Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Computació d'Altes Prestacions

El impacto de las aplicaciones intensivas de E/S en la planificación de trabajos en clusters no-dedicados

Memoria del trabajo de ”Iniciación a la Investigación. Trabajo de Fin de Máster” del “Máster en Computació d'altes prestacions”, realizada por Aprigio Augusto Lopes Bezerra, bajo la dirección del Dr. Porfidio Hernández Presentada en la Escuela de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

2010

Iniciación a la investigación. Trabajo de fin de máster

Máster en Computación de Altas Prestaciones.

El impacto de las aplicaciones intensivas de E/S en la planificación de trabajos en clusters no-dedicados

Autor: Aprigio Bezerra

Director: Porfidio Hernández

en la Escuela de Ingeniería, en el Departamento Arquitectura de Computadores y Sistemas Operativos

Escuela de Ingeniería

Universidad Autónoma de Barcelona

Firmado

Autor	Director
Aprigio Bezerra	**Porfidio Hernández**

Agradecimientos

Deseo expresar mi agradecimiento al Dr. Porfidio Hernández Budé, por su paciencia, comprensión y constante disponibilidad para la crítica y valoración del trabajo realizado. Así como al Dr. Juan Carlos Moure y al Dr. Antoni Espinosa por sus incalculables aportes.

A Eduardo Argollo por su confianza y ayuda a mi llegada a Barcelona.

A mis compañeros, especialmente a Sandra Mendez, Eduardo Cabrera, João Gramacho y Leonardo Fialho.

A Daniel Ruiz y Javier Navarro por el suporte rápido y eficiente de todas solicitaciones.

Y muy especialmente, a mi esposa e hijos por la comprensión de mí prolongada ausencia.

Abstract

With the increased capacity of processing nodes in relation to computing power, increasingly data-intensive applications such as applications of bioinformatics, will be run on non-dedicated clusters. The non-dedicated clusters are characterized by their ability to combine the implementation of local user applications with applications, scientific or commercial, executed in parallel. Learn what effect intensive applications to access given for mixed produce other (batch, interest, SRT, etc) in the non-dedicated environment allows the development of more efficient planning policies.

Some intensive applications E / S are based on the MapReduce paradigm where environments that use them, such as Hadoop, dealing with data locality, load balancing automatically and work with distributed file systems. Hadoop's performance can be improved without increasing the costs of hardware, tune several key settings to the specifications of the cluster, for the size of the input data and complex processing. The timing of these timing parameters may be too complex for the user or administrator but seeks to ensure more adequate benefits.

This master thesis proposes the evaluation of the impact of intensive applications E / S in planning work on non-dedicated clusters under the MPI, MapReduce paradigm.

Keywords: planning, non-parallel, MPI, MapReduce

Resum

Amb la major capacitat dels nodes de processament en relació a potència de còmput, cada vegada més aplicacions intensives de dades com les aplicacions de la bioinformàtica, es duran a executar en clústers no dedicats. Els clústers no dedicats es caracteritzen per la seva capacitat de combinar l'execució d'aplicacions d'usuaris locals amb aplicacions, científiques o comercials, executades en paral.lel. Saber quin efecte les aplicacions amb accés intensiu a daus produeixen respecte a la barreja d'un altre tipus (batch, interès, SRT, etc) en els entorns no-dedicats permet el desenvolupament de polítiques de planificació més eficient.

Algunes de les aplicacions intensives d'E / S es basen en el paradigma MapReduce on els entorns que les utilitzen, com Hadoop, s'ocupen de la localitat de les dades, balanceig de càrrega de forma automàtica i treballen amb sistemes d'arxius distribuïts. L 'exercici de Hadoop es pot millorar sense augmentar els costos de maquinari, en sintonitzar diversos paràmetres de configuració claus per a les especificacions del clúster, per la mida de les dades d'entrada i per al processament complex. La sincronització d'aquests paràmetres de sincronització pot ser massa complexa per a l'usuari i / o administrador però procura garantir prestacions més adequades.

Aquest treball proposa l'avaluació de l'impacte de les aplicacions intensives d'E / S en la planificació de treballs en clústers no-dedicats sota els paradigmes MPI i MapReduce.

Paraules clau: planificació, no paral.lels, MPI, MapReduce

Resumen

Con la mayor capacidad de los nodos de procesamiento en relación a la potencia de cómputo, cada vez más aplicaciones intensivas de datos como las aplicaciones de la bioinformática, se llevarán a ejecutar en clusters no dedicados. Los clusters no dedicados se caracterizan por su capacidad de combinar la ejecución de aplicaciones de usuarios locales con aplicaciones, científicas o comerciales, ejecutadas en paralelo. Saber qué efecto las aplicaciones con acceso intensivo a dados producen respecto a la mezcla de otro tipo (batch, interativa, SRT, etc) en los entornos no-dedicados permite el desarrollo de políticas de planificación más eficientes.

Algunas de las aplicaciones intensivas de E/S se basan en el paradigma MapReduce donde los entornos que las utilizan, como Hadoop, se ocupan de la localidad de los datos, balanceo de carga de forma automática y trabajan con sistemas de archivos distribuidos. El rendimiento de Hadoop se puede mejorar sin aumentar los costos de hardware, al sintonizar varios parámetros de configuración claves para las especificaciones del cluster, para el tamaño de los datos de entrada y para el procesamiento complejo. La sincronización de estos parámetros de sincronización puede ser demasiado compleja para el usuario y/o administrador pero procura garantizar prestaciones más adecuadas.

Este trabajo propone la evaluación del impacto de las aplicaciones intensivas de E/S en la planificación de trabajos en clusters no-dedicados bajo los paradigmas MPI y Mapreduce.

Palabras claves: planificación, no-paralelos, MPI, MapReduce

Índice

Lista de Figuras

Lista de Tablas

Lista de Ecuaciones

Capítulo 1 - Presentación del problema y objetivos

1.1- Introducción

Los clusters no dedicados se caracterizan por su capacidad de combinar la ejecución de aplicaciones de usuarios locales con aplicaciones, científicas o comerciales, ejecutadas en paralelo. La infrautilización de los clusters no-dedicados que encontramos – en Departamentos o en aulas de laboratorios de Universidades – ha generado una tendencia importante de estudio, hacia el diseño de sistemas capaces de ejecutar, aplicaciones paralelas, junto con las cargas locales de cada nodo del cluster en particular.

Ahora bien, para hacer uso de los recursos ociosos en este tipo de entorno, es posible utilizar diferentes enfoques: técnicas basadas en servicios de ejecución remota (Piranha (1) y Process Server (2)); las basadas en migración de tareas (Mosix (3), Condor (4), Stealth (5) y Sprite (6)); la construcción de maquinas virtuales(VM) y la utilización de técnicas de panificación de trabajos, que permitan la coexistencia de diferentes tipos de aplicaciones (locales, paralelas) con prestaciones adecuadas.

La combinación de cargas locales y cargas en paralelo es necesario realizarla de manera que no comprometa el rendimiento de las aplicaciones locales o las prestaciones de las aplicaciones paralelas.

Un planificador de trabajos es un sistema que, desde la gestión de los recursos del clúster y, a través de políticas pre-establecidas, realiza la distribución de los recursos del clúster para las aplicaciones presentados al entorno. La Figura 1.1 presenta un entorno paralelo no-dedicado, con carga local y carga en paralelo inyectada al cluster por un planificador de trabajos.

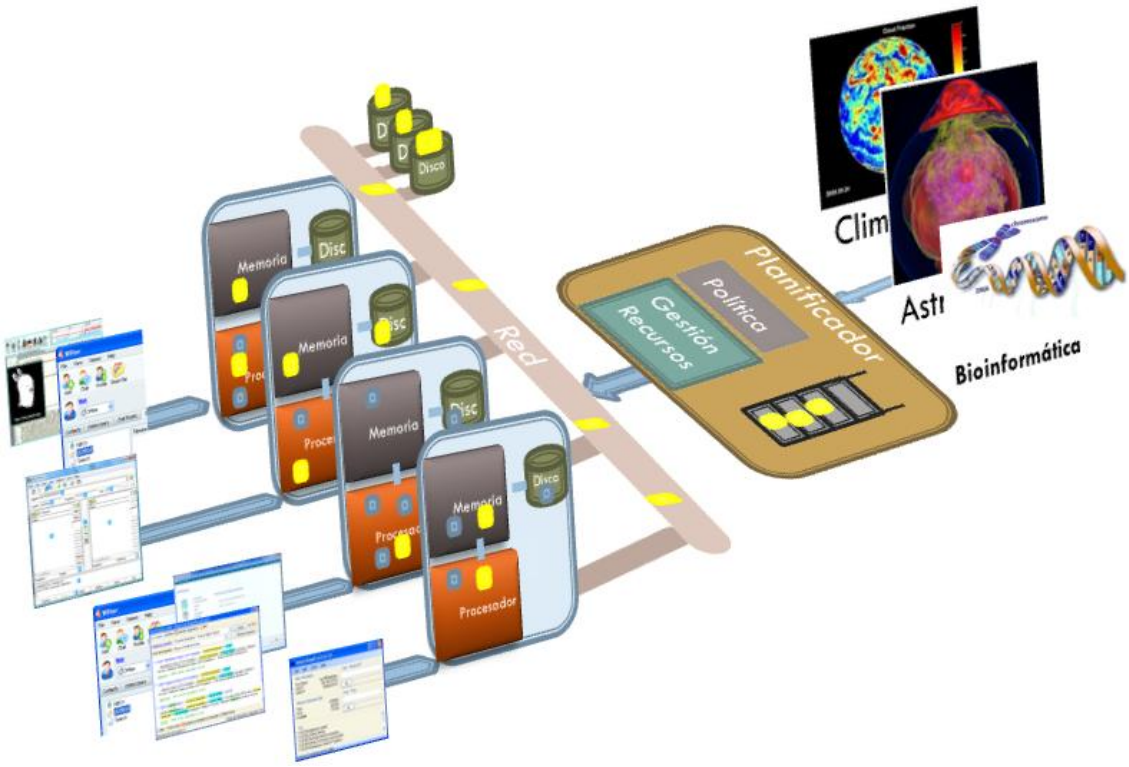

Figura 1.1.– Entorno paralelo no-dedicado

El trabajo de planificación se puede producir a dos niveles: a largo plazo (planificación espacial) y a corto plazo (planificación temporal). La planificación espacial tiene una granularidad más alta y es responsable por la distribución de los trabajos entre los nodos que componen el cluster. La planificación temporal tiene una granularidad más fina y es responsable de administrar y distribuir los recursos de cada nodo de cómputo entre las tareas que se ejecutan en el mismo nodo. La Figura 1.2 muestra los niveles en que pueden producirse la planificación de trabajos.

Varios factores influyen en el rendimiento de un planificador. Algunos de estos factores son estáticos, es decir, no cambian durante el proceso de planificación y pueden estar relacionados con el entorno o con la aplicación. Por ejemplo, el número de nodos disponibles en el clúster (suponiendo que ningún nodo se desconecta durante la ejecución de la planificación). Otros factores son dinámicos, es decir, son objeto de modificaciones durante el proceso de planificación, por ejemplo, la cantidad de carga que se distribuirá en un momento dado.

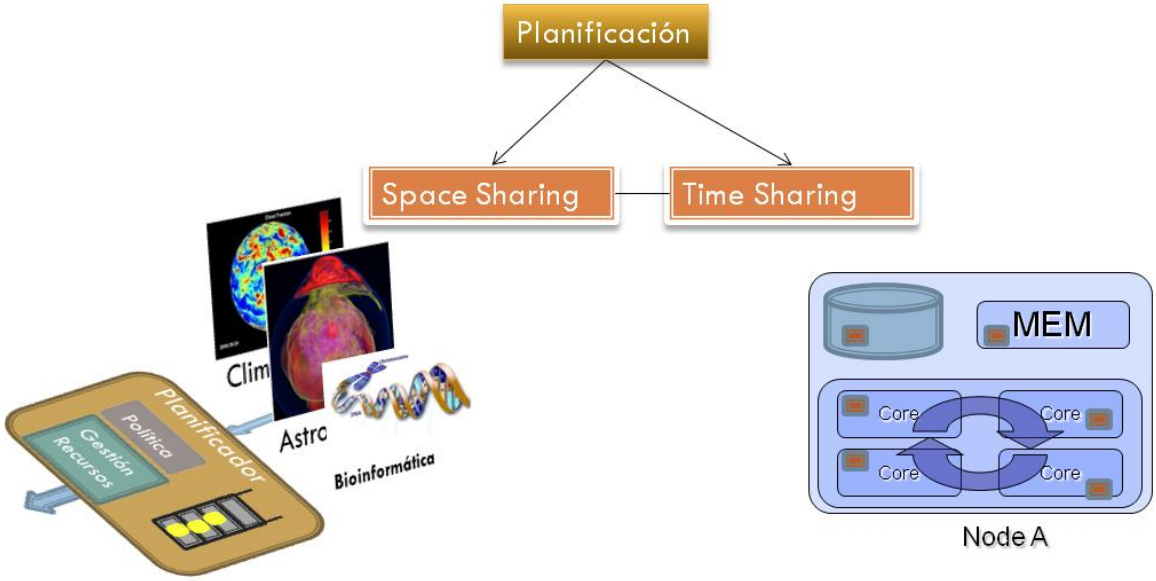

Figura 1.2- Niveles de la planificación de trabajos

Uno de los factores que contribuyen al proceso de planificación de los trabajos en los entornos no-dedicados es la tipología de aplicación que se ejecutará en el cluster. Las características de una aplicación determinan qué recursos del entorno serán utilizados por la misma. Por ejemplo, las aplicaciones con requerimiento de cómputo hacen uso intensivo del procesador, mientras aplicaciones con requerimiento de comunicación, hacen uso intensivo de la red de comunicación del entorno. Combinar adecuadamente la ejecución de estas aplicaciones, permite obtener prestaciones adecuadas en los clusters paralelos no-dedicados. La necesidad de ejecutar por parte de los usuários, aplicaciones de diferentes características, en el mismo cluster; así como la aparición de nuevas arquitecturas en los nodos de procesamiento; ha originado que esta línea de trabajo este en constante evolución.

Múltiples trabajos de investigación se han desarrollado, para evaluar los efectos que la combinación de diferentes cargas de diferentes características puede producir en el desempeño de los computadores paralelos: El efecto que las métricas, las cargas y la combinación de ellas producen en la evaluación de sistemas computacionales (7), la interacción de las cargas de trabajo paralelas y secuenciales en una red de estaciones de trabajo (8), la influencia de cargas

dinámicas en la planificación de trabajos paralelos (9), la planificación de tareas para ejecución balanceada de cargas en entornos distribuidos y heterogéneos (10), la reserva adaptativa de recursos en entornos paralelos para cargas locales y paralelas (11), la planificación de trabajos paralelos con demandas en QoS (12), la combinación de aplicaciones locales interactivas, ejecutándose junto con aplicaciones paralelas del tipo batch (13); son algunos de los más relevantes.

Nuestro grupo de investigación posee experiencia en la investigación y en el desarrollo de planificadores para clusters paralelos no-dedicados. Fruto de esta experiencia, ha sido desarrollado CISNE (Cooperative & Integral Scheduling for Non-dedicated Environments) [(13) (14) (15)] que construye una maquina virtual paralela (MVP) que mediante técnicas de planificación realiza la combinación de cargas locales del tipo Best-effort con aplicaciones paralelas en entornos no-dedicados.

A propuesta original de CISNE le fue extendida la siguiente funcionalidad: soporte a otros tipos de carga de trabajo. CISNE pasó a soportar no sólo cargas locales y paralelas del tipo Best-effort sino también cargas locales y aplicaciones paralelas del tipo SRT (SoftReal-Time) empezando a tratar la problemática derivada por la inclusión de múltiples cores en los nodos de cómputo (16).

El problema que impulsa esta investigación es evaluar el impacto de las aplicaciones intensivas de E/S a disco en la planificación de trabajos en el nivel espacial en clusters no dedicados. Como ejemplos de aplicaciones que requieren acceso a disco se pueden citar diversas aplicaciones científicas en las áreas de la física y la bioinformática. Ejecutar estas aplicaciones en un cluster no dedicado con los planificadores tradicionales requiere la transferencia de grandes cantidades de datos por la red y como resultado los tiempos de espera para acceder a estos datos. ¿Es posible tomar ventaja de estos períodos de inactividad de los nodos de cómputo, mientras que aplicaciones intensivas de E/S a disco esperan el acceso a los datos, e inyectar cargas en paralelo al entorno? ¿Cuál es la consecuencia de estas mezclas para el rendimiento de las aplicaciones? ¿Las mezclas pueden afectar

significativamente los indicadores de valoración de la gestión de los recursos distribuidos? (17) Estas son algunas preguntas que motivan la elaboración de este trabajo. También una multitud de trabajos se han centrado en este enfoque desde diferentes perspectivas. [(18), (19), (20), (21), (22)]

Entre los paradigmas más conocidos para la programación de aplicaciones en computadores paralelos/distribuidos, podemos mencionar: paso de mensajes (MPI), memoria compartida (Pthreads) y MapReduce entre otros. En función de las diferencias existentes entre ellos, la planificación de las aplicaciones construidas bajo estos paradigmas posee características distintas.

El paso de mensajes, a través de su interface MPI (message passing interface), es un paradigma que permite el intercambio de informaciones entre los procesos que componen la aplicación por medio de mensajes sin la necesidad de hacer uso de memoria compartida. La planificación de estas aplicaciones en un cluster se realiza de forma estática habitualmente y los procesos necesitan estar sincronizados (en ejecución o espera en un orden determinado). Esto requiere que la planificación de los procesos que forman la aplicación se presente de forma coordinada y conjunta.

El paradigma MapReduce se expresa mediante la construcción de dos funciones: Map y Reduce, definidas ambas con respecto a datos estructurados en pares (clave, valor). En la función Map los datos de entrada son procesados y es generada una lista de pares intermedios (clave, valor). A continuación la función Reduce se aplica sobre estas tuplas intermedias y hace la agrupación de los valores que tienen la misma clave. La planificación de las aplicaciones basadas en el paradigma MapReduce se realiza de forma dinámica y no requiere la comunicación entre los procesos en cada una de las etapas del Map y Reduce debido a que estos procesos son independientes. La Figura 1.3 muestra una comparación entre la planificación bajo los paradigmas MPI y MapReduce

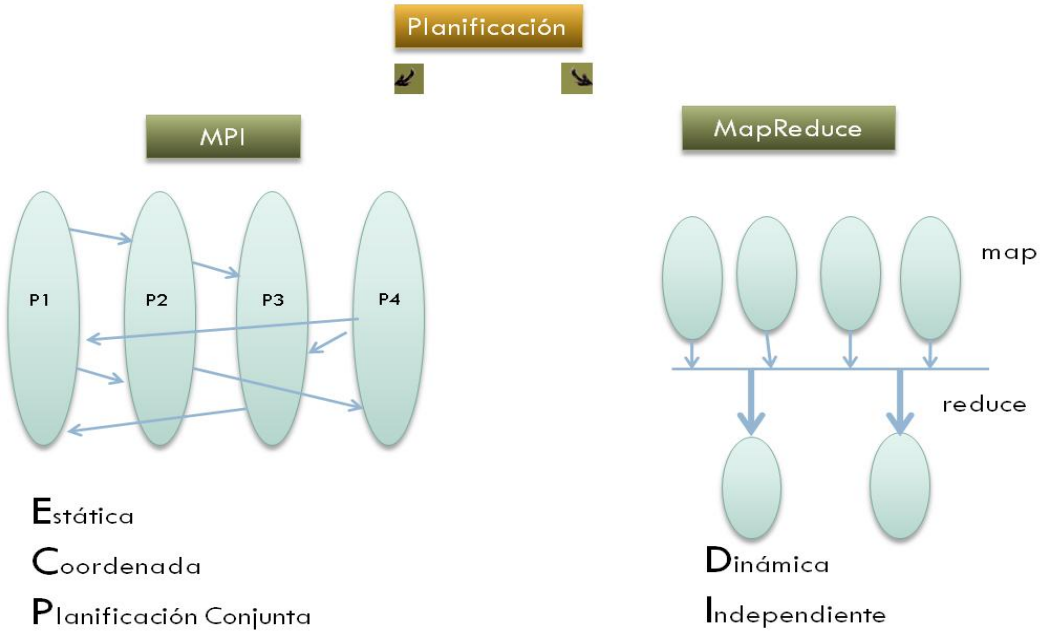

Figura 1.3– Planificación bajo los paradigmas MPI y MapReduce

Una de las características del paradigma MapReduce es la simplicidad para el programador que se centra en la construcción de las funciones Map y Reduce. Los planificadores de las aplicaciones basados en este paradigma implementan la tolerancia a fallos, se ocupan de la localidad de los datos, balanceo de carga de forma automática y trabajan con sistemas de archivos distribuidos.

Hadoop es un framework altamente configurable desarrollado en el proyecto Apache y que implementa MapReduce inspirado en la propuesta de Google (25) (27). Su implementación oculta del programador los detalles de paralelización, lo qué permite a programadores con poca experiencia en programación paralela, desarrollar aplicaciones para estos entornos.

El rendimiento de Hadoop se puede mejorar, sin aumentar los costos de hardware, al sintonizar parámetros de configuración claves para las especificaciones del cluster: para el tamaño de los datos de entrada y para el ajuste del entorno.

