Capítulo 2

Teoría y Fundamentos

* 1. Banco de datos

Los datos forman una parte importante del activo de todo ente, es por ello que el proceso de administración de los datos es considerado de amplia criticidad para la toma de decisiones y desde tiempos históricos la información obtenida del ambiente brindó las pautas necesarias para sobrevivir en el ambiente. De esta forma la elección de un gestor de Banco de datos, al igual que la misma base de datos, va ligada al acceso a los datos. En esta sección abordaremos los principios básicos de Banco de Datos, además, se pretende brindar las bases necesarias para la correcta elección de una base de datos a ser montada sobre un Clúster SSI para su evaluación en pruebas de rendimiento.

2.1.1 Concepto de Banco de Datos.

Primeramente de forma fundamental para iniciar este capítulo definimos la unidad mínima de la cual tendremos referencia, es por ello que el dato (del latín datum, lo que se da) es nuestro punto de partida.

**Dato**

La definición en un ambiente relacionado a la tecnología de computación para dato seria “Información dispuesta de manera adecuada para su tratamiento por un ordenador”. [1] Partiendo de esta premisa se pude afirmar que el dato es una información que por sí sola no posee valor agregado y solo luego de ser procesada pasa a convertirse en información, que no es más que el resultado de este proceso. De esta manera se forma un bucle que es la base del conocimiento. A lo largo del proceso de transformación del dato a información existen dos estados, los cuales para definición los llamaremos “Estado de Reposo” y “Estado de Transición” **Fig.1**. Es decir, un dato para un tiempo *t* se encuentra en Reposo y en un tiempo *t’* está en una transición o movimiento.

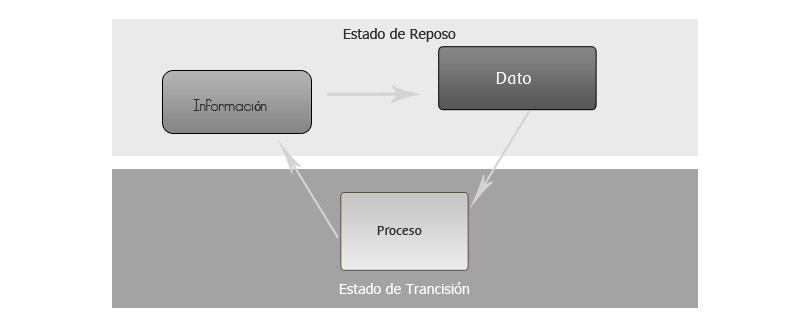


Fig. 1. Transición de un Dato hasta convertirse en Información.

**Base de Datos**

Apuntando a la definición de Base de Datos, se dispone que los datos en reposo deben ser alojados en un lugar lógico de forma ordenada a fin de tener un acceso a el mismo, estos lugares lógicos puede ser ficheros, archivos o lo que se puede denominar como base de datos. En resumen una base de datos es la disposición lógica de uno o más datos en reposo de forma ordenada.

**DBMS: Database Management System.**

La necesidad de incrementar la velocidad del acceso a datos y facilitar su administración provoco el desarrollo de herramientas que permitían visualizar de cierta forma las estructuras y datos dentro del banco de datos. Es por ello que el uso de estructuras de datos como las de Árbol-B para la organización de búsquedas en archivos físico/lógicos permitieron el nacimiento de los sistemas administradores de base de datos (DBMS, por sus siglas en ingles de Data Base Management System).

Las DBMS obedecían una arquitectura Cliente-Servidor, donde el motos de la DBMS era considerado como el servidor y las aplicaciones que lo utilizaban cumplían el rol de clientes. Esta estructura de servidor, internamente, se encontraba formada por partes que de forma síncrona realizaban tareas solicitadas desde el o los clientes.

**Bases de Datos Relacionales**

En 1970 el científico informático ingles Edgar F. Codd publicaba su modelo de banco de datos relacional para datos compartidos [2], modelo que posteriormente seria tomado como referencia para las bases de datos más utilizadas en la actualidad [3][4][5]6][7].

Una base de datos relacional no es más que un conjunto de relaciones organizadas mediante tablas de identificador únicos que abstraen un objeto. De esta manera las tablas están formadas por tuplas que a su vez están compuestas de atributos. La relación entre tablas para la obtención de atributos específicos se realiza mediante claves únicas que identifican a las tuplas de una tabla de manera inequívoca, de forma análoga a la teoría de conjuntos [9]. Las claves únicas forman un requisito común entre los conjuntos a ser cruzados. De la misma forma lo realizarían las DBMS dando lugar a las RDBMS o Relational Data Base System Management debido a sus siglas en ingles.

