Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Computació d'altes prestacions

Configuración de la Entrada/Salida Paralela para el Cómputo de Altas Prestaciones

Memoria del trabajo de "Iniciación a la Investigación. Trabajo de Fin de Máster" del "Máster en Computació d'altes prestacions", realizada por Sandra Adriana Men dez, bajo la dirección de Dolores Rexachs del Rosario Presentada en la Escuela de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos)

Barcelona

Julio, 2010

Iniciación a la Investigación. Trabajo Fin de Máster
Máster en Computación de Altas Prestaciones

Título: **Configuración de la Entrada/Salida Paralela para el Cómputo de Altas Prestaciones**

Realizada por **Sandra Adriana Méndez** en la Escuela de Ingeniería, en el Departamento de Arquitectura de Computadores y Sistemas Operativos

Dirigida por: **Dra. Dolores Rexachs del Rosario**

Firmado

Directora

Estudiante

A los amores de mi vida “mi querida mami” y ”mi hermosa hija Valeria”. Gracias por toda la paciencia, por acompañarme y entender cada locura que emprendo y por todo el amor que diariamente me hacen sentir.

Agradecimientos

A quién además de ser mi directora, es mi amiga, por hacerme sentir más cerca de mi hogar y ayudarme a creer que lo puedo lograr. Gracias por tu paciencia, tu comprensión, tus consejos y por exigirme diariamente; por todo eso, muchisimas gracias!

A mis compañeros de Máster: César, Aprigio, Eduardo, Carlitos, Andrea, Tharso y Rita; por su compañía en las horas que tuvimos que dedicarle al Máster y porque de todos ustedes aprendí algo. A César, Aprigio y Eduardo por cuidarme y estar a mi lado cuando la tristeza me quiere ganar.

A mis compañeros de los grupos de investigación de “Arquitectura Paralelas” y de “Tolerancia a Fallos” porque con ustedes aprendí a investigar en grupo, a escuchar críticas y aprender de ellas y aunque algunas veces son un poco “duros” para decir las cosas, me prepararon para transmitir el trabajo de investigación y saber responder cuestiones. Gracias, aprendí mucho de ustedes.

Resumen

La E/S Paralela es un área de investigación que tiene una creciente importancia en el cómputo de Altas Prestaciones. Si bien durante años ha sido el çuello de botella"de los computadores paralelos en la actualidad, debido al gran aumento del poder de cómputo, el problema de la E/S se ha incrementado y la comunidad del Cómputo de Altas Prestaciones considera que se debe trabaja en mejorar el sistema de E/S de los computadores paralelos, para lograr cubrir las exigencias de las aplicaciones científicas que usan HPC.

La Configuración de la Entrada/Salida (E/S) Paralela tiene una gran influencia en las prestaciones y disponibilidad, por ello es importante "Analizar configuraciones de E/S paralela para identificar los factores claves que influyen en las prestaciones y disponibilidad de la E/S de Aplicaciones Científicas que se ejecutan en un clúster".

Para realizar el análisis de las configuraciones de E/S se propone una metodología que permite identificar los factores de E/S y evaluar su influencia para diferentes configuraciones de E/S formada por tres fases: Caracterización, Configuración y Evaluación.

La metodología permite analizar el computador paralelo a nivel de Aplicación Científica, librerías de E/S y de arquitectura de E/S, pero desde el punto de vista de la E/S.

Los experimentos realizados para diferentes configuraciones de E/S y los resultados obtenidos indican la complejidad del análisis de los factores de E/S y los diferentes grados de influencia en las prestaciones del sistema de E/S.

Finalmente se explican los trabajos futuros, el diseño de un modelo que de soporte al proceso de Configuración del sistema de E/S paralela para aplicaciones científicas. Por otro lado, para identificar y evaluar los factores de E/S asociados con la disponibilidad a nivel de datos, se pretende utilizar la Arquitectura Tolerante a Fallos RADIC.

Palabras Claves: E/S Paralela, Arquitectura de E/S, Sistemas de Archivos Paralelos, Librerías de E/S, Patrón de E/S, Almacenamiento para HPC

Abstract

Parallel I/O is a research area with growing importance in the world of High Performance Computing (HPC). For many years parallel I/O has been considered super computers “bottleneck” and this problem has increased, with the advent of major augments in computing power in modern computers, until it became an important subject to be considered at the parallel I/O research community, in order to meet the requirements of scientific applications using HPC.

Considering Parallel Input/Output configurations have a great influence on the performance and availability of an I/O System, as main goal was proposed ”to Analyze parallel I/O settings to identify key factors influencing the performance and I/O availability for Scientific Applications running on a HPC cluster.”

In order to perform I/O configuration analysis a methodology is proposed, to identify I/O factors and assess their influence to different I/O configurations, consisting of three phases: Characterization, Setting and Evaluation.

The methodology allows super computer analysis at many levels, as the scientific application, I/O libraries and architecture, from the I/O point of view. Experiments performed for different I/O configurations and results obtained indicate the complexity of I/O factors analysis and different influence degrees on I/O system performance.

Finally, discussion about future work is given, as well as the design of a model that supports I/O system configuration process for parallel scientific applications. On the other hand, to identify and evaluate I/O factors associated with level of data availability, RADIC Fault Tolerant Architecture is proposed as an alternative to use.

Keywords: Parallel I/O, I/O Architecture, I/O Libraries, Access Patterns, High Performance Mass Storage

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. La Entrada/Salida Paralela en el Cómputo de Altas Prestaciones

En las Ciencias de la Computación, la Entrada/Salida (E/S) ha sido el *"hijo abandonado"*, especialmente en el Cómputo de Altas Prestaciones (HPC). De hecho, llamarlo computación, en lugar de manipulación o gestión de datos refleja el sesgo en el vocabulario. Se habla de unidad central de proceso y memoria principal, pero la E/S, hacemos referencia a ella como periféricos o almacenamiento secundario y terciario. Este punto de vista centrado en el cómputo deja constancia en el sentido de que el cómputo científico esta cada vez más en la gestión inteligente de datos . Los computadores más rápidos y potentes, junto con la importancia de su uso en la ciencia como el tercer miembro del triunvirato; teoría, experimento y simulación computacional, significa que la extracción significativa de información, tanto experimental como generada por computador, es el centro para el descubrimiento científico.

La simulación en la epoca de la teraescala puede producir una cantidad enorme de datos, y proporcionar herramientas para visualización de datos o reducción inteligente de datos requiere hardware y software de E/S que puedan mover los datos a través de la red y desde los arrays de dispositivos de almacenamiento [1].

En ciencias de la computación se dice que una aplicación esta limitada por E/S cuando el tiempo total que lleva realizar el cómputo esta determinado principalmente por el período de tiempo que espera que las operaciones de E/S sean completadas. En HPC el estado limitado por E/S no es deseable porque esto significa que la CPU debe detener su operación mientras espera que los datos sean cargados o descargados desde el almacenamiento.

En las últimas dos décadas los sistemas de E/S, tanto para sistemas paralelos como distribuidos, han dejado de ser ese "hijo abandonado" para pasar a ser un centro de atención, debido a

que se ha hecho evidente que son parte de los factores claves que limitan el rendimiento en los sistemas paralelos y distribuidos es la E/S. Esto ha llevado a un creciente y sistemático estudio de la E/S como el cuello de botella de los sistemas paralelos y distribuidos.

Son tres las razones fundamentales por las cuales la E/S se ha convertido en el cuello de botella. Primero, mientras las velocidades de las CPUs se han incrementando en las últimas décadas, la velocidad de los dispositivos de E/S, limitada por la velocidad de los componentes electromecánicos de los discos, ha ido incrementándose pero a un ritmo mucho más lento. Segundo, tanto en paralelo como en distribuido, las múltiples CPU, se emplean simultáneamente, lo que agrava el desfase de velocidad cómputo-E/S. Por último, los nuevos dominios de aplicación, tales como multimedia, visualización, y los problemas de gran desafío, están creando cada vez más demanda para la E/S [2].

Dado los antecedentes y no siendo ajenos a la importancia creciente de la E/S, especialmente en HPC, en este trabajo se presenta un "Análisis de configuraciones de E/S Paralela para identificar los factores claves que influyen en las prestaciones y disponibilidad de la E/S de Aplicaciones Científicas que se ejecutan en un Cluster", este es el primer paso para poder definir un "modelo que de soporte al proceso de configuración del sistema de E/S paralela para aplicaciones científicas". Este modelo permitirá a los consumidores y productores de clústers determinar el sistema de E/S más apropiado para su sistema paralelo de manera de obtener un mayor rendimiento y disponibilidad a un coste y eficiencia aceptable.

Este capítulo se organiza como sigue; en la sección 2 se presenta brevemente que es una Sistema de E/S paralelo, en el sección 3 se presenta la motivación de este trabajo, en la Sección 4 se presenta los objetivos y finalmente en la sección 5 se presenta la organización de esta memoria.

1.2. Sistema de E/S Paralelo

Estos conceptos se explican más profundamente en capítulos posteriores pero una explicación breve en esta sección permitirá al lector entender mejor el objetivo de este trabajo.
Si bien en la literatura se encuentran diversas formas de estructurar el Sistema de E/S Paralelo, para este trabajo se optó por estructurarlo en tres partes, considerando una estructura vertical: Aplicación Científica, Librerías de E/S y la Arquitectura de E/S Paralela.

1.2.1. Aplicación Científica

Para las aplicaciones científicas es importante considerar algunas características que impactan la E/S:

- Carga de trabajo: Tamaño de la solicitud de transferencia y de las respuestas (MBytes/seg),
- Tipo de accesos (secuencial, aleatoria)
- Patrón de acceso (lectura, escrituras, contiguas, no contiguas, colectivos, independientes)

Esto incluye los patrones en los cuales todos los procesos leen/escriben sus datos desde/hacia archivos separados o regiones separadas de un gran archivo compartido. Las E/S pequeñas y no-contiguas es otro patrón común para las aplicaciones científicas, especialmente cuando las aplicaciones usan estructuras de alto nivel, estructura de datos jerárquicas, como un arreglo multidimensional con descomposiciones de datos complejos entre los procesos paralelos.
Otros estudios de carga de trabajos de E/S paralelas [3] muestran patrones de acceso que varían ampliamente entre aplicaciones. Muchas aplicaciones generan grandes secuencias de solicitudes que acceden únicamente a cientos o miles de bytes. Incluso cuando una aplicación mueve datos en grandes bloques, la configuración del hardware de E/S y el software (incluyendo las políticas de caching y prefetching de los sistemas de ficheros) puede causar que el sistema entregue mucha más performance para algunos patrones de acceso que otros.

1.2.2. Librerías de E/S

Las librerías de E/S proporcionan la interfaz para que el programador pueda realizar las operaciones de lectura y escritura de forma más eficiente. Las librerías más usados son POSIX I/O y MPI-IO en la versión implementada más usada ROMIO. Esta librerías nos permiten hacer las operaciones de E/S (lecturas, escrituras).

1.2.3. Arquitectura de E/S Paralela

La Arquitectura de la E/S Paralela implica:

- Conexión: definen el rol que tendrán los nodos del sistema paralelo, estos pueden ser nodos de cómputo, nodos de E/S o nodos de computo-E/S. Además que tipo de red de interconexión se usará para la E/S, distinguiendo si existe una red exclusiva para la E/S (red de E/S) o el tráfico de los datos será compartida con la red de cómputo (red compartida)

- Gestión: es importante conocer que sistemas de ficheros se usará en los diferentes niveles. En HPC se espera utilizar sistemas de ficheros paralelos como por ejemplo GPFS, LUSTRE, PVFS pero también existen cluster que usan sistemas de ficheros distribuidos como por ejemplo NFS

- Colocación: donde se ubicarán los nodos, los dispositivos de almacenamiento y la red de interconexión. Existen básicamente 4 formas de organizar la colocación de los dispositivos: Direct Attached Storage(DAS), Network Attached Storage (NAS), Network Attached Storage Device (NASD), Storage Area Network (SAN)

- Buffer/Cache: la ubicación de los buffer y cache en los nodos de cómputo o de E/S. Esto tiene influencia en las prestaciones dependiendo de la localidad de los datos que tenga la aplicación

- Disponibilidad: garantizar la fiabilidad del almacenamiento, por ejemplo a través de niveles de RAID y garantizar que se van a disponer de los datos cuando se los requiera incluso cuando un nodo de E/S falle, para lo cual se deberá incorporar alguna forma de proteger a los servidores de datos, para este tipo de disponibilidad se utiliza la redundancia de componentes

1.3. Motivación del Trabajo

El aumento del número de unidades de procesamiento en los cluster, los avances tanto en velocidad como en potencia de las unidades de procesamiento y las crecientes demandas de las aplicaciones científicas que utilizan cómputo de altas prestaciones trae mayores exigencias a los sistemas de E/S paralelos.
En muchos casos, el cuello de botella de los sistemas paralelos es la E/S dada las exigencias que debe afrontar. Para poder disminuir la brecha entre CPUs-E/S se deben identificar los factores que influyen en las prestaciones y proponer nuevas soluciones.
Esto lleva a plantear las siguientes preguntas:

- ¿Deben cambiar/adaptarse las aplicaciones o los diseñadores y programadores de sistemas deben mejorar el rendimiento de E/S?

- ¿Qué factores de E/S influyen en el rendimiento?

- ¿Qué factores influyen en la disponibilidad?

Responder a estas preguntas no es trivial. Las aplicaciones tienen un comportamiento de acuerdo a sus propósitos y si bien los programadores o diseñadores pueden realizar las modificaciones necesarias para que la aplicación realice de forma eficiente las operaciones de E/S, estas modificaciones están realizadas para un computador paralelo en particular. Por otro lado sacar el mayor provecho al sistema de E/S requiere que el programador sea un experto en los detalles de los sistemas de E/S, lo que es realmente muy difícil. Las respuestas a las siguientes preguntas:

- ¿Cómo funcionan los dispositivos de almacenamiento?
- ¿Cómo se transfieren los datos entre los dispositivos y la memoria principal?
- ¿Qué interfaces están disponibles para leer y escribir datos?
- ¿Cómo se comportan las aplicaciones?
- ¿Qué patrones de E/S tienen las aplicaciones?
- ¿Cómo podemos hacer las peticiones de E/S para ejecutarlas más eficientemente?
- ¿Cómo maneja ficheros el sistema de ficheros?

pueden ayudar a identificar los factores que influyen en el rendimiento y disponibilidad del sistema de E/S. Estos son numerosos y además una variación en ellos implica un aumento o diminución significativa en rendimiento y disponibilidad. Por esta razón es tan importante disponer de algún instrumento que permita a los usuarios del cómputo de altas prestaciones definir qué sistema de E/S es el más conveniente para su sistema paralelo.

1.4. Objetivos

Dentro de este contexto el objetivo es “Analizar configuraciones de E/S paralela para identificar los factores claves que influyen en las prestaciones y disponibilidad de la E/S de aplicaciones Científicas que se ejecutan en un clúster de HPC”.

Este análisis permitirá identificar los factores de E/S que tienen mayor inlfuencia en las prestaciones y disponibilidad, servirá como punto inicial para diseñar un modelo que ayude a la Configuración de la E/S paralela para Aplicaciones Científicas y permita sintonizarlas. La configuración de E/S paralela debe tener en cuenta tanto la aplicación científica, la técnicas de E/S que las librerias de E/S permiten implementar y la arquitectura de E/S.

Debido a la gran cantidad de factores que influyen en la E/S y lo complejo de la tarea de analizar su influencia en la Configuración es que un primer paso es definir una metodología que permita obtener información sobre el impacto en prestaciones y disponibilidad teniendo en cuenta los factores de E/S.
Tanto los factores identificados como la metodología seguida son base para los trabajo a largo plazo:

- Diseñar un modelo de configuración de E/S paralela para aplicaciones científicas
- Incluir herramientas de simulación para validar el modelo configuraciones de E/S
- Seleccionar y configurar una arquitectura tolerante a fallos a nivel de servidores de datos para garantizar la disponibilidad del Sistema de E/S Paralelo

1.5. Organización de la Memoria

En este capítulo se han presentado de manera sucinta el problema que presentan muchas aplicaciones científicas limitadas por el sistema de E/S paralelo, la motivación y el objetivo del presente trabajo. A continuación se presenta la estructura de esta memoria:

- Capítulo 2: Conceptos de la E/S paralela

  En este capítulo se presentará los conceptos básicos de la E/S Paralela. Se hará una breve descripción de las aplicaciones que son intensivas de E/S. El estudio de Patrones de E/S sobre las Aplicaciones Científicas y las Técnicas de E/S utilizadas para tratar con los patrones de E/S. Además se explica los dispositivos, Redes y Librerias para la E/S, y Sistemas de Ficheros Paralelos junto con una explicación de los benchmarks usados en E/S para estos componentes.

- Capitulo 3: Sistema de E/S Paralela

  En este capítulo se presenta un esquema de organización para el Sistema de E/S paralela que incluye Aplicación Científica, Librerías de E/S y Arquitectura de E/S Paralela. Además se presenta un esquema de los niveles de E/S Paralela que es también una forma de ver a la E/S Paralela.

- Capítulo 4: Estado del Arte

  En este capítulo se presenta un estudio de lo últimos 20 años de investigación en el área de E/S Paralela. Se incluyen los principales autores de E/S Paralela y los grandes aportes. Y se termina con una visión de los desafíos que la E/S deberá afrontar para la era de la exascala.

- Capítulo 5: Metodología de la Configuración de E/S Paralela

  En este capítulo se presenta la estructura y fases de la Metodología propuesta para la Configuración de E/S Paralela. Con explicación detallada de los factores de E/S que se deben considerar, los indices de E/S a utilizar y la herramientas que se necesitan para cada fase.

- Capítulo 6: Evaluación Experimental

  En este capítulo se muestran los experimentos realizados y los resultados obtenidos siguiendo las fases de la Metodología propuesta.

- Capítulo 7 Conclusiones y Trabajos Futuros

  Finalmente en este capítulo se presentan las conclusiones en base a los experimentos realizados. También se presentan los trabajos futuros.

Capítulo 2

Conceptos de la E/S paralela

La E/S Paralela es un factor que limita de manera significativa las prestaciones de las aplicaciones de HPC. Por lo tanto, comprender que se entiende por E/S paralela es fundamental para poder definir una configuración de la E/S paralela que logre mejorar la *performance*. En este capítulo se presentan los conceptos de E/S paralela en los que se sustenta este trabajo.

2.1. Dispositivos de E/S

Los computadores almacenan datos en una variedad de formas, incluyendo memoria electrónica, discos magnéticos, discos ópticos y cintas. Estos son clasificados en tres niveles de jerarquía, las cuales se distinguen por su volatilidad, costo, tiempo de acceso y uso típico (figura 2.1).