El framework Hadoop posee más de 190 parámetros ajustables. Hacer la sintonización de estos parámetros no es una tarea sencilla, porque un único parámetro puede tener efectos importantes en los parámetros de rendimiento

del sistema. Algunos de estos parámetros serán presentados en el capítulo 3 de este trabajo. El sistema posee una configuración por defecto de estos parámetros, y la sintonización entre ellos, puede ser realizada por el programador y/o administrador del sistema, a fin de obtener del sistema, las prestaciones más adecuadas.

Una posible forma de ejecutar de manera eficiente las aplicaciones con acceso a disco en los clusters no dedicados, es utilizando el paradigma MapReduce y su implementación en uno de los frameworks más conocidos (Hadoop). Cada vez más aplicaciones de Bioinformática son construidas a través de este paradigma, lo que genera la necesidad de planificadores capaces de manejarlas, junto con las aplicaciones basadas en otros paradigmas.

1.2 - Objetivos

Como objetivo general, y a largo plazo, se propone: *Diseñar nuevas políticas de planificación que tengan en cuenta: diferentes clases de aplicaciones y diferentes paradigmas de programación en entornos no dedicados.*

Y como objetivo general a corto plazo al alcance de esta tesina, se propone: *Estudiar la interferencia entre las aplicaciones intensivas de E/S y aplicaciones científicas con necesidades de cómputo y comunicaciones bajo el paradigma MPI y comenzar los estudios con los planificadores construido bajo el paradigma MapReduce .* A continuación se muestran la siguientes subtareas a abordar:

- Analizar el rendimiento de una aplicación paralela MPI, con requerimientos de E/S a disco, con la localización de los datos en un servidor remoto, o en los discos locales de los nodos de cómputo.
- Estudiar el rendimiento de una aplicación con requerimientos de E / S a disco construida bajo el paradigma MapReduce.

- Familiarizarse con las herramientas de planificación y de medida de prestaciones sobre un cluster, tanto de aplicaciones MPI como aplicaciones MapReduce.

1.3 - Organización de este trabajo

Este trabajo está organizado en cuatro capítulos. En el capítulo 2, discutiremos la planificación de aplicaciones paralelas en entornos no dedicados. Se realizará una descripción de sistemas paralelos, se presentarán los planificadores de trabajo en clusters no-dedicados y se discutirá el impacto de la E / S en el comportamiento de los programas paralelos y en la planificación del trabajo.

El capítulo 3 presentará una breve descripción sobre el paradigma MapReduce, y se describirá la planificación de los trabajos en entornos Hadoop

En el capítulo 4 se presentará una descripción de los experimentos. Se describirán las aplicaciones utilizadas en los experimentos, los ambientes en los que fueron ejecutados y los indicadores de evaluación que se han aplicado. A continuación, se mostrarán los resultados obtenidos mediante la ejecución de los experimentos, y se realizará un análisis de los mismos. Por último, se describirán las conclusiones y se propondrán líneas futuras de continuación.

Capítulo 2 - Planificación de aplicaciones paralelas MPI en entornos no dedicados

Las necesidades de cómputo de las aplicaciones han crecido a lo largo del tiempo de manera ininterrumpida. Para atender estas demandas, la investigación en computación ha seguido principalmente por dos caminos: incrementar la capacidad de cómputo de los ordenadores, y desarrollar técnicas de cómputo distribuido y paralelo. Por otro lado, han surgido nuevas aplicaciones paralelas con requerimientos más estrictos de tiempo de retorno (turnaround), de calidad de servicios (QoS) y con acceso a volúmenes de datos cada vez más grandes.

Estas aplicaciones poseen necesidades propias de recursos de memoria, CPU y ancho de banda de red e disco. Combinar la ejecución de estas aplicaciones paralelas, cada vez más complejas con cargas locales en entornos paralelos no-dedicados, es un desafio para los investigadores en computación de alto rendimiento (HPC).

2.1 - Introducción a los entornos paralelos no dedicados

A continuación, nos proponemos situar los cluster de naturaleza no-dedicada, dentro de los entornos de cómputo paralelo más habituales. Esta caracterización, nos permitirá fijar aspectos críticos de los mismos, en relación a los tipos de planificadores más adecuados en cada caso.

La Figura 2.1 presenta una taxonomía para los sistemas paralelos propuesta por Hanzich (15), basada en las taxonomías de Flynn y Tanenbaum, donde es posible clasificar diferentes sistemas disponibles para el cómputo distribuido y paralelo.

En el ámbito de los multicomputadores una opción son las maquinas tipo Massive Parallel Processors (MPP). Estos computadores están basados en máquinas con un gran número de procesadores y un alto nivel de acoplamiento entre ellos que proporcionan elevada capacidad de cómputo. Para su funcionamiento, estas maquinas necesitan de hardware y software especifico con costos económicos elevados.

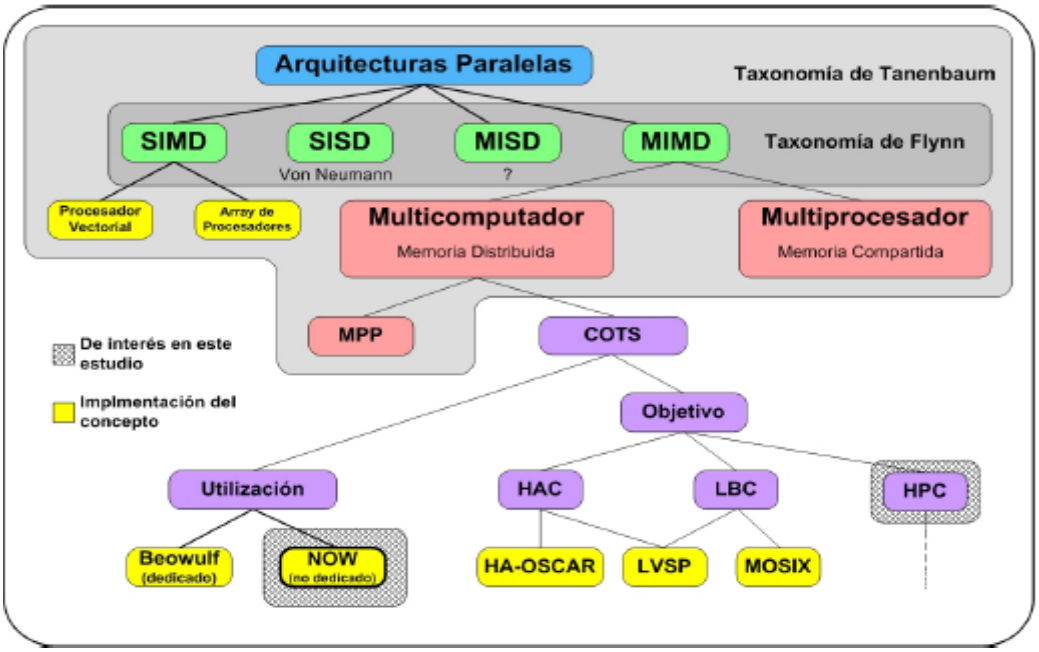

Figura 2.1 - Taxonomía aplicada a los sistemas paralelos.

Por este motivo la utilización de computadores de alto rendimiento basados en máquinas MPPs ten disminuido a revés da utilización de clusters. La Figura 2.2, presenta la distribución de los distintos sistemas paralelos, dentro de la lista del Top500, en las dos últimas décadas.

Es posible construir clusters a partir de componentes comerciales fácilmente accesibles y de bajos costos económicos. Estos son clusters del tipo COTS

(Commodity Of-The-Shelf) cada vez más comunes pero poseen niveles de acoplamiento más bajos que las MPPs.

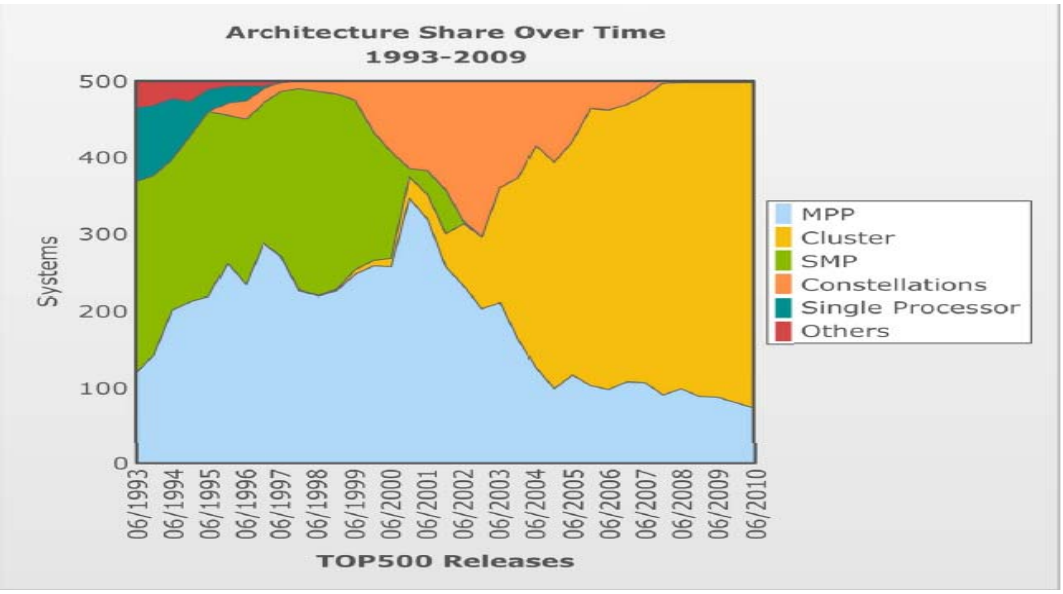

Figura 2.2 – Distribución de la arquitectura de los computadores en la lista Top500 de las últimas dos décadas

Los clusters tipo COTS pueden ser clasificados, a partir de la taxonomía propuesta, de acuerdo con la utilización de sus recursos o de acuerdo con su objetivo. De acuerdo con su objetivo pueden ser de alta disponibilidad (HAC – High Availability Clusters) como HA-OSCAR (23), con foco en el balanceo de carga (LBC, Load Balancing Clusters) como MOSIX (24) o de altas prestaciones (HPC, High Performance Computing) como Maui (25)

Y de acuerdo con la utilización de sus recursos los clusters tipo COTS pueden ser clasificados en dedicados o no-dedicados. Los clusters dedicados son conocidos como Beowulf y poseen como características la ejecución solamente de cargas controladas por el sistema de gestión del cluster y los recursos sintonizados para una única aplicación. Los clusters no-dedicados poseen como característica la ejecución de cargas no controladas por el sistema de gestión del cluster. En estos clusters las aplicaciones paralelas comparten los recursos disponibles con cargas locales. Esto hace con los clusters no-dedicados tengan costo económico aún más bajo que las máquinas

MPPs. Los clusters no-dedicados son muy utilizados por universidades en laboratorios de clases, aulas de informática y por empresas.

La posibilidad de que los clusters no-dedicados ejecuten cargas locales y aplicaciones paralelas exige, la investigación de técnicas que permitan la combinación de estas cargas sin comprometer los requerimientos de las aplicaciones locales y ni las prestaciones paralelas.

Una de las técnicas utilizadas a efecto de eliminar los periodos de inactividad en los clusters, y permitir la ejecución de aplicaciones paralelas, es mediante la implementación de planificadores, que analizan las características de las aplicaciones y el estado de los recursos del entorno durante la planificación de las aplicaciones paralelas, generando políticas que permiten la compartición de recursos.

2.2 - Planificación de trabajos

La planificación de trabajos tiene como objetivo proporcionar una utilización más eficiente de los recursos computacionales, a través de políticas que determinan los criterios de elección de los trabajos a ser ejecutados. El paradigma de programación utilizado para construir la aplicación influencia en esta planificación. Las aplicaciones construidas bajo el paradigma de paso de mensaje en clusters, habitualmente son planificadas de forma estática, la planificación de trabajo se realiza antes de la ejecución; en tiempo de carga o compilación.

Como se ha visto en el capítulo anterior, la planificación de aplicaciones en un cluster no-dedicado ocurre a dos niveles: Planificación Espacial donde se decide en qué procesadores se ejecutan los procesos, y Planificación Temporal que define la política de planificación en cada procesador individual decidiendo que proceso se ejecuta. En el ámbito de este trabajo de investigación está la

planificación espacial. La Figura 2.3 presenta una taxonomía para el proceso de planificación de procesos en clusters no dedicados.

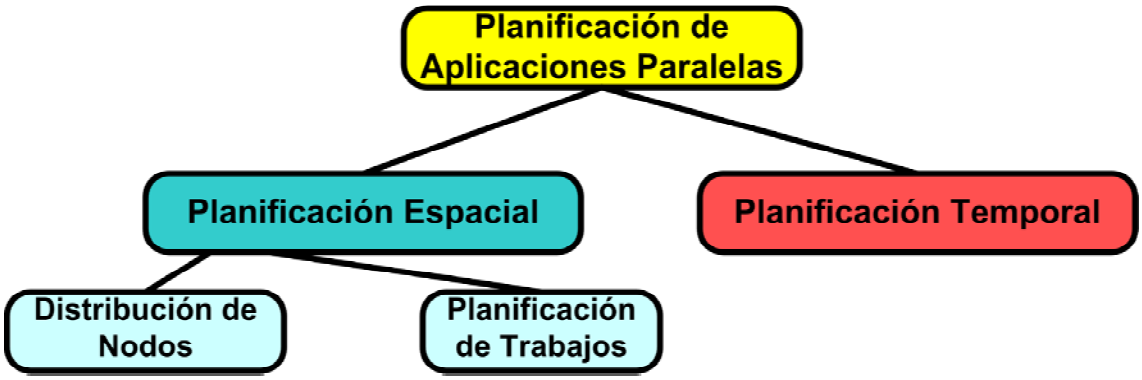

Figura 2.3 – Taxonomía de La Planificación de Procesos Paralelos

La planificación espacial posee dos retos efectivos: repartir los nodos entre las aplicaciones que serán ejecutadas y hacer la asignación de los recursos a estas aplicaciones en cada nodo.

2.2.1 – Distribución de los nodos

La Figura 2.4 presenta de forma esquemática como ocurre la distribución de los nodos en un planificador de aplicaciones en un cluster no-dedicado. En esta fase se reparten los nodos del cluster éntre las aplicaciones y después se eligen los nodos para cada una de las aplicaciones que serán ejecutadas

Entre las alternativas con las cuales se pude hacer el particionamiento de los nodos, se puede citar la estática, y la variable que permite un mejor balanceo de la carga en el sistema.

Después que los nodos están asignados, se puede elegir donde se ejecutará la aplicación. La manera más simple de seleccionar un nodo es de forma binaria. En esta política de selección un nodo solamente puede ser asignado a

una aplicación caso esté libre, es decir, el grado de multiprogramación (Multi Programming Level – MPL) de esta política es 0.

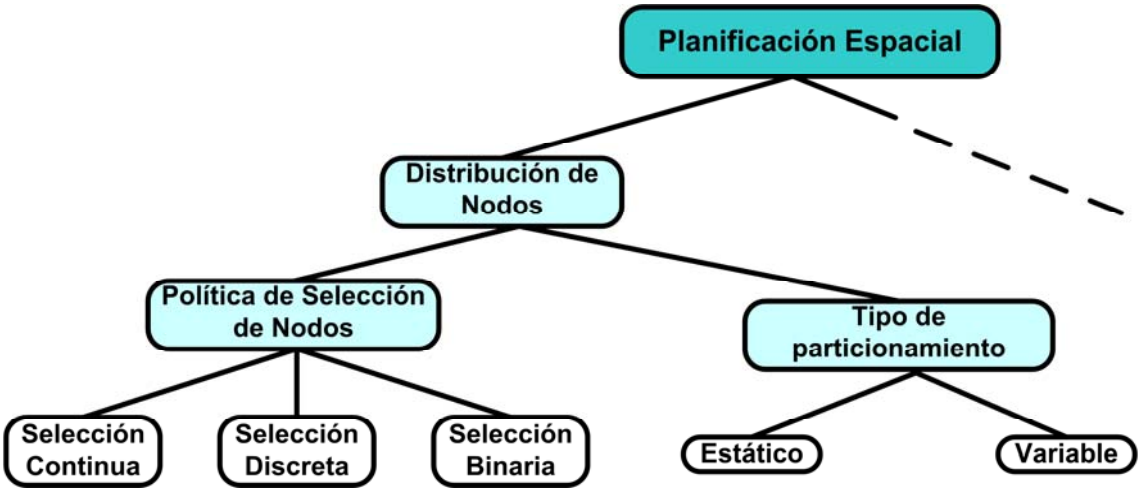

Figura 2.4 – Etapa de distribución de nodos

La política que permite un MPL más grande que uno es llamada de selección discreta. Como en esta política más de una aplicación puede ser asignada para el mismo nodo, existe la necesidad de combinar el espacio compartido con el tiempo compartido.

La política de selección continua de nodos es aquella que permite la asignación de aplicaciones paralelas a nodos donde es posible ejecutarse cargas locales.

2.2.2 – Planificación de Trabajos

La Figura 2.5 presenta las etapas en que ocurre la planificación de trabajos. En esta etapa son definidas la orden en que los trabajos permanecen en espera de ser ejecutados y de qué forma serán seleccionados de esta cola.

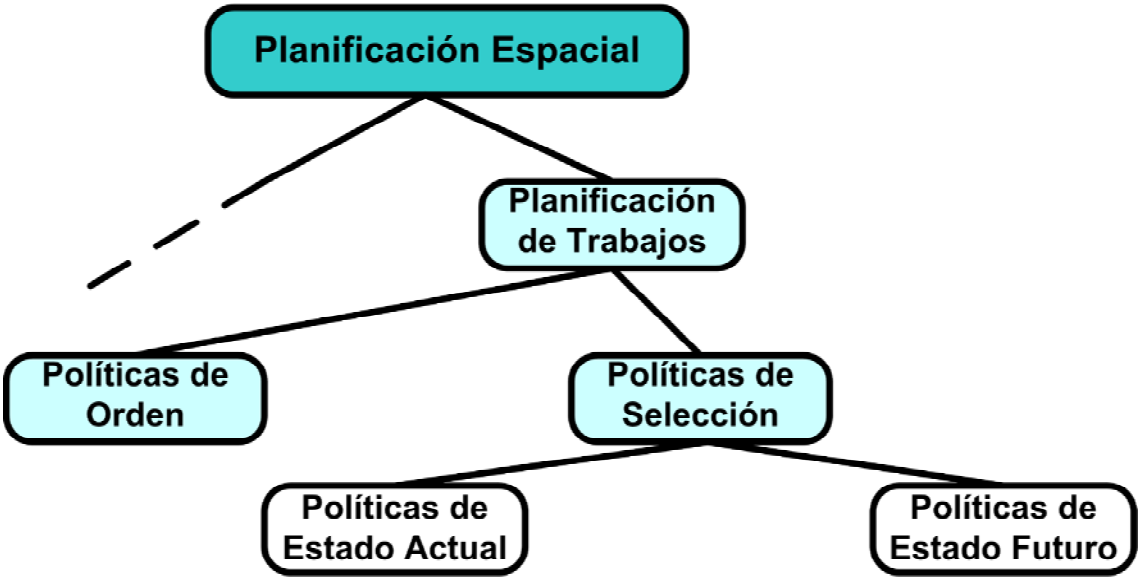

Figura 2.5 – Etapa de Planificación de Trabajos

Cuando un nuevo trabajo es inyectado en el cluster y no puede ser ejecutado en el momento, este es añadido a una cola de espera en el planificador. La manera como este trabajo será añadido a cola es definido por la política de orden. Algunas de las políticas de orden son: FCFS (First Come Firt Served) donde el ordenamiento obedece la orden de inyección en el sistema; SJF (Shrotest Job First) donde la cola es ordenada de forma creciente en relación al tiempo de ejecución del trabajo; y SNPF (Smallest Number of Processes First) donde la cola es ordenada de acuerdo con el número de procesadores que solicita

Una vez que la cola de trabajo esté ordenada se puede seleccionar el trabajo a ser ejecutado en el cluster. Aunque se pueda elegir el primero trabajo de la cola existen políticas de selección que llevan en consideración el estado actual del cluster o su estado futuro. De acuerdo con el resultado de estas consideraciones estas políticas pueden elegir un trabajo para ser ejecutado que no sea el primero trabajo en la cola. Estas políticas que procuran utilizar el conocimiento del estado del cluster son las más usuales

Un ejemplo de planificador de aplicaciones es Sun Grid Engine (SGE) desarrollado por SUN MicroSystem y que está disponible en el Departamento CAOS de la UAB. SGE en su versión 6.1, fue uno de los entornos utilizados para el desarrollo de este trabajo de investigación.

Sun Grid Engine es un planificador de aplicaciones que permite la gestión de los recursos distribuidos. Él procura acomodar las necesidades de software y hardware del usuario, a los recursos distribuidos en la red, respetando las políticas de planificación definidas por el administrador del sistema.

SGE agrupa los recursos de un cluster en colas. Un cluster posee una o muchas colas. En cada una de ellas están definidos los recursos disponibles (nodos, CPU, memoria,…), el tipo de particionamiento y la política de selección de estos nodos. Para cada cola también están definidas las políticas de ordenamiento y selección que serán aplicadas a los trabajos sometidos a cola. Todas estas definiciones son responsabilidad del administrador del sistema. Un nodo de un cluster puede hacer parte de más de una cola estando sometido a las políticas de planificación de cada una de ellas.