**Sentencias SQL.**

Las DBMS, y a su vez las RDBMS, interactúan con las bases de datos mediante el estándar SQL, siglas del ingles Structured Query Language que fue considerado como tal con su primera versión de SQL-86 y no hasta su segunda versión revisada en 1992, conocida como SQL-92, fue ampliamente aceptada ya que se consideraba que cubría todas las necesidades. La revisión del 2003 permitió la adopción del XML [8] al estándar SQL permitiendo que funciones lean y formen estructuras XML a partir de datos de un SQL [10]

A modo de definir a las Sentencias SQL por su comportamiento podemos agruparlas en dos grandes grupos Lenguaje de Manipulación de Datos (DML de sus siglas del ingles Data manipulation Language) y el Lenguaje de Definición de Datos (DDL de sus siglas del ingles Data Definition Language).

**DDL**: El Lenguaje de Definición de Datos es utiliza para administrar y definir objetos de los bancos de datos y se encuentra definidas principalmente por las instrucciones *Create, Alter y Drop*

**DML**: El Lenguaje de Manipulación de Datos es utilizado para la manipulación de los objetos existentes en los diferentes objetos de base de datos. Las instrucciones utilizadas en el DML, principalmente, son *Select, Update y Delete*.

Las instrucciones SQL de forma genérica se forman por palabras reservadas [11] en forma conjunta con los nombres de atributos, tablas y otros objetos de base. Esta notación nos permite crear instrucciones que dispuestos de forma correcta luego de su ejecución permite obtener los datos deseados. A modo de ejemplo, se muestra un grupo de tuplas simples contenidas en una tabla, con un atributo único como clave primaria y otros atributos que complementan la información **fig.2.**



Fig. 2. Ejemplo de registros de una tabla

De forma de obtener según un criterio los datos podemos realizar una instrucción SQL que solicite a la tabla los datos que cumplan con las condiciones **fig.3**.



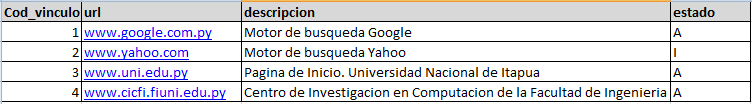


Fig.3. Sentencia SQL y el resultado arrojado.

2.1.2 Banco de Datos en la industria

En la actualidad el uso de banco de datos, para todo lo considerado con el flujo de la información, es frecuente ya que todas las empresas que cuente con datos a ser almacenados cuentan con un banco de datos. El acceso a este es restringido por normas que regulan el acceso ya que es considerado un patrimonio del ente, es por eso que salvaguardar los datos del ente en contra de terceros es una preocupación valida.

El nacimiento de los recursos y datos compartidos genero facilidades al momento de tener acceso inmediato a los datos de forma precisa, pero a su vez necesito que los canales de transmisión sean más seguros, teniendo en cuenta el uso mal intencionado que podría generarse si alguien así lo deseara.

La otra cara de la moneda es el aprovechamiento de la red global para la gestión de datos en glandes conglomerados. De esta forma nace lo que hoy conocemos como DaaS (del ingles Database as service) [12] o Base de Datos como servicio, el cual aprovechando su recursos ofrece servicios de alojamiento de datos en internet.

De forma análoga a las DBMS, DaaS considera el modelo cliente-servidor donde el Servidor o host provee el servicio que los clientes pueden consumir. Este modelo permite el acceso centralizado a los datos, pero a su vez abrió una brecha de seguridad dejando a nuestros datos vulnerables de robos, alteraciones y otros elementos que no son garantizados por el prestador del servicio.

2.1.3 Banco de Datos y Paralelismo

La utilización del procesamiento paralelo [13] en los bancos de datos tradicionales busca acelerar la ejecución de las instrucciones dividiéndolos en fragmentos de modo que una instrucción pueda ser dividida en N fragmentos siendo N el numero de procesos o hilos concurrentes que puede realizar la ejecución al mismo tiempo, es decir que multiplicamos nuestro poder computacional en N veces.

Los sistemas de bases de datos en paralelo utilizan el cómputo paralelo aplicado a la administración de datos. EL concepto se basa en un conjunto de nodos de bases en los cuales se procesan los datos y cada nodo mantiene una copia replicada de la BD original, un manejador principal para la distribución de la carga y las necesidades de comunicación entre nodos.

Los factores más resaltantes de las Bases de Datos Paralelas se canalizan principalmente en:

* Consideraciones en el diseño de la Distribución de información entre los nodos.
* Consideraciones en la elaboración de consultas manteniendo el costo de transmisión de la información a los nodos y el retorno de los resultados presente en el cálculo.
* Manejo de concurrencia, bloqueo, durabilidad y atomicidad de transacciones.
* Manejo transparente de datos para usuarios.

