imamagic, B. Radic y D. Dobrenic. Job Management Systems Analysis. Presentado en CARNet Users' Conference, 2004. Disponible en línea en `http://www.srce.hr/dokumenti/crogrid/radovi/Job_Management_Systems_Analysis.pdf`.

[3] Maui, Cluster Resources. Maui Cluster Scheduler manual. Disponible en línea en `http://www.clusterresources.com/products/maui/`. Consultado Julio de 2007.

[4] TORQUE, Cluster Resources. TORQUE Resource Manager manual. Disponible en línea en `http://www.clusterresources.com/products/torque/`. Consultado Julio de 2007.

[5] William Gropp, Ewing Lusk y Thomas Sterling. Beowulf Cluster Computing with Linux, Second Edition. Publicado por MIT, 2003.

[6] Joseph D. Sloan. High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI. Publicado por O'Reilly, 2004.

[7] M. Michels y W. Borremans. Clustering with openMosix. University of Amsterdam, 2005. Disponible en línea en `http://staff.science.uva.nl/~delaat/snb-2004-2005/p20/report.pdf`. Consultado en Junio de 2007.

[8] Dansk Center for Scientific Computing. CSC's Super Computer documentation. Disponible en línea en `http://www.dcsc.sdu.dk/doc.php`. Consultado en Febrero de 2008.

[9] Stony Brook University. Seawulf Cluster tutorials. Disponible en línea en `http://www.sunysb.edu/seawulfcluster/tutorials.shtml`. Consultado en Febrero de 2008.

[10] University of Heidelberg. Helics Cluster documentation. Disponible en línea en `http://helics.uni-hd.de/doc/`. Consultado en Febrero de 2008.

[11] University of Glasgow. Glasgow computer cluster guide. Disponible en línea en `http://www.gla.ac.uk/services/it/whatwedo/computecluster/userguide/`. Consultado en Febrero de 2008.

[12] DRMAA, Open Grid Forum. Distributed Resource Management Application API Specification Documents. Disponibles en línea en `http://www.drmaa.org/documents.php`. Consultado en Junio de 2007.

[13] Sergio Nesmachnow. Algoritmos genéticos paralelos y su aplicación al diseño de redes de comunicaciones confiables. Facultad de Ingeniería, Universidad de la República, Uruguay, 2004. Disponible en línea en `http://www.fing.edu.uy/~sergion/Tesis.pdf`.

[14] Pablo Ezzatti. Mejora del desempeño de modelos numéricos del Río de la Plata. Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. Disponible en línea en `http://www.fing.edu.uy/inco/grupos/cecal/hpc/mej_des/tesis.pdf`.

[15] Cristian Perfumo, Gerardo Mora y Lucas Rojas. Algoritmos genéticos paralelos aplicados a la resolución de problemas de asignación de frecuencias en redes celulares. Universidad Nacional de la Patagonia, Argentina, 2006. Disponible en línea en `http://www.fing.edu.uy/inco/grupos/cecal/hpc/AG_MIFAP/AG_MIFAP.pdf`.

[16] Sergio Nesmachnow. Una versión paralela del algoritmo evolutivo para optimización multiobjetivo NSGA-II y su aplicación al diseño de redes de comunicaciones confiables. Facultad de Ingeniería, Universidad de la República, Uruguay, 2003. Disponible en línea en `http://www.fing.edu.uy/~sergion/MOEA/PNSGAII.pdf`.

[17] Sebastián Baña, Gonzalo Ferres y Nicolás Pepe. Proyecto MPI.net. Facultad de Ingeniería, Universidad de la República, Uruguay, 2003. Disponible en línea en `http://www.fing.edu.uy/~sergion/mpinet/`.

[18] Federico Dominioni y Pablo Musso. Proyecto algoritmos genéticos incrementales. Facultad de Ingeniería, Universidad de la República, Uruguay, 2003. Disponible en línea en `http://www.fing.edu.uy/~sergion/agi/`.

[19] R. Baron y L. Higbie. Computer Architecture. Publicado por Addison Wesley, 1992.

[20] M. De Balsi. Computer Architecture. Publicado por Addison Wesley, 1990.

[21] I. Foster. Designing and Building Parallel Programs. Publicado por Addison Wesley, 1995. Disponible en línea `http://www-unix.mcs.anl.gov/dbpp/`. Consultado en Agosto de 2007.

[22] Apache, The Apache Software Foundation. Documentación del Servidor HTTP Apache 2.0. Disponible en línea `http://httpd.apache.org/docs/2.0/`. Consultado en Junio de 2007.

[23] Ganglia. Ganglia 3.0 Installation and Configuration. Disponible en línea en `http://sourceforge.net/docman/display_doc.php?docid=128915&group_id=43021`. Consultado en Junio de 2007.

[24] PostgreSQL, The PostgreSQL Global Development Group. PostgreSQL 8.2 Documentation. Disponible en línea en `http://www.postgresql.org/docs/8.2/interactive/index.html`. Consultado en Julio de 2007.

[25] MySQL, Sun Microsystems. Documentación técnica. Disponible en línea en `http://www.mysql.com/`. Consultado en Julio de 2007.

[26] PHP, The PHP Group. Disponible en línea en `http://www.php.net/`. Consultado en Julio de 2007.

[27] Job Monarch. Documentación técnica. Disponible en línea en `https://subtrac.sara.nl/oss/jobmonarch/`. Consultado en Setiembre de 2007.

[28] PBS Python. Documentación técnica. Disponible en línea en `https://subtrac.sara.nl/oss/pbs_python/`. Consultado en Setiembre de 2007.