El almacenamiento primario, es el nivel superior de la jerarquía, incluye todos los tipos de memorias electrónicas. Los computadores usan almacenamiento primario para almacenar datos e

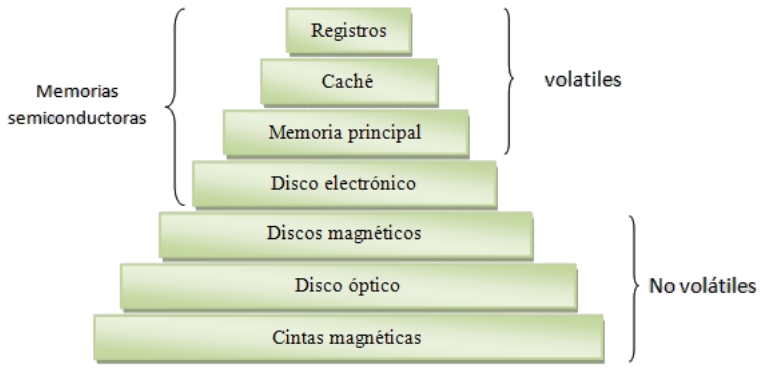


Figura 2.1: Jerarquía de Almacenamiento

instrucciones para que los programas puedan ejecutarse. Es volatil y los tiempos de acceso estan en el orden de los microsegundos, el costo de almacenmiento del los niveles superiores es muy alto en comparación a los otros niveles. El almacenamiento terciario incluye a cintas magnéticas y algunos medios ópticos. El almacenamiento secundario incluye a los discos mágneticos, uno de los dispositivos más usados en la E/S Paralela, éstos son medios de almacenamiento no volatil, con una relación costo-capacidad aceptable y es por esta razón el medio más usado para E/S en los cláster de computadores. Los sistemas de múltiples discos son también muy usados para lograr una mayor tasa de transferencia. Los discos debido a sus componentes mecánicos, tienen tiempo de accesos elevados, es por esta razón que las operaciones de E/S tienen una alta latencia debido a que los accesos a disco penalizan el rendimiento de la aplicación.

2.1.1. Dispositivos de Almacenamiento

Los dispositivos de almacenamiento [1] son descritos en base a un conjunto de características básicas, tales como capacidad, tasa de transferencia y tiempo de acceso. La capacidad es la cantidad de datos que un dispositivo puede almacenar. La unidad básica de capacidad es el *byte* o *bit*. Los discos actuales pueden almacenar de cientos de GBytes o incluso algunos TBytes. La velocidad con la que un disco puede leer o escribir datos se denomina tasa de transferencia, los discos modernos manejan tasas de transferencias del orden de los GBytes/segundo aunque, como veremos, en el momento de evaluar la rendimiento no pasan de los cientos MBytes/segundos debido a la sobrecarga que provocan otros factores, como por ejemplo el sistema de fichero provocan. Cuando un computador realiza una solicitud para leer o escribir un dato, siempre hay un retardo hasta que el primer byte se mueve, este retardo se conoce como tiempo de acceso.

Los componentes principales de una unidad de disco magnético son un conjunto de platos que rotan cabezas que leen y escriben los datos, un conjunto de brazos que mueven a las cabezas aproximadamente a los largo del radio del disco, y un actuador que pivotea a los brazos.

Las pistas en los discos son lógicamente dispuestas en anillos concéntricos. Cada pista consiste de varios sectores que contienen un número fijo de dominios y por lo tanto un número fijo de bytes. Un tamaño común de un sector es de 512 Bytes de datos de usuario, aunque está cambiando. Los sectores de un disco son también llamados bloques. Sin embargo, un bloque podría también hacer referencia a una unidad de tamaño fijo de datos que un sistema operativo o un subsistema de disco maneja en una operación de E/S. Un bloque puede contener el mismo número de bytes que un sector o un multiplo de ese número.

2.1.2. Performance de los Discos

La capacidad de la unidad de disco depende del número y tamaño de los platos; y de la densidad de area de los datos. La densidad de área se ha incrementado rápidamente en las décadas pasadas, mientras que los platos se incrementaron más lentamente. El número de platos es usualmente elegido para adaptar el tamaño físico y capacidad necesaria de una unidad de disco. Sin embargo, incluso aunque el espacio lo permitiera, usar muchos platos es impracticable por dos razones. Primero, un simple actuador mueve todas la cabezas, asi que incrementar el número de cabezas incrementa la masa en la que el actuador debe moverse, lo que trae un incremento en el tiempo de acceso. Segundo, unidades con más platos, incrementan el riesgo que cualquiera de estos sufran una caída de la cabeza, lo cual podría provocar averías.

La densidad de área es el producto de la densidad de pista y la densidad lineal. La densidad de pista es el número de pista por unidad de radio de disco, y la densidad lineal es el número de bits por unidad de longitud en la pista.

Una importante limitación en el diseño de las unidades de disco es que el incremento de la densidad de área por un factor dado no incrementa la tasa de transferencia en el mismo factor. De los dos factores que influyen en la densidad de área (densidad lineal y densidad de pista) únicamente la densidad lineal contribuye a la tasa de transferencia. La tasa de transferencia no ha mejorado tan rápidamente como la densidad lineal, porque la velocidad rotacional ha incrementado también lentamente. El principal limitante parece ser la cabeza y su soporte electrónico, que no puede leer y escribir datos arbitrariamente a alta velocidad. El resultado es, como la capacidad de las unidades de disco incrementa, también se incrementa el tiempo para acceder a los datos [1].

2.1.3. Necesidad del Paralelismo

La rapidez con la que las unidades de discos leen o escriben datos esta en el orden de los cientos de MB/seg que es muy baja en comparación a la velocidad con la que trabajan la unidades de cómputo (miles de GB/seg), una solución obvia es usar sistemas paralelos de E/S, como usan los supercomputadores paralelismo de procesamiento. La capacidad y *throughput* de una simple unidad de disco esta limitada para cumplir con las necesidades de las aplicaciones de altas prestaciones. Una solución es juntar varias unidades de discos que trabajen juntas. Los *disk arrays* son una colección ordenada de múltiples unidades de discos que proporcionan varias características de rendimiento y confiabilidad. Una de las principales técnicas de distribución de datos sobre múltiples dispositivos es el *stripping* de disco (figura 2.2).

El *stripping* se puede hacer a nivel de byte, o en bloques. El *stripping* a nivel de byte significa

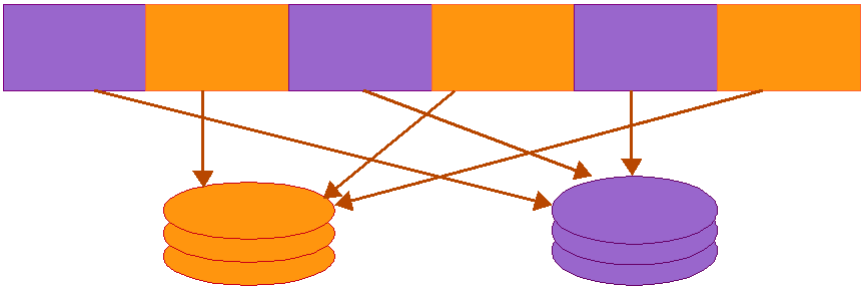


Figura 2.2: *Disk Stripping*, se envian de forma alternada bloques de datos desde la memoria a discos separados en paralelo

que el archivo se divide en ”trozos de tamaño de byte”. El primer byte del archivo es enviado a la primera unidad, luego la segunda a la segunda unidad, y así sucesivamente. A veces el *stripping* a nivel de byte se realiza como un sector de 512 bytes. El *stripping* a nivel de bloque significa que cada archivo se divide en bloques de un tamaño determinado y se distribuyen a las diversas unidades. Un computador que escribe una gran cantidad de datos podría dividir los archivos en trozos y escribirlos simultáneamente en los *disk arrays*. Además de multiples discos el computador debe tener canales de E/S separados para cada disco, hardware y software apropiado para enviar sobre todos los canales simultaneamente [1].

El primer parámetro clave en los *disk arrays* es el *stripe width*. El *stripe width* es el número de *stripes* que pueden ser leidos o escritos de forma simultánea y es igual al número de discos del *disk arrays*. El segundo parámetro es *stripe size*, también conocido como *block size, chunk size, stripe length*, este hace referencia al tamaño de *stripes* escritos en cada disco. Este puede ser tan pequeño como un byte o varios sectores de disco. La elección apropiada depende de como el *disk arrays* será usado. En principio, un sistema podría distribuir los *stripe* sobre cientos de discos y mover los datos en muchos GBytes/segundos.

El problema del esquema de *striping* es la Fiabilidad. Si un archivo es dividido sobre multiples discos, la perdida de un disco podría causar que el archivo completo sea inútil, y el riesgo de la perdida de datos es aproximadamente proporcional al número de discos. Para sistemas que manejan datos críticos, el riesgo es muy alto.

Para tratar el problema de la fiabilidad en los *disk arrays* y mejorar la rendimiento de E/S, un grupo de investigación de la Universidad de California, Berkeley, a propuesto un arreglo de discos redundantes de bajo costo (RAID (*Redundant Array of Independent Disk*)) [4]. La idea central en RAID es replicar datos sobre varios discos de manera que los datos no se pierdan si alguno de

Figura 2.3: JBOD o RAID lineal

los discos falla. El artículo original de RAID estableció cinco estrategías denominadas niveles de RAID, con diferentes características en rendimiento y diferentes formas de replicación de datos. En las siguientes secciones se presentan los diferentes niveles de RAID.

JBOD

JBOD (también conocido como RAID Lineal, 2.3). Aunque la concatenación de discos (también llamada JBOD, de Just a Bunch Of Drives, 'Sólo un Montón de Discos') no es considerado un RAID, sí es un método popular de combinar múltiples discos duros físicos en un solo disco virtual. Como su nombre indica, los discos son meramente concatenados entre sí, de forma que se comportan como un único disco. En este sentido, la concatenación es como el proceso contrario al particionado: mientras éste toma un disco físico y crea dos o más unidades lógicas, JBOD usa dos o más discos físicos para crear una unidad lógica.

Una ventaja de JBOD sobre RAID 0 es que, en caso de fallo de un disco, en RAID 0 suele producirse la pérdida de todos los datos del conjunto, mientras en JBOD sólo se pierden los datos del disco afectado, conservándose los de los restantes discos. Sin embargo, JBOD no supone ninguna mejora de rendimiento.

RAID 0

El RAID 0 es el nombre que recibe el *disk stripping* (figura 2.4). Este nivel de RAID requiere un mínimo de 2 discos y no ofrece redundancia,por lo tanto, la perdida de un disco provoca la corrupción de los datos. Sin embargo, en algunos casos es usado en combinación con otros niveles de RAID para mejorar su rendimiento.

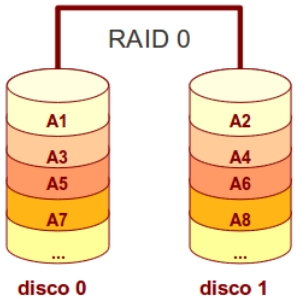


Figura 2.4: Nivel de RAID 0

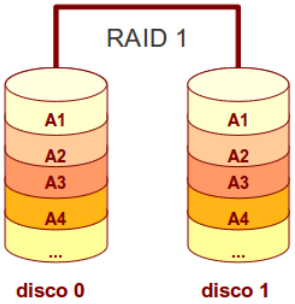


Figura 2.5: Nivel de RAID 1

RAID 1

En RAID 1 (figura 2.5), también llamado *mirroring*, se duplican todos los datos de un disco a otro. De esta manera se asegura la integridad de los datos y la tolerancia al fallo, pues en caso de avería, la controladora del RAID sigue trabajando con los discos no dañados sin detener el sistema. Los datos se pueden leer desde la unidad o matriz duplicada sin que se produzcan interrupciones. Se necesita un mínimo de dos unidades para implementar un RAID 1.

RAID 2

El RAID 2 es un nivel que utiliza la la técnica de código Hamming, usada para detectar y corregir errores en memorias de estado sólido. El código ECC *Error Correction Code* se intercala a través de varios discos a nivel de bit. Puesto que el código Hamming se usa tanto para detección como para corrección de errores, RAID 2 no hace uso completo de las amplias capacidades de detección de errores contenidas en los discos. Las propiedades del código Hamming también restringen las configuraciones posibles para RAID 2, particularmente el cálculo de paridad de los discos. Por lo tanto, RAID 2 no ha sido implementado en productos comerciales, debido a que requiere características especiales en los discos y no usa discos estándares.

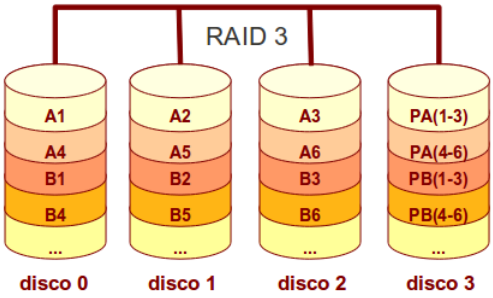


Figura 2.6: Nivel de RAID 3

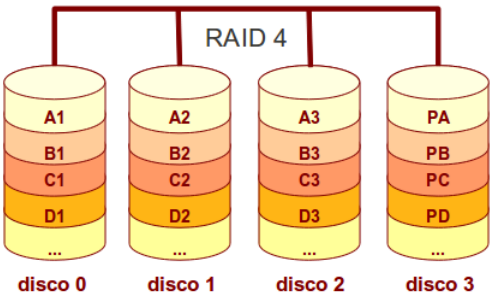


Figura 2.7: Nivel de RAID 4

RAID 3

El RAID 3 (figura 2.6) dedica un disco a información de paridad. La información de ECC *error checking and correction* se usa para detectar errores. La recuperación de datos se consigue calculando la función O exclusivo (XOR) de la información registrada en los otros discos. En RAID 3 se accede a todos los discos de una vez, por lo tanto, el tiempo para realizar un operación de E/S dependerá del la mayor latencia de cualquier disco del RAID. Se necesita un mínimo de tres unidades para implementar una solución RAID 3.

RAID 4

RAID 4 (figura 2.7), similar a RAID 3, almacena bytes de paridad para N discos en un disco de paridad separado, pero en lugar distribuir las solicitudes sobre todos los discos, este accede a los discos individualmente. Basa su tolerancia al fallo en la utilización de un disco dedicado a guardar la información de paridad calculada a partir de los datos guardados en los otros discos. En caso de avería de cualquiera de las unidades de disco, la información se puede reconstruir en tiempo real mediante la realización de una operación lógica de O exclusivo. El problema que presenta RAID 4 es que almacena los datos de paridad en un disco que se convierte en el cuello de botella para este RAID.

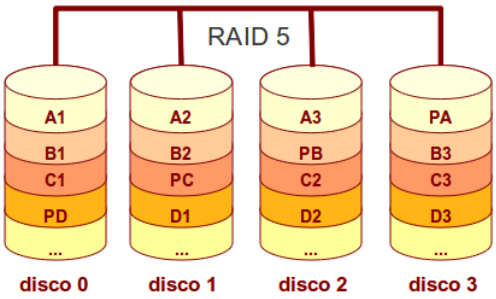


Figura 2.8: Nivel de RAID 5

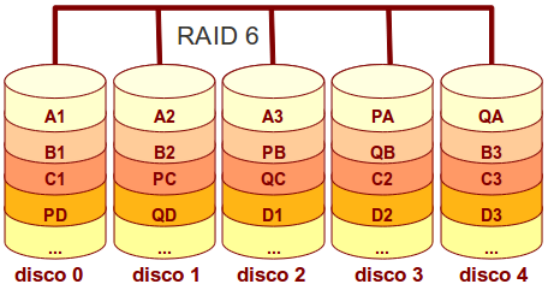


Figura 2.9: Nivel de RAID 6

RAID 5

Para evitar el problema de cuello de botella del RAID 4 con el disco de paridad, el RAID 5 (figura 2.8) no asigna un disco específico a la paridad sino que asigna un bloque alternativo de cada disco a esta misión de escritura. Al distribuir la función de comprobación entre todos los discos, se disminuye el cuello de botella y con una cantidad suficiente de discos puede llegar a eliminarse completamente, proporcionando una velocidad equivalente a un RAID 0. Se necesita un mínimo de tres unidades para implementar una solución RAID 5.

RAID 6

Similar al RAID 5, RAID 6 (figura 2.9) incluye un segundo esquema de paridad distribuido por los distintos discos y por tanto ofrece tolerancia extremadamente alta a los fallos y a las caídas de disco, ofreciendo dos niveles de redundancia. Hay pocos ejemplos comerciales en la actualidad, ya que su coste de implementación es mayor al de otros niveles RAID, ya que las controladoras requeridas que soporten esta doble paridad son más complejas y caras que las de otros niveles RAID. Así pues, comercialmente no se implementa.

Cuadro 2.1: Comparación de los diferentes niveles de RAID, N es la cantidad de discos usados, excepto para RAID 6 donde donde MN es la capacidad

Nivel de RAID	**Método de Protección**	**Uso de Espacio**	**Bueno en...**	**Malo en...**
0	ninguno	N	prestaciones	protección de datos
1	espejo	2N	prestaciones para lecturas y protección de datos	uso eficiente de espacio
2	código hamming	apróx. 1.5N	tasa de transferencias de datos	uso eficiente de espacio y throughput (solicitudes atendidas)
3	paridad	N+1	tasa de transferencias de datos	throughput (solicitudes atendidas)
4	paridad	N+1	throughput (solicitudes de lectura atendidas)	prestaciones para escrituras
5	paridad	N+1	throughput (solicitudes atendidas)	tasa de transferencias de datos
6	P+Q (paridad)	N+2 or MN+M+N	protección de datos	prestaciones para escrituras
10	espejo	2N	prestaciones	uso eficiente de espacio

RAID 0+1 ó RAID 0/1 ó RAID 10

RAID 10 es una combinación de los arrays anteriores que proporciona velocidad y tolerancia al fallo simultáneamente. El nivel de RAID 0+1 fracciona los datos para mejorar el rendimiento, pero también utiliza un conjunto de discos duplicados para conseguir redundancia de datos. Al ser una variedad de RAID híbrida, RAID 0+1 combina las ventajas de rendimiento de RAID 0 con la redundancia que aporta RAID 1. Sin embargo, la principal desventaja es que requiere un mínimo de cuatro unidades y sólo dos de ellas se utilizan para el almacenamiento de datos. Las unidades se deben añadir en pares cuando se aumenta la capacidad, lo que multiplica por dos los costes de almacenamiento. Una resumen de los diferentes niveles de RAID se presenta en la tabla 2.1

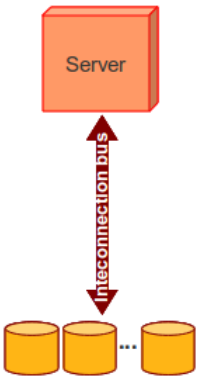


Figura 2.10: Manera tradicional de conexión

2.1.4. Redes de Interconexión en la E/S Paralela

La manera tradicional de interconectar los dispositivos de almacenamiento con los computadores ha sido a través de una arquitectura de bus (figura 2.10).