Cuando un usuario inyecta una aplicación en el cluster utilizando SGE, especifica los requisitos de la aplicación como el tipo de cpu, cantidad de disco y memoria necesarios o el tiempo máximo de ejecución de la aplicación y SGE inyecta esta aplicación en la cola más apropiada. El usuario puede también someter un trabajo directamente en una cola específica

La definición de las políticas de distribución de nodos, y de las políticas de planificación de trabajos, afectan directamente las prestaciones. Ajustar la definición adecuada de estas políticas, a la carga de trabajos del sistema, es una tarea compleja, debido a la gran cantidad de parámetros de ajuste que proporciona el sistema.

.

Capítulo 3 - Planificación de trabajos en clusters Hadoop

3.1 - El paradigma de programación MapReduce

MapReduce es un modelo de programación que permite el procesamiento y la gestión de grandes volúmenes de datos. Los dados de entrada son divididos en fragmentos de tamaños menores para que sean procesados de forma intermedia generando resultados parciales. Estos resultados parciales son procesados una vez más para generar el resultado final. El paradigma MapReduce es utilizado por empresas como Google (26) para el procesamiento de archivos de log o busqueda de textos en documentos, por ejemplo; Facebook (27) para detección de spam, data-mining y optimización.

La programación es dividida en dos funciones desarrolladas por el programador: Map y Redulce. El programador define una estructura de dados del tipo <clave, valor> y la función Map es aplicada, de forma paralela y distribuida, sobre cada fragmento en que los dados de entrada tengan sido divididos. Lo objetivo de la función Map es recorrer cada fragmento de dados buscando las coincidencias con la <clave> definida por el programador. Para cada coincidencia encontrada por la función Map es generada una tupla también en el formato <clave, valor> definida por el programador. Estas tuplas son almacenadas de forma temporaria y serán procesadas a seguir por la función Reduce. Para cada fragmento de dados en que la entrada tenga sido dividida es generada una lista de tuplas por la función MAP. Estas listas son entonces agrupadas y procesadas por la función Reduce.

La función Reduce actúa sobre los datos temporarios que han sido generados por la función Map, haciendo el procesamiento de los valores que poseen la misma <clave>. La salida de la función Reduce también es una lista de <clave, valor> representando el resultado final de la aplicación. La Figura 3.1 presenta un esquema de MapReduce.

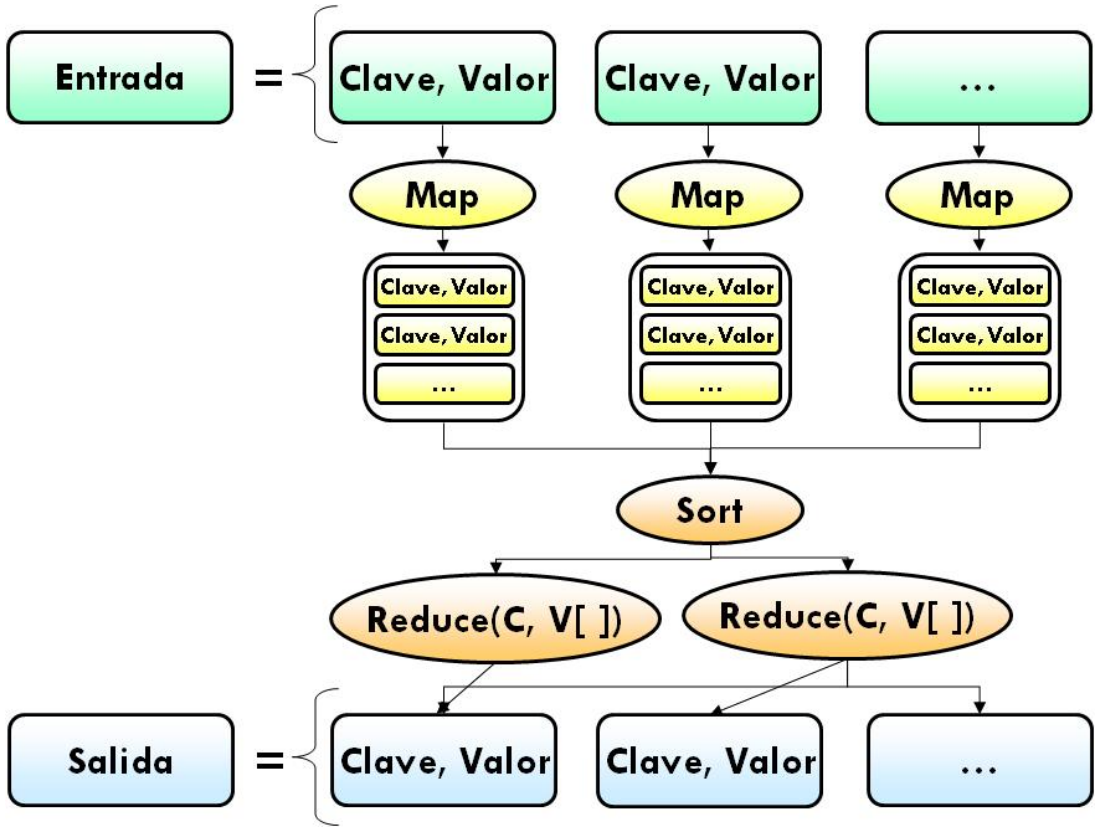

Figura 3.1 – Esquema de Mapreduce

Para ilustrar el paradigma MapReduce usamos una aplicación de tipo WordCount que recibe como entrada un fichero de datos y genera como salida el número de ocurrencias de cada palabra al respecto. La estructura de entrada definido por el programador es una tupla<w,1> donde <w> es cualquier palabra que se encuentre en el archivo de texto. El archivo de entrada se divide en fragmentos más pequeños y a cada fragmento se aplicará la función Map en

paralelo y distribuido. Para cada fragmento del archivo se generará una lista de tuplas en el formato <palabra encontrada, 1>. La Figura 3.2 muestra la función Map que actúa sobre fragmentos del archivo de entrada y genera como salida una lista temporal de tuplas.

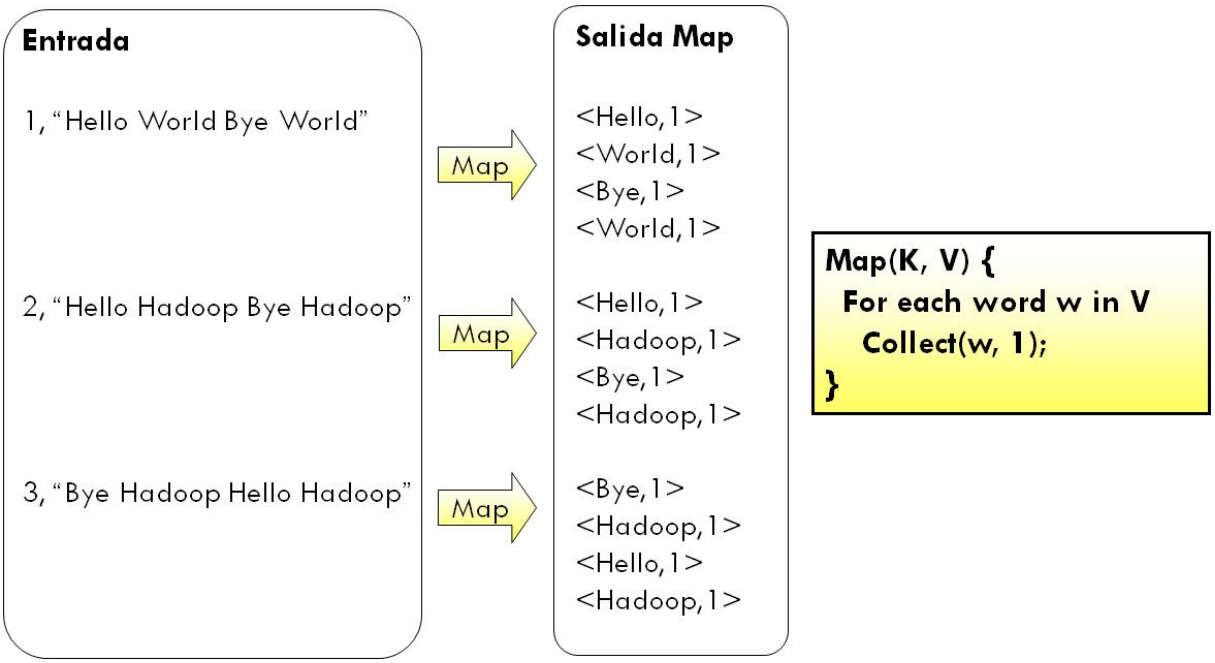

Figura 3.2 – Ejemplo de función Map

Las listas de tuplas generadas por la función Map pasan entonces por un proceso de agregación de los valores que poseen la misma chave que ha sido definida por el programador, en esto ejemplo <cada palabra>. El resultado de la agregación es un conjunto de tuplas en el formato <cada palabra, lista de valores iguales a 1>.

El resultado de la agregación de las listas de tuplas generadas por la función Map se presentan entonces a la función Reduce. La función Reduce cuenta el número de ocurrencias de cada palabra generando así el resultado final. La Figura 3.3 presenta un esquema para la función Reduce.

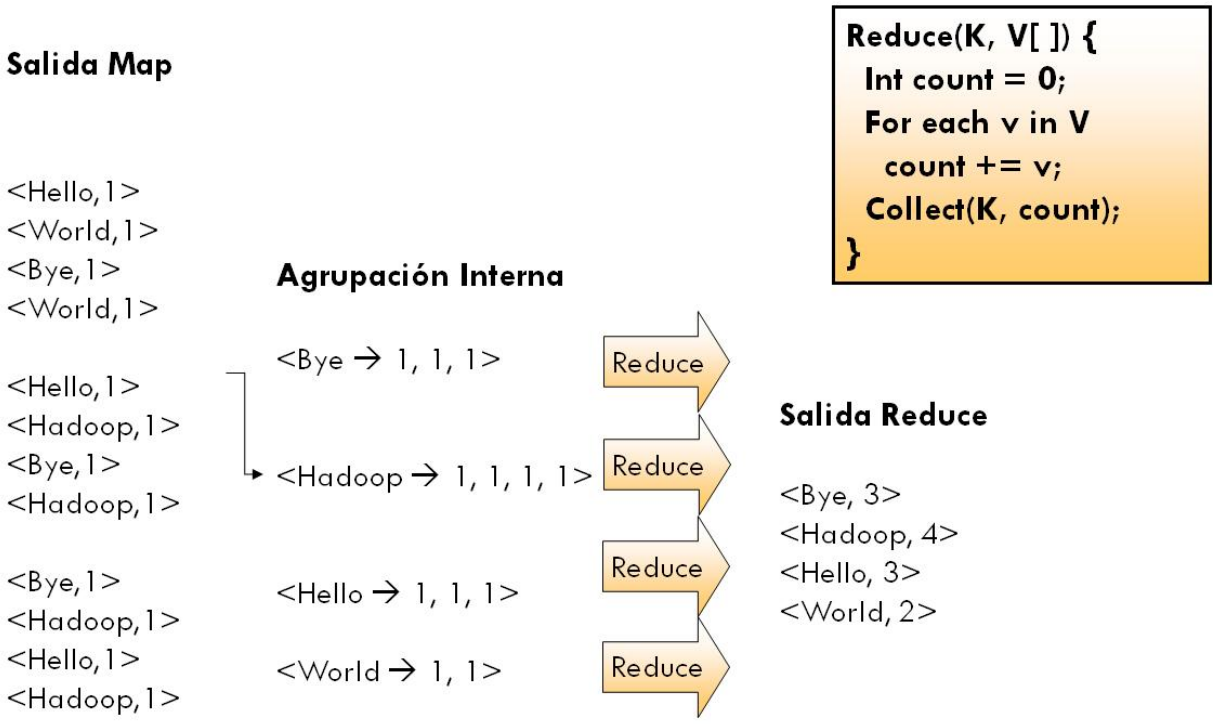

Figura 3.3 – Ejemplo de la función Reduce

Cada una de las funciones Map y Reduce pueden ser divididas en varios procesos y cada uno de ellos actúa de forma independiente sin necesidad de comunicación con los demás. Una de las ventajas del paradigma MapReduce es permitir que el programador tenga entonces el foco en el desarrollo de las funciones Map y Reduce sin necesidad de desarrollar código para el intercambio de mensaje entre los procesos. Otra ventaja del paradigma MapReduce es que evita el tráfico de una cantidad considerable de datos a través de la red, un cuello de botella para los entornos paralelos.

Cada vez más el paradigma MapReduce es utilizado para la construcción de aplicaciones con acceso a disco y, en consecuencia, es natural explorar su uso en plataformas HPC.

3.2 - Planificación de trabajos en clusters Hadoop

Hadoop es un framework desarrollado en el proyecto Apache y que implementa MapReduce inspirado en la propuesta de Google (26) (28). Es un sistema de código abierto y después de su implementación original para clusters han sido implementadas versiones para varias arquitecturas incluso para Cell B.E (29), GPUs (30) y procesadores multi-core (31).

El framework Hadoop hace de forma automática la fragmentación y distribución de los archivos de entrada, la planificación de los trabajos entre los nodos del entorno paralelo, hace el controle de fallos de los nodos y gestiona la necesidad de comunicación entre los nodos del cluster.

Hadoop se ejecuta sobre un sistema de archivos distribuidos, Hadoop Distributed File System – HDFS que es capaz de almacenar archivos de tamaños grandes a través de muchas máquinas. La fiabilidad es obtenida por la replicación de los dados a través de las máquinas del cluster sin necesidad de implementación de Raid.

El desarrollo de Hadoop está basado en el modelo master/work donde es implementado el motor MapReduce: Job Tracker (el nodo master) y Task Tracker (los nodos Works).

Los trabajos MapReduce son inyectados al sistema en una cola gestionada por Job Tracker. Los trabajos en colados son divididos por Hadoop en un conjunto de tareas. Estas tareas son asignadas a cada uno de los Task Trackers. Cuando un Task Tracker finaliza su tarea informa su estado al Job Tracker que hace la asignación de una nueva tarea. La Figura 3.4 presenta el sistema de cola de trabajos de Hadoop.

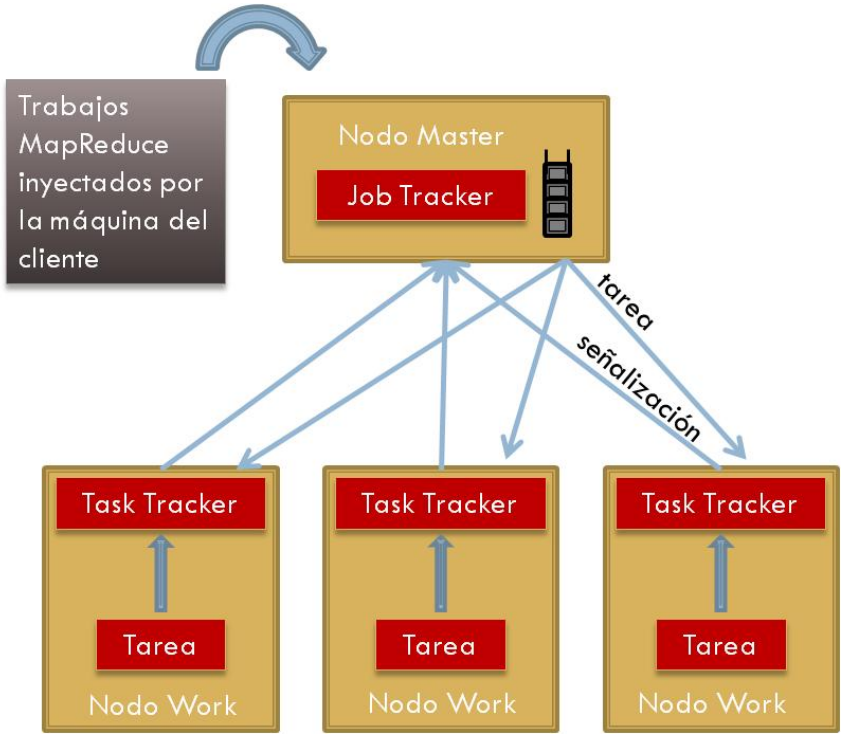

Figura 3.4 – Componentes del Scheduler en Hadoop

Las primeras tareas asignadas a los Task Trackers son las tareas de Map. Estas tareas aplican la función Map a los fragmentos del archivo de dados de entrada que están distribuidos por los nodos Works del cluster. Hadoop hace la distribución de las tareas a los TaskTrakers, de modo que estén más cerca de los nodos cuanto sea posible. Las tuplas <clave, valor> generadas por cada tarea Map, son almacenadas temporalmente en los discos locales de cada Task Traker.

Cuando todos los Maps han terminado, el Job tracker bajo petición de los Task Traker, empieza a distribuir las tareas Reduce. Estas tareas aplican la función Reduce a las listas de tuplas generadas por las tareas Map. La Figura 3.5 ilustra una ejecución MapReduce en el framework Hadoop.

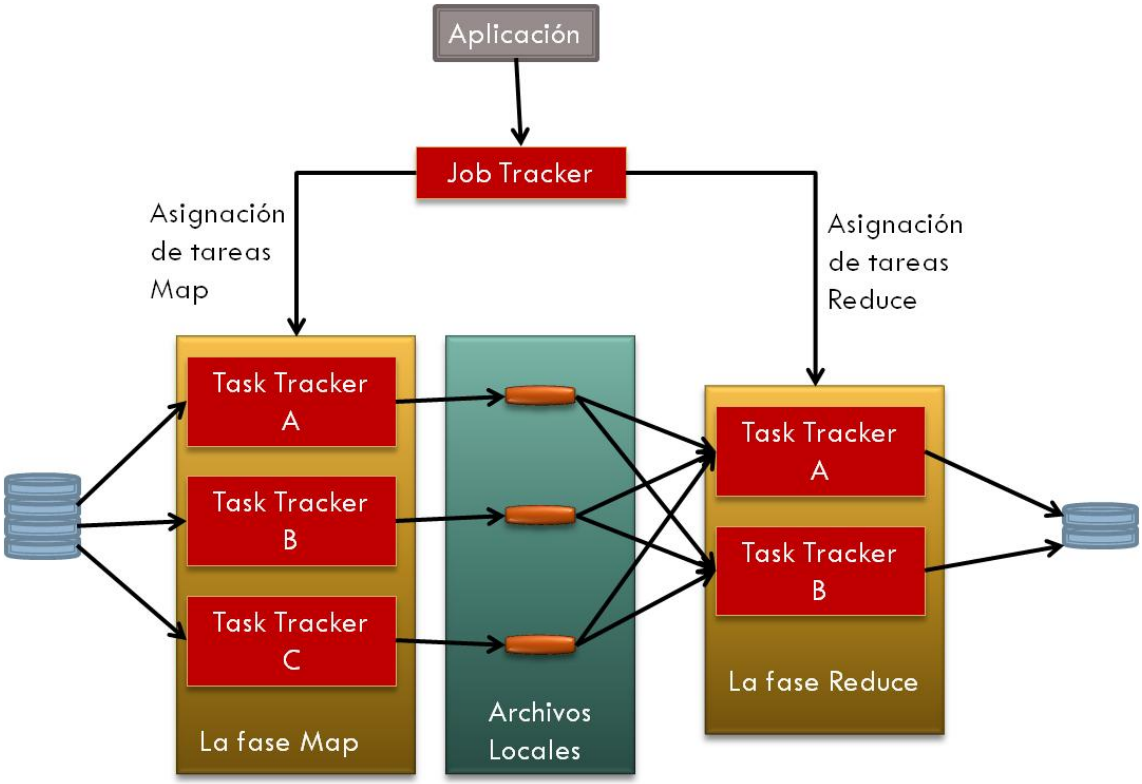

Figura 3.5 – Ejecución de Hadoop

Hadoop ejecuta varias tareas Map y Reduce al mismo tiempo en cada Work - dos en cada uno de forma predeterminada – para que se solapen la computación y E / S. Cada Task Tracker le dice al Job Tracker cuando éste ha finalizado sus tareas. El planificador entonces asigna nuevas tareas al Task Tracker. La planificación de las tareas se realiza de forma dinámica.

Como MapReduce es altamente paralelizable, Hadoop necesita hacer control de fallos. El Master hace la verificación periódica de la finalización de las tareas a ejecutar. En el caso de que una tarea a realizar, no conteste al Master en un intervalo de tiempo determinado, esta es marcada como en estado de fallo. Todas las tares de Map que han sido concluidas pero no finalizadas, serán planificadas una vez más. Además todas las tareas de Map y Reduce en ejecución en el nodo que ha fallado también serán planificadas otra vez. La necesidad de replanificar las tareas de Map ya concluidas, es porque los dados intermediarios generados por la función Map, son almacenados en el disco local de cada máquina. Así, en caso de que haya ocurrido fallo en una maquina, no es posible acercar los dados a su disco local.

Las versiones más recientes de Hadoop, también hacen checkpoints periódicos de las estructuras de datos existentes en el Master. En caso de que ocurra un fallo en el Master, una nueva ejecución a partir del punto de checkpoint puede ser comenzada en un nuevo Master elegido en el cluster.

3.3 - Sintonización

El rendimento de Hadoop se puede mejorar sin aumentar los costos de hardware, al sintonizar varios parámetros de configuración claves para las especificaciones del cluster, para el tamaño de los datos de entrada y para el procesamiento complejo.

El framework Hadoop posee más de 190 parámetros ajustables. Realizar la sintonización de estos parámetros, no es una tarea sencilla porque un único parámetro puede tener efecto sobre los demás y sobre el rendimiento del sistema. La Figura 3.6 presenta algunos de estos parámetros.