Otro concepto muy ligado a las bases de datos paralelas es el balanceo de carga. Esta técnica permite dividir un trabajo de forma tal que todos los subprocesos terminen en el mismo tiempo considerando la capacidad de cada nodo.

El balanceo de carga disminuye el tiempo ocioso de los procesadores y mediante su distribución de carga de trabajo realiza un aprovechamiento optimo del procesador en relación capacidad de carga / tiempo. Esta configuración de trabajo realiza un mejor desempeño cuando ambos componentes (nodos, procesos, etc.) son de similares características (velocidad de proceso y velocidad de acceso y retorno de datos) de modo a dividir la capacidad total entre ellos.

2.2 Clúster

2.2.1 Procesamiento Paralelo.

Movido por las disponibilidades y limitaciones de los semiconductores, la industria informática debió establecer nuevas metodologías que permitan realizar de forma más optima los grandes conglomerados de información.

En el inicio del siglo XXI los procesadores dieron un giro que mas en adelante se convirtió en tendencia y de esta forma la posibilidad de realizar más de un proceso mediante la inclusión de un núcleo independiente dentro de la misma unidad promoviendo la independencia de tarea por núcleos y por ende un paralelismo a nivel de bits [14].

Toda implementación a nivel de hardware requiere un software que pueda interpretar las órdenes de ejecución y por ende realice la distribución del trabajo en forma paralela. Ambos componentes en conjunto permitieron la serialización de las cargas de trabajo.

Si bien el procesamiento paralelo no es algo sumamente nuevo, ya que tiene sus inicios en la década del 70 [15], hoy es considerado como esencial para la computación de altas prestaciones.

En palabras generales el procesamiento paralelo busca acelerar la ejecución de tareas dividiéndolo en paquetes para luego procesarlo en forma simultánea en procesadores o núcleos diferentes.

A finales de la década del 60 e inicio de los 70 Michael Flynn sugirió una forma de organizar las arquitecturas con el motivo de estudiarlas [16], estas divisiones se basan sobre las instrucciones de control de secuencias de datos dispuestas a la arquitectura. Ellas están divididas en:

* SISD: Del ingles, Single Instruction and Single Data. Tomando como ejemplo la Arquitectura de Von Neumann, corresponde a una única instrucción operando con una única secuencia de datos.
* SIMD: Del Ingles, Single Instruction and Multiple Data. Una misma instrucción aplicada a múltiples secuencias de datos.
* MISD: Del Ingles, Multiple Instruction and Single Data. Múltiples Instrucciones simultaneas aplicados a una única secuencia de datos.
* MIMD: Del ingles, Multiple Instruction and Multiple Data. Esta ultima trata sobre las arquitecturas que permiten un múltiple acceso a secuencias de datos por instrucciones múltiples. Tratando de organizar esta ultima a finales de la década del 90 se procede a subdividirla en dos secciones [17][18]:
  + SMP (Symetric MultiProcesing). Arquitecturas que comparten una memoria física única y cuya velocidad disminuye según se incrementan los procesadores.
  + DMM (Distibuited Memory Multiprocessing). Son sistemas en las cuales el procesador posee su propia memoria, evitan problemas de bloqueos y evita el acceso a la memoria física principal.

2.2.2 Concepto Clúster.

En forma general podemos definir la palabra clúster como un numero de elementos del mismo tipo agrupados entre si. Su definición proviene del idioma ingles [19] y es utilizado para definir un conjunto de elementos, de igual manera Tomas Sterling lo señalaba en su libro [20] en el cual realiza analogías con los sistemas biológicos, organizaciones humanas y estructuras de computadores.

En las ciencias de la computación el término clúster fue asociado a un conjunto de ordenadores que interconectados mediante un medio o red trabajan en forma conjunta. Muchas investigaciones y aplicaciones a problemas específicos tanto en la industria como en el ámbito académico, fueron desarrolladas con esta herramienta.

Con el transcurso de los años la tecnología de clúster cobro fuerza al convertirse en una alternativa viable y menos costosa para procesos de gran porte. La oportunidad de escalar esta arquitectura, la flexibilidad en cuanto a configuración y los avances en alta disponibilidad lograron esta posición.

Un clúster brinda flexibilidad de configuración. El número de nodos, la memoria disponible por nodo, el número de procesadores por nodo y la topología de red utilizada para la interconexión de las partes son los parámetros mas considerados a la hora de especificar el poder computacional del clúster. Otra característica muy bien vista por la comunidad tecnológica es la capacidad de adaptación a nuevas tecnologías de hardware y por tanto el incremento de la capacidad de cómputo no se vería ligado de forma unilateral a la compra de un nuevo equipo y de forma condescendiente el desecho del anterior como sucede con los mainframes.