Estas eran conexiones del tipo Server-Bus-Dispositivos. En estas conexiones puede haber 1 ó varios servidores dedicados a gestionar los datos desde y hacia los dispositivos de E/S (1 ó varios discos). Este modelo se denomina DAS (*Direct Attached Storage*, almacenamiento directamente conectado). Este modelo ha evolucionado a lo largo del tiempo dotando de inteligencia a la gestión del almacenamiento: memorias caché propias, RAID. El estandar SCSI (Small Computer System Interface) define un conjunto de normas para la conexión física y la transferencia de datos entre los nodos de cómputo o los nodos de E/S y dispositivos de E/S. SCSI es una interfaz inteligente, *peripheral*, *buffered*, *peer to peer*. SCSI permite hasta 16 dispositivos conectados con un solo cable. El cable y el adaptador host forman el bus SCSI, y funciona de manera independiente del resto del computador. Cada uno de los ocho equipos se le da una dirección única del BIOS SCSI, que van de 0 a 7 para un bus de 8 bits o de 0 a 15 para un bus de 16 bits. Los Dispositivos de los procesos que hacen la solicitud de E/S se llaman procesos iniciadores. Los *targets* son los dispositivos que llevan a cabo las operaciones solicitadas por los iniciadores. Cada *target* puede acomodar hasta ocho dispositivos de otras, conocidas como unidades lógicas, y cada uno se le asigna una LUN (*Logic Unit Number*). Los comandos que se envían a la controladora SCSI identifican los dispositivos en función de su LUN. Otras versiones de SCSI como SSA (*Serial Storage Architecture*), FCP (*SCSI-over-Fibre Channel Protocol*, SAS (*Serial Attached SCSI*, ADT(*AutomationDrive Interface-Transport Protocol* y UAS (*USB Attached SCSI* dejan la forma tradicional paralela SCSI y realizan la transferencia a través de comunicación serial. El SCSI serial tiene tiene tasas de transferencias más rápidas, *hot swapping* y una mejora en el aislamiento

de fallas. Otra variante es iSCSI que conserva el paradigma SCSI, especialmente el conjunto de comandos, a través de un SCSI-3 embebido en TCP/IP. SCSI es muy usado en nodos de computo y de E/S de altas prestaciones, siendo los más usados en los RAIDs de los servidores.

En las redes de almacenamiento (SAN) los elementos que entran a interactuar son los mismos que en el modelo anterior, pero esta vez cambia la forma de interconexión: lo que era un bus paralelo se convierte en una infraestructura de red que permite ir más allá en el número de dispositivos y servidores y en las distancias entre los estos. La tecnología de interconexión que hace posible una SAN es el estándar Fiber Channel. Existe otro modelo para compartir almacenamiento a través de una red: el denominado NAS (Network Attached Storage, almacenamiento conectado a red). Un dispositivo NAS se conecta directamente a las redes de datos tradicionales basadas en TCP/IP a través de interfaces Ethernet y pone a disposición de los equipos de esta red el almacenamiento que gestiona mediante un protocolo de sistema de ficheros en red (NFS, CIFS o incluso HTTP). A continuación se presentan con un poco más de detalles estos modos de interconexión.

Direct Attached Storage (DAS)

Es el método tradicional de almacenamiento y el más sencillo. Consiste en conectar el o los dispositivos de almacenamiento directamente al servidor o estación de trabajo, es decir, físicamente conectado al dispositivo que hace uso de él. En DAS las aplicaciones y programas de usuarios hacen sus peticiones de datos al sistema de ficheros directamente. Los protocolos principales usados en DAS son SCSI,SAS y Fibre Channel, tradicionalmente un sistema DAS habilita capacidad extra de almacenamiento a un servidor. En la figura 2.11 se muestra un esquema de DAS.

Storage Area Network (SAN)

Una SAN (figura 2.12) es una red para el área de almacenamiento, es una red para conectar servidores, RAIDs y librerías de soporte. Principalmente, está basada en tecnología Fibre Channel y más recientemente en iSCSI. Su función es la de conectar de manera rápida, segura y fiable los distintos elementos que la conforman. Una SAN permite a varios servidores acceder a varios dispositivos de almacenamiento en una red compartida. En SAN el almacenamiento es remoto. SAN utiliza diferentes protocolos de acceso como Fibre Channel y Gigabit Ethernet. Las SAN proveen conectividad de E/S a través de las computadoras host y los dispositivos de almacenamiento combinando los beneficios de tecnologías Fibre Channel y de las arquitecturas de redes.

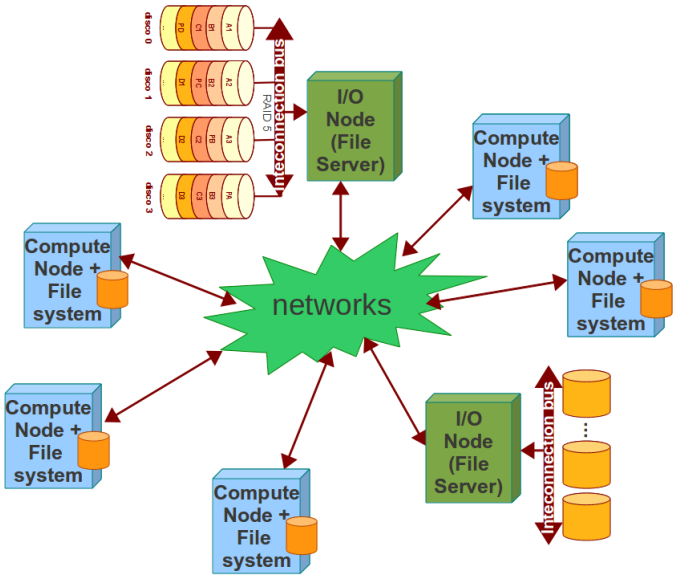


Figura 2.11: Un ejemplo de un Esquema DAS (Direct Attached Storage)

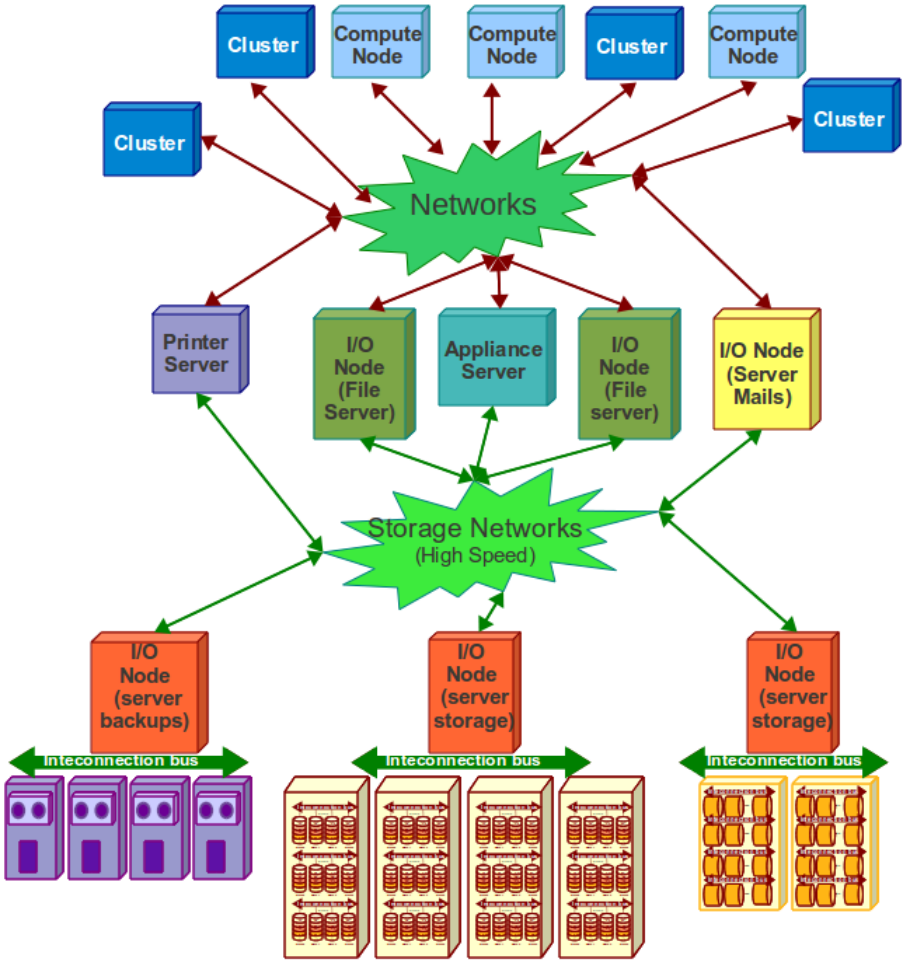


Figura 2.12: Esquema SAN (Storage Area Network)

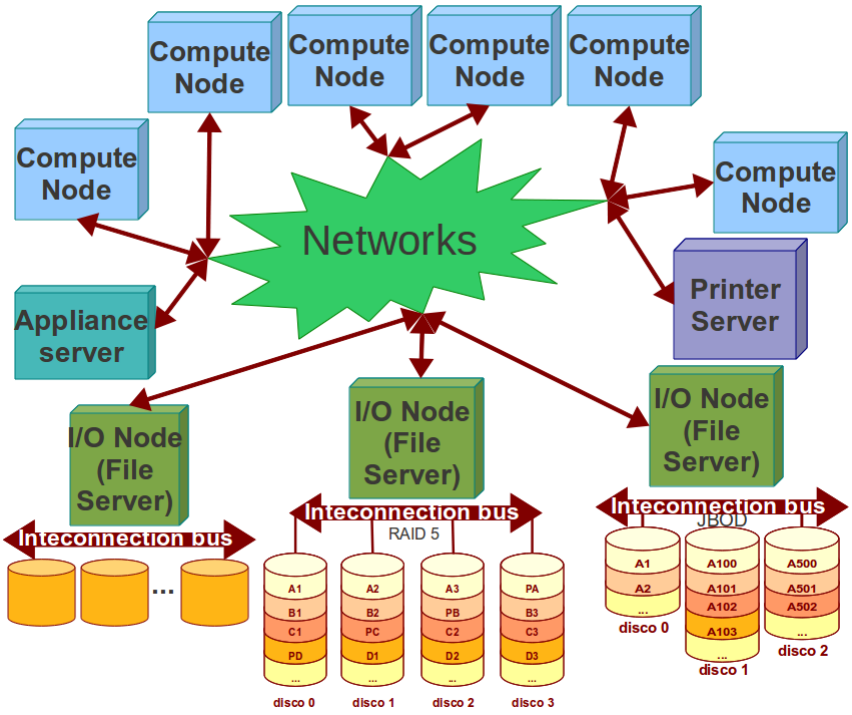


Figura 2.13: Esquema NAS (Network Attached Storage)

Network Attached Storage (NAS)

Los sistemas NAS (figura 2.13) son dispositivos de almacenamiento a los que se accede desde los computadores a través de protocolos de red (normalmente TCP/IP). Los protocolos de comunicaciones NAS son basados en ficheros por lo que el cliente solicita el fichero completo al servidor y lo maneja localmente, están por ello orientados a información almacenada en ficheros de pequeño tamaño y gran cantidad. Los protocolos usados son protocolos de compartición de ficheros como NFS, Microsoft Common Internet File System (CIFS). Muchos sistemas NAS cuentan con uno o más dispositivos de almacenamiento para incrementar su capacidad total. Normalmente, estos dispositivos están dispuestos en RAID.

NetworkAttached Storage Devices (NASD)

La forma más común para los computadores paralelos de acceder a archivos compartidos es a través de un nodo de E/S o servidor, un computador que actua como una puerta al dispositivo de almacenamiento. El nodo acepta solicitudes desde otros nodos de cómputos en la red para acceder a datos almacenados en sus discos. Un servidor puede gestionar muchos discos, pero cada solicitud debe pasar por el servidor. Parte del trabajo del nodo de E/S es autenticar las solicitudes y asegurar que el usuario tiene permiso para acceder al archivo solicitado. También puede cachear datos en

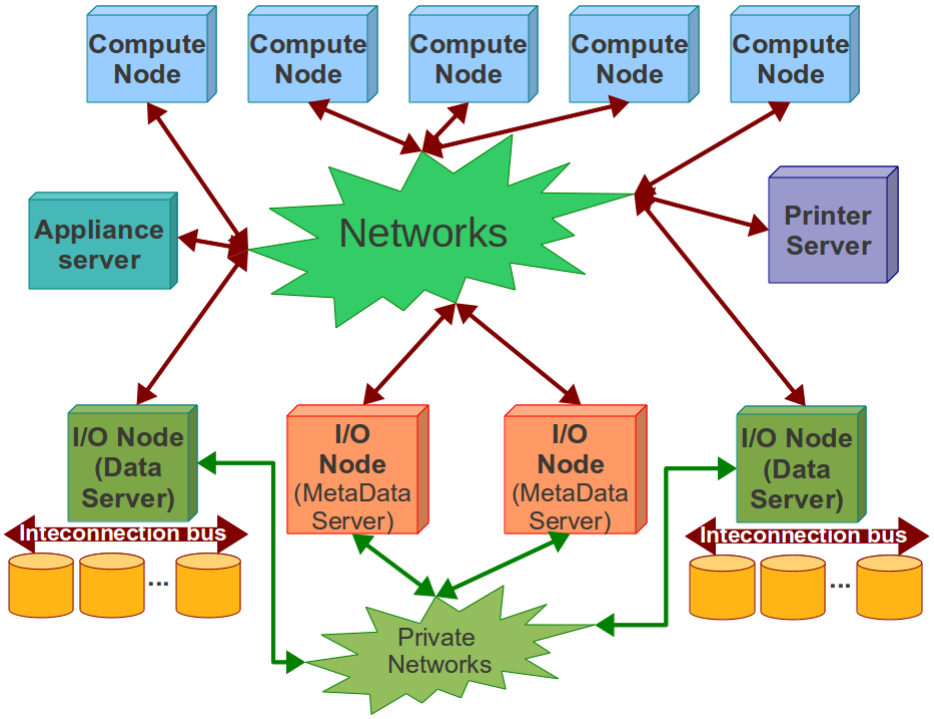


Figura 2.14: Una propuesta NASD (Network Attached Storage Devices)

almacenamiento principal para mejorar la rendimiento.

Sin embargo, ya que todas las solicitudes y todos los datos se mueven a través del servidor este puede ser el cuello de botella cuando el tráfico de E/S es muy alto. Una forma de disminuir este cuello de botella es adjuntar los dispositivos de almacenamiento directamente a la red y usar el servidor solo para algunas operaciones. El servidor de archivo aceptará y autenticará las solicitudes pero solo mediará la transferencia de los datos, simplemente le indicará al disco que debe leer o escribir datos directamente en la red. Este tipo de arquitectura de almacenamiento requiere un periférico inteligente llamado *NetworkAttached Storage Devices (NASD)*, este sistema ofrece dos ventajas:

- una vez que el servidor inicia la transferencia de datos entre el dispositivo de almacenamiento y el nodo cómputo, su trabajo en la transferencia ha concluido, de esta forma esta libre para atender otras solicitudes.
- el servidor ya no es el cuello de botella para la transferencia de datos. Múltiples discos pueden mover los datos sobre la red al mismo tiempo, tanto como el ancho de banda de la red lo permita.

En la figura 2.14 se presenta una propuesta de NASD.

En la tabla 2.2 se presentan un resumen de los 4 tipos de conexiones.

Cuadro 2.2: Comparación de los diferentes tipos de conexiones

	DAS	**SANs**	**NAS**	**NASD**
Tipo de Transporte	SCSI; Fibre Channel	Fibre Channel; IP (iSCSI)	IP	iSCSI, TCP/IP-RPC
Tipo de datos	Nivel Bloque	Nivel Bloque	Nivel fichero	Nivel objeto
Aplicaciones Típicas	Cualquiera	Bases de datos; OLTP; ERP	Servido de ficheros; software de desarrollo; CAD/CAM	Cómputo científico
Pros	Facil de usar; Menos problemas de compatibilidad	Escalable; Caracterísitcas de disponibilidad; Mejor uso; Permite continuidad; Gestión centralizada	Fácil de instalar; Servidores de propósito general	Escalable; Seguridad
Contras	Gestión compleja; Escalabilidad limitada	Comparativamente caro; Requiere conocimientos especializados; Difícil de configurar; Interoperabilidad	No adecuado para algunos tipos de aplicaciones; Puede tener dificultades para escalar	Requiere OSD y MSD

2.1.5. Benchmarks de Dispositivos de E/S

Existen diversos benchmarks para evaluar las prestaciones de los discos pero casì siempre estos benchmarks evaluan también las prestaciones de los sistemas de ficheros debido a la influencia que tienen estos en las prestaciones de las unidades de discos. A continuación se presentarán tres benchmark muy usados y que fueron empleados para caracterizar el sistema de E/S para el presente trabajo.

Bonnie++

Bonnie++ es utilizado para comprobar el rendimiento de discos duros y sistemas de archivos, permite la creación de tests de lectura, escritura y borrado de archivos de diversos tamaños. Inicialmente realiza una serie de pruebas en un archivo (archivos) de tamaño conocido (200 MB). Por cada prueba, Bonnie reporta el número de KBytes/seg procesados, y el %CPU usado (tanto del sistema como del usuario). También realiza pruebas (seis últimas mediciones) para grandes servidores y máquinas que manejan decenas de miles de archivos de mail. El patrón de acceso a disco crea un fichero grande, el cual debe ser como mínimo el doble de la RAM del sistema sobre el que realiza las siguientes acciones de Entrada/Salida:

- *write* secuencial por caracteres: Utiliza la macro *putc* de la biblioteca stdio con un loop que realiza las escrituras lo suficientemente pequeñas para que quepan en cualquier cache. La carga de CPU corresponde al código stdio y al espacio que tiene el sistema operativo para asignar archivos.

- secuencial por bloques: Utiliza la función *write* con bloques de 16KB por defecto, aunque se puede variar en su ejecución. La carga del CPU corresponde al espacio que tiene el sistema operativo para asignar archivos.

- Reescritura secuencial: del archivo creado se lee cada bloque (de 16KB) con la función read y se reescribe con *write*; la función *lseek* reposiciona luego el puntero de lectura/escritura del fichero. Como no se realizan asignaciones de espacio y la E/S está bien localizada, esta operación evalúa específicamente la capacidad de *caching* del fichero y la velocidad de transferencia de datos.

- Lectura secuencial de caracteres: Utiliza la macro *getc* de la biblioteca stdio. La carga del CPU corresponde al código *stdio* y a la entrada secuencial.

- Lectura secuencial en bloques: Utiliza la función *read*. Evalúa el rendimiento de entradas secuenciales a disco.
- Búsquedas aleatorias: Lanza 3 procesos (por defectos) en paralelo que realizan en conjunto 8000 búsquedas a posiciones aleatorias del archivo. En cada caso el bloque es leído con read y en el 10 % de dichas búsquedas se reescribirá el bloque con *write*.