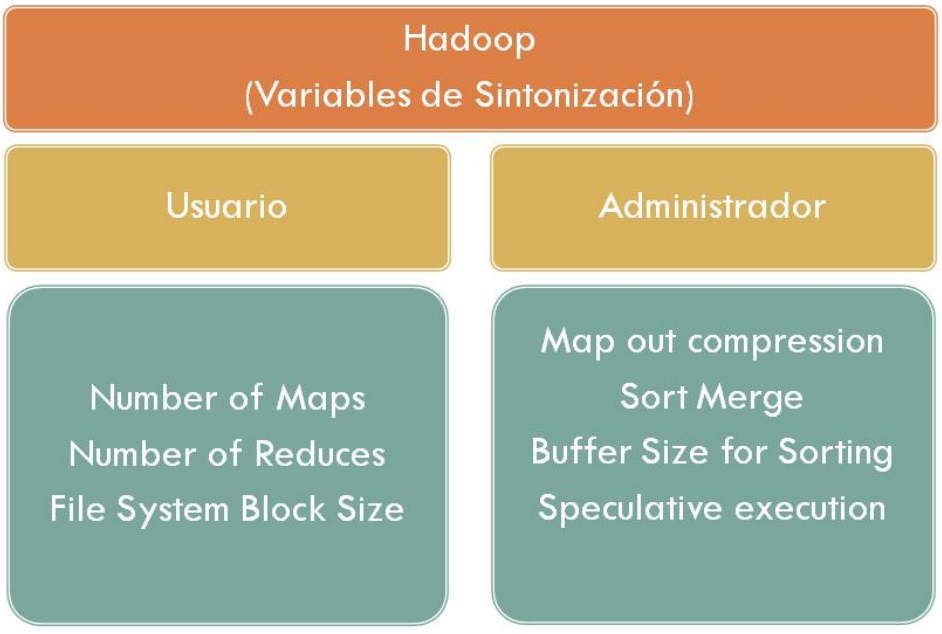

Figura 3.6 - Parámetros de Sintonización HADHOOP

El sistema posee una configuración por defecto de estos parametros y la sintonización entre ellos, puede ser realizada por el programador y/o administrador del sistema para obtener prestaciones más adecuadas.

El número de veces que las funciones Map y Reduce serán aplicadas en cada una de las etapas de la ejecución, deberá ser mucho mayor que el número de nodos. La granularidad de las tareas permite un balanceo más equilibrado de cargas dinámicas y además puede acelerar la recuperación cuando ocurre un fallo en un nodo, ya que los datos temporales pueden estar dispersos en el cluster.

El tamaño de los fragmentos del archivo de datos de entrada, y el número de veces que cada uno de estos fragmentos será replicado en los nodos del cluster influyen en la localidad de los datos, y en el consumo del ancho de banda de la red.

El número de tareas que se asignan a cada nodo al mismo tiempo, permite el solapamiento de tareas de cómputo con tareas de acceso a datos. Sin embargo, si este número es demasiado grande, puede causar una sobrecarga en las tareas que se están ejecutando.

Los planificadores de las aplicaciones basados en este paradigma, implementan la tolerancia a fallos, se ocupan de la localidad de los datos, balanceo de carga de forma automática y trabajan con sistemas de archivos distribuidos.

La sintonización de estos parámetros, puede ser demasiado compleja para el usuario y/o administrador, pero procura garantizar prestaciones más adecuadas.

Capítulo 4 - Experimentación realizada y resultados obtenidos

El objetivo de los experimentos llevados a cabo durante la investigación, fue evaluar el impacto que la mezcla de diferentes tipos de aplicaciones provoca, en sus tiempos de ejecución, cuando estas aplicaciones son inyectadas a un entorno paralelo no-dedicado. Consideramos dos escenários; cuando los datos están almacenados en los discos locales de los nodos de cómputo, y cuando los datos están en un servidor de datos accedido de forma remota. En la misma línea, proponemos evaluar los efectos que esta mezcla de aplicaciones provoca, en relación al throughput del sistema, a partir de un conjunto de experimentos.

La investigación se dividió en cuatro fases:

- En la primera fase realizamos experimentos, para evaluar la influencia de la mezcla de aplicaciones en el tiempo de ejecución en relación al throughput. Se han utilizado aplicaciones con requerimientos de cómputo (NAS-ep.B) y con requerimientos de comunicación (NAS-lu.B).
- En la segunda fase, sustituimos la aplicación NAS-lu.B, por una aplicación del tipo WordCount intensiva de E/S a discos, y diseñamos experimentos a efectos de evaluar: la influencia en el tiempo de ejecución, y en relación al throughput cuando esta aplicación es ejecutada en paralelo con la aplicación NAS-ep.B.
- En la tercera fase, se llevaron a cabo experimentos para evaluar la influencia de la ubicación de los datos. Dos escenários fueron evaluados; considerando que los datos están almacenados en los discos locales de los nodos de cómputo, y cuando los datos están en un servidor accedido de forma remota a través de NFS.

- Y en la cuarta y última fase, se ha desarrollado una aplicación (WordCount), bajo el paradigma MapReduce y se evalua su escalabilidad, en un entorno paralelo con planificación de trabajos bajo el paradigma anterior.

4.1 - Métricas

Durante la investigación fueron utilizadas las siguientes medidas de prestaciones:

Tiempo de Ejecución: tiempo necesario para ejecutar un trabajo submetido al cluster no-dedicado.

Throughput: definido como el número total de trabajos enviados al cluster no-dedicado durante la realización del experimento, dividido por el tiempo necesario para la completa ejecución de todos estos trabajos. La Ecuación 4.1 presenta cómo es efectuado el cálculo de throughput.

$$Throughput = \frac{N}{T_N}$$

Ecuación 4.1 – Cálculo de troughput

Donde,

N = Número total de trabajos enviados al cluster

T_N = Tiempo total necesario para la ejecución de todos los trabajos en el cluster no-dedicado.

4.2 - Entorno

Para la ejecución de los experimentos fueron utilizados dos entornos distintos. El primer entorno paralelo utilizado, fue el cluster IBM del Departamento de Arquitectura de Computadores y Sistemas Operacionales de la Universitat Autònoma de Barcelona, representado por la Figura 4.1.

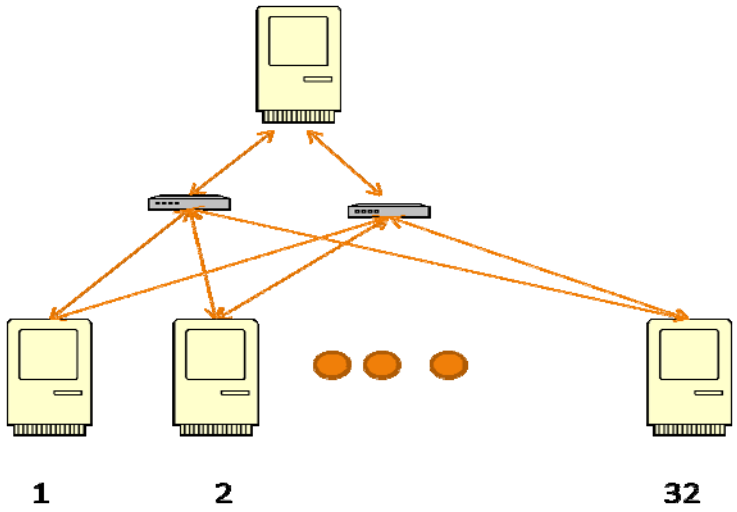

Figura 4.1 – Representación del entorno 1

En el cluster IBM fueron ejecutados los experimentos utilizando las aplicaciones construidas bajo el paradigma MPI. El cluster está constituído por un nodo servidor de datos del tipo Dual-Core Intel(R) Xeon(R) CPU 5150 y por 32 nodos de cómputo del tipo 2 x Dual-Core Intel(R) Xeon(R) CPU 5160. En el nodo servidor de datos existen 6 discos con capacidad individual de almacenamiento de 300GB, totalizando 1,8TB de capacidad de almacenamiento, y está implementado con RAID 5. La gestión y el acceso a los datos son hechos utilizando el protocolo NFS – Network File System, en la versión 3, que fornece acceso remoto transparente al sistema de archivos a través de la red de interconección. El sistema operacional del nodo servidor de datos es el Linux 2.6.16 en su distribuición SUSE.

También está implementado en el nodo servidor de datos el planificador de trabajos SGE - Sun Grid Engine en su versión 6.1. SGE se encarga de la

aceptación, envío y gestión de la ejecución remota y distribuida de tareas en espacio de usuario ya sean individuales, paralelas o interactivas. Entre las funcionalidades de SGE, también se encuentra la posibilidad de gestionar y programar la reserva de recursos distribuidos a lo largo de la plataforma; como procesadores, memoria, espacio de disco y licencias de software.

Cada uno de los nodos de cómputo, posee como sistema operacional Linux 2.6.16 en su distribuición SUSE y posee también un disco local de 160GB. Los nodos de cómputo están ligados al servidor de datos a través de dos redes Gigabit Ethernet, una de ellas solamente para la transferencia de datos y la otra para la comunicación de las aplicaciones y administración del sistema. Para llevar a cabo los experimentos fueron utilizados cuatro de estos nodos, y la ubicación del archivo de datos accedido por las aplicaciones utilizadas en los experimentos, fue alternada entre el servidor de datos y el disco local de cada máquina.

El cluster IBM no puede ser utilizado para realizar experimentos utilizando aplicaciones construidas bajo el paradigma MapReduce, ya que las aplicaciones construidas bajo este paradigma, necesitan de un sistema de archivos distribuidos, y el planificador que hace la distribuición de las tareas debe ser capaz de soportarlo. La versión de Sun Grid Engine instalada, no soporta la planificación de aplicaciones construidas bajo el paradigma MapReduce, y la versión del protocolo NFS no soporta la gestión de archivos distribuidos.

El segundo entorno paralelo (Figura 4.2) utilizado para llevar a cabo los experimentos con las aplicaciones del tipo MapReduce, fue un cluster no dedicado, compuesto por PC´s e interconectado mediante una red de interconexión a 100Mbits/segundo.

Este cluster esta constituido por cuatro máquinas. Desde la perspectiva Hadoop, el sistema esta formado por un nodo Master y 3 máquinas realizando tareas de Worker. La planificación de trabajos, y la gestión del sistema de archivos por HDFS - Hadoop Distributed File System, se centraliza en el nodo master. A partir de la metainformación localizada en el nodo master, HDFS crea múltiplas replicas de bloques de datos y los distribuye por los nodos de

cómputo, permitiendo que el acceso a los mismos pueda ocurrir de forma paralela.

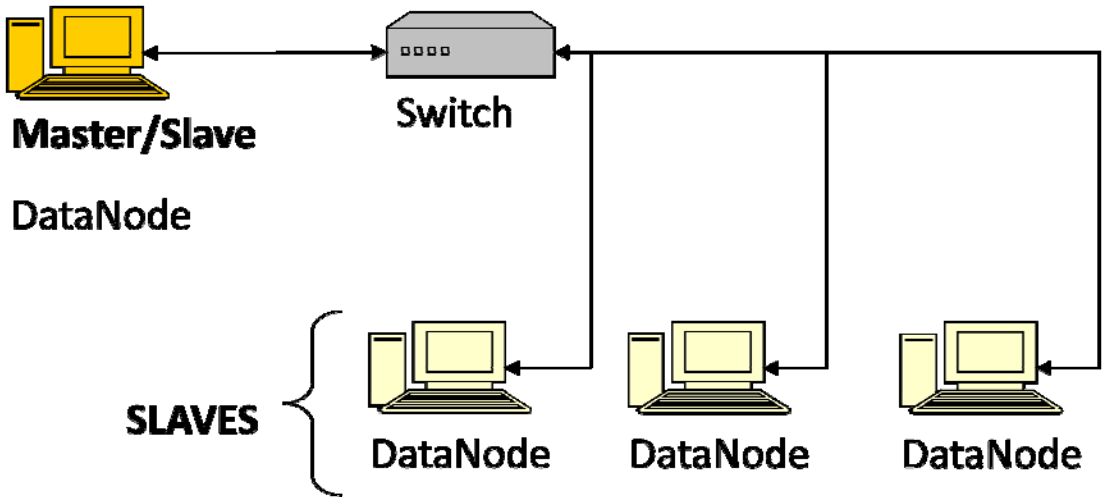

Figura 4.2 – Representación del entorno 2

4.3 - Aplicaciones

Para realizar la investigación fueron utilizadas tres aplicaciones durante los experimentos. Una aplicación del tipo WordCount intensiva de aceceso a disco, aplicaciones con requerimientos de cómputo (NAS-ep.B) y de comunicación (NAS-lu.B).

4.3.1 - NAS Parallel Benchmarks

Los benchmarks paralelos NAS fueron desarrollados por el programa NAS - Numerical Aerodynamic Simulation, ubicado en la NASA Ames Research Center, con el objetivo de medir el rendimiento de ordenadores paralelos. Estos

benchmarks, fueron desarrollados a partir de códigos de dinámica de fluidos computacional – CFD y han tenido gran aceptación entre la comunidad científica, para evaluar el rendimiento de los superordenadores.

Todos los resultados experimentales estan basados en la implementación MPI de los benchmarks NAS. Los códigos utilizados fueron EP (Embarrassingly Parallel) y LU (LU solver) desarrollados en Fortran 77 y compilados con la clase de tamaño B.

La aplicación LU utiliza SSOR (symmetric successive over-relaxation) para resolver un sistema de ecuaciones en tres dimensiones, resultante de la discretización por diferencias finitas de las ecuaciones de Navier-Stockes. El método de resolución genera un “pipelining” diagonal llamado por sus autores de “wavefront”. Como resultado del método de implementación, se genera el direccionamiento de un número relativamente grande de pequeños mensajes (40 bytes), hecho que torna la aplicación intensiva en comunicación. El patrón de comunicación de la aplicación LU cuando es ejecutada como 4 procesos es presentado en la Figura 4.3. Este código requiere un número de procesadores que sea potencia de dos.

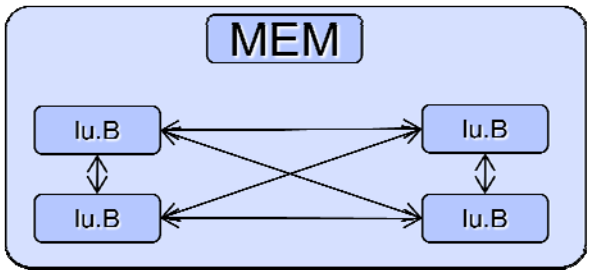

Figura 4.3 – Patrón de comunicación NAS-LU

El benchmark EP proporciona una estimación de la capacidad de cómputo, en operaciones de tipo punto flotante, sin necesidad de comunicación significativa entre los procesos. EP es embarazosamente paralelo. Basicamente, el benchmark hace la acumulación de estadísticas bi-dimensionales, a partir de la generación de gran cuantidad de números pseudo-aleatórios de Gaussian, creados a partir de un esquema particular. El

patrón de comunicación utilizado por el benchmark EP, cuando es ejecutado por cuatro (4) procesos es presentado por la Figura 4.4.

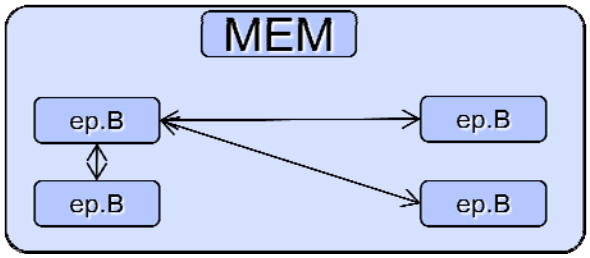

Figura 4.4 – Patrón de comunicación NAS-EP

4.3.2 - Aplicación WordCount

Se construyó una aplicación del tipo WordCount bajo la arquitectura master/worker utilizando las funciones MPI de la biblioteca MPI – IO. La aplicación es intensiva de acceso a disco. Las funciones elegidas para su construcción se ejecutan con llamadas colectivas, utilizandose punteros independientes y la aplicación posee un patrón de acceso secuencial a los datos.

La aplicación cuenta los números de ocurrencia de cada palabra existente en un archivo de entrada, que puede estar en un servidor de datos remoto, o en el disco local de cada uno de los nodos del cluster que ejecutarán la aplicación (Figura 4.5)

La aplicación WordCount fue dividida en tres etapas distintas. En la primera etapa, ocurre todo el proceso necesario para la inicialización de la aplicación, como la asignación de memoria que será utilizada, la apertura del archivo de texto, la división del archivo en partes más pequeñas y la distribuición de estas partes menores para cada proceso con el objetivo de ser procesadas. En la segunda etapa, ocurre el cómputo propiamente dicho efectuado por cada proceso. En esta etapa, cada proceso identifica las palabras existentes en la

porción del archivo de texto que está bajo su responsabilidad, y genera una lista de tuplas formada por las palabras y el número de ocurrencia de cada una de ellas. En esta etapa, ocurre el acceso de forma intensiva al archivo donde está el texto a ser procesado. Sin embargo, es importante resaltar, que el archivo puede estar ubicado en un servidor de datos remoto en el cluster no-dedicado, o estar ubicado en el propio nodo de cómputo en su disco local. Esta distinción permitió que se evaluara el efecto de la ubicación de los datos, en la mezcla de aplicaciones, aprovechándose el tiempo de espera de los recursos, mientras el sistema realizaba la lectura de los datos en el archivo de texto.

Figura 4.5 – Aplicación WordCount

La tercera y última etapa, es donde se recogen los resultados por parte del proceso master. Cada una de las listas de tupla generadas por los procesos workers, es agrupada en una única lista ordenada, ocurriendo la suma de palabras iguales presentes en listas distintas. En este etpa, también se genera el cierre del archivo texto, y la liberación de los espacios de memoria reservados anteriormente. La Figura 4.6 presenta el diseño de estas etapas de la aplicación WordCount.

La aplicación WordCount fue construida utilizando las funciones MPI de la biblioteca MPI-IO. El acceso a archivos através de funciones MPI, permite la creación de imágenes, que definen los datos que están disponibles para cada proceso. Estas imágenes poseen tres elementos básicos: el desplazamiento

que es un offset a partir del início del archivo, el tipo elemental de datos existente en el archivo, y un modelo de acceso.

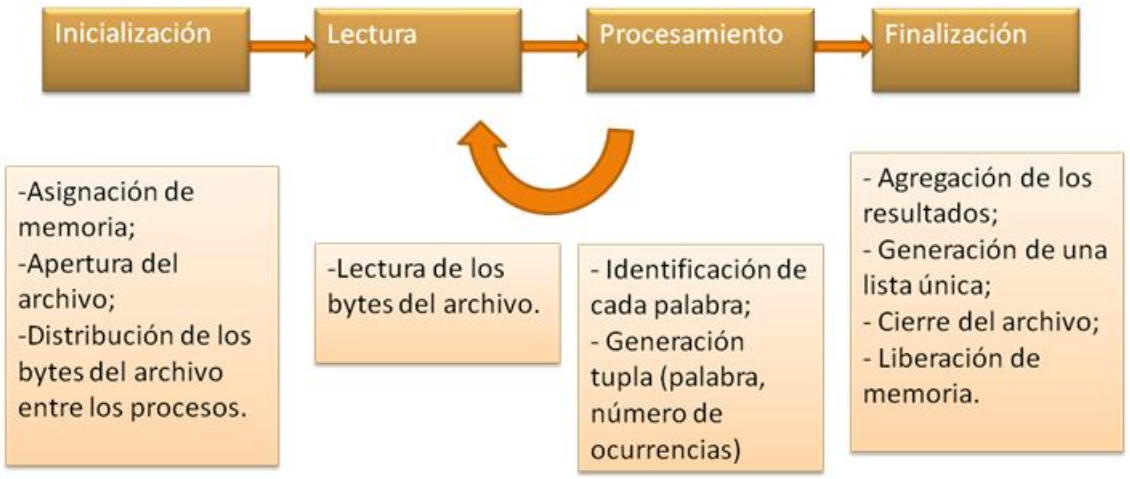

Figura 4.6 – Aplicación Wordcount

Las funciones MPI-IO de acceso a datos pueden ser clasificadas, de acuerdo con la forma como son coordinadas en: no-colectivas que se restringen al acceso a los datos por un único procesador, o colectivas que permiten el acceso a los datos por todos los procesos de una única vez. Como en las llamadas colectivas todos los procesos actúan de forma coordinadas, las solicitaciones de tamaño pequeño pueden ser mezcladas permitiendo una ganancia en el desempeño cuando comparado con las llamadas de acceso a los datos no-colectivas. Todos los experimentos realizados en el entorno de esta investigación utilizaran llamadas colectivas.

Las rutinas de acceso a datos soportadas por las funciones MPI-IO permiten el uso de punteros individuales mantenidos de forma independiente por cada proceso. Cada proceso tiene su propio puntero de acceso a los datos que es incrementado después de cada operación de llectura/escrita de forma independiente. Para permitir un solapamiento entre E/S y cómputo, todas las funciones MPI-IO utilizadas son del tipo no-bloqueantes que significa que retornan antes mismo de finalizar la transferencia de datos. La Figura 4.7

resume la distribución de las funciones MPI-IO utilizadas por la aplicación wordcount.