La clave del comportamiento del clúster se basa principalmente en el manejo de los planificadores de plazo [21] y en la capacidad de interconexión entre los componentes de la red conocidos también como nodos del clúster.

**Nodos**

Siguiendo con la analogía podemos considerar a un nodo como el punto de intersección de la red interna **Fig.4**.

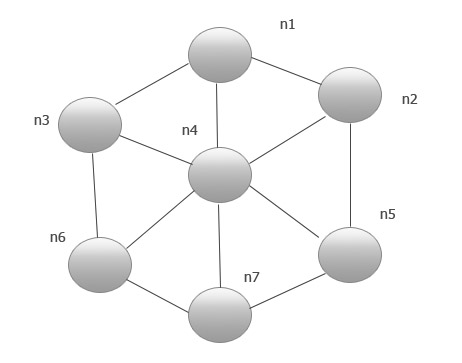


Fig. 4. Disposición de nodos Interconectados

En un Clúster un Nodo es la intersección de la red, pudiendo ser un nodo conector o terminal.

Un nodo provee de forma independiente las mismas características que un ordenador personal, es decir, un sistema operativo, memoria independiente, procesador y espacio de almacenamiento. Cada nodo, por ende, es considerado un entorno independiente que forma parte del clúster. Extendiendo esta ideología existe lo que hoy conocemos como “Clúster de clúster”, en donde un clúster pasa a ser el nodo de otro conglomerado más grande.

Un nodo, al igual que un ordenador, debe poder realizar operaciones de forma independiente, es por ello que debe poseer ciertas características similares a las de un ordenador.

2.2.3 Características de un clúster.

Es un hecho que la mayoría de los más grandes problemas estratégicos en las cuales se aplica el cálculo numero en relación con las ciencias de la computación, solo puede ser resueltas mediante un maquina cuya ingeniería permita realizar un computo complejo.

De manera desafortunada para usuarios o empresas el costo de estos ordenadores no compensa la necesidad en un factor de costo o tiempo de uso. Es por ello que las alternativas viables para tener acceso a esta disponibilidad fueron siempre aceptadas y estudiadas.

El punto en el cual se unen las necesidades de disponibilidad, rendimiento y bajo costo de forma sinérgica dieron inicio a una tendencia que se expandió de acuerdo a la demanda de las necesidades. Es punto es conocido como Computación en Conjunto (del ingles Clúster Computing) cuya características principales se basan en la unión de una capa física de nodos, una abstracción del manejo del paso de mensajes y la aplicación de tecnologías Distribuidas.

1. Middleware SSI

El middleware puede ser considerado como una capa software que reside entre las aplicaciones y los niveles subyacentes como son el sistema operativo, las pilas de protocolo y el hardware [22]. Históricamente el concepto de middleware no proviene de la investigación académica, sino de la industria.

Una imagen de sistema único SSI (por sus siglas en ingles) es una propiedad de un sistema que oculta la distribución y heterogeneidad de sus recursos. y los vuelve disponibles a los usuarios y aplicaciones como un recurso computacional unificado. SSI provee a los usuarios una vista globalizada de los recursos disponibles en el sistema, desconsiderando al nodo al cual este asociado físicamente dentro del entorno distribuido. Además, SSI puede asegurar la continua operatividad del sistema luego de una falla (alta disponibilidad) así también garantiza que el sistema se cargue uniformemente, provea multiprocesamiento y gestión de los recursos.

Los objetivos de un diseño SSI para un sistema basado en Clúster se enfocan principalmente en la transparencia complete de la gestión de recursos, el rendimiento escalable, y la disponibilidad del sistema para el soporte de las aplicaciones para usuarios [19]. SSI puede ser definido como una ilusión creada por hardware o software, el cual presenta una colección de recursos como un único recurso mucho más poderoso [23].