IOzone

IOzone es una herramienta destinada a comprobar el rendimiento de un sistema de archivos. La aplicación genera y mide una gran cantidad de operaciones sobre ficheros (escritura, reescritura, lectura, relectura, lectura/escritura aleatoria, etc.) El benchmark devuelve el *bandwidth* en MBytes/seg de la entrada/salida para las siguientes operaciones: read, write, reread y rewrite. Las operaciones se explican a continuación:

- Escritura: este test mide el *bandwidth* para escribir un nuevo archivo. Cuando un nuevo archivo es creado se necesita los datos a ser almacenados y se agrega información adicional para hacer un seguimiento de donde se encuentra la información en el disco. Esta información adicional es llamada "metadato". Consiste de una información de directorio, espacio de ubicación y cualquier otra información asociada con el archivo que no es parte de los datos contenidos en el archivo. Es normal entonces que rendimiento de la primera escritura sea menor que el rendimiento de rescritura.
- Reescritura: mide el rendimiento de la escritura de un archivo ya existente.
- Lectura: mide el rendimiento de la lectura de un archivo existente.
- Relectura: mide el rendimiento de la lectura de un archivo que fue recientemente leído. Es normal que el rendimiento sea mayor debido a que generalmente el sistema operativo mantiene una cache de datos para los archivos que fueron recientemente leídos. Esta cache puede ser usada para satisfacer las lecturas y mejorar el rendimiento.
- Lectura aleatoria: mide el rendimiento de la lectura de un archivo con accesos realizados sobre posiciones aleatorias dentro del archivo. El rendimiento sobre este tipo de actividad puede verse influenciado por el tamaño de cache del sistema operativo, número de discos, latencia de búsqueda, y otros.
- Escritura aleatoria: Mide el rendimiento de la escritura de un archivo con accesos realizados sobre posiciones aleatorias dentro del archivo. El rendimiento sobre este tipo de actividad

puede verse influenciado por el tamaño de cache del sistema operativo, número de discos, latencia de búsqueda, y otros.

hdparm

hdparm es un programa usado para modificar los parámetros de los discos duros en Linux. Entre otras cosas puede usarlo para activar o desactivar el UDMA (Ultra-Direct Memory Access) para un dispositivo y comprobar su tasa de transferencia sostenida. Las opciones para usarlo como benchmark son:

- T realiza varias lecturas a cache para usarla como punto de referencia y propósitos de comparación. Para que los resultados sean significativos, esta operación debería ser repetida 2 a 3 veces en una forma en la que el sistema esté inactivo (que no estén activos otros procesos) con al menos un par de *Megabytes* libres de memoria. Esta medida es esencialmente una indicación del throughput del procesador, la cache, la memoria del sistema bajo prueba.
- t realiza varias lecturas al dispositivo para usarlo como punto de referencia o comparación. Para que los resultados sean significativos, esta operación debería ser repetida 2 a 3 veces con al menos un par de *Megabytes* de memoria libre. Esta muestra la velocidad de lectura a través de buffercache para el disco sin ningún caching previo de datos. Esta medida es una indicación de que tan rápido el controlador puede sostener la lectura de datos secuencial bajo Linux, sin ningún overhead del sistema de archivo.

Esta es la linea de comando que permite ver qué opciones activadas para un disco

```
hdparm /dev/hda
```

y este para medir la tasa de transferencia de un disco

```
hdparm -Tt /dev/hda
```

2.2. Sistemas de Ficheros

Los sistemas de archivos o ficheros (file system en inglés), estructuran la información guardada en una unidad de almacenamiento (normalmente un disco duro de una computadora), que luego será representada ya sea textual o gráficamente utilizando un gestor de ficheros. La mayoría de los sistemas operativos manejan su propio sistema de archivos. Lo habitual es utilizar dispositivos de almacenamiento de datos que permiten el acceso a los datos como una cadena de bloques de un

mismo tamaño, a veces llamados sectores, usualmente de 512 bytes de longitud. El software del sistema de archivos es responsable de la organización de estos sectores en archivos y directorios y mantiene un registro de qué sectores pertenecen a qué archivos y cuáles no han sido utilizados. En la práctica, un sistema de archivos también puede ser utilizado para acceder a datos generados dinámicamente, como los recibidos a través de una conexión de red (sin la intervención de un dispositivo de almacenamiento). Los sistemas de fihceros también deben:

- mover los datos de forma eficiente entre los dispositivos de almacenamiento y la memoria principal
- coordinar los accesos concurrentes para multiples procesos a un mismo archivo
- colocar lo bloques de datos en el dispositivo de almacenamiento para archivo específicos, y liberar los bloques de los archivos que fueron borrados
- recuperar la mayor cantidad de datos como sea posible si el sistema de archivo llegara a corromperse

Los sistemas de archivos modernos cumplen con estas funciones. A continuación se presentan una descripción de los Sistemas de Archivos Distribuidos y Paralelos debido a que estos son lo que más se usan en HPC.

2.2.1. Sistemas de Ficheros Distribuidos

Los Sistemas de archivos distribuidos estan diseñados para permitir a los procesos de multiples computadores acceder a un conjunto común de archivos. Aunque los sistemas de archivos distribuidos tienen algunas caraterísticas comunes con los sistemas de archivos paralelos, estos no son una solución completa para la E/S paralela. No están diseñados para permitir a multiples procesos acceder en forma concurrente y eficiente al mismo archivo.

El sistema de archivo distribuido más conocido es NFS (Network File System), entregado por Sun MicroSystems en 1985. NFS permite a un computador compartir una colección de sus archivos con otros computadores en la red. El computador donde residen los archivos se denomina servidor, y el computador que remotamente accede a estos archivos recibe el nombre de cliente. En la actualida existen cluster de computadores pequeños (menor a 125 nodos) que usan NFS como sistema de archivo.

2.2.2. Sistemas de Ficheros Paralelos

Los sitemas de Ficheros Paralelos deben tratar con varias cuestiones importantes:

- ¿Cuántos procesos (cientos o miles) pueden acceder concurrentemente al mismo archivo?
- ¿Cómo podrían trabajar los punteros de archivo?
- ¿Puede la semántica de consistencia secuencial de UNIX ser conservada?
- ¿Cómo podrían los bloques ser colocados en la cache y buffer?

Aunque los sistemas de ficheros actuales están bastante avanzados, muchas aplicaciones paralelas usan una de dos tipos E/S: acceso secuencial puro a un archivo, en el cual los programas envían todos sus accesos a archivos a través de una simple tarea, y múltiples accesos a archivos, donde cada tarea accede a su propio archivo.

El acceso secuencial puro trabaja bien en computadores de memoria compartida. Un simple *thread* puede copiar datos entre la memoria principal y uno o más canales de E/S. El sistema de archivo puede automáticamente ordenar los *strips* de archivo sobre múltiples dispositivos de almacenamiento, y con un hardware apropiado de acceso directo a memoria, los datos se pueden mover en paralelo entre las diferentes regiones de la memoria y el almacenamiento. Sin embargo, el acceso secuencial puro es menos probable en los computadores de memoria distribuida. Los datos que van a ser escritos a un archivo, desde cada proceso, deben moverse en la memoria de los procesos de E/S antes de ir a los dispositivos de almacenamiento.

El acceso múltiple a archivos es una alternativa para enviar todos lo datos a un proceso. Para esta técnica, usada principalmente en programas de paso de mensajes, cada proceso escribe datos en un archivo separado. Si el archivo reside en el disco del nodo local, el acceso al archivo desde el propio nodo será muy rápido debido a que los datos no necesitan viajar a través de la red de paso de mensaje. Si los archivos son temporales o si el usuario no necesita los datos para postprocesamiento, el acceso múltiple a un archivo es un buena elección. Sin embargo, algunas aplicaciones necesitan producir un simple conjunto de datos. El postprocesamiento requerido para reunir muchos archivos separados en un gran archivo puede fácilmente llevar a reducir el rendimiento de los múltiples acceso a archivos. En sistemas donde los discos locales no son accesibles por otros nodos, mezclar archivos requiere otros programas paralelos que se ejecuten en el mismo conjunto de nodos que el programa que ha generado los datos.

Los sistemas de archivos paralelos tratan de dar solución a los problemas del acceso secuencial y acceso múltiple. Ellos combinan las altas prestaciones y escalabilidad de los accesos multiples

con el beneficio de coleccionar los datos en un archivo simple. Para hacer esto, ellos deben permitir a múltiples tareas acceder al archivo al mismo tiempo, aunque no todas las tareas accederán a las misma ubicaciones de un archivo dado. Entre los sistemas de archivos paralelos más usado en HPC se destacan: Lustre creado por Sun MicroSystems pero que en la actualidad es propiedad de Oracle, GPFS de IBM y PVFS que es un sistema de paralelo libre creado por comunidad de investigadores en E/S Paralela.

2.3. E/S Paralela de Aplicaciones Científicas

Tres categorías de aplicaciones tienen una gran demanda de E/S, los Sistemas de Gestión de Bases de Datos (DBMs), las Aplicaciones Multimedia y las Aplicaciones Científicas que se usan principalmente para simulación [1]. A continuación se da un breve descripción de cada una de ellas en la sección 2.3.1 se presenta con más detalle las aplicaciones científicas que son el objeto de estudio de este trabajo.

Sistemas de Getión de Base de Datos

Los DBMs gestionan colecciones de registros de datos, usualmente almacenados en disco. La colección puede ser bastante grande, conteniendo millones de registros o más, y las DBMs deben buscar frecuentemente registros individuales que reúnan ciertos criterios. Dependiendo de la operación que se esté realizando, un DBMs puede examinar cada registro en la base de datos o puede ver un conjunto de registros divididos en la base de datos. Las escrituras o lecturas en pequeñas *strided* (aproximadamente, menos de 1000 bytes) llamadas acceso de grado fino, y este puede ser menos eficiente que el acceso a grandes *strided*.

Aplicaciones Multimedia

Las Aplicaciones Multimedia pueden procesar sonido, imágenes y datos de video. A diferencia de un DBMs, una aplicación multimedia pueden acceder frecuentemente a grandes bloques de datos en un secuencia predecible. Por ejemplo, si una aplicación está presentando un video, este puede determinar bastante bien que datos serán los próximos en leerse, así que se puede planificar los accesos al disco en una secuencia eficiente. El usuario puede buscar hacia delante y hacia atrás a través del vídeo o tomar diferentes ramas a través de una historia interactiva, pero incluso entonces el programa por lo general puede acceder a los datos en grandes bloques. Sin embargo, a diferencia de muchas otras aplicaciones, los programas multimedia a menudo requieren el sistema E/S para leer los datos en más de una tasa mínima determinada, esta tasa debe ser sostenida para que el sonido y las imágenes se muevan a la misma velocidad.

Aplicaciones Científicas

En aplicaciones científicas, la granularidad puede ser fina o gruesa, y los patrones de acceso pueden ser predecibles o aleatorios. Muchas aplicaciones científicas leen y escriben datos en fases bien definidas: el programa leerá algún dato, realizará cómputo, escribirá datos. Usualmente, un programa continuará procesando en varios pasos, escribiendo datos cada vez que pasan un número específico de pasos. En una aplicación paralela, las tareas individuales en un trabajo escribirán sus datos al mismo tiempo. Las librerías de E/S paralelas pueden aprovechar este sincronismo para mejorar el rendimiento.

Una importante diferencia entre las aplicaciones científicas, las multimedias y la DBMs es que las dos últimas son diseñadas especificamente para hacer E/S. Los diseñadores de estas aplicaciones reconocen que el almacenamiento externo es esencial para la prestación de la aplicación. En Aplicaciones Científicas, por otro lado, la tarea central es el cómputo, se asume que los datos están en la memoria. El movimiento de los datos entre la memoria y el almacenamiento es un problema secundario. Otra diferencia es que las bases de datos y aplicaciones multimedia se centran más en la lecturas mientras que las aplicaciones científicas en la escritura.

Las aplicaciones científicas de áreas como estudios del clima, fusión y dinámica molecular usan E/S para varios propósitos, tales como la obtención inicial de condiciones y parámetros de ejecución, como una forma persistente de almacenamiento de las salidas del programa, y como resguardo contra fallas del sistema. Este último propósito está teniendo un crecimiento importante: el deseo de alcanzar una capacidad computacional marcada en los sistemas de cómputo de altas prestaciones ha producido una tendencia a sistemas con un número incremental de componentes, y la confiabilidad de estos sistemas disminuye a medida que el número de componentes aumenta.

2.3.1. Categorías de Aplicaciones Científicas

En un esfuerzo por clasificar las aplicaciones científicas según su patrón de E/S, Miller and Kantz [5] las dividen en tres categorías: E/S requeridas, checkpoint y staging data.

E/S Requeridas

Las E/S es inherente en la aplicaciones y estas hacen[6]:

- *Input:* La mayoria de las aplicaciones necesitan datos de entrada para poder iniciar el cómputo. Las entradas de datos varian desde pequeños archivos conteniendo pocos parámetros importantes a grandes archivos de datos que inicializan matrices claves. Una aplicación

puede requerir leer datos al inicio, alternativamente puede leer datos en intervalos durante todo el cómputo.

- *Output:* Las salidas de la aplicaciones toman diversas formas, estas pueden ser desde pequeños archivos con pocos resultados o grandes archivos con varios matrices. La salida puede ocurrir al finalizar la aplicación o en intervalos durante el cómputo cuando estan involucrados datos dependientes del tiempo.

Checkpoint

Las aplicaciones que requieren tiempos de ejecución prolongados, usan la E/S para guardar su estado de cómputo para continuar el cómputo desde un punto determinado. El Checkpoint es una técnica que se usa para guardar el estado de un programa periódicamente para poder recuperarse de una falla de hardware o software. Si la aplicación se detiene antes de que finalice el cómputo, esta puede leer los datos del checkpoint y reiniciar el cómputo.

Staging de Datos

Las aplicaciones pueden necesitar grandes cantidades de datos que no caben en memoria principal para realizar su trabajo. Estos datos pueden se intercambiados a disco y son llamados *out-of-core*. Estos datos pueden ser colocados en memoria principal por partes y los resultados parciales son transferidos a disco. Hay máquinas que proporciona memoria virtual, por hardware o software, para tratar este tipo de estructuras aunque las prestaciones de estas aplicaciones se ve degradada.

2.3.2. Patrones de E/S en Aplicaciones Científicas

El estudio de las cargas de E/S paralela muestran que los patrones de E/S varían ampliamente entre aplicaciones. Muchas aplicaciones generan grandes secuencias de solicitudes que acceden a unas decenas o centenas de bytes. Incluso cuando una aplicación mueve datos en grandes bloques, la configuración de la E/S (tanto hardware, software, políticas de buffering y caching) pueden causar que el sistema entregue mucho más prestaciones para algunos patrones de acceso que para otros. Por ejemplo, las prestaciones mejoran en algunos sistemas de archivos cuando los nodos de cómputo hacen un correspondencia entre el tamaño y alineación de sus archivos al tamaño de *stripping*. Los desarrolladores pueden modificar sus aplicaciones para manejar de forma eficiente sus solicitudes de E/S, pero esto es un trabajo tedioso, y el código resultante solo es valido para una configuración específica de un Sistema Paralelo [1].

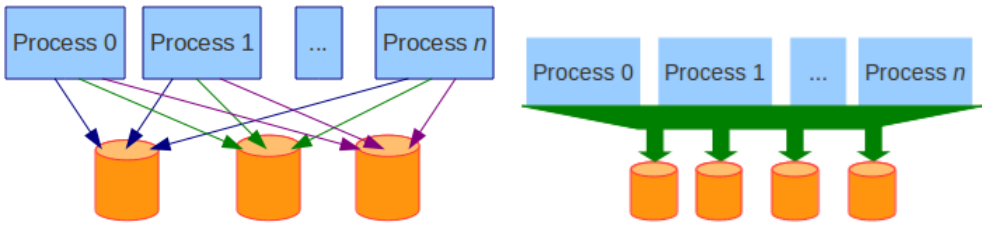
Los detalles de las estructuras de datos de la aplicación, por supuesto, pero los siguientes dos ejemplos representan los casos más comunes.

Un programa puede gestionar una matriz rectangular. Los elementos de la matriz podrían ser simples números o una colección de valores relacionados (tuplas), como por ejemplo temperatura, presión y velocidad de un fluido en un punto del espacio. Para el último caso algunas aplicaciones almacenan estos datos en memoria como arreglos separados y otros los disponen como un arreglo de tuplas. La unidad más pequeña que un programa accede en una operación de E/S se denomina registro. Para un arreglo multidimensional este registro puede ser un simple número o una tupla.

En memoria y en muchos sistemas de archivos, un arreglo multidimensional es almacenado como un arreglo unidimensional. Por lo tanto, dos elementos de arreglos que son lógicamente contiguas podrían no serlo en memoria o en archivo. Esto no sería un problema si las aplicaciones acceden al arreglo completo, ya que acceden a este en el orden que fue almacenado. Sin embargo, cuando se accede a una porción del arreglo, puede ser que se realicen escritura o lecturas a posiciones no contiguas de memoria o de archivo. Esta clase de acceso es común en aplicaciones out-of-core, ya que leen y escriben un subconjunto de grandes estructuras de datos. Sin embargo, a pesar de ser discontiguos, estos accesos siguen un patrón regular. Por ejemplo, un programa que lee una columna de un arreglo de dos dimensiones almacenado por filas realizará una serie de solicitudes de lectura en un *offset* de archivo que será regular. Una secuencia de accesos a intervalos fijos es llamado *strided access*; el stride es el número de elementos (o bytes) entre el inicio de dos elementos sucesivos.

Una estructura de datos más compleja es un grid irregular. Los grid regulares definen puntos en un espacio computacional en intervalos fijos. Los grid irregulares tienen puntos en posiciones arbitrarias. Un grid irregular puede ser representado como un conjunto de arreglos de nodos, aristas, superficies, volúmenes, etc. Los nodos adyacentes en un grid pueden ser almacenados en intervalos arbitrarios del arreglo, de forma que el acceso a un elemento del grid requiere un serie de accesos aleatorios en lugar de un acceso por *strided*.

Los grid regulares son particionados en patrones regulares, y cada proceso procesa un subconjunto de tamaño fijo del grid global. Una técnica de distribución común es particionar cada dimensión del arreglo rectangular en una de dos formas: como un conjunto de bloques de igual tamaño (*block distribution*) o como serie repetitiva de *strided* (*cyclic distribution*). Cada tira en un distribución cíclica puede ser uno o varios elementos. Dependiendo de la distribución, un programa que lee o escribe un arreglo completo puede producir un serie de accesos no contiguos en cada proceso. A pesar que la aplicación como unidad esta accediendo al arreglo completo, los procesos



(a) Entrada/Salida independientes (b) Entrada/Salida colectivas

Figura 2.15: Patrones globales

deben saltar a través del archivo para acceder a filas o columnas individuales.