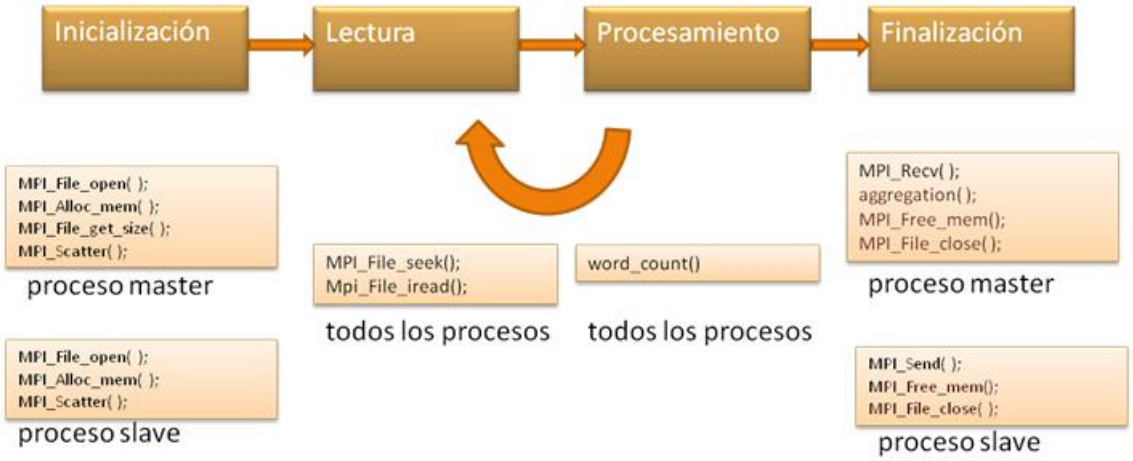

Figura 4.7 – Funciones MPI-IO utilizadas en WordCount.

4.4 - Experimentos

Los experimentos fueron divididos en cuatro etapas. En la primera etapa, se buscó evaluar el impacto en relación al tiempo de ejecución y throughput, provocados por la mezcla de aplicaciones intensivas en comunicación con las aplicaciones intensivas en cómputo. Las aplicaciones NAS-lu.B y NAS-ep.B fueron utilizadas. La metodología utilizada durante el proceso fue: Se ejecutó primero la aplicación lu.B en forma serial y despues en paralelo con 2 y 4 procesos en un único nodo de cómputo, con 8 procesos en dos nodos de cómputo y finalmente, con 16 procesos en 4 nodos de cómputo.

La Figura 4.8 presenta el escenario de los experimentos con la aplicación lu.B. Es importante la comprensión de la distribuición de los procesos, para permitir una mejor interpretación de los resultados obtenidos con los experimentos. Se observa, que cuando la aplicación fue dividida en dos procesos distribuidos en un mismo nodo de cómputo, cada uno de estos

procesos ocupó un core en una cpu distinta del nodo. Esta distribución permitió por parte de cada proceso, un mejor uso de los recursos disponibles en cada nodo de cómputo. Cuando la aplicación fue dividida en cuatro procesos distribuidos en un mismo nodo de cómputo, cada uno de ellos ocupó un core distinto del nodo de cómputo. La ocupación de todos los cores disponibles en un mismo nodo de cómputo, también ocurrió cuando la aplicación fue dividida en 8 y 16 procesos.

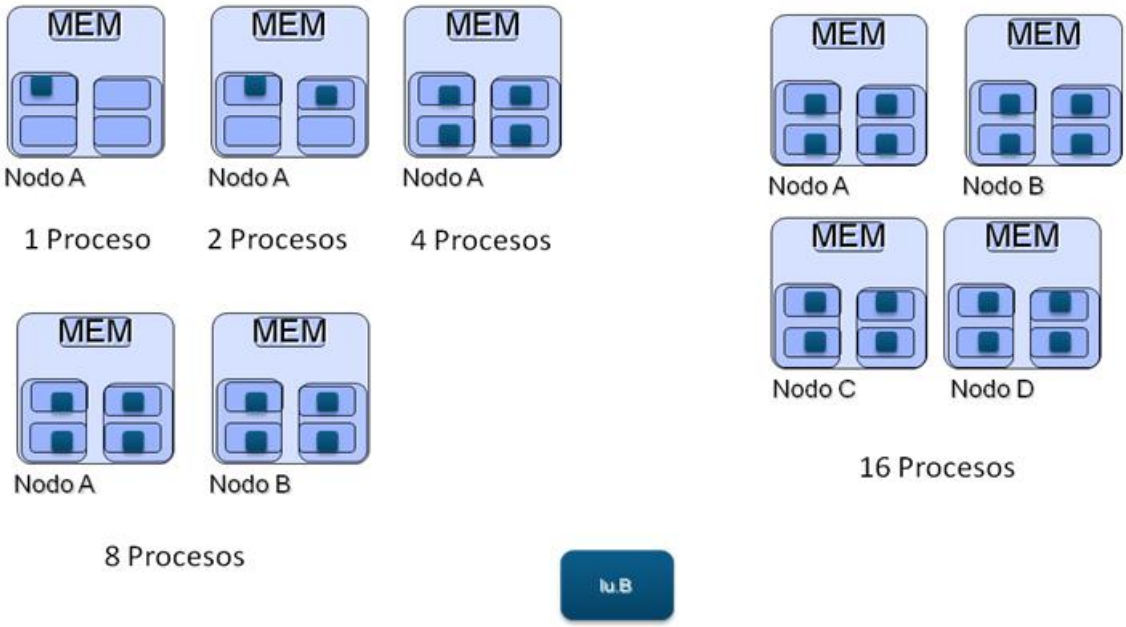

Figura 4.8 – Escenario del experimento con lu.B

La aplicación NAS-lu.B fue ejecutada con control de carga en los nodos de cómputo. Es decir, no hubo carga local en cada uno de los nodos utilizados durante los experimentos, tampoco hubo la existencia de aplicaciones en paralelo excepto la carga provocada por la propia aplicación NAS-lu.B. La inyección de la aplicación en el cluster no paralelo ocurrió a través del planificador de trabajos SGE, que tiene un sistema de cola para trabajos submetidos al cluster. La política de atribuición de trabajos implementada en el cluster y utilizada durante todos los experimentos fue la First Come First Serve (FCFS).

Cada ejecución de la aplicación NAS-lu.B fue repetida ocho veces. Los resultados obtenidos fueron la média de los valores medidos, descartándose los tres peores valores.

Al finalizar los experimentos con la aplicación NAS-lu.B se repetieron los mismos experimentos con la aplicación NAS-ep.B. Se ejecutó la aplicación ep.B en forma serial y después en paralelo con 2 y 4 procesos en un único nodo de cómputo, con 8 procesos en dos nodos de cómputo y por fin, con 16 procesos en 4 nodos de cómputo.

La Figura 4.9 presenta el escenario de los experimentos con la aplicación ep.B. La distribución de los procesos por cada nodo de cómputo también repetió el mismo patrón de la aplicación lu.B.

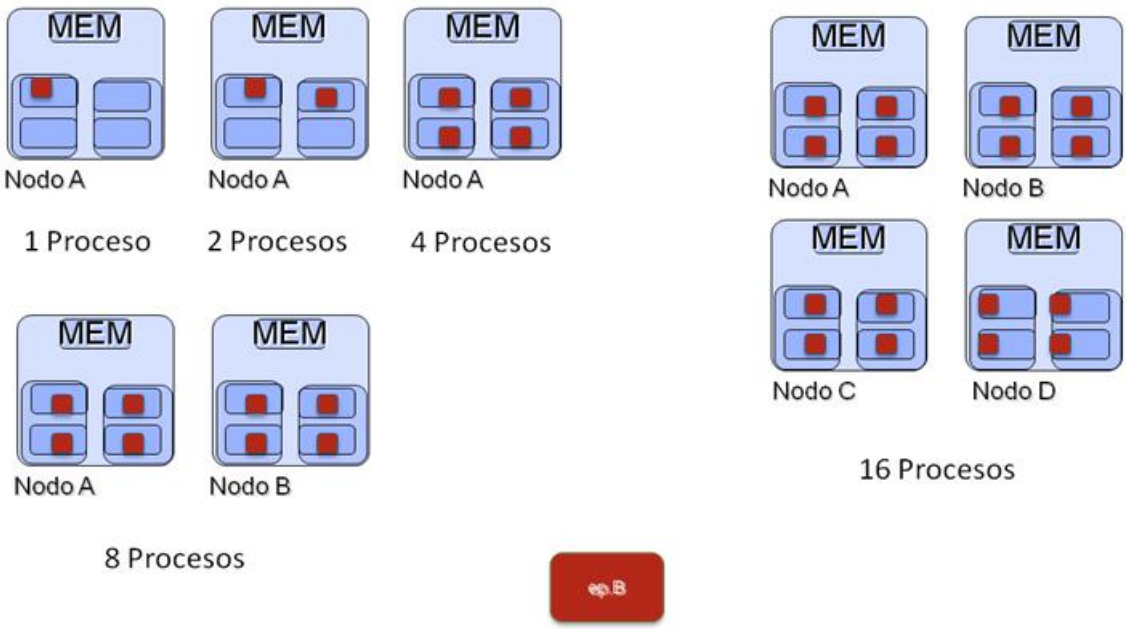

Figura 4.9 – Escenario del experimento con ep.B

Cada ejecución de la aplicación NAS-ep.B fue repetida ocho veces. Los resultados obtenidos fueron la média de los valores medidos, descartándose los tres peores valores.

Después de la ejecución de cada una de las aplicaciones NAS de forma independiente, se iniciaron los experimentos ejecutándose las dos aplicaciones

de forma concurrentes para evaluar el impacto de la mezcla de ambas aplicaciones sobre el tiempo de ejecución y el throughput.

Se ejecutaron las aplicaciones ep.B e lu.B en forma serie y concurrente, dividiendo el mismo nodo de cómputo. Después se ejecutaron ambas aplicaciones en paralelo, y de forma concurrente, con 2 y 4 procesos en un único nodo de cómputo, con 8 procesos en dos nodos de cómputo y por fin, con 16 procesos en 4 nodos de cómputo.

La Figura 4.10 presenta el escenario de los experimentos con las aplicaciones lu.B y ep.B. Cuando ejecutadas de forma serial y concurrente en el mismo nodo de cómputo, cada aplicación ocupó una CPU distinta haciendo, en función de la arquitectura de las máquinas que componen el cluster, con que la ejecución tuviese las mismas características de cuando las aplicaciones fueron ejecutadas isoladamente. Se observa que cuando las aplicaciones fueron divididas en dos procesos distribuidos en un mismo nodo de cómputo, cada aplicación ocupó una CPU distinta del nodo y los procesos de cada aplicación ocuparan así los cores de una misma CPU. Esta distribución permitió, por parte de cada proceso, un mejor uso de los recursos disponibles en cada nodo de cómputo. Cuando las aplicaciones fueron divididas en cuatro procesos distribuidos en un mismo nodo de cómputo, cada un de ellos ocupó un core distinto del nodo de cómputo, hecho que provocó la competencia de todos los recursos del nodo de cómputo. Todos los cores disponibles en un mismo nodo de cómputo estuvieron ocupados cuando las aplicaciones fueron divididas en 8 e 16 procesos.

Las aplicaciones NAS-lu.B y NAS-ep.B también fueron ejecutadas con control de carga en los nodos de cómputo. Así no hubo carga local en cada uno de los nodos utilizados durante los experimentos, tampoco hubo la existencia de aplicaciones en paralelo excepto las aplicaciones provocadas por las propias aplicaciones. Para asegurar el efecto de la competencia sobre ambas aplicaciones durante todo el tiempo de ejecución, el benchmark NAS-ep.B, que tiene un tiempo de ejecución menor del que la aplicación NAS-lu.B, tubo de ser ejecutado más veces en cada uno de los experimentos. La Tabla 4.1 presenta el número de ejecución de cada uno de los benchmarks.

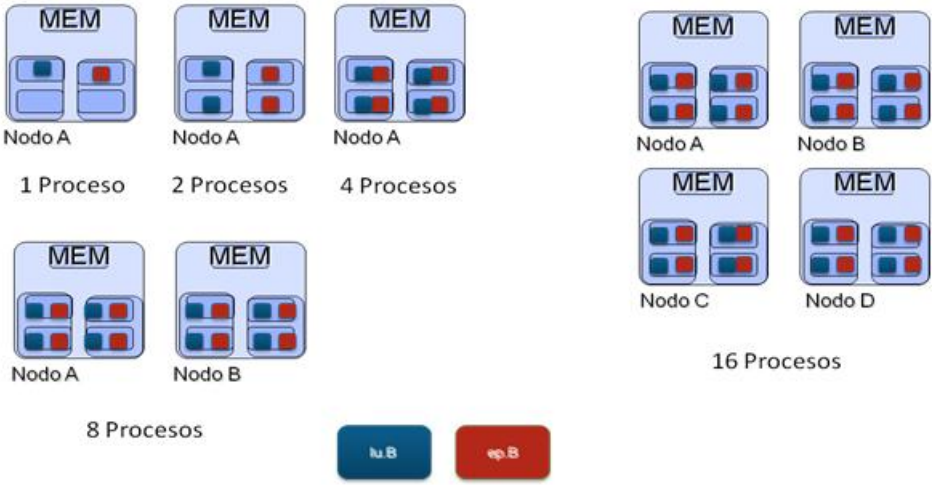

Figura 4.10 – Escenario del experimento con lu.B y ep.B

La inyección de las aplicaciones en el cluster no paralelo también ocurrió a través del planificador de trabajos SGE 6.1, usando la política First Come First Served (FCFS).

Procesos	lu.B	ep.B
1	1	6,86
2	1	6,77
4	1	8,41
8	1	11,56
16	1	23,32

Tabla 4.1 – Número de ejecuciones de cada Benchmark

Cada experimento también fue repetido ocho veces. Los resultados obtenidos fueron la média de los valores medidos, descartándose los tres peores valores.

En la segunda etapa de experimentos el obetivo era evaluar el impacto en relación al tiempo de ejecución y throughput provocados por la mezcla de aplicaciones intensivas en acceso a disco con aplicaciones intensivas en cómputo. Se utilizó la aplicación WordCount, construida para este propósito, y el benchmark NAS-ep.B. La metodología utilizada durante el proceso fue la misma utilizada en la primera etapa de experimentos: se ejecutó primero la aplicación wordcount en forma serial y después en paralelo con 2 y 4 procesos en un único nodo de cómputo, con 8 procesos en dos nodos de cómputo y por fin, con 16 procesos en 4 nodos de cómputo. El archivo de entrada conteniendo el texto a ser procesado con tamaños de 5MB, 50MB, 500MB y 1GB y estaban ubicados en el nodo servidor de datos remoto gestionado por el protocolo NFS.

La Figura 4.11 presenta el escenario de los experimentos con la aplicación WordCount.

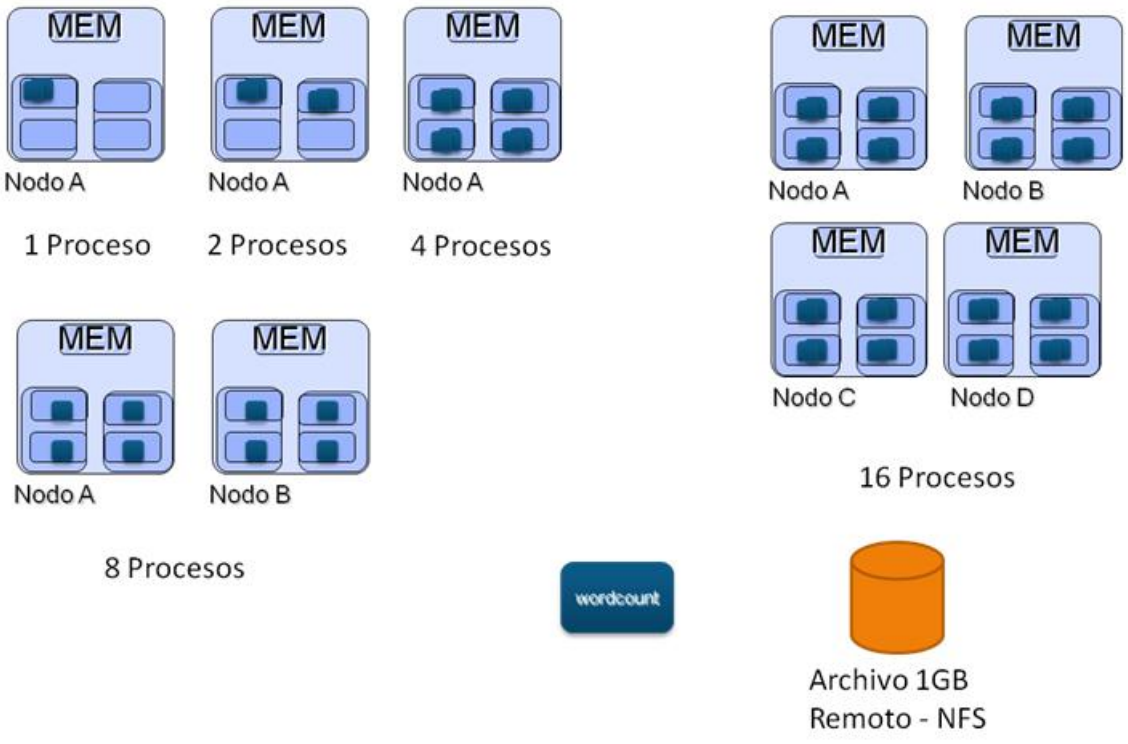

Figura 4.11 – Escenario del experimento con wordcount: acceso remoto

El planificador de trabajos SGE 6.1 mantuvo su configuración en relación a las políticas de planificacion.

Cada ejecución de la aplicación WordCount fue repetida ocho veces y los resultados obtenidos fueron la média de los valores medidos, descartándose los tres peores valores.

Con los valores medidos para el benchmark NAS-ep.B a través de la primera etapa de experimentos y la ejecución aislada de la aplicación wordcount en manos, se iniciaron los experimentos ejecutándose las aplicaciones WordCount y NAS-ep.B de forma concurrente para evaluar el impacto de la mezcla de las aplicaciones sobre el tiempo de ejecución y el throughput. El archivo de entrada (1 GB) que contiene el texto a ser procesado se mantuvo ubicado en un servidor de datos remoto bajo la gestión del protocolo NFS.

Se ejecutaron las aplicaciones ep.B y WordCount en forma serial y concurrente, dividiendo el mismo nodo de cómputo. Posteriormente, se ejecutaron ambas aplicaciones en paralelo, y de forma concurrente, con 2 y 4 procesos en un único nodo de cómputo, con 8 procesos en dos nodos de cómputo y por fin, con 16 procesos en 4 nodos de cómputo.

La Figura 4.12 presenta el escenario de los experimentos con las aplicaciones WordCount y ep.B. La distribución de los procesos en el cluster siguió el mismo patrón presentado en la mezcla de aplicaciones cuando ejecutada la primera etapa de experimentos.

Siguiendo el patrón de ejecución de los experimentos en la primera etapa, las aplicaciones WordCount y NAS-ep.B también fueron ejecutadas con control de carga en el nodo de cómputo.

Para asegurar el efecto de la competencia sobre ambas aplicaciones durante todo el tiempo de ejecución, el benchmark NAS-ep.B, que tiene un tiempo de ejecución menor de lo que la aplicación WordCount, se ejecutó más veces en cada uno de los experimentos.

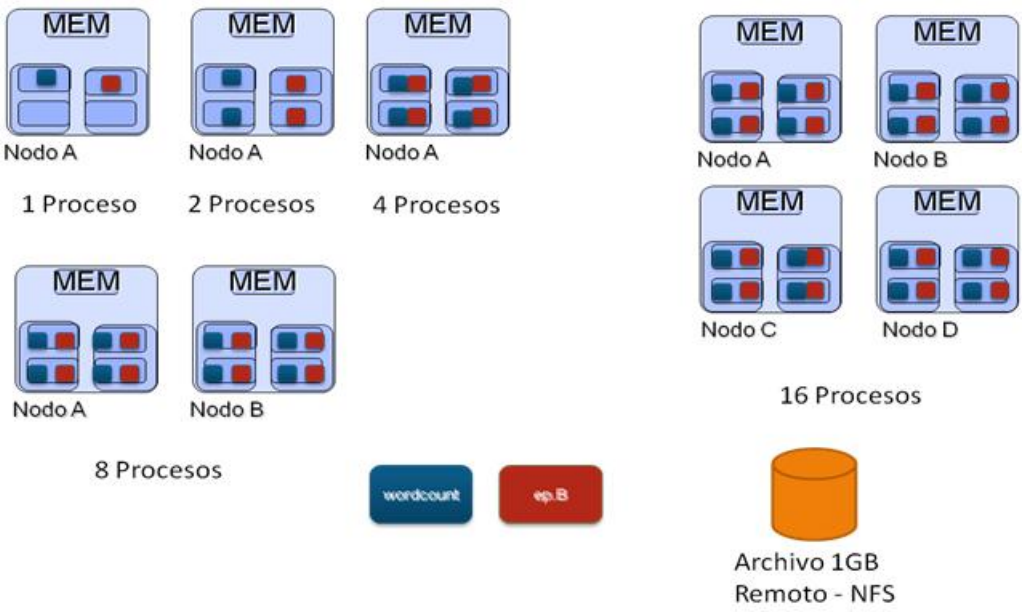

Figura 4.12 – Escenario del experimento con WordCount y ep.B

La Tabla 4.2 presenta el número de ejecuciones de cada una de las aplicaciones.

Procesos	Wordcount	(NAS) ep.B
1	1	1,64
2	1	1,78
4	1	0,87
8	1	1,23
16	1	2,61

Tabla 4.2 – Número de ejecuciones de cada aplicación: acceso remoto

Cada experimento también fue repetido ocho veces. Los resultados obtenidos fueron la media de los valores medidos, descartándose los tres peores valores.