1. Servicios y Beneficios

Los principales servicios de una imagen de sistema único SSI incluyen los siguientes [19]:

* Proporciona una visión simple y directa de todos los recursos y actividades del sistema desde cualquier nodo del clúster.
* Libera al usuario final de tener que saber en que parte del clúster se ejecutará la aplicación.
* Permite el uso de los recursos de manera transparente con independencia de su ubicación física.
* Permite el trabajo del usuario por medio de una interfaz amigable y permite al administrador gestionar al clúster como una sola identidad.
* Ofrece la misma sintaxis de comandos como de otros sistemas y por lo tanto reduce el riesgo de errores de operaciones y con este resultado los usuarios finales observan un mayor rendimiento, fiabilidad y mayor disponibilidad del sistema.
* Permite centralizar/descentralizar la gestión del sistema y el control para evitar la necesidad de administradores expertos para la gestión del sistema.
* Simplifica de gran manera la gestión del sistema y reduce así el costo de propiedad.
* Proporciona la comunicación de mensajes sin dependencia de localización
* Beneficia a los desarrolladores de sistemas en la reducción del tiempo, esfuerzo y conocimientos necesarios para la realización de tareas que permiten al personal actual manejar grandes o más complejos sistemas.
* Promueve el desarrollo de herramientas estándares y servicios.

Un buen SSI se obtiene normalmente mediante la cooperación entre todos estos niveles, un nivel más bajo pueden simplificar la implementación de un ser superior.

1. SSI al nivel del sistema operativo

Los sistemas operativos de los Clúster soportan una eficiente ejecución de aplicaciones paralelas en entorno distribuido con aplicaciones secuenciales. El objetivo es reunir los recursos en un clúster para proporcionar un mejor rendimiento tanto para las aplicaciones secuenciales y paralelas.

Para lograr este objetivo, el sistema operativo debe de ser compatible con la programación planificada en masa de algoritmos paralelos [24], identificación de los recursos ociosos en el sistema, tales como procesadores, memoria y redes; además ofrecen acceso globalizado a ellos. Debe soportar de manera optima el proceso de migración para proporcionar equilibrio de carga dinámica, así como la comunicación entre procesos rápidos, tanto par alas aplicaciones del sistema y de nivel de usuario. El sistema operativo debe asegurarse de que estas características estén disponibles para el usuario sin la necesidad de llamadas adicionales al sistema.

A continuación se listan os sistemas operativos más representativos que soportan SSI al nivel de kernel:

**SCO Unix Ware Non Stop Cluster**

Es el software de alta disponibilidad de SCO. Amplia significativamente el soporte por hardware, por lo que es más fácil y menos costoso de implementar el software de clustering mas avanzado para sistemas Intel. Es una extensión para el sistema operativo UnixWare en el cual todas las aplicaciones se ejecutan de manera eficaz y fiable dentro de un entorno SSI el cual elimina toda la gestión de carga. Cuenta con IP estándar como la interconexión, eliminando la necesidad de algún hardware propietario. La arquitectura clúster de Unix Ware Non Stop ofrece soporte integrado para aplicaciones de conmutación por error mediante un enfoque de *n + 1*. Con este enfoque, la copia de seguridad de la aplicación puede ser reiniciada en cualquiera de los varios nodos dentro del clúster. Esto permite que un nodo actúe como un nodo de copia de seguridad para los demás nodos dentro del clúster [25].

**Sun Solares-MC**

Es una extensión prototipo de un kernel Solaris de nodo único. Proporciona imagen de sistema único y alta disponibilidad al nivel del kernel. Solaris MC esta implementado mediante técnicas de orientación a objetos. Utiliza de manera amplia el lenguaje de programación orientado a objetos de C++, el modelo estándar de objeto y el lenguaje de definición de interfaz de COBRA. Solaris MC utiliza un sistema de archivos global llamado Proxy del Sistema de Archivos PXFS (las siglas en ingles de Proxy FileSystem). Las características principales incluyen la imagen de sistema único, la semántica coherente y de alto rendimiento. El PXFS hace que los accesos a ficheros se vuelva transparente para los procesos. PXFS logra una imagen de sistema único por medio de la interceptación de las operaciones de accesos a archivos a nivel de la interfaz vnode/VFS (WritingFileSystem, instancias para la operatividad de los FileSystems). Solaris MC asegura que las aplicaciones de red no necesiten ser modificados y observen la misma conectividad de red, independiente de en que nodo se este ejecutando la aplicación [26].

**GLUnix**

Otra alternativa disponible para que el sistema operativo soporte SSI es la implementación de una capa superior en el sistema operativo existente, la cual realiza las asignaciones globales de los recursos. Este es el enfoque seguido por GLUnix de Berkeley. Esta estrategia vuelve portable al sistema operativo y reduce el tiempo de desarrollo. GLUnix es una capa del sistema operativo diseñado para proveer soporte en la ejecución remota transparente, trabajos paralelos y secuenciales, balanceo de carga y compatibilidad con versiones anteriores de binarios de aplicaciones existentes. GLUnix esta completamente implementado en el nivel de usuario y no necesita modificación en el kernel, por lo que se vuelve mas fácil de implementar. Las principales características proporcionadas por GLUnix incluyen técnicas de planificación en paralelo de algoritmos paralelos; detección de recursos ociosos, proceso de migración, y balanceo de carga; comunicación rápida a nivel de usuario, y soporte de disponibilidad [27].