Los grid irregulares tienen distribuciones más complejas, y éstas generalmente cambian en tiempo de ejecución. La descripción de como una parte del grid global se adapta a un procesador se denomina *local-to-global mapping*. Esta mapping especifica una ubicación en el grid global para cada punto en el grid local de los procesos. Cualquier acceso a una estructura compartida globalmente en un archivo usará este mapping.

Los esfuerzos por mejorar las prestaciones de la E/S debe emperzar por entender los patrones de E/S. Muchos archivos son pequeños y temporales, que son creados y borrados en un corto tiempo. Además en muchos archivos son más frecuentes las lecturas que las escrituras. Muchas lecturas son secuenciales y recorren todo el archivo. Para estos archivos el caching y buffering trabajan bien. Desafortunadamente, las aplicaciones científicas de HPC no son programas UNIX típicos, y los patrones de E/S de computadores de memoria compartida difieren de los de memoria distribuida []. Este comportamiento da la E/S en las aplicaciones de HPC trae como consecuencia que sea complicado realizar una taxonomía de los patrones de acceso. En este trabajo se optará por presentar una clasificación de los patrones de acuerdo a [7]. Los patrones de aplicaciones paralelas pueden ser divididas en patrones locales y patrones globales. Los patrones locales estan determinados por el proceso (o thread), mostrando como un archivo es accedido por un proceso local. Los patrones globales (figura 2.15) son sobre la aplicación paralela, representado como los multiples procesos acceden a un archivo. En [7] se hace una clasificación en cinco dimensiones para un proceso local. Las cinco dimensiones son espacial, tamaño de solicitud, comportamiento repetitivo, temporal y tipo de operación de E/S. A continuación se explica las 5 dimensiones:

- Patrón espacial: la secuencia en la que se accede a posiciones de un archivo se denomina patrón espacial de una aplicación. Ellos puede ser contiguos (figura 2.16(a)) o discontiguos (Fig. 2.16(b)) o una combinación de ambos (Fig. 2.16(c), 2.16(d)). Los accesos discontiguos hacen referencia a huecos en los accesos a un archivo. Estos huecos pueden ser de tamaño

fijo o variable. Los huecos variables pueden serguir un patrón de dos o más dimensiones. Otro patrón puede ser con *strides* decrecientes. Algunos accesos no tiene un patrón regular, los *strides* son aleatorios.

- Tamaño de la Solicitud: este puede ser pequeño (*small*), mediano (*medium*) y grande (*large*). El tamaño de una solicitud puede ser fija o variable. Una solicitud se considera pequeña cuando es solo una fracción de una página. Y será grande cuando el tamaño sea varias veces mayor al tamaño de una página. Dada la alta latencia de la E/S, las solicitudes pequeñas pueden causar cuellos de botellas en las prestaciones si se accede a los discos por una pequeña cantidad de bytes.

- Comportamiento Repetitivo: Una aplicación tiene un comportamiento repetitivo cuando los bucles o una función con bucles tiene operaciones de E/S. Cuando un patrón de accesos es repetitivo, el *caching* y el *prefetching* pueden efectivamente enmascarar la latencia de los accesos.

- Intervalo Temporal: los patrones temporales capturan las regularidad de las rafagas de E/S en una aplicación. Ellos pueden ocurrir periódicamente como irregularmente.

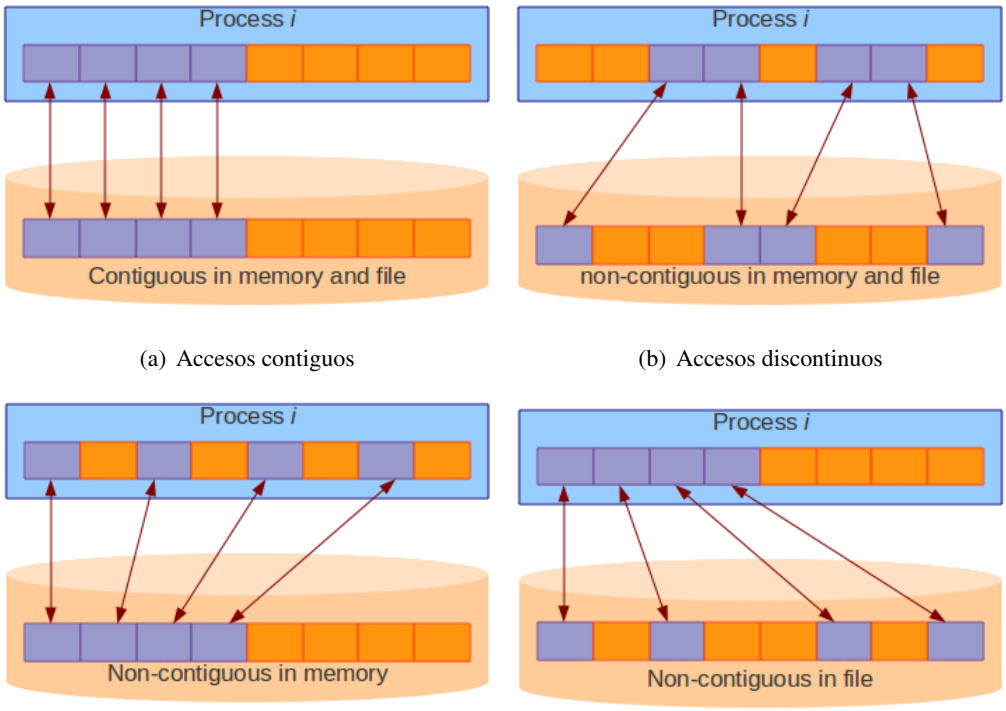
- Operación de E/S: las operaciones de E/S pueden ser de *write only*, *read only* y *read/write*.

2.3.3. Benchmarks de Aplicaciones Científicas Paralelas

Para evaluar las prestaciones de las aplicaciones intensivas de E/S se han desarrollado benchmarks de E/S que miden las prestaciones de acuerdo a los patrones de E/S. Si bien no existe una suite de Benchamrk de E/S como ocurre para la evaluación del cómputo. En esta sección se presentan los Benchamrk de E/S que corresponden a Kernels de aplicaciónes que son intensivas de E/S y que son los más usados para evaluar las prestaciones de la E/S paralela en diferentes configuraciones de E/S.

NAS BT-IO

El NAS BT-IO [8], una extensión del benchmark NAS BT, simula los requisitos de E/S de BT. BTIO presenta un patrón de particionado tridiagonal en una arreglo de tres dimensiones en un número cuadrado de procesos. Cada proceso es responsable de multiplicar un subconjunto cartesiano de datos enteros, cuyo número se incrementa como la raíz cuadrada de los procesos participantes en el cómputo. En BTIO, 40 arreglos son escritos consecutivamente a un archivo



(a) Accesos contiguos

(b) Accesos discontinuos

(c) Accesos discontinuos en memoria

(d) Accesos discontinuos en archivo

Figura 2.16: Patrón espacial de una aplicación

compartido agregando uno después de otro. Cada arreglo debe ser escrito en un formato canónico y por filas en el archivo. Los cuarentas arreglos son leídos nuevamente para verificación usando el mismo particionado de datos. El tamaño del archivo es fijo, sin tener en cuenta el número de procesos. Por lo tanto la cantidad de E/S se irá decrementando para cada proceso a medida que se aumenten los procesos. El patrón de acceso al archivo de cada proceso es discontiguo con *strides* variables. El patrón de acceso del NAS BT-IO es típico de las aplicaciones científicas, por esta razón es uno de los benchmarks más usado [9, 10, 11] en la evaluación de las prestaciones de la E/S.

MadBench

MADBench2 es un derivado del código de análisis de datos MADspec. El MADspec estima el espectro de energía regular de la microondas de la radiación cósmica en el cielo de un conjunto de datos pixelados. Como parte de sus cálculos, el MADspec realiza muchas operaciones sobre una matriz *out-of-core* , requeriendo sucesivas escrituras y lecturas de una gran cantidad de datos contiguos de archivos compartidos o individuales. Debido a grandes, contiguas y mezcladas operaciones de lectura y escritura que MADbench2 realiza y su capacidad para probar una var-

iedad de parámetros, éste es elegido como benchmark en la comunidad de la E/S paralela [10, 12]. MADBench2 realiza cuatro pasos

- construye recursivamente una secuencia del polinomio de *Legendre* basado en la matriz componente de la correlación pixel-pixel y la señal CMB (*Cosmic Microwave Background*), escribiendo cada uno a disco.
- Forma e invierte la matriz de componentes señalCMB+ruido (cálculo y comunicación)
- A su vez, lee cada componente de la matriz de correlación señal CMB del disco, multiplicando este por la matriz de correlación de datos CMB inversa, y escribe la matriz resultante a disco.
- A la vez, lee cada par de resultados de la matriz desde disco y calcula el signo de su producto.

Debido a la gran cantidad de memoria del dominio computacional en los cálculos reales, todas las matrices requeridas generalmente no entran en memoria simultáneamente. De esta forma, un algoritmo *out-of-core* es utilizado, el cual requiere suficiente memoria como para almacenar solo 5 matrices en cualquier momento.

FLASH IO

El suit benchmark FLASH IO es un kernel de la aplicación FLASH, un código de hidrodinámica de una maya adaptiva de bloque estructurado que resuelve ecuaciones de hidrodinámica reactiva, desarrollado principalmente para estudio de los flashes de los neutrones de las estrellas y las enanas blancas. El dominio computacional es dividido en bloques en un número de procesos. Un bloque es un arreglo de tres dimensiones con cuatro elementos adicionales que hacen de celdas protectoras en cada dimensión para almacenar información de sus vecinos. FLASH IO produce un archivo de checkpoint y dos archivos conteniendo datos centrales y de las esquinas. El archivo de mayor tamaños es el de checkpoint, el tiempo de E/S que domina el benchamrks. FLASH IO usa una librería de alto nivel de E/S (HDF5) para almacenar los datos con sus metadatos.

2.4. Técnicas de E/S Paralela

Existen técnicas permiten mejorar las prestaciones de la E/S en aplicaciones paralelas y simplifican el acceso a archivos paralelos. Algunas técnicas están implementadas en los sistemas de archivos y otras usan una capa intermedia de software que reside entre la aplicación y el sistema de ficheros. Estas reciben el nombre de software de E/S. Las técnicas se clasifican en tres categorías:

2.4.1. Métodos de Accesos Discontiguos

Las Aplicaciones de memoria distribuidas frecuentemente comparten un archivo entre los procesos, cada proceso puede necesitar realizar una serie de pequeñas solicitudes para leer o escribir datos. Los accesos pequeños son ineficientes por si mismos, combinarlos en una simple solicitud puede mejorar las prestaciones. Los Accesos Discontiguos mezclan solicitudes de trozos separados en una simple operación. Una ventaja de combinar pequeñas solicitudes es que que el software de E/S puede enviar un grupo pequeño de solicitudes o items de datos entre el nodo de cómputo y el de E/S como un simple mensaje, reduciendo el costo de comunicación. Minimizar la cantidad de pequeñas solicitudes es importante en sistemas que no usan buffering en los clientes. Ya que los sistemas *client-buffered* mueven bloques de datos completos entre los nodos de cómputo y los nodos de E/S sin tener en cuenta el tamaño de la solicitud, agrupando las solicitudes cercanas para datos discontiguos para evitar mensajes separados.
Una segunda ventaja es que se puede mejorar la planificación de acceso a disco. Por ejemplo, si un nodo de cómputo lee una serie de registros muy distantes entre si en una archivo, el software de E/S puede no ser capaz de *prefetch* los datos efectivamente si recibe las solicitudes en forma independiente. Los registros pueden estar separados por bloques que no contienen los datos solicitados, así la prebusqueda de los bloques de datos sería un desperdicio. Sin embargo, si el software de E/S recibe una solicitud que describe el patrón de acceso completo, este solo prebuscará aquellos bloques que contienen datos útiles.

La interfas de E/S puede soportar los accesos no contiguos de dos formas [1].

- Descripción Algorítmica: La descripción algorítmica trabaja bien para accesos que siguen un patrón regular. Un descripción algorítmica incluye en cada solicitud de E/S un conjunto de parámetros que describen los *chunks* individuales de datos. Una alternativa para especificar los patrones de acceso en cada solicitud de E/S es una partición lógica del archivo para cada proceso. Esto define un template a través del cual cada proceso accede a los datos en el archivo. Todos los procesos pueden usar la misma plantilla, o pueden usar plantillas intercaladas. La ventajas que tienen las plantillas es que los desarrolladores solo deben preocuparse del *local-to-global mapping* cuando abren el archivo y no en todas las operaciones de E/S. Sin embargo,las plantillas definen estructuras que son discontiguas solo en el archivo no en memoria.

- Lista de E/S: Una solicitud de E/S basada en una lista de E/S incluye una serie de descriptores de definen accesos individuales. Por ejemplo, la interfaz de lista de E/S de UNIX usa

una serie de estructuras C que incluyen los siguientes campos.

```
struct aiocb {
   int aio_filedes;    /* file descriptor */
   void *aio_buf;      /* memory buffer*/
   size_t aio_nbytes; /* size of chunk */
   off_t aio_offset;  /* file location */
   /*...otros campos */
 };
```

Como se muestra en esta definición, cada estructura lista un descriptor de archivo *target*, la ubicación en memoria de donde los datos serán leidos o escritos, la ubicación correpondiente en el archivo y el número de bytes a ser movidos. Cada una de estas estructuras describe una simple transferencia de datos contiguos.

Las aplicaciones pasan a la función de Lista E/S un arreglo de punteros para estas estructuras, uno por cada elemento en la transferencia. Una interfaz basada en lista se puede usar tanto en patrones regulares como en irregulares, pero una interfez algorítmica puede describir una serie de grandes accesos de forma más compacta, el software de E/S puede comunicar descripciones algorítmicas entre los nodos de cómputo y los nodos de E/S más eficientemente que las listas. Por lo tanto, las interfaces algorítmicas son mejor opción que la listas de E/S para transferencias regulares.

2.4.2. Colectivas de E/S

Las Colectivas de E/S, como los métodos de accesos discontiguos, es una clase de optimización que mejora las prestaciones mezclando varias solicitudes de E/S. Sin embargo, a diferencia del acceso discontiguo, las colectivas de E/S mezclan solicitudes de múltiples procesos. En muchos casos, las colectivas de E/S fuerzan la localidad temporal de manera que el software de E/S puede explicitameante coordinar las solicitudes.

Las operaciones de escritura colectivas reúnen datos de múltiples procesos en un *chunk* grande y contiguo antes de almancenarlo en disco. Las operaciones de lectura colectivas recuperan *chunk* grandes de datos y los distribuyen a los múltiples procesos solicitantes. Esto reduce el número de accesos a discos y hace cada acceso más eficiente. La *false-sharing* y la secuancialidad de los accesos son eliminados. Aunque en principio las colectivas de E/S no necesitan sincronizar las tareas participantes, en la práctica muchas operaciones colectivas tienen el efecto de barrera.

Las colectivas de E/S se pueden dar a:

Cuadro 2.3: Método Disk-Directed

Nodo de Cómputo (NC)	**Nodo de E/S (NIO)**
Lectura Colectiva de NC (archivo,parámetros de lectura, dirección destino): ya que los datos llegan después en un mensaje asíncrono se registra la dirección destino para uso del gestor de mensajes	Lectura Colectiva de NIO (archivo, parámetros de lectura): determina los archivos de datos locales para este NIO, determina el conjunto de bloque de discos necesarios
Barrera (NC usando estos archivos), para asegurar que todos los buffers están listos para cualquier NC: multicast (Lectura Colectiva, archivo, parámetros de lectura) para todos los NIOs, esperan para todos los NIOs para responder que hacen	Ordena los bloques de disco para optimizar los movimientos del disco. Usa un double-buffering para cada disco, solicita bloques desde los discos a medida que cada bloque arriba desde el disco, envía las pieza(s) al NC apropiado
Barrera (NCs usando este archivo), esperando que todas las E/S se completen	Cuando completa, envía el mensaje al NC que hizo la solicitud

Nivel de Disco (*disc-directed E/S*) [13]

Las solicitudes de E/S son pasadas a los nodos de E/S, los cuales ordenan la transferencia de los datos como se muestra en la tabla 2.3. Esta técnica pone los discos bajo el control del orden y el tiempo del flujo de los datos. Es decir que los accesos para lectura y escritura se ajustan al diseño físico de los datos en el disco.

Nivel de Usuario (*client-based E/S*)

Este tipo de colectivas usa la red de paso de mensaje del nodo de cómputo para reordenar los datos de manera que se adapte el diseño de acuerdo a las necesidades de la aplicación y el diseño para mejorar las prestaciones de E/S. El intercambio de datos recibe le nombre de *shuffle*. Las escrituras de los procesos primero se mezclaran para formar grandes bloques y entonces enviar los bloques a los nodos de E/S (figura 2.18). Para las operaciones de lectura (figura **??**), primero determinaran el rango total entre todas las solicitudes que deben ser accedidas. Después cada proceso leerá una subconjunto de datos de un gran bloque. Cuando los datos llegan, los procesos tendrán datos que otros procesos han solicitado, así que ellos se encargarán de enviar los datos al proceso que lo solicito [1].