En la tercera fase, se llevó a cabo experimentos para evaluar la influencia de la ubicación de los datos en relación al tiempo de ejecución y throughput causado por la combinación de aplicaciones intensivas de acceso a disco, con

aplicaciones intensivas de cómputo. Repetimos los experimentos realizados en la segunda fase, pero alternamos la ubicación del archivo de lectura de la aplicación WordCount. El archivo de lectura ya no se localiza más en el servidor de datos remoto, accedido via NFS, sino en el disco local en el nodo de cómputo.

Se utilizó la aplicación WordCount, construida para este propósito, y el benchmark NAS-ep.B. La metodología utilizada durante el proceso fue la misma utilizada en las etapas anteriores de experimentos. El archivo de entrada conteniendo el texto a ser procesado con tamaños de 5MB, 50MB, 500MB y 1GB y estaban, de esta vez, ubicados en el disco local de cada nodo de cómputo.

La Figura 4.13 presenta el escenario de los experimentos con la aplicación WordCount.

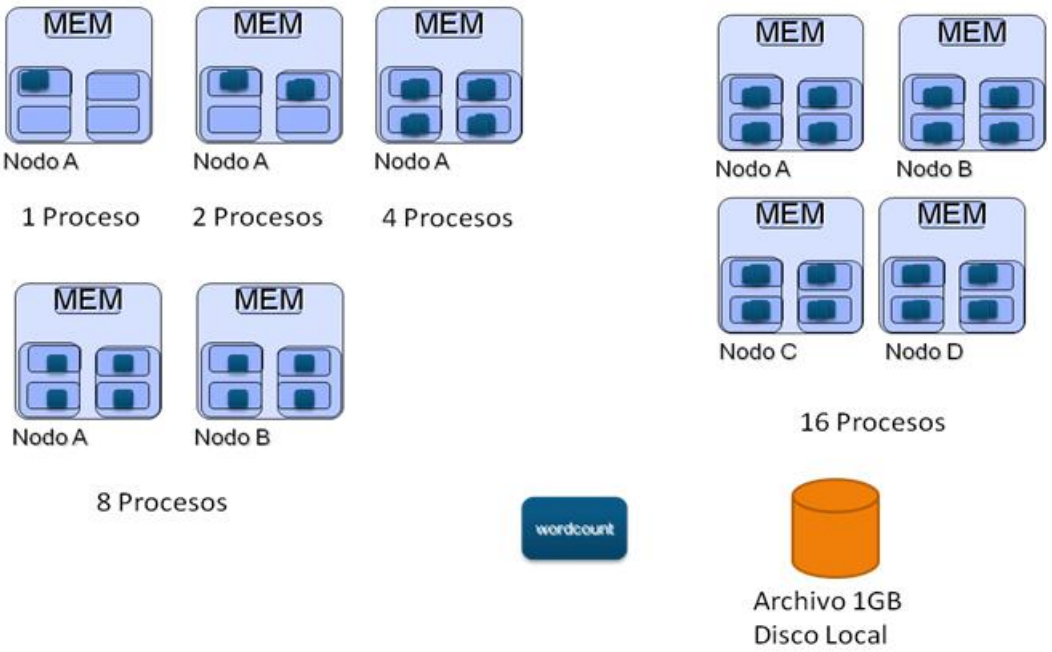

Figura 4.13 – Escenario del experimento con WordCount: acceso local

El mismo control de carga en los nodos de cómputo, fue promovido cuando de la ejecución de la aplicación wordcount. No existió carga local en cada uno

de los nodos utilizados durante los experimentos, ni tampoco cargas en paralelo, excepto la carga provocada por la propia aplicación wordcount.

Cada ejecución de la aplicación wordcount fue repetida ocho veces, y los resultados obtenidos, fueron la media de los valores medidos, descartándose los tres peores valores.

Después de la ejecución aislada de la aplicación WordCount y de poseer los valores medidos para el benchmark NAS-ep.B a través de la primera etapa de experimentos, se iniciaron los experimentos ejecutándose las aplicaciones WordCount y NAS-ep.B de forma concurrente, para evaluar el impacto de la mezcla de ambas las aplicaciones sobre el tiempo de ejecución y el throughput. El archivo de entrada conteniendo el texto a ser procesado, se ubicó en el disco local de cada uno de los nodos de cómputo.

La Figura 4.14 presenta el escenario de los experimentos con las aplicaciones WordCount y ep.B para esta etapa de experimentos. Las mismas observaciones realizadas en los esenarios anteriores se repiten en este experimento.

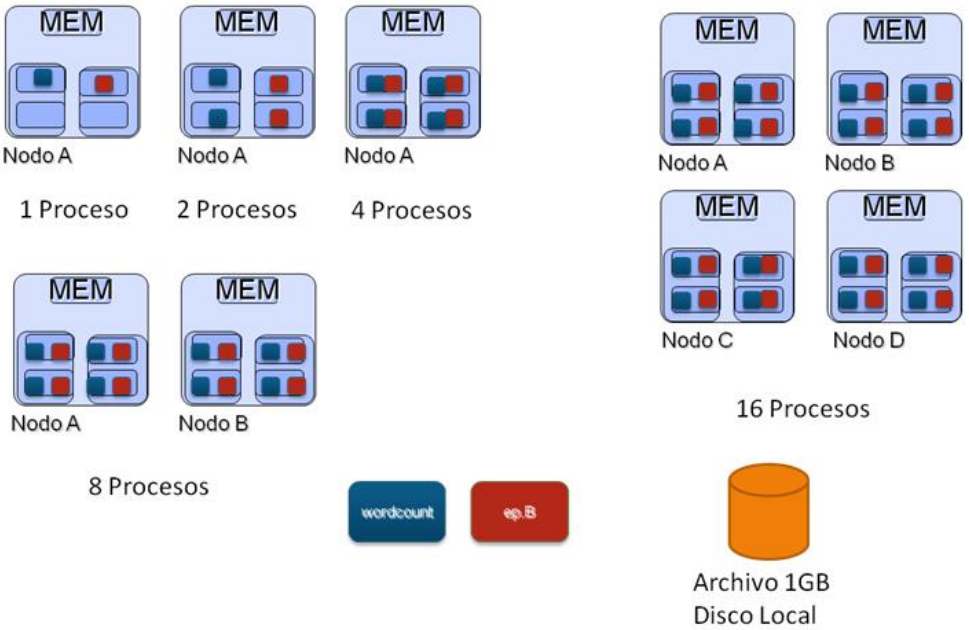

Figura 4.14– Escenario del experimento con WordCount y ep.B

Siguiendo el patrón de ejecución de los experimentos en las etapas anteriores, las aplicaciones WordCount y NAS-ep.B también fueron ejecutadas con control de carga en los nodos de cómputo. Para asegurar el efecto de la concurrencia sobre ambas aplicaciones durante todo el tiempo de ejecución, el benchmark NAS-ep.B, que tiene un tiempo de ejecución menor de que la aplicación wordcount, necesitó ser executado más veces en cada uno de los experimentos. La Tabla 4.3 presenta el número de ejecuciones de cada una de las aplicaciones.

Procesos	Wordcount	NAS(ep.B)
1	1	1,18
2	1	1,16
4	1	1,12
8	1	1,02
16	1	1,02

Tabla 4.3 – Número de ejecuciones de cada aplicación: acceso local

Cada experimento también fue repetido ocho veces. Los resultados obtenidos fueron la media de los valores medidos, descartándose los tres peores valores.

Y en la cuarta y última fase de pruebas, empezamos los experimentos con los planificadores bajo el paradigma MapReduce a fin de evaluar la influencia del modelo de programación sobre el tiempo de ejecución. En esta fase, hemos construido una aplicación WordCount bajo el paradigma MapReduce y evaluamos su escalabilidad en un entorno paralelo con planificación de trabajos bajo el paradigma MapReduce. La aplicación fue construida utilizando el lenguaje de programación Java.

En el entorno utilizado para la realización de los experimentos la planificación de los trabajos fue realizada a través del framework Hadoop. Es

tolerante a fallos, escalable y muy fácil de escalar. La gestión de archivos fue realizada de forma distribuída por Hadoop Distributed File System. HDFS es altamente tolerante a fallos y está diseñado para ser implementado en hardware de bajo costo.

La aplicación fue ejecuta primeramente de forma serie en un único nodo de cómputo, y posteriormente en dos y cuatros nodos. En cada ejecución fue medido el tiempo de ejecución de la aplicación. Los resultados obtenidos reflejan la media de las ejecuciones, descartándose los tres peores valores, después de cada experimento ser repetido ocho veces.

4.5 - Resultados Obtenidos

Los resultados obtenidos son presentados y discutidos en este capítulo. Dos parámetros serán el foco de análisis, para cada etapa de la experimentación: **el tiempo de ejecución** de las aplicaciones cuando son ejecutadas de forma individual, y de forma concurrente. Así como **el througput** bajo las condiciones anteriores de ejecución. Con la medición de los tiempos de ejecución, se puede evaluar la sobrecarga que la ejecución concurrente ejerció sobre las ejecuciones serie.

En la primera etapa de experimentos se ejecutaron los benchmarks NAS-lu.B, intensivo en comunicación, y NAS-ep.B, intensivo en cómputo. El objetivo de los experimentos fue evaluar el impacto provocado por la mezcla de aplicaciones en los parámetros investigados.

Los tiempos de ejecución medidos para el Benchmark lu.B son presentados por la Figura 4.15 y la sobre carga sobre el tiempo de ejecución es presentada por la Figura 4.16.

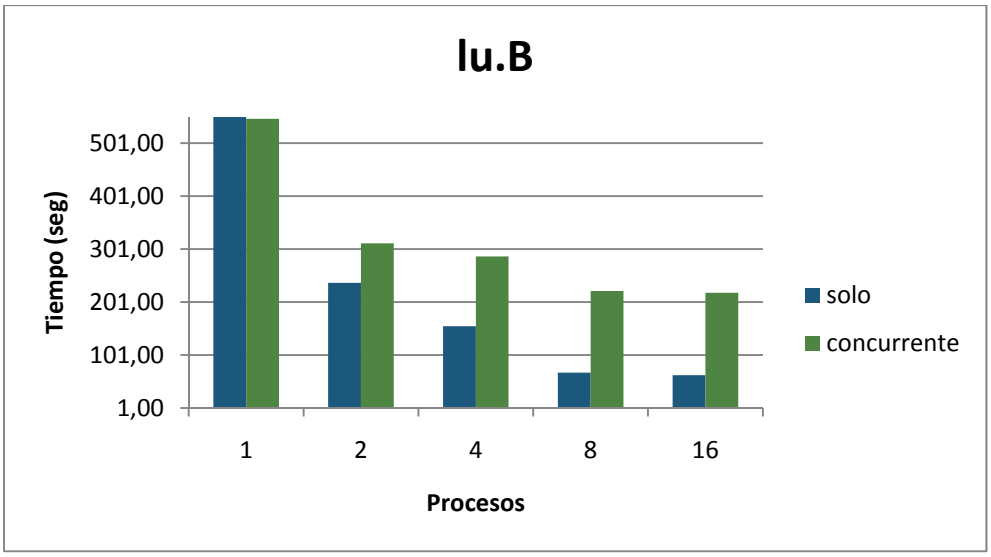

Figura 4.15 –Tiempo de ejecución NAS-lu.B

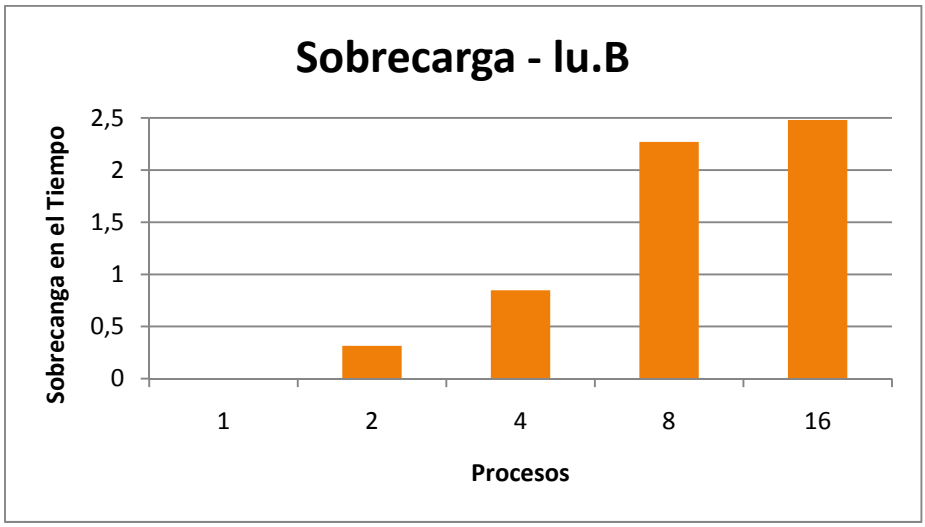

Figura 4.16 – Sobrecarga en el tiempo de ejecución – NAS-lu.B

Analizando la sobrecarga sobre el tiempo de ejecución de la aplicación lu.B provocada por la ejecución de forma concurrente con el benchmark ep.B podemos observar, que presentó un comportamiento creciente, a medida que se aumentó el número de procesos ejecutando las aplicaciones. Este comportamiento, puede ser explicado en función de las características de la aplicación lu.B. Como lu.B es una aplicación intensiva de comunicación, a medida que aumenta el número de procesos ejecutando la aplicación, mayor es la necesidad de comunicación entre estos procesos y la sincronización entre los mismos. Esta sincronización sufre mayor dificultad de ser atendida cuando

la aplicación lu.B está siendo ejecutada de forma concurrente con la aplicación ep.B.

La ausencia de sobrecarga, cuando las aplicaciones fueron ejecutadas de forma serie, puede ser explicada en función de la distribución de los procesos en el nodo de cómputo. Cada aplicación, ocupó un core en una CPU distinta del nodo, pudiendo usar completamente los recursos disponibles en el nodo.

Para el Benchmark ep.B los tiempos de ejecución medidos durante los experimentos son presentados por la Figura 4.17 y la sobrecarga sobre el tiempo de ejecución es presentada por la Figura 4.18.

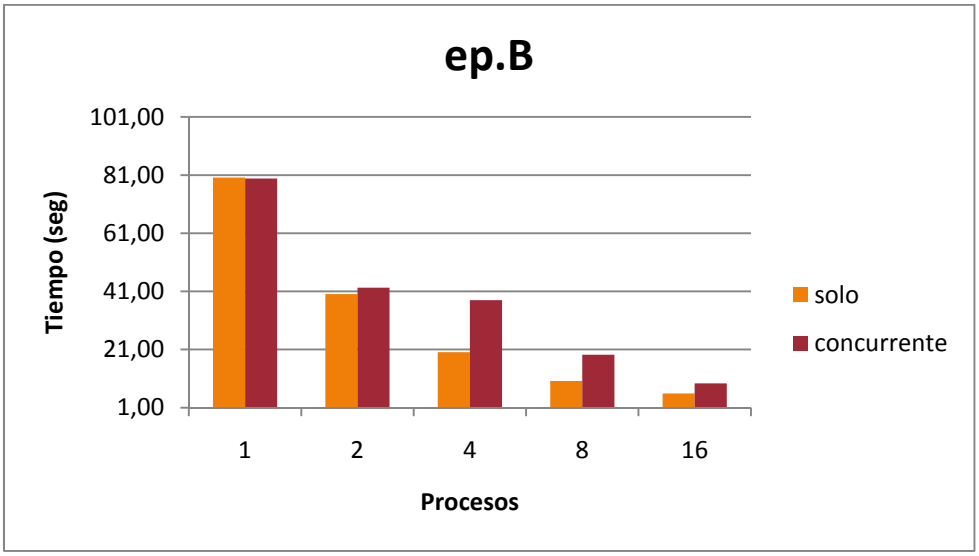

Figura 4.17 –Tiempo de ejecución NAS-ep.B

El análisis de los resultados obtenidos nos permite observar que también hay una sobrecarga sobre el tiempo de ejecución del benchmark ep.B cuando executado de forma concurrente con la aplicación lu.B. Entretanto, esta sobrecarga es relativamente más baja que la sufrida por lu.B. La explicacón para este resultado viene del hecho del benchmark ep.B ser intensivo en cómputo y poder usar mejor los recursos computacionales durante los tiempos de espera de la aplicación lu.B.

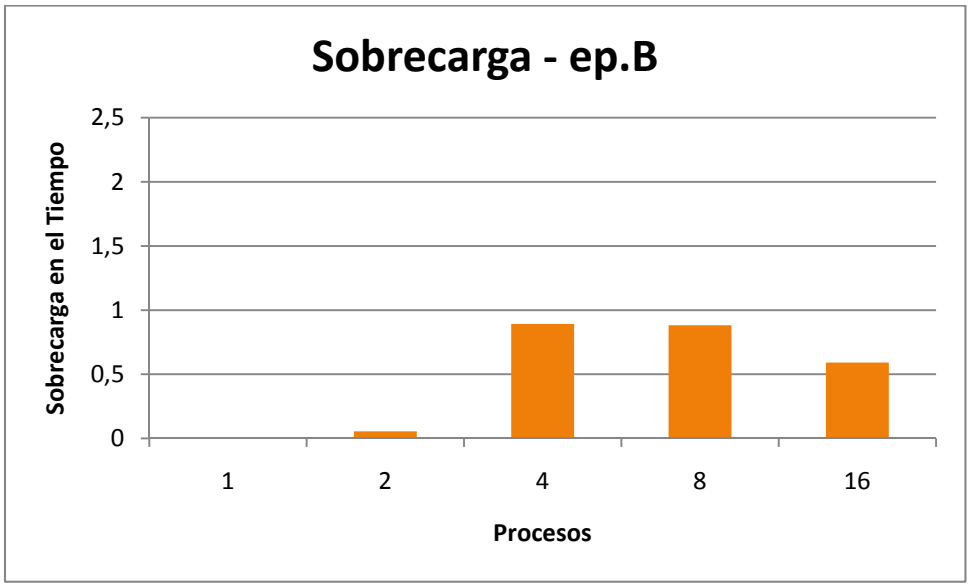

Figura 4.18 - – Sobrecarga en el tiempo de ejecución – NAS-ep.B

Podemos observar también que no hubo sobrecarga cuando las aplicaciones fueron ejecutadas de forma serial. Este hecho ocurrió en función de la distribución de los procesos en el nodo de cómputo. Cada aplicación ocupó un core en una CPU distinta del nodo, pudiendo usar completamente los recursos disponibles en el nodo. Lo mismo ocurrió para la ejecución en dos procesos. Cada aplicación tuvo sus dos procesos ocupando la misma CPU pudiendo así, usar mejor los recursos disponibles en el nodo de cómputo.

La Figura 4.19 presenta el throughput del sistema calculado para las aplicaciones cuando fueron ejecutadas de forma concurrente y secuencial. Podemos observar que a pesar de una sobrecarga en el tiempo de ejecución de las aplicaciones cuando fueron ejecutadas de forma concurrente, el solapamiento de sus ejecuciones permitió una ganancia en relación al throughput.

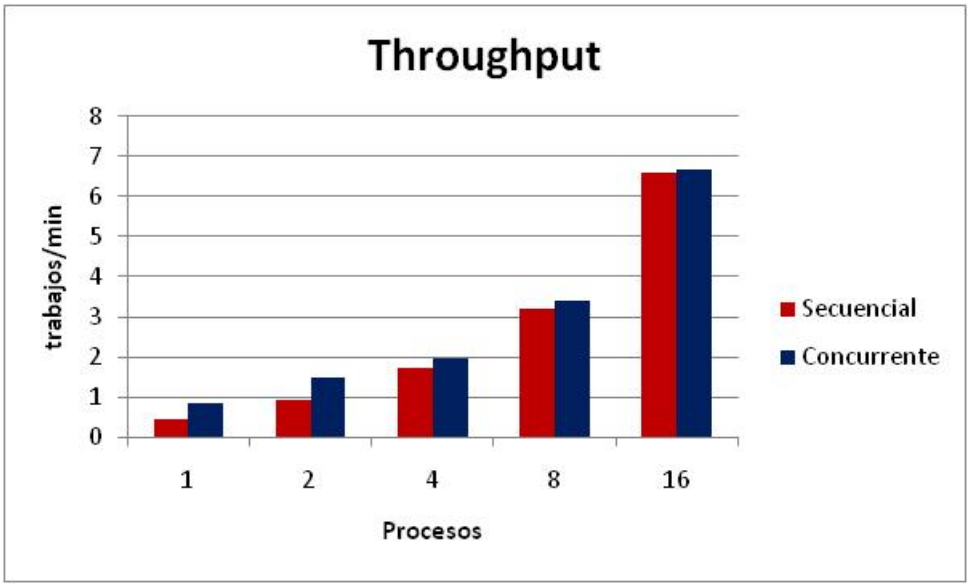

Figura 4.19 – Throughput – NAS-lu.B y NAS-ep.B

En la segunda etapa de experimentos se ejecutó la aplicación WordCount, intensiva en E/S y el benchmark NAS-ep.B, intensivo en cómputo. Los datos presentados a continuación son para el archivo de entrada de la aplicación WordCount de tamaño 1,0GB, ubicado en el servidor de datos remoto, y gestionado por el protocolo NFS. El objetivo de los experimentos fue evaluar el impacto provocado por la mezcla de las aplicaciones en los parámetros investigados.

Los tiempos medios de ejecución medidos para la aplicación WordCount son presentados a través de la Figura 4.20 y la sobrecarga reflejada sobre el tiempo de ejecución es presentada por la Figura 4.21.