**MOSIX**

Mosix es una extensión del kernel de Linux que permite ejecutar aplicaciones no paralelizadas en un Clúster. Una de las posibilidades de MOSIX es la migración de procesos, que permite trasladar los procesos de nodo en nodo. Mosix opera de manera inadvertida y sus operaciones son transparentes para las aplicaciones. Esto significa que se pueden ejecutar aplicaciones secuenciales y paralelas al igual que lo haría un SMP (Sistema de Multiprocesamiento simétrico, Symmentricmultiprocessing). No es necesario prestar atención en donde el proceso se esta ejecutando o lo que otros usuarios podrían estar realizando. Poco después de que un nuevo proceso haya sido creado, Mosix intenta asignarlo al mejor nodo disponible en ese momento. Luego de esto, Mosix continua monitoreando el nuevo proceso, asi como también los otros procesos existentes, evaluara los nodos para maximizar el rendimiento global. Todo esto es realizado sin necesidad de alterar la interfaz de Linux. Esto significa que todos los procesos pueden ser monitoreados y controlados como si estuviesen ejecutándose en un nodo en particular. La última versión de Mosix, llamada MOSIX2 es compatible con Linux 2.6. MOSIX2 es implementada como una capa virtualizada del sistema operativo, lo que provee al usuario y a las aplicaciones una imagen de sistema único SSI [28].

**OpenMosix**

El sistema OpenMosix esta basado en Mosix, la principal diferencia, se encuentra en su licencia GPL. OpenMosix es un conjunto de parches al kernel y unas utilidades y bibliotecas de área de usuario que permiten tener un sistema SSI completo para Linux. Al estar basado en el código de MOSIX, comparte algunas de sus características y limitaciones. OpenMosix hace uso del parche de Rik van Riel de mapeado inverso de memoria, que permite que el proceso que consume más recursos de OpenMosix pase de tener una complejidad computacional de *O(n)* a una complejidad *k*. En la práctica, elimina una de las partes del código de OpenMosix que pueden consumir una cantidad apreciable de procesador. OpenMosix fue lanzado como un parche para el kernel de Linux, pero también estuvo disponible en Live CDs especializados. El desarrollo de OpenMosix ha sido detenido por sus desarrolladores, pero el proyecto LinuxPMI continúa su desarrollo por medio del código OpenMosix [29].

**Kerrighed**

Kerrighed es el resultado de un proyecto de investigación iniciado en 1999. Su objetivo es presentar un único SMP (Sistema de Multiprocesamiento simétrico, Symmentricmultiprocessing) por encima del clúster. Kerrighed se compone de un conjunto de servicios distribuidos del kernel a cargo de la gestión general de los recursos del cluster [29][30].

**OpenSSI**

OpenSSI surge a inicios del año 2001, esta basado en los proyectos Non-Stop Cluster de UnixWare [25]. Bruce Walker, director del proyecto y principal desarrollador de OpenSSI, demarco claramente que el objetivo era crear una plataforma para poder integrar tecnologías de clúster, con código abierto. La ultima versión de OpenSSI dispone de varios archivos del sistema y manejo de sistemas de disco en código abierto, como por ejemplo GFS, OpenGFS, Lustre, OCFS, DRBD, también integra un mecanismo de bloqueo distribuido (OpenDLM) y una política para obtener balanceo de carga, siendo esta una característica importante heredada de Mosix. OpenSSI permite el balanceo de carga de un clúster dinámicamente mediante el uso de migración de procesos, otra característica también heredada de Mosix. El modulo de migración de procesos de OpenSSI utiliza un mecanismo de acceso al recurso remoto manteniendo los recursos de acceso del sistema bloqueados para que estos no puedan ser migrados. Este mecanismo es usado principalmente en IPC (interfaz de la tarjeta de red), CFS (sistema de archivos del clúster) y también para algunas llamadas al sistema [29].

1. Optando por OpenSSI como un clúster de alto desempeño y balanceo de carga

En los clústeres SSI de alto rendimiento los programas no tienen por que notar que se están ejecutando en uno de ellos. El único requisito es que los programas desplieguen múltiples procesos, los cuales se asignan de forma transparente entre los ordenadores del clúster. Aunque los clústeres SSI no son tan escalables como algunos otros tipos, ofrecen sin embargo la significativa ventaja de que los programas que se ejecutan en el sistema no tienen que conocer la existencia del clúster en el que se estén ejecutando. Otras plataformas de alto rendimiento basadas en clústeres requieren que el código fuente de los programas contengan código propio del clúster, o al menos que el programa se enlace con alguna librería de este. OpenSSI es una solución de clúster SSI extensa de código abierto para Linux basada en la plataforma clúster NonStop de HP. NonStop deriva de Locus, que se desarrollo en los ochenta.