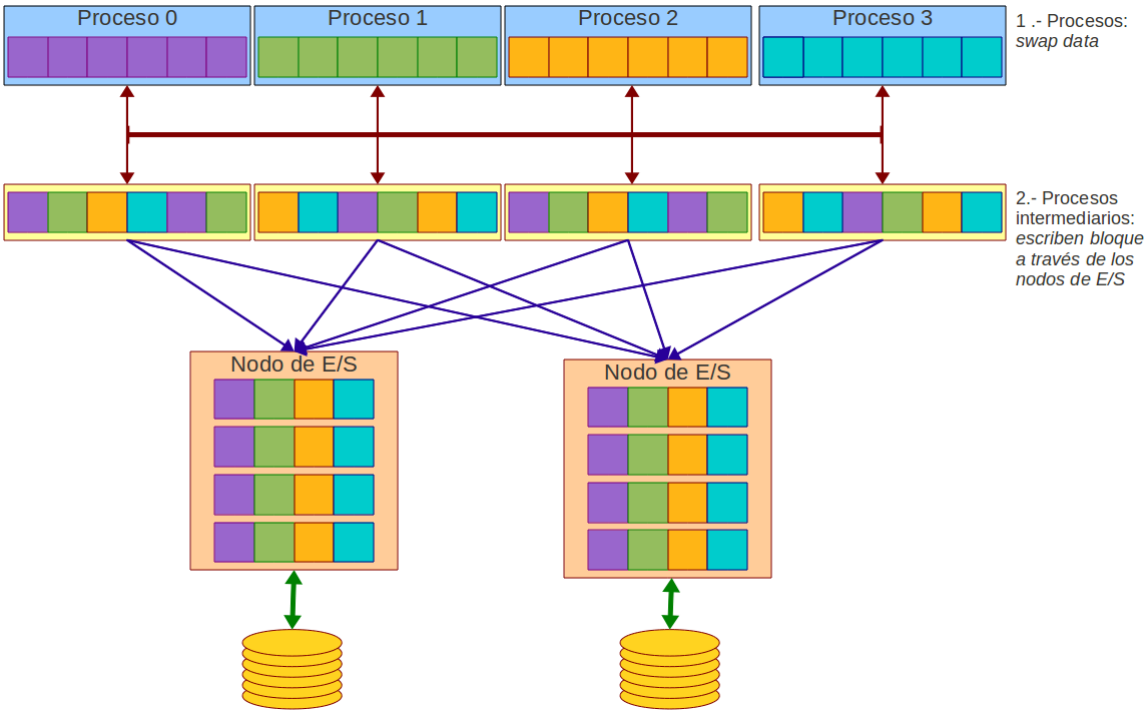


Figura 2.17: Operación de escritura para colectivas de E/S basada en el cliente

Hay varias implementaciones de esta técnica. Una del más usadas es la técnica de *Two-phase* [14, 15]. Esta consta de una fase de E/S y otra de intercambio de datos. En la fase de E/S, los agregadores de E/S designados de un grupo de comunicadores colectivos son exclusivamente responsables por los accesos a las posiciones que no se superponen del archivo, y realizan la E/S en nombre de todos los procesos. La determinación de la asignación del *file realm* se la deja para hacerlo en la implementación o incluso se la puede delegar al usuario. Para mayor claridad, los *client* hace referencia a los procesos que realizan son fuente de las solicitudes de E/S y los *aggregators* hacen referencia a los procesos que tienen la capacidad de reunir las solicitudes de E/S. Todos los procesos que hacen la llamada colectiva de E/S son *client* pero no todos son necesariamente *aggregators* de E/S.

En la fase de comunicación se mueven los datos de los archivos desde y hacia los procesos apropiados. Si la llamada de E/S es de lectura o escritura determina el orden de las fases. En el caso de la lectura, los datos son leídos primero por los agregadores y después transferidos a los procesos solicitantes. En el caso de la escritura, los datos primero son enviados a los agregadores y escritos en el archivo por los agregadores. Una combinación de la solicitudes de E/S en los agregadores puede mejorar la eficiencia de la E/S.

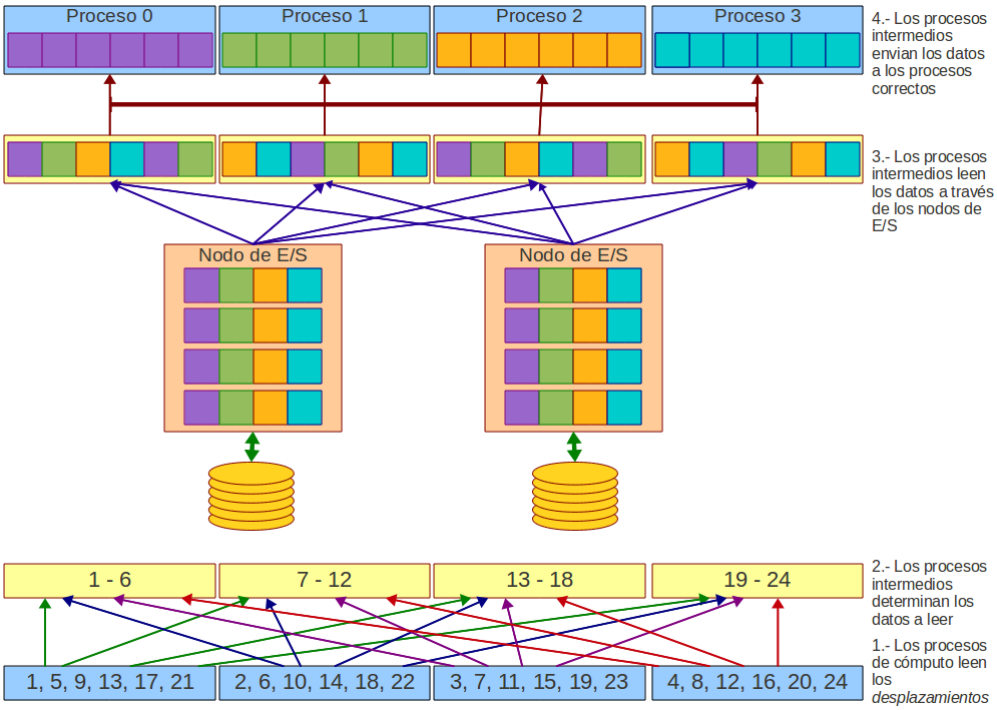


Figura 2.18: Operación de lectura para colectivas de E/S basada en el cliente

Las Colectivas *client-based E/S* tienen mejores prestaciones sobre el acceso directo a los archivos paralelos, pero tienen varios detalles de fondo: requieren espacio de buffer adicional en los procesos intermedios para la operación de mezcla (*suffle*), mueve los datos dos veces sobre la red (una por cada fase), y requiere una buena distribución de datos intermedias para lograr un rendimiento óptimo.

Nivel de Server (*server-based E/S*) [16]

Las *server-based E/S* intentan resolver los problemas de las colectivas basada en el cliente. Al igual que la *client-based E/S* , las *server-based E/S* reúnen y mezclan pequeñas solicitudes desde múltiples procesos de cómputo como una fase separada de la E/S, y reune las pequeñas solicitudes directamente en los nodos de E/S. Para una operación de escritura (figura 2.19), los nodos de cómputo envián una descripción de transferencia a todos los nodos de E/S. Sin embargo, antes de que ellos hagan esto, cada nodo se prepara para recibir solicitudes de los nodos de E/S para las porciones de datos actuales. Dependiendo de la implementación, la preparación puede involucrar *setting up* de un socket o el *posting* de solicitudes no bloqueantes para recibir mensajes. Una vez que los nodos de E/S han recibido una descripción de la transferencia de todos los nodos de cómputo, este determina que bloques de archivo bajo su control debe escribir. Entonces determina que nodo de cómputo almacena el dato que llenará cada bloque. Con frecuencia, varios nodos de

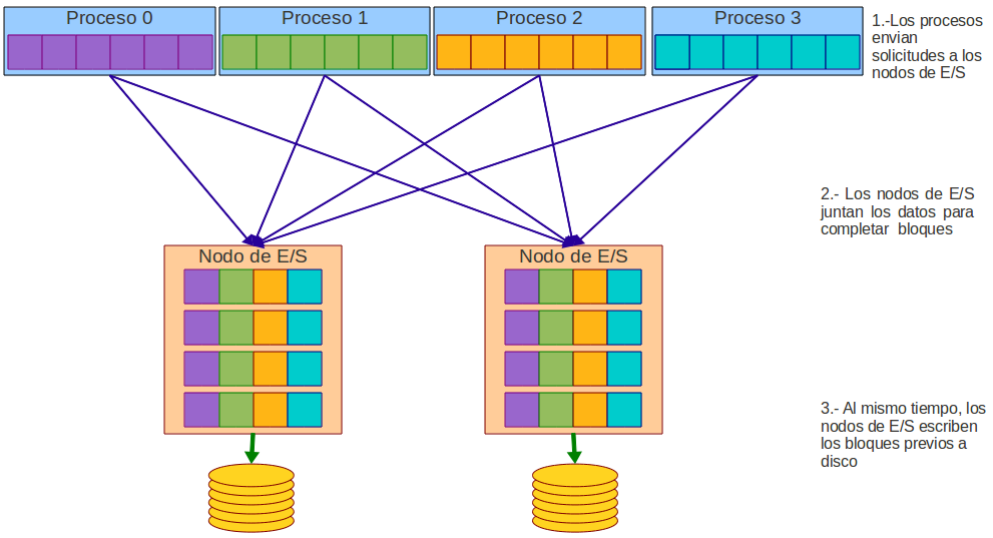


Figura 2.19: Operación de escritura para las colestivas de E/S basada en el servidor)

cómputo almacenan datos para el mismo bloque. Por cada bloque, el nodo de E/S solicita los datos al nodo de cómputo (probablemente en paralelo) y llena los bloques a medida que llegan. Una vez que los bloques estan completos, un nodo de E/S puede escribir este dato al disco y continuar trabajando con el siguiente bloque. Para ganar tiempo, un nodo de E/S puede completar varios bloques a la vez, o puede usar un doble buffer para ir escribiendo un bloque mientras completa el siguiente bloque. Sin embargo, el espacio total de buffer en cada nodo de E/S es pequeño debido a que los nodos de E/S solo transfieren de los nodos de cómputos los datos que son necesarios. Los nodos de cómputo no requieren un sistema de buffer total. Sin embargo, ellos deben estar preparados para transferir datos de cualquier tamaño que solicitan los nodos de E/S y para recibir solicitudes desde múltiples nodos de E/S concurrentemente.

Una operación de lectura se inicia con los nodos de cómputo preparándose para conectarse con los nodos de E/S. Entonces, como en una solicitud de escritura los nodos de cómputo envian sus solicitudes a los nodos de E/S. Los nodos de E/S determinan que bloques de los archivos leer, una vez que los datos llegan desde los discos, éste los distribuye a los nodos de cómputo que esperan. Como en la escritura, se puede usar doble buffer y los requisitos son los mismos. Para la gestión del reordenamiento de los datos desde el diseño del proceso al diseño de archivo en los nodos de E/S, las *server-based E/S* eliminan la necesidad de buffer extra en los nodos de cómputo, y

requiere un tamaño relativamente pequeño de buffer en los nodos de cómputo [1].

Medidas de prestaciones de las Colectivas de E/S

Debido a que las prestaciones de las Colectivas de E/S es la característica medible más importante a continuación se presenta como se puede medir las prestaciones en estos sistemas. Las operaciones colectivas implican varios procesos de cómputo, hay varias formas de definir su duración:

- Primero-En-Iniciar-a-Último-En-Terminar (*Earliest start to latest finish*)

  El tiempo transcurrido es el intervalo desde el inicio del primer proceso hasta que el último proceso haya terminado. Esta medida requiere un reloj global sincronizado para asegurarse que los tiempos en los procesos son comparables. Esta medida muestra el efecto de la operación colectiva en la aplicación. Si la sincronización de las operaciones colectivas es fundamental para una aplicación entonces esta es la medida adecuada.

- Mayor tiempo de los procesos (*Longest elapsed time of any process*)

  En este caso, como en el caso anterior, también se exponen efectos de sincronización pero en este caso no necesita un reloj global.

- Tiempo Medio de Finalización (*Average elapsed time*) Esta medida produce una aparente tasa de transferencia alta y da una vista más exacta de la cantidad de recursos de cómputo que una operación usa, pero oculta los efectos de sincronización y desbalance de carga en el resto del programa.

2.4.3. Métodos Adaptativos

Un software de E/S debe trabajar bien en diferentes situaciones, y debe adaptarse a las circunstancias cambiantes; una simple configuración de un sistema de E/S no trabaja bien en todas las situaciones. Por lo tanto, algunos software de E/S reúnen información de la aplicación que le ayuda a reconocer patrones de accesos. Un *hint* es información que una aplicación pasa explicitamente al software de E/S para mejorar las prestaciones. Algunos software de E/S pueden disernir por si mismo los patrones de acceso y adaptarlo automáticamente. Los *hints* cubren un amplio rango de técnicas para la configuración del sistema de E/S. Algunos *hints* imponen requisitos de semántica a los programas: si un programa realiza una solicitud que contradice los *hints* iniciales, la solicitud operará incorrectamente. Por ejemplo, si una aplicación asegura que nunca escribirá regiones de archivos que se suporponen desde diferentes procesos, y luego procede a hacer esto, el contenido

del archivo resultante puede ser ilegible. Otros *hints* afectan únicamente las prestaciones y no los resultados. Si una aplicación no puede leer consecutivamente, las prestaciones se veran afectadas ya que el sistema de E/S desperdiciará tiempo leyendo bloques innecesarios, pero la aplicación recibirá los datos correctos.

Los *hints* pueden darse a alto nivel, describiendo los patrones de E/S de la aplicación y utilizando éste por el software de E/S para seleccionar las optimizaciones adecuadas, o pueden ser a bajo nivel, directamente controlando un aspecto específico de la configuración del sistema. Los programadores deben elegir el *hints* de bajo nivel basado en comprender como los patrones de acceso se utilizarán en el sistema de archivo y configuración determinados.

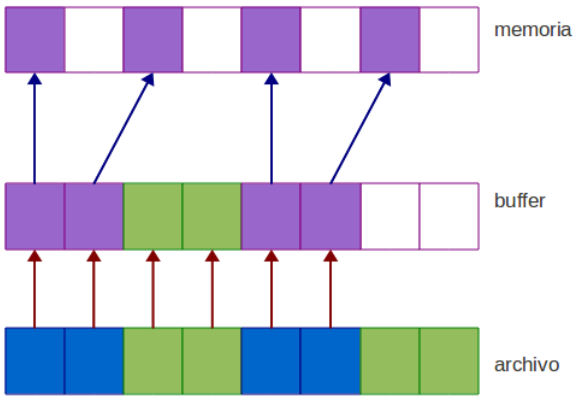
2.5. Librerías de E/S

Las librerías de E/S son interfaces que se proporcionan, a nivel de software, que permiten realizar las operaciones de E/S de forma más eficiente. Existen dos tipos de Librerías de E/S las de bajo nivel y las de alto nivel. A continuación se presentan las más usadas.

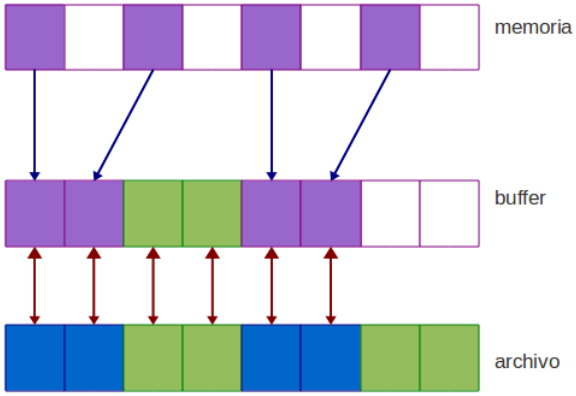
2.5.1. Bajo Nivel

Los accesos discontiguos, las colectivas de E/S y los *hints* necesitan interfaces de programación especializadas (APIs). El estandar UNIX E/S incluye una interface orientada a listas que puede acceder a regiones discontiguas de archivos y memoria, pero tiene inconvenientes para accesos *strided* ya que el programa debe listar cada de elemento discontiguo que debe transferir en forma separada. La interfaz de control de archivo de UNIX podría ser extendida para utilizarse por los *hints*, pero por ahora no lo soporta. Incluso las interfaces de E/S de FORTRAN carecen de esta capacidad. Ni UNIX I/O o FORTRAN I/O pueden indentificar un grupo de procesos participantes en una operación colectiva. Las librerías de E/S de bajo nivel soportan los accesos a archivos paralelos e implementan las diferentes técnicas de E/S. Estas interfaces trabajan en diferentes niveles de abstracción, son interfaces de bajo nivel en el sentido que no necesitan ninguna información acerca de los datos que se van a almacenar, y la aplicación debe seguir la pista de los datos que residen en el archivo.

Uno de las interfaces de E/S más usados y un estándar en E/S Paralela es MPI-IO.



(a) operación de lectura



(b) operación de escritura

Figura 2.20: Operaciones en *data sieving*

MPI-IO

MPI-IO es un extensión del estandar MPI-2, es una interfaz diseñada especificamente para E/S de altas prestaciones y portable. Permite al usuario especificar patrones de accesos discontiguos y leer o escribir todos los datos con una simple llamada de función de E/S. También permite al usuario especificar colectivamente las solicitudes de E/S de un grupo de procesos, proporcionando mayor información de acceso y alcance de optimización. Una de las implementaciones más usada de MPI-IO es ROMIO [17].

ROMIO realiza *data sieving* (figura 2.20) para solicitudes discontiguas de un proceso y *collective E/S* para solitudes discontiguas desde multiples procesos. Para reducir el efecto de la alta latencia de la E/S, es crítico hacer tan pocas solicitudes al sistema de fichero como sea posible. Cuando un proceso hace solicitudes independientes para datos discontiguos, ROMIO, para ello, no accede de forma separada a cada porción contigua del archivo. En lugar de eso este usa una técnica denominada *data sieving* [18]. Esta es una forma de combinar varias pequeñas solicitudes

de E/S en pocas solicitudes de gran tamaño para reducir el efecto del alto tiempo de latencia de la E/S. La principal ventaja de este método es que requiere muy pocas llamadas de E/S; muchas de la veces esta esperando transferir datos desde la memoria principal. La desventaja es que requiere memoria extra y lee más datos de disco de los que realmente necesita. *Data sieving* es similar a realizar *caching* en el nodo de cómputo. El procesador efectivamente tiene en cache un *chunk* de dato completo, iniciando desde el primer elemento en la sección, en el nodo de cómputo, y todos los elementos requeridos son proporcionados desde esta cache. Para escribir secciones *strided* usando *data sieving* primero se lee una gran *chunk* de dato desde el archivo en un buffer temporal, entonces almacena la sección *strided* en el buffer, y finalmente escribe el buffer en el archivo. Si no se lee del buffer primero, los datos en el buffer entre los elementos *strided* sobreescribirán los datos correspondientes en el archivo. En consecuencia, la escritura de las secciones *strided* requiere dos veces la cantidad de E/S necesaria para leer secciones *strided*.

ROMIO usa parámetros que el usuario puede controlar que definen la cantidad máxima de datos contiguos que un proceso puede leer durante el tiempo de *data sieving*. Este valor representa el tamaño máximo del buffer temporal. El tamaño por defecto es 4MBytes por proceso, pero el usuario puede cambiarlo en tiempo de ejecución por medio de mecanismos de *hints* de MPI-IO. ROMIO también le permite al usuario definir parámetros para realizar la escritura, que definen la cantidad máxima de datos contiguos que el proceso puede escribir durante el tiempo que dura el *data sieving*. Debido a que las escrituras requieren bloquear las porciones del archivo que estan siento accedidas, ROMIO usa por defecto el menor tamaño de buffer (512 Kbytes) para reducir la contensión debido a los bloqueos.

ROMIO usa el método de *two-phases* (figura 2.21) que puede manejar cualquier solicitud de E/S discontigua, tanto las secciones de arreglos como los tipos de datos derivados. ROMIO usa dos parámetros para las colectivas de E/S que puede controlar: el número de procesos que pueden realizar la fase de E/S y el tamaño máximo en cada proceso del tamaño del buffer temporal necesario para la E/S de *two-phases*. Por defecto, todos los procesos realizan E/S en la fase de E/S, y el tamaño máximo del buffer es de 4MBytes por proceso. El usuario puede cambiar estos valores por medio de los mecanismos de *hints* que ofrece MPI-IO.