El análisis de los resultados obtenidos nos permite observar que también hay una sobrecarga sobre el tiempo de ejecución del benchmark ep.B cuando executado de forma concurrente con la aplicación lu.B. Entretanto, esta sobrecarga es relativamente más baja que la sufrida por lu.B. La explicacón para este resultado viene del hecho del benchmark ep.B ser intensivo en cómputo y poder usar mejor los recursos computacionales durante los tiempos de espera de la aplicación lu.B.

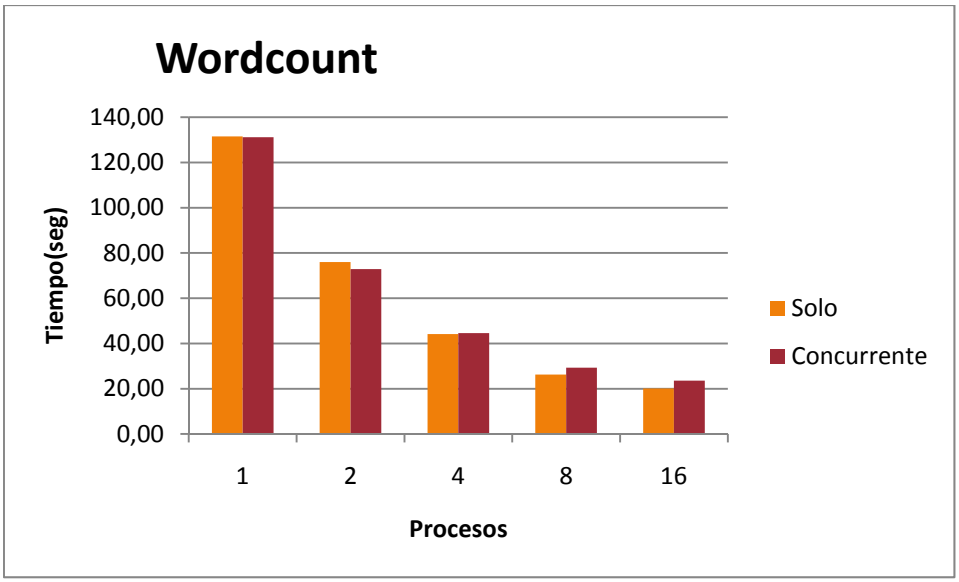

Figura 4.20 –Tiempo de ejecución WorCount – Acceso Remoto

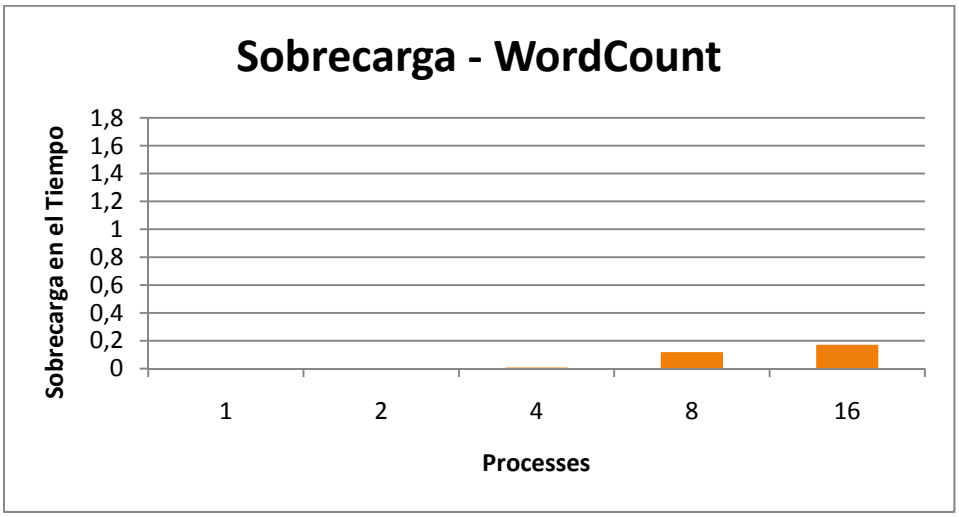

Figura 4.21 – Sobrecarga en el tiempo de ejecución – WordCount – acceso remoto

La sobrecarga empieza a aparecer, pero de forma muy discreta, cuando aumenta el número de procesos. A medida que aumenta el número de procesos, disminuye el volumen de datos accedidos por cada proceso, y por consiguiente disminuye el tiempo de espera para acceder a los datos.

Para el Benchmark ep.B, los tiempos de ejecución medidos durante los experimentos son presentados por la Figura 4.22 y la sobrecarga sobre el tiempo de ejecución es presentada por la Figura 4.23.

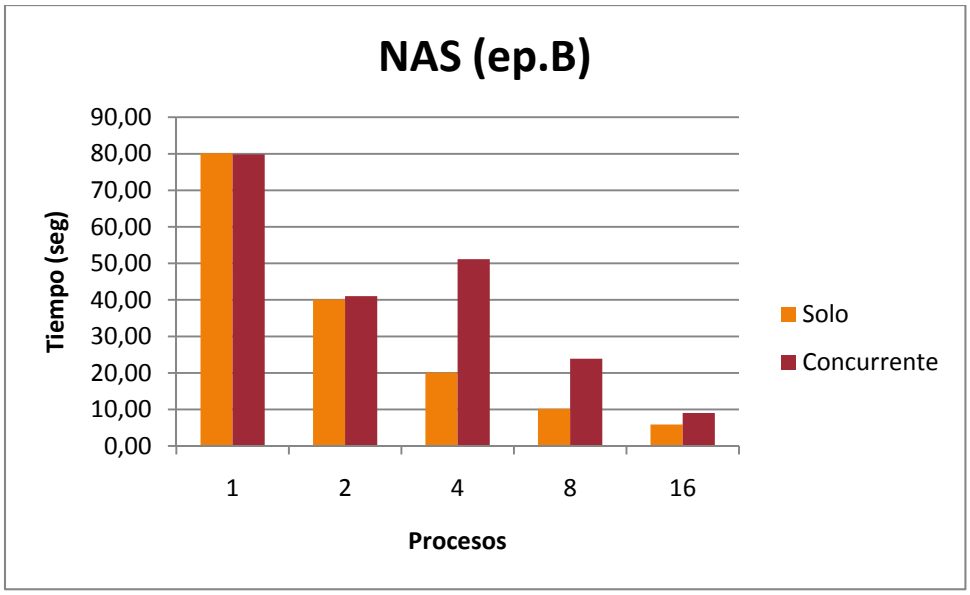

Figura 4.22 –Tiempo de ejecución NAS-ep.B

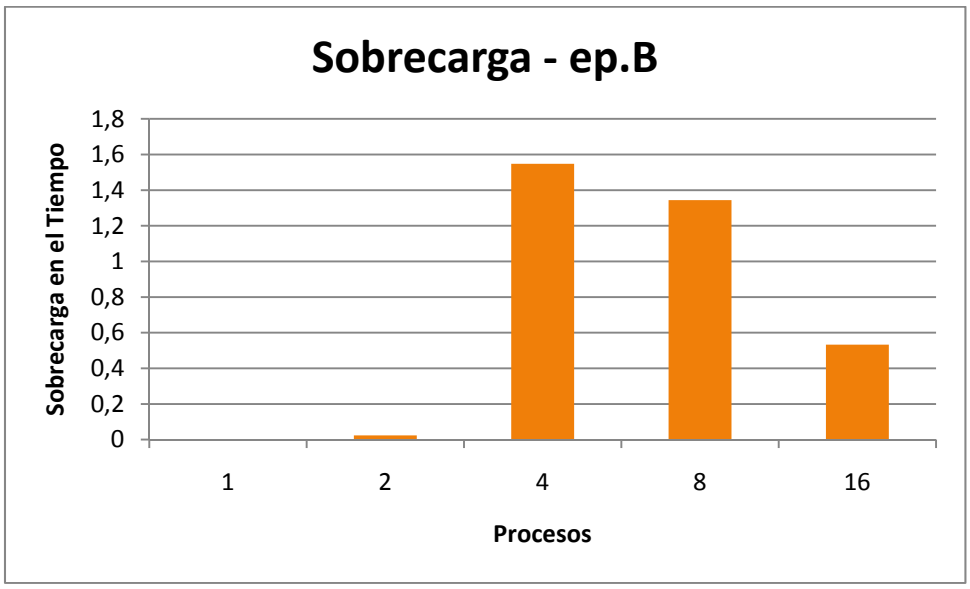

Figura 4.23 – Sobrecarga en el tiempo de ejecución – NAS-ep.B

El análisis de los resultados obtenidos nos permite observar, que ocurre una sobrecarga sobre el tiempo de ejecución del benchmark ep.B, es ejecutado de forma concurrente con la aplicación WordCount, a partir de la distribución en 4 procesos. La explicación para este resultado, viene derivada del hecho de que el sistema prioriza la ejecución de la aplicación WordCount, en los constantes estadios de espera.

Podemos observar también, que no hubo sobrecarga cuando las aplicaciones fueron ejecutadas de forma serie. Este hecho, ocurrió en función de la distribución de los procesos en el nodo de cómputo. Cada aplicación ocupó un core en una CPU distinta del nodo pudiendo usar completamente los recursos disponibles en el nodo. Lo mismo ocurrió para la ejecución en dos procesos. Cada aplicación tuvo sus dos procesos ocupando la misma CPU, pudiendo así usar mejor los recursos disponibles en el nodo de cómputo.

La Figura 4.24 presenta el throughput del sistema calculado para las aplicaciones cuando fueron ejecutadas de forma concurrente y cuando fueron ejecutadas de forma secuencial. Podemos observar, que a pesar de una sobrecarga en el tiempo de ejecución de las aplicaciones cuando fueron ejecutadas de forma concurrente, el solapamiento de las mismas, continuó permitiendo una ganancia en relación al throughput.

En la tercera etapa de experimentos se modificó la posición del archivo de entrada de la aplicación WordCount, pasando a posicionárlo directamente en los discos locales de cada nodo de cómputo. Una vez más, se ejecutó la aplicación WordCount, intensiva en E/S y el benchmark NAS-ep.B, intensivo en cómputo. Los datos presentados a continuación, son para el archivo de entrada de la aplicación WordCount de tamaño 1,0GB. El objetivo de los experimentos, fue evaluar el impacto provocado por la modificación de la ubicación del archivo, cuando se produce la mezcla de las aplicaciones en los parámetros investigados.

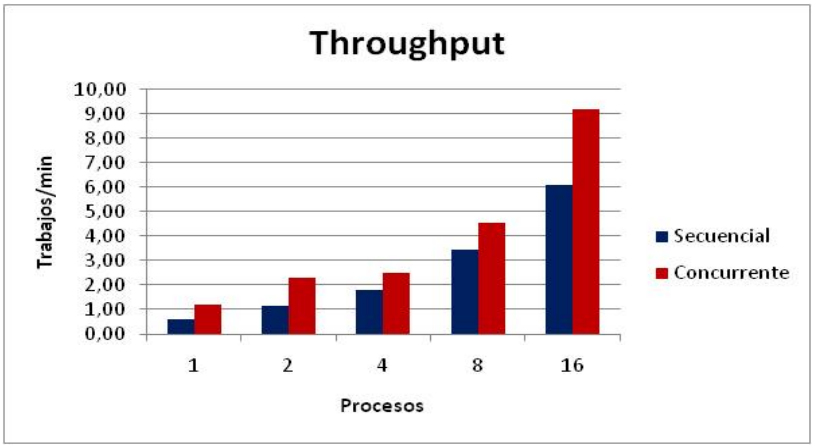

Figura 4.24 – Throughput – NAS-ep.B y WordCount – Acceso remoto

Los tiempos medios de ejecución medidos para la aplicación WordCount, son presentados mediante la Figura 4.25 y la sobrecarga reflejada sobre el tiempo de ejecución es presentada por la Figura 4.26.

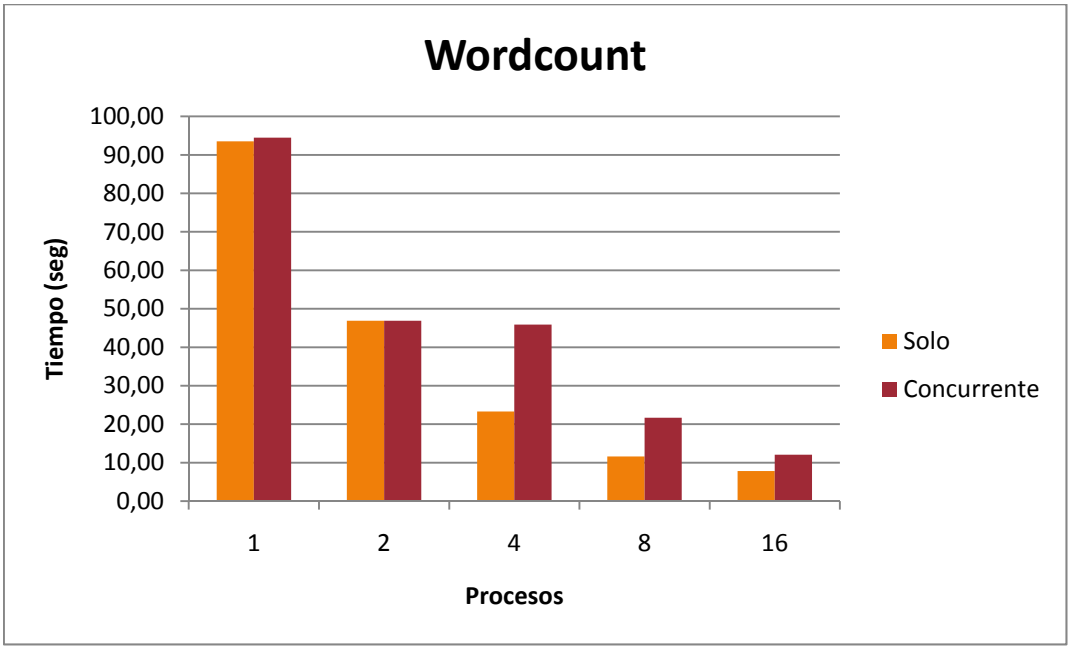

Figura 4.25 –Tiempo de ejecución WorCount – Acceso Local

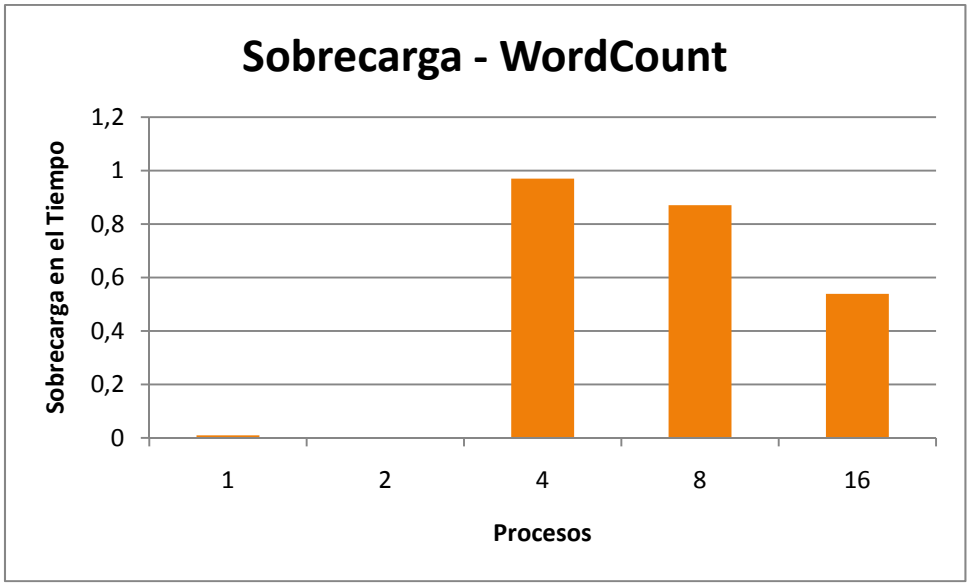

Figura 4.26 – Sobrecarga en el tiempo de ejecución – WordCount – acceso local

Podemos observar que en este caso, la aplicación WordCount pasó a sufrir el impacto en el tiempo de ejecución, cuando fue ejecutada de forma concurrente, con el benchmark ep.B a partir de cuatro procesos. Este comportamiento, puede ser explicado en función de la ubicación del archivo de E/S accedido por la aplicación WordCount. Como el archivo de E/S está ubicado en los discos locales de los propios nodos de cómputo, los tiempos de espera por el acceso a los datos, se redució cuando lo comparamos con el experimento anterior.

Podemos observar también, que no hubo sobrecarga cuando las aplicaciones fueron ejecutadas de forma serie. Este hecho también ocurrió, en función de la distribución de los procesos en el nodo de cómputo. Cada aplicación ocupó un core en una CPU distinta del nodo, pudiendo usar completamente los recursos disponibles en el nodo. Lo mismo ocurrió para la ejecución en dos procesos. Cada aplicación tuvo sus dos procesos ocupando la misma CPU, pudiendo así usar mejor los recursos disponibles en el nodo de cómputo.

Para el Benchmark ep.B los tiempos de ejecución medidos durante los experimentos son presentados por la Figura 4.27 y la sobrecarga sobre el tiempo de ejecución es presentada por la Figura 4.28.

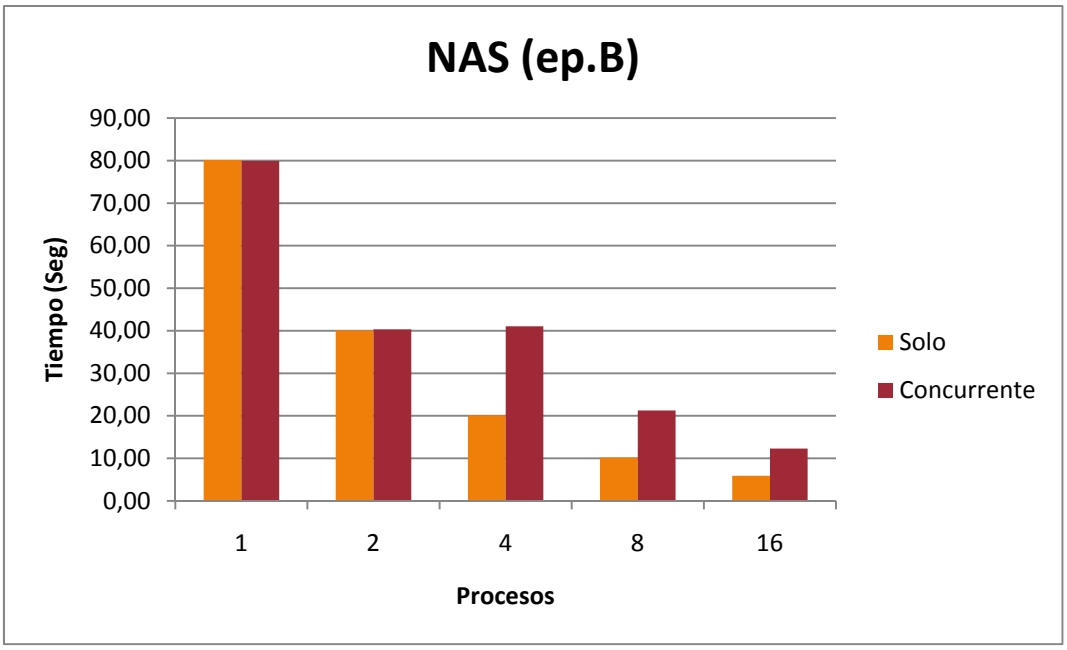

Figura 4.27 –Tiempo de ejecución NAS-ep.B

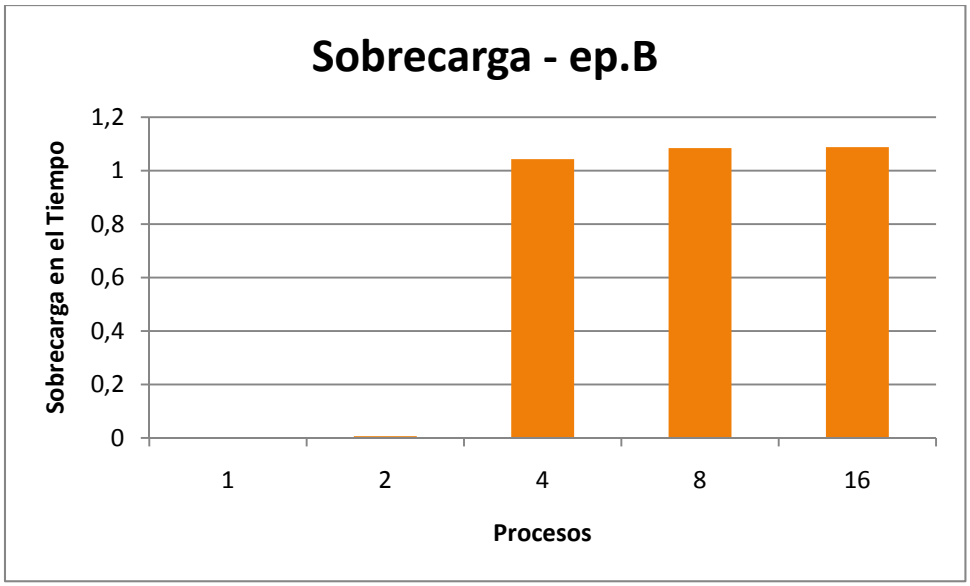

Figura 4.28 – Sobrecarga en el tiempo de ejecución – NAS-ep.B

El análisis de los resultados obtenidos, nos permite observar que, así como en el experimento anterior, ocurre una sobrecarga sobre el tiempo de ejecución del benchmark ep.B, cuando es ejecutado de forma concurrente con la aplicación WordCount, a partir de la distribución en 4 procesos. Como el benchmark ep.B es de cómputo intensivo, y la ejecución de la aplicación WordCount accede a los datos de una manera más rápida, no hay un solapamiento significativo entre las aplicaciones, hecho que se refleja en la sobrecarga medida.