OpenSSI puede repartir los procesos de forma transparente entre múltiples máquinas, característica conocida como nivelado de carga [31]. Otras plataformas de clúster SSI para Linux capaces de realizar el nivelado de carga son OpenMosix y Kerrighed. OpenMosix es el clúster SSI mas popular de Linux. En julio del 2007 se anuncio que el proyecto OpenMosix terminaría en marzo de 2008. Kerrighed es relativamente nuevo, encontrándose actualmente en una fase de desarrollo rápido.

OpenSSI monitoriza constantemente la carga en los ordenadores del clúster y migra automáticamente los procesos entre los nodos. Este sistema es capaz de migrar una aplicación multihilo, pero no es capaz de migrar hilos individualmente. Un proceso migrado puede continuar con las mismas operaciones que estaba realizando en la máquina original, puede leer y escribir en los mismos ficheros o dispositivos e incluso puede continuar una comunicación interproceso (IPC) sobre un socket.

La mayoría de las aplicaciones se ejecutan en OpenSSI sin ninguna modificación, con algunas excepciones, tal y como se indica en la pagina de ayuda del comando migrate. OpenSSI también proporciona una librería (API) que los programadores pueden utilizar para controlar el clúster.

Sabemos OpenMosix es el sistema de clustering más conocido y usado, sin embargo, aunque es difícil luchar contra un sistema plenamente acogido, desarrollado y mejorado por muchísima gente de diferente índole, OpenSSI, con cada actualización de Kernel nueva va ganando más adeptos y haciéndose un hueco entre el mundo de la supercomputación.

1. Características de OpenSSI

A continuación se muestran las características más importantes de OpenSSI:

* Manejo y administración del dominio de forma sencilla.
* Sistema de archivos de la raíz generado a través del clúster de forma sencilla.
* Copia sencilla de archivos binarios, librerías y administradores (como una contraseña).
* Se puede manejar los procesos por todo el clúster de manera sencilla según su PID.
* El nombre del clúster está disponible para todos los objetos del IPC.
* Contiene un dispositivo de acceso al clúster
* Nombrado de entidades consistente a lo largo de los nodos.
* Los archivos de acceso a todos los nodos son completos y públicos.
* La tecnología del sistema de archivos del clúster la tiene integrada y le provee flexibilidad y elección.
* Las interfaces del núcleo permitirán otra tecnología que esté desarrollada en código abierto.
* Está disponible un sistema de archivos del clúster con un failover público.
* El nombre y la dirección del clúster será altamente disponible, pudiéndose acceder a el siempre.
* La migración de procesos se puede realizar de forma completa incluyendo las llamadas al sistema.
* El balanceo de carga de un proceso se realiza en tiempo de ejecución.
* Incluye un conjunto de características de disponibilidad.
* Monitorización y reinicio del proceso.
* Servicio automático de failover.
* Sistema de archivos automáticos del failover.
* Dirección IP del clúster, manejo de la conexión y la habilidad de perder el nodo inicio sin matar a los procesos que están corriendo.
* Garantía y un API para los miembros que usan una infraestructura de un Clúster bajo Linux.
* Un API para migrar los procesos según los métodos: rexec() y fork()
* Nodos que no necesitan discos gracias al arranque por red.

1. Limitaciones de OpenSSI

En OpenSSI hay que tener en cuenta ciertas limitaciones para que funcione correctamente, a continuación se muestra un listado de las limitaciones que se deben cumplir:

* Miembros del clúster: El número máximo de nodos en un clúster es de 125.
* El sistema de archivos de un clúster (CFS): Tamaño máximo de ficheros, número máximo de ficheros, direcciones y sistemas de archivo físicos inherentes.
* UID/GID: 16-bit en OpenSSI-1.2. 32-bit en OpenSSI-1.9.
* Tipos de dispositivos principales: 255 en OpenSSI-1.2. 4095 en OpenSSI-1.9.
* Tipos de dispositivos secundarios: Número máximo de puntos de montaje por tipo de sistema de archivos. 8-bit in OpenSSI-1.2. 20-bit in OpenSSI-1.9.
* IPC: Máximo número de semáforos compartidos.
* Sockets: Máximo número de sockets.
* Procesos: Máximo número de procesos. 32,000 en OpenSSI-1.2 y 1 billón(americano) en OpenSSI-1.9.
* PTY: Máximo número de pty’s.
* HA-LVS: Máximo número de conexiones, de directorios, de CVIP. Igual que el original LVS, millones de conexiones.
* DRBB-SSI: Máximo tamaño del volumen por dispositivo de drbd, máximo de dispositivos drbd. Igual que el original DRBD.