2.5.2. Alto Nivel

Las librerías alto nivel también conocidas como librerías de datos científicos gestionan los datos a un nivel más alto. Los programas pueden manipular estructuras de datos complejas di-

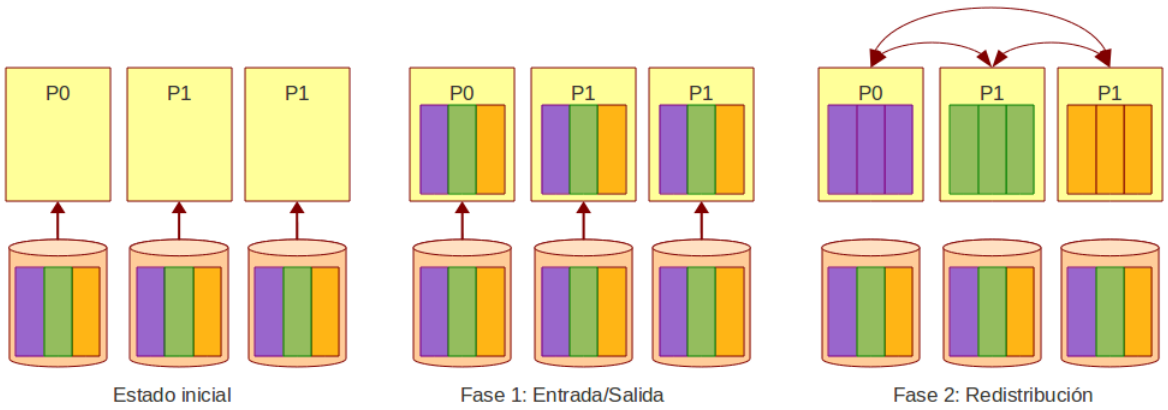


Figura 2.21: Operación de lectura para *two-phase*

rectamente, y la librería registra automáticamente el tipo de información y otros metadatos útiles. Las librerías de datos científicos ofrecen otras características útiles, por ejemplo, las aplicaciones pueden consultar los archivos para determinar el tamaño y forma de las estructura de datos antes de leerlo en memoria. Esto permite que la aplicación pueda ubicar un espacio correcto de memoria para almacenar los datos. Las librerías científicas también almacenan la información en formato numérico de datos en un archivo, así que un archivo escrito en una máquina puede ser leído en otra (esta habilidad es un paso más allá de característica de conversión de datos de MPI-I/O, ya que las librerías de datos científicos no sólo hacen una conversión de datos, sino que además registran la representación usada en el archivo). Las librerías científicas permiten a las aplicaciones leer y escribir archivos a través de los nombres en vez de la ubicación de los archivos. Una aplicación puede abrir un archivo, buscar los nombres de las estructuras de datos almacenadas en el archivo, seleccionar la estructura de interés, y recuperarlo, sin que conozca como el dato está definido en el archivo.

Parallel netCDF (PnetCDF)

Parallel netCDF [19] es una extensión de la librería netCDF desarrollada en Unidata [20], éste proporciona para el almacenamiento multidimensional, un conjunto de tipos de datos en un formato portable, así como almacenamiento de atributos en estos datos.

NetCDF (Network Common Data Form) es un conjunto de librerías de formato de datos independiente de las máquinas y autodescritptivo que soporta la creación, acceso y compartición de datos científicos orientado a arreglos. El formato de dato es *self-describing*, esto significa que hay una cabecera en la cual se describe el diseño del resto del archivo, en particular de los arreg-

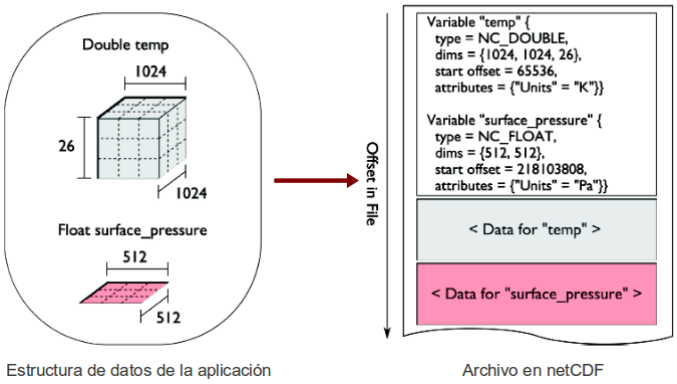


Figura 2.22: Ejemplo de un archivo en netCDF

los de datos, así como los metadatos de los archivos en la forma nombre/valor de los atributos (figura 2.22). El formato es independiente de la plataforma. Los arreglos son rectangulares, no desordenados, y almacenados en una forma simple y regular que permite un ajuste eficiente.

PnetCDF es construido sobre MPI-IO. En la arquitectura de PnetCDF, esta librería se ejecuta como una librería entre el espacio de usuario y el sistema de ficheros. Éste procesa la solicitudes paralelas netCDF desde los nodos de cómputo y, después de las optimizaciones, pasa la solicitudes de E/S a la librería MPI-IO, y entonces los servidores de E/S reciben las solicitudes MPI-IO y realizan la E/S sobre el almacenamiento final para todos los usuarios.

HDF5 Parallel

HDF5 [21] simplifica la estructura de un archivo al incluir dos tipos principales de objetos:

- *Datasets*, que son arreglos multidimensionales de un tipo homogéneo y
- *Groups*, que son estructuras contenedoras que pueden almacenar *Datasets* y otros *groups*.

Esto resulta realmente en una estructura jerárquica (figura 2.23), como el formato de datos de los sistemas de ficheros. De hecho los archivos en HDF5 son incluso accedidos usando sintaxis como POSIX. El Metadato es almacenado en la forma definida por el usuario, nombre de atributos adjuntado a los *datasets* y *groups*. HDF5 presenta un sistema de tipo mejorado, objetos de espacios de datos los cuales representan selecciones sobre regiones de *datasets*. El API es también orientado a objetos con respecto a *datasets*, *groups*, los atributos, tipos, espacio de datos y lista de propiedades. HDF5 también presenta su versión paralela HDF5 Parallel. HDF5 Paralelo abre un archivo con un comunicador. Éste retorna un manejador para ser usado para futuras accesos al archivo.

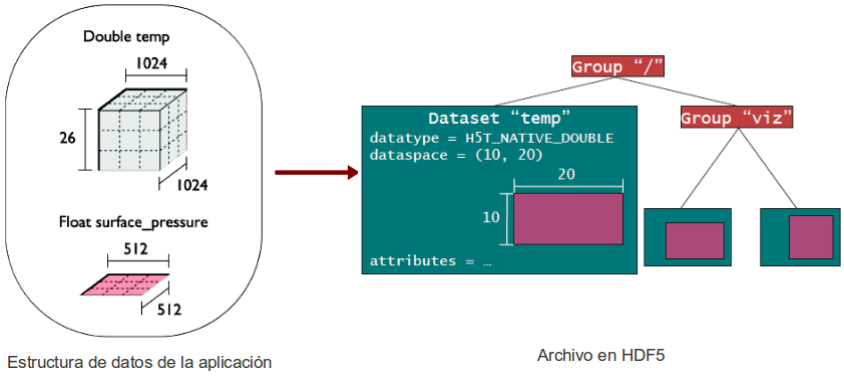


Figura 2.23: Ejemplo de un archivo en HDF5

Ejemplos de lo que se puede hacer con HDF5 Paralelo:

- Crear, abrir y cerrar un archivo
- Crear, abrir y cerrar un *dataset*
- Extender un *dataset* (incrementar el tamaño de dimensión)
- Escribir desde y hacia un *dataset* (la transferencia de los arreglos pueden ser colectivos o independientes)

Una vez que un archivo es abierto por los procesos de un comunnicador:

- todas las partes del archivo son accesibles por todos los procesos
- todos los objetos en el archivo son accesibles por todos los procesos
- multiples procesos escriben en el mismo *dataset*
- cada proceso escribe en un *dataset* individual

2.5.3. Benchamrks de Librerías de E/S

Debido a la influencia que tienen las librerías de E/S en las prestaciones de la E/S, se han desarrollado Benchmarks que permiten evaluar las prestaciones para algunas librerías. Aquí se presentarán dos Benchamrks muy usados para evaluar las prestaciones de la operaciones de E/S realizadas por MPI-IO, se eligieron estas porque las librerías de alto nivel funcionan sobre MPI-IO.

Benchmark de ancho de banda efectivo de E/S (b_eff_io)

El benchmark b_eff_io [22] tiene dos objetivos:

- lograr caracterizar el ancho de banda medio de las operaciones de E/S para aplicaciones paralela en MPI-IO y
- obtener información detallada de varios patrones de acceso y tamaños de buffers.

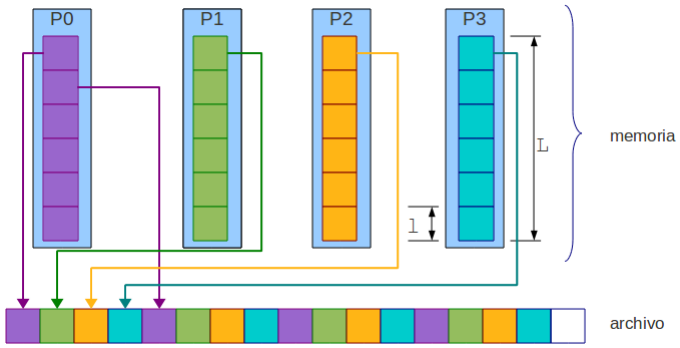
El benchmark examina los siguientes aspectos:

- un conjunto de particiones
- los métodos de acceso *initial write*, *re-write* y *read*
- los tipos de patrones (figura 2.24)
  - (0) *strided collective access*, distribución de un gran *chunks* en memoria (L en la figura 2.24(a)) para y desde el disco (l en la figura 2.24(a)),
  - (1) *strided collective access*, pero una llamada de lectura o escritura por *chunks* de disco,
  - (2) *noncollective access*, a un archivo por cada proceso MPI, por ejemplo: en archivos separados,
  - (3) igual a (2), pero los archivos individuales son ensamblados a un archivo segmentado,
  - (4) igual a (3), pero los accesos al archivo segmentado es realizado aplicando operaciones colectivas.

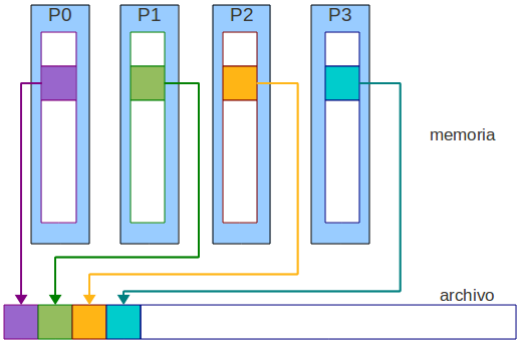
  Por cada tipo de patrón se usa un archivo.
- el tamaño de *chunk* continuo es elegido *wellformed*, por ejemplo: como potencia de 2, y *non-wellformed* que se obtiene agregando 8 bytes al tamaño *wellformed*,
- diferentes tamaños de *chunks*, principalmente 1kB, 32 kB, 1 MB y el máximo entre 2MB y 1/128 de la memoria de un nodo que esta ejecutando un proceso MPI.

MPI-Tile-IO

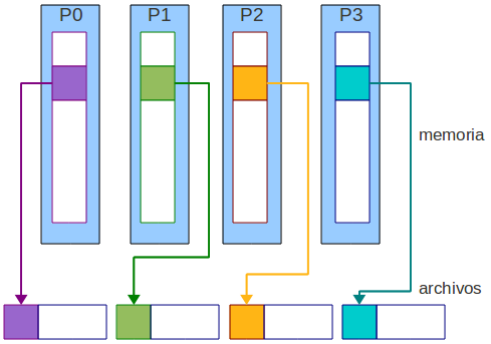
MPI-Tile-IO [23] es un benchmark de MPI-IO que prueba el rendimiento del acceso a los datos *tiled*. En esta aplicación, la E/S es sobre datos discontiguos y realizada en un simple paso usando colectivas de E/S. Este mide el acceso a un *tiled* a un conjunto de datos denso de dos dimensiones, simulando el tipo de carga que existe en algunas aplicaciones científicas y de visualización.



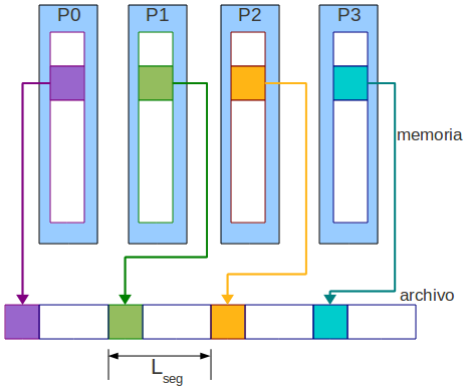
(a) Patrón tipo 0



(b) Patrón tipo 1



(c) Patrón tipo 2



(d) Patrón tipo 3 y 4

Figura 2.24: Patrones de transferencia de datos usados en b_eff_io. Cada diagrama muestra los datos transferidos por una llamada *write* en MPI-I/O.

Capítulo 3

Sistema de E/S Paralela

3.1. Estructura del Sistema de E/S Paralela

El sistema de E/S Paralelo comprende dos niveles: nivel de Librería de E/S Paralela y nivel de Arquitectura de E/S. Pero considerar al Sistema de E/S Paralelo sin tener en cuenta la aplicación, no es apropiado debido a que las prestaciones de la E/S paralela para una configuración de E/S varía de acuerdo al patrón de E/S de la aplicación, por esta razón se incorporá un nivel de aplicación. Debido a que no existe un consenso sobre los componentes de un sistema de E/S paralelo, es necesario establecer una estructura del sistema de E/S que será la base para establecer la metodología de configuración de la E/S Paralela (que se explica en el capítulo 5) y será útil para enmarcar el trabajo de tesina. En la figura 3.1 se muestra la estructura del sistema de E/S Paralelo para este trabajo y sus componentes. Un sistema de E/S debe satisfacer los siguientes requisitos[24]:

- Proporcionar el ancho de banda y capacidad que le exigen la aplicaciones actuales. Las prestaciones del sistema de E/S debe ser balanceado con la velocidad de cómputo y el ancho de banda de las redes de interconexión de los computadores paralelos.

- Minimizar la latencia de los accesos para reducir el tiempo de los procesos en espera. Esto es posible conectando los nodos de E/S a través de una red de altas prestaciones.

- Proporcionar fiabilidad frente a fallas, y mantener la disponibilidad de los datos cuando algún elemento falle o sea reemplazado.

- Conseguir efectividad de los Costos. En muchas ocasiones, agregar hardware de E/S de altas prestaciones es muy costoso. Sin embargo, el costo puede ser justificado si el sistema de E/S

Nivel	Parámetro
Aplicación Científica	◦Patrón de Acceso •I/O Contiguas •I/O Discontinguas •I/O Colectivas •I/O Independientes
Librería de I/O	◦Alto Nivel •NetCDF, PnetCDF •HDF4, HDF5 ◦Bajo Nivel •MPI-IO •POSIX-IO
Arquitectura de I/O	◦Conexión ◦Gestión ◦Colocación ◦Buffering/Caching ◦Disponibilidad

Figura 3.1: Sistema de E/S Paralelo

permite al computador, como una unidad, usarse más efectivamente, especialmente cuando consideramos que los sistemas de E/S internos reducen el espacio de los servidores de datos externos.

- Soportar operaciones complejas. Los sistemas de ficheros deben controlar de diseño de datos de los archivos, proporcionar la habilidad de particionar los datos entre los procesos de la aplicación y soportar la compartición de datos entre los procesos. Estas características deberían estar también en los modelos de programación y de particionado de datos usado en los computadores paralelos.

Además un sistema de E/S paralelo debe tener en cuenta la Arquitectura de E/S y también las librerías de E/S necesarias para realizar los accesos en forma eficiente. Debido a que las prestaciones de una configuración de E/S esta muy influenciada por los patrones de acceso de la aplicación paralela, se debe también considerar el tipo de aplicación y sus patrones de accesos.

3.2. Arquitectura de E/S Paralela

La arquitectura de los computadores paralelos deben proporcionar un balance entre cómputo, ancho de banda y capacidad de memoria, capacidad de comunicación e E/S. En el pasado, mucha parte del esfuerzo del diseño se centraba en el hardware y software del cómputo básico y la comunicación. Esto ha llevado a computadores desbalanceados con pobres prestaciones de E/S. En la actualidad se dedica esfuerzo en el diseño de hardware y software para el sistema de E/S en los computadores paralelos. Las arquitecturas que surgieron para los computadores paralelos se basa

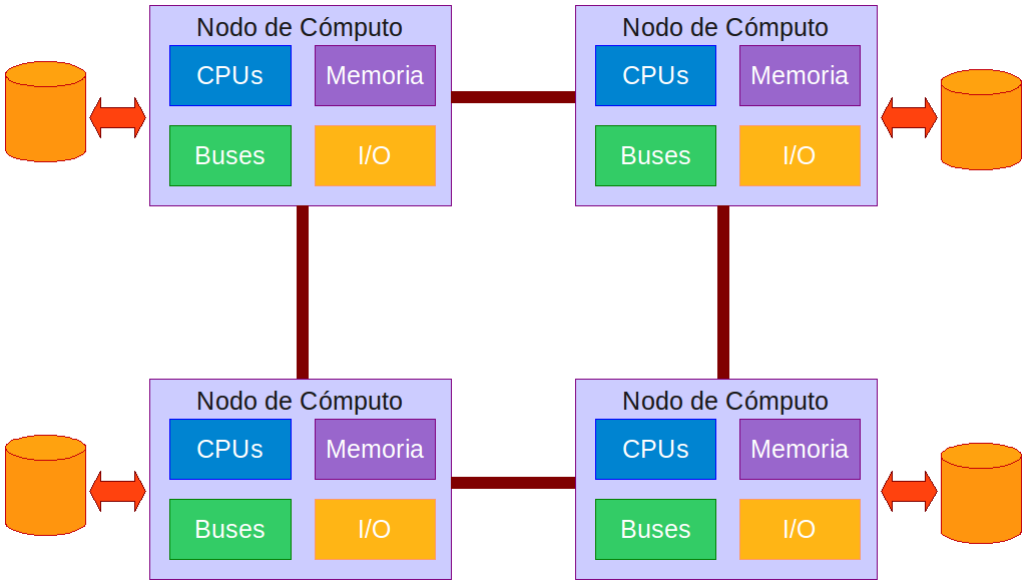


Figura 3.2: Cada procesador esta conectado a su propio disco local o a discos remotos

en una sistema de E/S interno, acompañado con una colección de nodos de E/S dedicados, cada uno gestionando y proporcionando accesos a un conjunto de discos. Los nodos de E/S estan conectados a otros nodos por una red de interconexión de la misma forma que lo nodos de cómputo se conectan con otros nodos. En esta sección se explica arquitectura de E/S paralela [25].