Podemos observar también, que no hubo sobrecarga cuando las aplicaciones fueron ejecutadas de forma serie. Este hecho ocurrió en función de la distribución de los procesos en el nodo de cómputo. Cada aplicación ocupó un core en una CPU distinta del nodo pudiendo usar completamente los recursos disponibles en el nodo. Lo mismo ocurrió, para la ejecución en dos procesos. Cada aplicación tuvo sus dos procesos ocupando la misma CPU, pudiendo así usar mejor los recursos disponibles no nodo de cómputo.

La Figura 4.29 presenta el throughput del sistema calculado para las aplicaciones, cuando fueron ejecutadas de forma concurrente y secuencial. Podemos observar que continúa ocurriendo el solapamiento en la ejecución de las aplicaciones. Esto permite la continuidad de ganancia en relación al

throughput, a pesar de una sobrecarga en el tiempo de exejución de las aplicaciones, cuando fueron ejecutadas de forma concurrente.

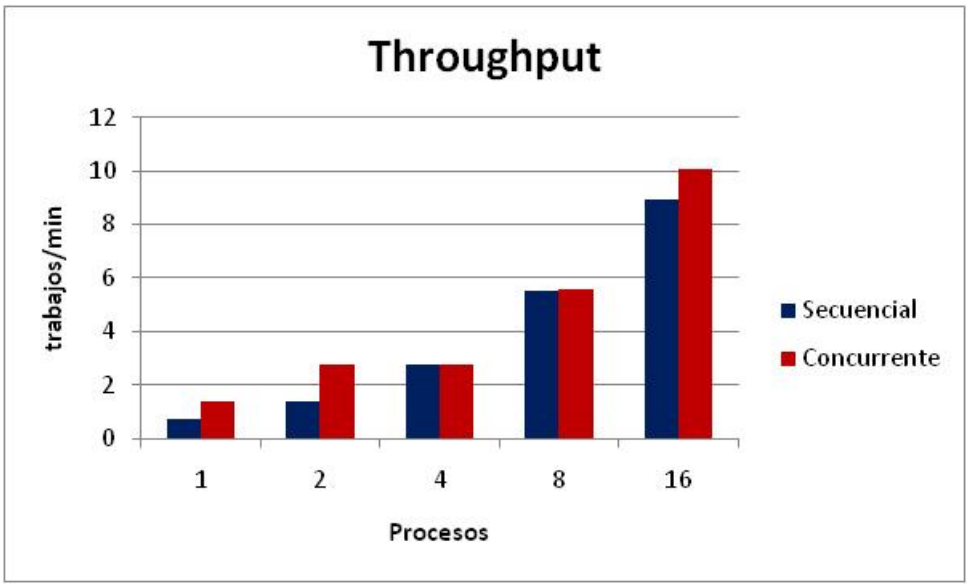

Figura 4.29 – Throughput – NAS-ep.B y WordCount – Acceso local

La Figura 4.30 nos permite comparar, los tiempos de ejecución de la aplicación WorCount, cuando fue alternado el acceso al archivo texto de E/S de su ubicación remota gestionado por el protocolo NFS, para la ubicación en los propios discos locales de los nodos de cómputo.

Podemos observar, que la ejecución con acceso a archivo local, fue sistemáticamente realizada en un tiempo menor que la versión con acceso remoto. La sobrecarga no es observada en ambos casos, cuando la aplicación es ejecutada por uno y dos procesos, en función de como ocurre en la distribución de los procesos en los nodos de cómputo del cluster evaluado. La distribución se realiza de modo que cada aplicación, utilize una CPU diferente del nodo, maximizando la utilización de los recursos computacionales. Para la aplicación con acceso remoto, no ocurre sobrecarga en el tiempo de ejecución, en función de los tiempos de espera para acceder los datos remotamente. En la versión con acceso local a los datos, la sobrecarga es observada, a partir de la distribución en cuatro procesos, en función del acceso a los datos puede ocurrir de una manera más rápida.

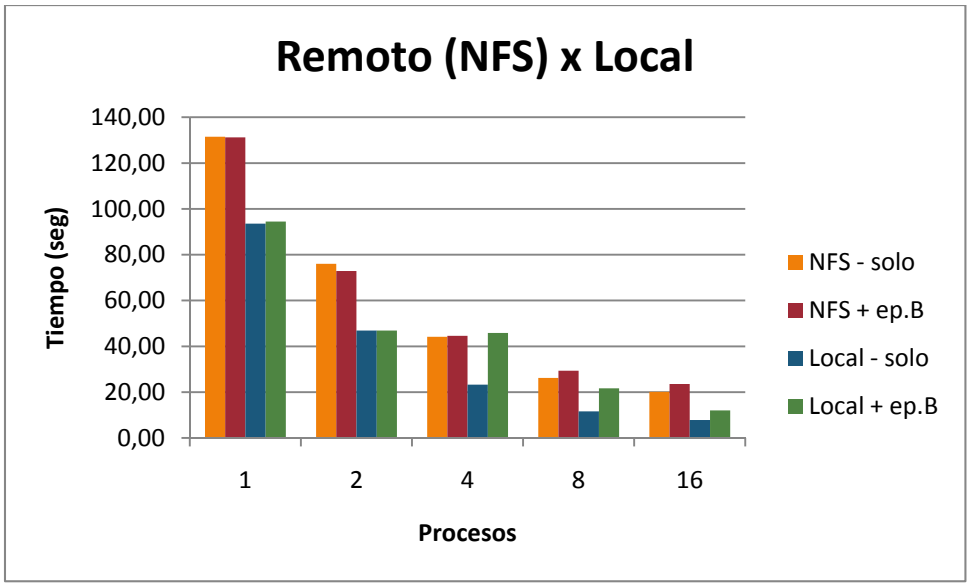

Figura 4.30 – WordCount – acceso remoto x local

En la cuarta etapa de experimentos, se buscó evaluar la escalabilidad de la aplicación WordCount, contruída sobre el paradigma MapReduce, y ejecutada en un cluster no-dedicado con planificación a través del framework Hadoop. Durante la ejecución de los experimentos, hubo control de la carga ejecutada en los nodos de cómputo, de tal forma que no existiera carga local y tampoco cargas distintas a las cargas de la aplicación WordCount. Cada ejecución fue repetida ocho veces, y los valores presentados, son la media de los valores medidos exceptuando los tres peores valores. La Figura 4.31 presenta el Speed-up de la aplicación y la Figura 4.32 la eficiencia obtenida.

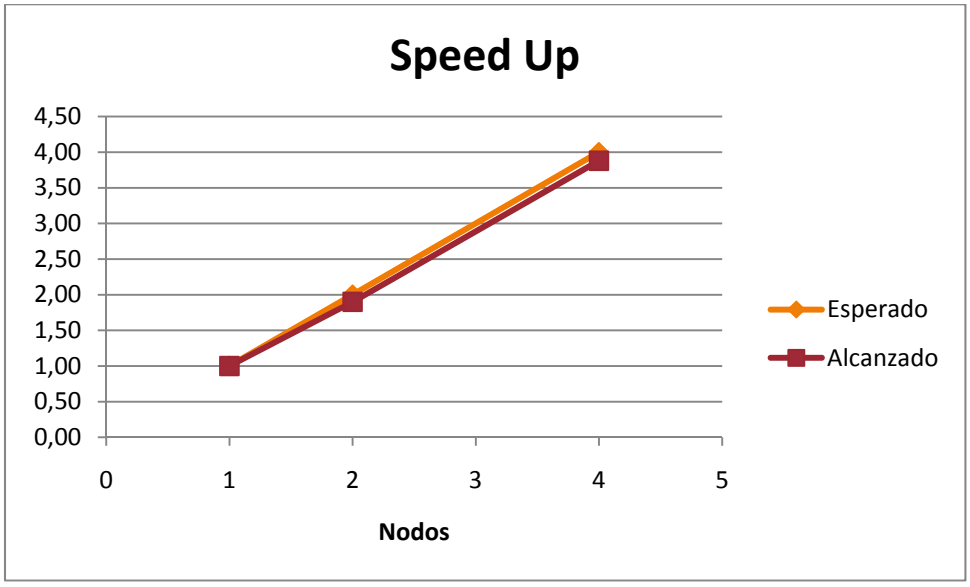

Figura 4.31 – Speed Up – WordCount - Mapreduce

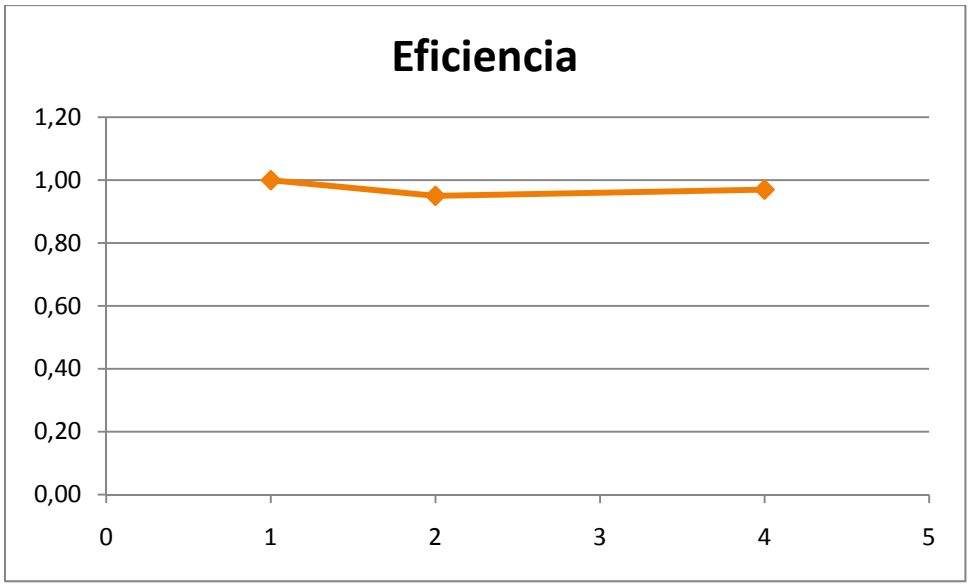

Figura 4.32 – Eficiência – WordCount - Mapreduce

Podemos observar que, la aplicación presentó una escalabilidad y eficiencia próximas a la ideal.

4.6 - Conclusiones

Los experimentos ejecutados nos permitien concluir que para las aplicaciones con requerimientos de E/S, la mezcla con aplicaciones en paralelo no generó sobrecarga significativa. Pero es necesario hacer más pruebas con diferentes aplicaciones, con archivos más grandes y también mezclar con otras aplicaciones intensivas de E/S. Así se podrá tener una visión más precisa, de la influencia que la mezcla de aplicaciones provoca, en las prestaciones de un cluster paralelo no-dedicado.

Para las aplicaciones CPU-bound, la mezcla con aplicaciones en paralelo generó una cierta sobrecarga, pero permitió ganancias en relación al throughput. La mezcla de aplicaciones, provocó en las aplicaciones con requerimientos de comunicación, una sobrecarga crecinete en sus tiempos de ejecución, a la medida que se aumentó el número de procesos. Los procesos cooperantes, necesitan de una mayor coordinación durante sus ejecuciones, y la mezcla de aplicaciones, genera una dificultad para la realización de esta coordinación. Como consecuencia, hay la sobrecarga en su tiempo de ejecución.

La ubicación de los datos influyó de forma significativa, en la sobrecarga sobre las aplicaciones de E/S. Los tiempos de espera, provocados por el acceso a los datos de forma remota, provocaron en el cluster investigado, la posibilidad de mezclar aplicaciones en paralelo sin generar sobrecarga en la aplicación con requerimientos de E/S. Cuando el acceso a los datos ocurrió de forma más rápida, a través de un archivo ubicado en el disco local de cada nodo de cómputo, hubo sobrecarga en el tiempo de ejecución de la aplicación.

Los primeros pasos en el manejo MapReduce y ejecución en cluster Hadoop fueron dados. Aunque los experimentos se realizaron en una pequeña cantidad de nodos, el speed-up y la eficiencia obtenidos, nos hacen intuir prestaciones de escalabilidad lineales, en relación al uso del paradigma MapReduce, para aplicaciones intensivas de E/S en clusters de tamaño medio. Es importante destacar también, la facilidad de construcción de la aplicación utilizando el

paradigma MapReduce, sin que haya necesidad de un control de bajo nivel, para el acceso a los datos, como ocurre con las aplicaciones que utilizan funciones de la biblioteca MPI-IO.

Como línea abierta de investigación, esperamos proponer el diseño de nuevos planificadores, que tengan en cuenta: diferentes clases de aplicaciones y diferentes paradigmas de programación en clusters no dedicados.

Referencias

1. **Carriero, Nicholas, et al.** *Adaptive Parallelism and Piranha.* 1, s.l. : IEEE Computer Society Press, 1995, Computer, Vol. 28, pp. 40-49.

2. **Hagman, R.** *Process server: Sharing processing power in a workstation environment.* 1988.

3. **Barak, Amnon and La'adan, Oren.** *The MOSIX multicomputer operating system for high performance cluster computing.* 4-5, s.l. : Elsevier Science Publishers B. V., 1998, Future Gener. Comput. Syst., Vol. 13, pp. 361-372.

4. **Litzkow, Living M. and Mutka, M** *Condor- a hunter of idle workstations.***.** 1988.

5. **Krueger, P. and Babbar, D.** Stealth: a liberal approach to distributed scheduling for network of workstations. *Stealth: a liberal approach to distributed scheduling for network of workstations.* 1993.

6. **Ousterhout, John K., et al.** The Sprite Network Operating System. *The Sprite Network Operating System.* s.l. : University of California at Berkeley, 1987.

7. **Feitelson, Dror G.***Metric and Workload Effects on Computer Systems Evaluation.* 9, s.l. : IEEE Computer Society Press, 2003, Computer, Vol. 36, pp. 18-25.

8. **Arpaci, Remzi H., et al.** *The interaction of parallel and sequential workloads on a network of workstations.* s.l. : ACM, 1995. pp. 267-278.

9. **Frachtenberg, Eitan.** *Process Scheduling for the Parallel Desktop.* s.l. : IEEE Computer Society, 2005. pp. 132-139.

10. **Chronopoulos, Anthony T., et al.** *An efficient 3D grid based scheduling for heterogeneous systems.* 9, s.l. : Academic Press, Inc., 2003, J. Parallel Distrib. Comput., Vol. 63, pp. 827-837.

11. **Sodan, Angela C., et al.** *Time and space adaptation for computational grids with the ATOP-Grid middleware.* 6, s.l. : Elsevier Science Publishers B. V., 2008, Future Gener. Comput. Syst., Vol. 24, pp. 561-581.

12. **He, Ligang, et al.** *Dynamic Scheduling of Parallel Jobs with QoS Demands in Multiclusters and Grids.* s.l. : IEEE Computer Society, 2004. pp. 402-409.

13. **Mnaouer, Adel Ben and Al-Riyami, Badriya.** *Effective scheduling of local interactive processes and parallel processes in a non-dedicated cluster environment.* 6, s.l. : Academic Press, Inc., 2005, J. Parallel Distrib. Comput., Vol. 65, pp. 755-766.

14. **Hanzich,M; Giné, F.; Hernández, P.; Soloma, F and Luque, E.** *3DBackflling: A space sharing approach for non-dedicated clusters.* 2005, In Parallel and Distributed Computing and Systems (PDCS'05), Vol. 17, p. 131_138.

15. **Hanzich, Mauricio.** *A Temporal and Spatial Scheduling System for Nondedicated Clusters.* PhD Thesis. Universidad Autónoma of Barcelona. 2006.

16. **García, Jose.** *Planificación de Aplicaciones Best-Effort y Soft real-Time on NOWs.* Master Thesis Universidad Autónoma of Barcelona. 2007.

17. **El-Ghazawi, Tarek, et al.***A performance study of job management systems: Research Articles.* 13, s.l. : John Wiley and Sons Ltd., 2004, Concurr. Comput. : Pract. Exper., Vol. 16, pp. 1229-1246.

18. **Isaila, Florin, et al.** *A Scalable Message Passing Interface Implementation of an Ad-Hoc Parallel I/o system.* 2, s.l. : Sage Publications, Inc., 2010, Int. J. High Perform. Comput. Appl., Vol. 24, pp. 164-184.

19. **Marjanovic, Vladimir, et al.** *Overlapping communication and computation by using a hybrid MPI/SMPSs approach.* s.l. : ACM, 2010. pp. 5-16.

20. **Hoefler, Torsten, Lumsdaine, Andrew and Dongarra, Jack.** *Towards Efficient MapReduce Using MPI.* s.l. : Springer-Verlag, 2009. pp. 240-249.

21. **Hsu, Ching ;Hsien and Chen, Tai ;Lung.** *Performance\&\#45; and economisation\&\#45;oriented scheduling techniques for managing applications with QoS demands in grids.* 4, s.l. : Inderscience Publishers, 2010, Int. J. Ad Hoc Ubiquitous Comput., Vol. 5, pp. 219-226.

22. **Dodonov, Evgueni and de, Rodrigo Fernandes** *A novel approach for distributed application scheduling based on prediction of communication events.***.** 5, s.l. : Elsevier Science Publishers B. V., 2010, Future Gener. Comput. Syst., Vol. 26, pp. 740-752.

23. **Scott, S. L. and Leangsuksun, C.** *Oscar clusters.* 2003.

24. **McClure, S.; Wheeler R.** *Mosix: How linux clusters solve real-world problems.***.** 2000.

25. **Jackson, D.; Snell, Q.; Clement., M.** *Core algorithms of the maui scheduler.* 2001.

26. **Dean, Jeffrey and Ghemawat, Sanjay** *MapReduce: simplified data processing on large clusters.***.** 1, s.l. : ACM, 2008, Commun. ACM, Vol. 51, pp. 107-113.

27. **Joseph, Anthony D** *Improving MapReduce Performance in Heterogeneous Environments.***.** 2008.

28. **Lämmel, Ralf.** *Google's MapReduce programming model -- Revisited.* 1, s.l. : Elsevier North-Holland, Inc., 2008, Sci. Comput. Program., Vol. 70, pp. 1-30.

29. **Kruijif, M.; Sankaralingam, K.** *MapReduce for the CELL B.E. Architecture.* 2007, IBM Journal of Research and Development, p. 52(4).

30. **He, Bingsheng, et al** *Mars: a MapReduce framework on graphics processors.***.** s.l. : ACM, 2008. pp. 260-269.

31. **Ranger, Colby, et al.** *Evaluating MapReduce for Multi-core and Multiprocessor Systems.* s.l. : IEEE Computer Society, 2007. pp. 13-24.

32. **Tan, G., et al** *A Study of Architectural Optimization Methods in Bioinformatics Applications*.**.** 3, s.l. : Sage Publications, Inc., 2007, Int. J. High Perform. Comput. Appl., Vol. 21, pp. 371-384.

33. **Rosti, Emilia, et al.** *The impact of I/O on program behavior and parallel scheduling.* 1, s.l. : ACM, 1998, SIGMETRICS Perform. Eval. Rev., Vol. 26, pp. 56-65.

34. **Cannataro, Mario, Talia, Domenico and Srimani, Pradip K** *Parallel data intensive computing in scientific and commercial applications*.**.** 5, s.l. : Elsevier Science Publishers B. V., 2002, Parallel Comput., Vol. 28, pp. 673-704.

35. **Tourio, Juan and Doallo, Ramon** *Characterization of Message-Passing Overhead on the AP3000 Multicomputer*. s.l. : IEEE Computer Society, 2001. pp. 321-330.

36. **Tourio, Juan et al** *Performance analysis of MPI-I/O primitives on a PC cluster*.**.** s.l. : ACM, 2002. pp. 907-912.

37. **Vanneschi, M. and Veraldi, L** *Dynamicity in distributed applications: issues, problems and the ASSIST approach*.**.** 12, s.l. : Elsevier Science Publishers B. V., 2007, Parallel Comput., Vol. 33, pp. 822-845.

38. **Tian, Chao, et al** *A Dynamic MapReduce Scheduler for Heterogeneous Workloads*.**.** s.l. : IEEE Computer Society, 2009. pp. 218-224.

39. **Subhlok, Jaspal, Venkataramaiah, Shreenivasa and Singh, Amitoj** *Characterizing NAS Benchmark Performance on Shared Heterogeneous Networks*.**.** s.l. : IEEE Computer Society, 2002. p. 91.

40. **Moreto, Miquel, et al** *Load balancing using dynamic cache allocation*.**.** s.l. : ACM, 2010. pp. 153-164.

41. **Jin, Hai, et al** *An adaptive meta-scheduler for data-intensive applications*.**.** 1, s.l. : Inderscience Publishers, 2005, Int. J. Grid Util. Comput., Vol. 1, pp. 32-37.

42. **Frachtenberg, Eitan, et al** *Parallel Job Scheduling Under Dynamic Workloads*.**.** 2003.

43. **Cappello, Franck and Etiemble, Daniel** *MPI versus MPI+OpenMP on IBM SP for the NAS benchmarks*.**.** s.l. : IEEE Computer Society, 2000. p. 12.