Bibliografía

1. Diccionario de la Real Academia española. <http://buscon.rae.es/draeI/SrvltConsulta?TIPO\_BUS=3&LEMA=dato/>.
2. Codd, E.F. *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM ed. 13, Vol. 6 pp. 377–387. 1970.
3. *Oracle Database*. <http://www.oracle.com/index.html/>
4. *Postgresql Database*. <http://www.postgresql.org/>
5. *MySql Database*. <http://www.mysql.com/>
6. *SQLServer 2008 Database*. <http://www.microsoft.com/spain/sql/2008/default.aspx/>
7. *Firebird Database*. <http://www.firebirdsql.org/>
8. *XML Reference Markup*. <http://www.w3.org/XML/>
9. J. Ferrando, V. Gregori. *Matemática Discreta*. 2da. Edicion. Editorial Reverté. 1995.
10. A. Eisenberg, J. Melton*. SQL/XML is making good progress*. SIGMOD Rec. 31, 2 (June 2002), 101-108. DOI=10.1145/565117.565141 <http://doi.acm.org/10.1145/565117.565141/>
11. *Estándares Ansi SQL*. <http://fenix.dcaa.unam.mx/servidores/docs/bd/ANSI%20SQL.pdf/>
12. H. Hacigum us, S. Mehrotra, B. Iyer. *Providing database as a service*. In ICDE (2002), IEEE Computer Society, pp. 29–38.
13. J. Dongarra et all. *Sourcebook of parallel Computer*.
14. K. Asanovic et all. *The landscape of parallel computing research: A view from Berkeley*. Technical Report, EECS Department, University of California, Berkeley.
15. P. Denning, J. Dennis. *The resurgence of Parallelism*. Communications of the ACM. June 2010, vol 53 – Nº 6.
16. Flynn M. J. *Some Computer Organizations and Their Effectiveness*. IEEE Trans. Comput., Vol C-21, pp 948. 1972.
17. Ralph Duncan. *A Survey of Parallel Computer Architectures*. IEEE Computer Society Press. Vol 23, Issue 2.
18. K. Hwang, F. Briggs. *Computer Architecture and Parallel Processing*. McGraw-Hill. 1994.
19. Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems, Volume 1*. School of Computer Science and Software Engineering Monash University Melbourne, Australia
20. T. Sterling. *Beowolf Cluster Computing*. The MIT Press. Cambrige, Massachusetts – London,England. 2002.
21. Tanenbaum, Andrew S. *Sistemas operativos distribuidos*. Prentice Hall Hispanoamericana. México. MX. 1996.
22. R. Schantz and D. Schmidt. *Research Advances in Middleware for Distributed Systems: State of the Art*. In IFIP World Computer Congress, pages 1-36, Canadá, August 2002. Kluwer Academic Publishers.
23. Greg Pfister, *In Search of Clusters*, 2nd Edition, Prentice Hall 1998.
24. Gang Scheduling. *Timesharing on Parallel Computers*, SC98, November 1998
25. Walker, B., and Steel, D. 1999. I*mplementing a full single system image UnixWare cluster: Middleware vs. underware*. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications
26. Vlada Matena, Moti N. Thadani, Ken Shirriff, Yousef A. Khalidi and Jose M. Bernabeu. *Solaris MC: A Multi-Computer OS*. November 1995
27. Ghormley, D., Petrou, D., Rodrigues, S.,Vahdat, A., and Anderson, T. 1998. *GLUnix: A global layer Unix for a network of workstations*. Journal of Software Practice and Experience <http://now.cs.berkeley.edu/Glunix/glunix.html>
28. Barak, A., and La’adan, O. 1998. *The MOSIX multicomputer operating system for high performance cluster computing*. Journal of Future Generation Computer Systems <http://www.mosix.cs.huji.ac.il/>.
29. Lottiaux, R.; Gallard, P.; Vallee, G.; Morin, C.; Boissinot, B.; IRISA, Rennes. *OpenMosix, OpenSSI and Kerrighed: a comparative study*. France. May 2005
30. Christine Morin, Renaud Lottiaux, Geoffroy Valle, Pascal Gallard, David Margery, Jean-Yves Berthou, and Isaac Scherson. *Kerrighed and data parallelism: Cluster computing on single system image operating systems*. In Proceedings of Cluster 2004. IEEE, September 2004
31. *Introducción a OpenSSI*: <http://openssi.org/ssi-intro.pdf>