3.2.1. Conexión

Una red de interconexión es necesaria para mover los datos entre los multiples dispositivos de E/S (o nodos de E/S, controlodores, etc) y las multiples memorias. Existen dos modelos básico para la conexión:

- cada procesador está conectado su propio disco local o a discos compartidos. En el primer modelo, un procesador está conectado a su propio disco local. Cualquier procesador puede acceder a su propio disco local o a un disco remoto en un procesador remoto. Si un procesador está conectado con un disco local, el sirve como de controlador del disco local (ver figura 3.2)

- exiten dos clases de nodos: nodo de cómputo y nodo de E/S. Dependiendo de la posición de los nodos de E/S, el subsistema de E/S puede ser visible o independiente. El ratio de los nodos de E/S a los nodos de cómputo es muy importante para mantener el balance entre cómputo y subsistemas de E/S y como un resultado, decidiendo el rendimiento de E/S global

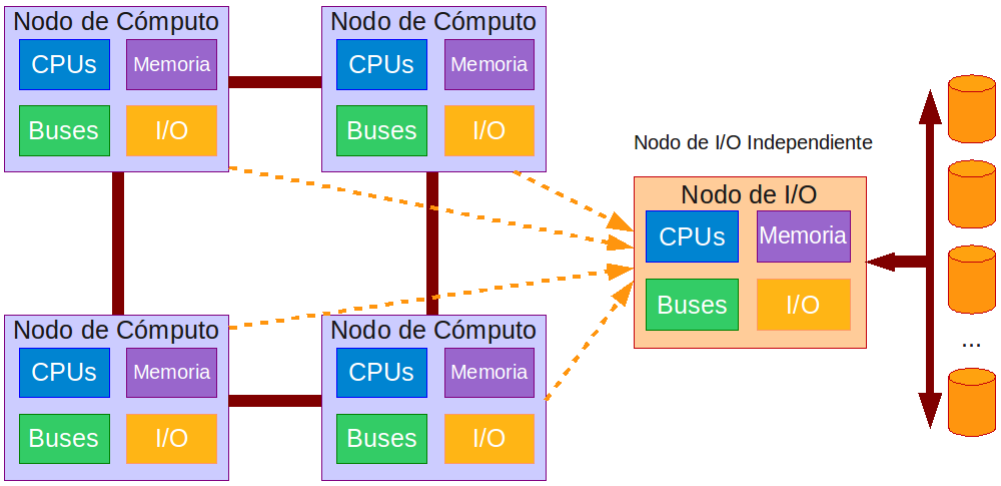


Figura 3.3: Nodo de E/S Independiente. Los nodos de cómputo acceden a disco por medio de los nodos de E/S.

(y el de la aplicación). Usualmente, los nodos de E/S constan del mismo hardware que los nodos de cómputo pero estos nodos pueden no ejecutar ninguna computación. Cada nodo de E/S actúa como un controlador dedicado para un conjunto de discos asociado. En este modelo, cada acceso a disco por un nodo de cómputo requiere intercambio de mensajes con el nodo de E/S controlando el disco. Cada nodo de E/S está conectado a un conjunto de discos a través de una red de interconexión. El subsistema de E/S está formado por los nodos de E/S, discos y red de interconexión. Dependiendo de la posición de los nodos de E/S, el subsistema de E/S puede ser, respecto de los nodos de cómputo, independiente o empotrado. En un subsistema de E/S independiente, los nodos y los discos están organizados físicamente como unidades separadas. El subsistema de E/S está conectado con el resto de la máquina (o parte de cómputo) a través de un canal de E/S (figura 3.3) En un subsistema de E/S empotrado, la E/S y los nodos de cómputo están organizados en una única unidad física. Los nodos de E/S utilizan la misma red de interconexión que los nodos de computo (Figura 3.4). Si múltiples procesadores envían peticiones de E/S al mismo controlador de E/S simultáneamente, resultará una congestión. Por tanto, es necesario proporcionar varios controladores de E/S los cuales pueden compartir la carga de trabajo. El ratio de los nodos de E/S a los nodos de cómputo es muy importante para mantener el balance entre cómputo y subsistemas de E/S y como un resultado, decidiendo el rendimiento de E/S global (y el de la aplicación).

Hay cuestiones a tener en cuenta en la conexión de los dispositivos de E/S y los nodos de cómputos:

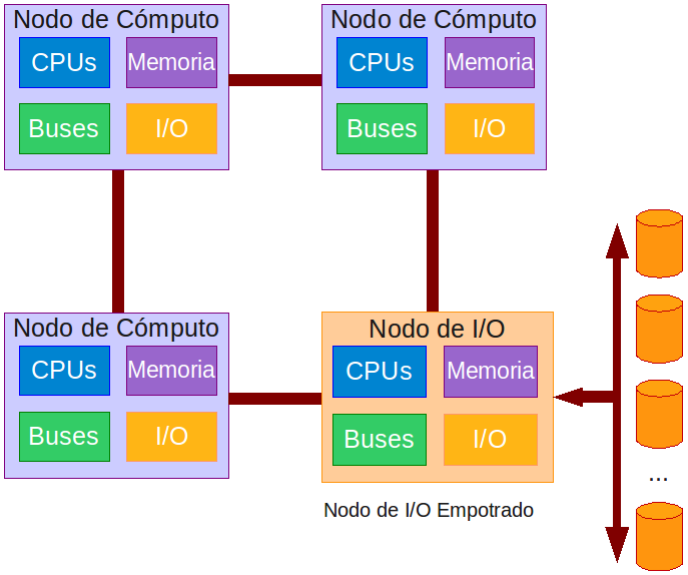


Figura 3.4: El nodo de E/S se encuentra integrado. Los nodos de Cómputo acceden a los discos por el nodo de E/S.

- ¿Hay una red o una subred dedicada al tráfico de la E/S? ¿El tráfico de E/S comparte la red de interconexión interprocesos?

  Un extremo es conectar los nodos de E/S, o incluso lo dispositivos de E/S, directamente a la red de cómputo. Otro extremo es proporcionar una red dedicada a la E/S. También se puede conectar cada nodo de E/S a unos pocos puntos en la red de cómputo usando enlaces extras. El tráfico de la E/S es diferente del tráfico de la red de cómputo. Los mensajes de E/S tienden a ser grandes y a ráfagas, mientras que los mensajes entre procesos tienden a ser pequeños. El *Throughput* es la meta de la comunicación relacionada con la E/S, mientras que la latencia es lo importante para los mensajes interprocesos. Cada tipo de comunicación puede causar congestión o contención que impactará negativamente en las prestaciones de la otra.

- ¿La interfaz de la red incluye DMA o memoria compartida? ¿Acepta accesos a nivel de usuario o requiere privilegios de usuarios? ¿El adaptador de E/S esta adjuntado directamente a la red de interconexión o al nodo de E/S?

  Esto es crítico debido a que los sitemas de E/S dependen de la habilidad de mover datos. En la actualidad el computador y la tarjeta se comunican entre sí para que puedan proceder al intercambio de información. De esta manera, el computador asigna parte de su memoria a las tarjetas que tienen DMA (acceso directo a memoria). La interfaz de la tarjeta indica que

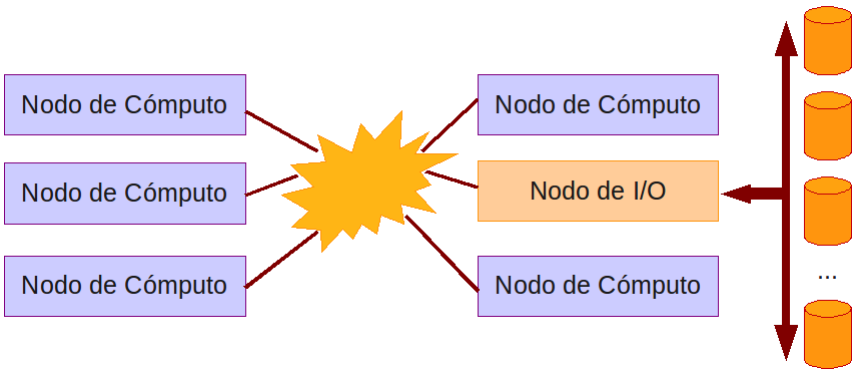


Figura 3.5: Gestión del accesos a los dispositivos Centralizada. Un nodo (Nodo de E/S) gestiona el acceso a los discos.

otro nodo está solicitando datos del nodo. El bus del nodo transfiere los datos de la memoria del nodo a la tarjeta de red. Si los datos se desplazan demasiado rápido como para que el adaptador proceda a su procesamiento, se colocan en la memoria del buffer de la tarjeta (RAM), donde se almacenan temporalmente mientras se siguen enviando y recibiendo los datos.

3.2.2. Gestión

La E/S implica el movimiento de los datos entre la memoria y los dispositivos periféricos y viceversa. En los multicomputadores hay varias memorias y varios dispositivos periféricos. Una cuestión clave, entonces, es la Gestión, ¿Qué procesadores gestionan el acceso a los discos? Hay tres soluciones comunes, donde la Gestión puede ser:

- Centralizada en un procesador: El enfoque centralizado (figura 3.5)es común en sistemas SPMD, donde la mayoría de la gestión es centralizada y el modelo de programación es síncrono. En grandes sistemas MPMD, sin embargo, representa un potencial cuello de botella, especialmente cuando son usados con un modelo de programación asíncrono.

- Distribuida entre todos los procesadores: Pocos sistemas eligen la gestión distribuida entre todos los procesadores (figura 3.6), prefiriendo concentrar el hardware de E/S en un subconjunto de nodos procesador que son usualmente dedicados a actividades de E/S. Cuando la gestión de los dispositivos distribuidos entre todos los nodos:

  - Un procesador está conectado a su propio disco local.

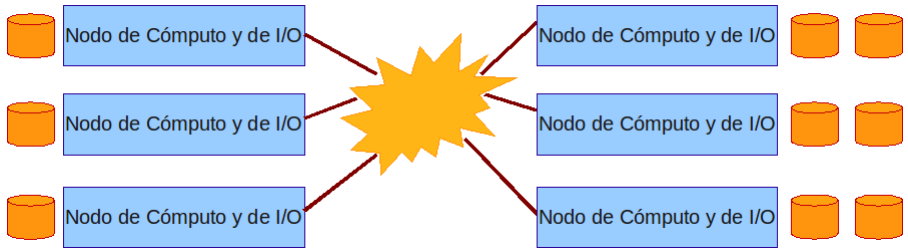


Figura 3.6: Gestión de Distribuida entre todos los procesadores

- Cualquier procesador puede acceder a su propio disco local o a un disco remoto en un procesador remoto
- Mejora la localidad si cada procesador puede enfocar su actividad de E/S en sus dispositivos locales
- Es útil para soporte de memoria virtual
- Es difícil caracterizar el impacto si cambia el workload
- Mejora la localidad de accesos, balanceo de carga y fiabilidad
- Posiblemente se presente interferencia con computaciones de usuario.
- Es aconsejable para NOWs

- Distribuida entre un subconjunto de procesadores que estan dedicados a la E/S (figura 3.7): El hardware de E/S se concentra en un subconjunto de nodos que estan dedicados a las actividades de E/S. La concentración de hardware de E/S en nodos de E/S tiene algunas ventajas sobre la distribución total:

  - El número de nodos de E/S y dispositivos pueden ser elegidos independientemente del número de nodos de cómputo, permitiendo una configuración del sistema más flexible.
  - Los nodos de E/S pueden ser diferentes, es decir, con una CPU diferente, más o menos memoria, hardware DMA especializado y desde luego adaptadores para periféricos y buses de E/S.
  - Pueden necesitarse menos adaptadores.
  - El sistema puede ser más simple, puesto que los nodos de cómputo pueden tener características físicas diferentes que los nodos de E/S. Además cada uno puede encontrarse en diferentes tipos de racks.
  - La actividad del servicio de E/S no impacta en aplicaciones de cómputo por robo de ciclos o memoria, o causando interrupciones inesperadas.

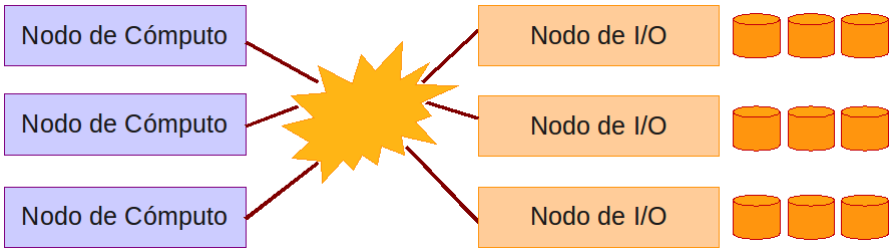


Figura 3.7: Gestión de los dispositivos a cargo de un subconjunto de nodos de E/S.

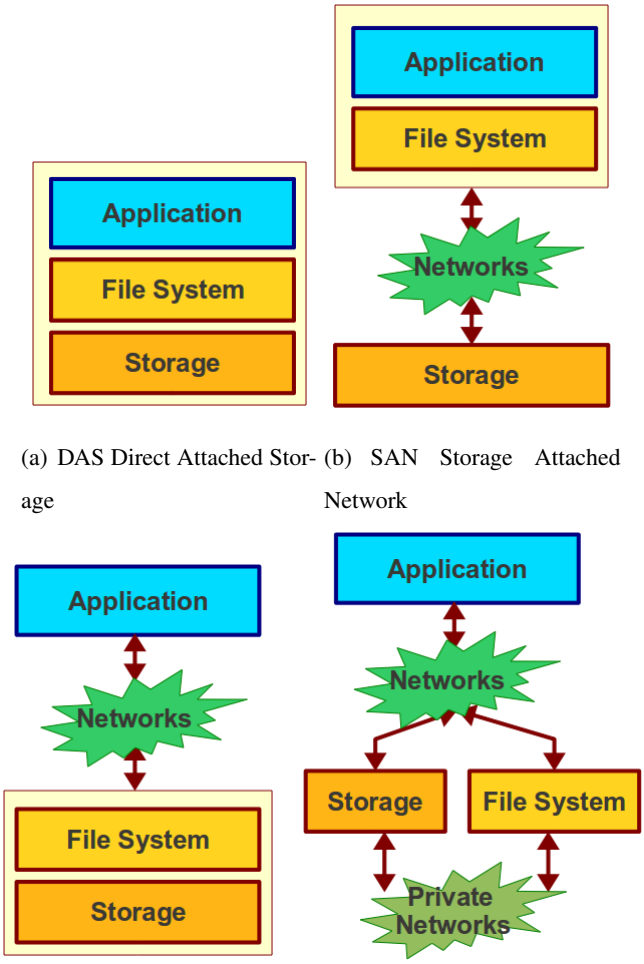
- Por otro lado, la gestión de la E/S distribuida entre todos los procesadores podría lograr mejor localidad, si cada procesador pudiera enfocar su actividad de E/S en sus dispositivos locales. Es difícil caracterizar el impacto en el rendimiento de su localidad dada la gran variedad de workloads y arquitecturas de redes de interconexión, pero parece adecuado que los discos locales sean útiles para paginación y otras formas de soporte de memoria virtual para computaciones *out-of-core*.

3.2.3. Ubicación

Todos los computadores paralelos tienen una red de interconexión con una topología determinada. La latencia de comunicación, el ancho de banda y la contención en estas redes a menudo depende de la posición relativa del destino de la comunicación. Entonces la posición de los nodos o dispositivos de E/S en la topología de la red pueden tener un impacto significativo en el rendimiento del sistema de E/S. Existen 3 enfoques típicos:

- La posición es ignorada; los nodos de E/S están colocados en cualquier sitio.
- Todos los nodos de E/S están agrupados en su propia "partición" de la red.
- Los nodos de E/S están distribuidos alrededor de la red, pero en posiciones elegidas cuidadosamente.

El desarrollo de las redes y de Internet han hecho que el compartir archivos sea algo sencillo, común y requerido. La posición de los discos o dispositivos de almacenamiento si estamos conectados a una red pueden tener un impacto significativo en el rendimiento del sistema. Hay 4 formas de acceder a los datos de acuerdo a la ubicación y gestión los dispositivos de almacenamiento: DAS (Figura 3.8(a)), SAN (Figura 3.8(b)), NAS (Figura 3.8(c)) y NASD (Figura 3.8(d)).



(a) DAS Direct Attached Storage

(b) SAN Storage Attached Network

(c) NAS Network Attached Storage

(d) NASD Network Attached Storage Devices

Figura 3.8: Acceso a los datos de acuerdo a la ubicación y gestión los dispositivos de almacenamiento

3.2.4. Buffering/Caching

El buffering y caching es un factor importante en cualquier sistema de E/S. El buffering es importante, por ejemplo, entre la unidad de disco y red de interconexión, para compensar las diferencias de velocidad y las diferente granularidad (bloques o paquetes), y las ráfagas dadas por las características de los dispositivos (búsqueda en los discos) o la carga (congestión de la red).
Un buffer/cache, almacena un pool de los bloques usados recientemente, este puede evitar accesos completos. Un buffer cache puede ser particularmente importante en los nodos de E/S de los computadores paralelos, porque pueden tomar ventaja de la localidad interproceso, cuando múltiples procesadores estan accediendo a diferentes partes del mismo bloque.

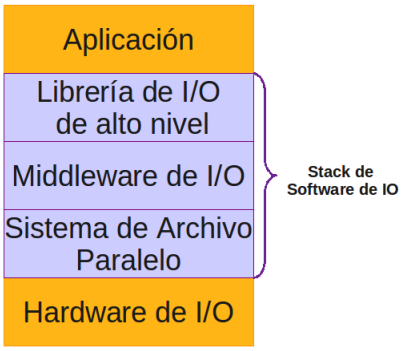


Figura 3.9: Stack de Software de E/S

3.2.5. Disponibilidad

Un computador paralelo esta formado por varios componentes, usados en paralelo para mejorar las prestaciones. Cuando los datos de los archivos son distribuidos sobre multiples dispositivos de almacenamiento, la falla de cualquier dispositivo causa la perdida del archivo. Por lo tanto, la distribución de los datos sobre mutiples dispositivos aumenta las prestaciones pero decrementa la fiabilidad. Para la falla en los discos normalmente es enmascarada con los sistemas RAID. La falla en los nodos de E/S pueden ser enmascarado con la redundancia de componente.

3.3. Stack de Software de E/S Paralela

Muchas avances se han hecho en los sistemas de E/S tanto en la comunidad de HPC como en grupos externos. Una categoría clave de mejora ha sido la organización del software de E/S y la definición de interfaces estándar para varias capas, tanto en sofware como en hardware. Otro avance ha sido el desarrollo y despliegue de implementaciones de multiples sistemas de archivos paralelos.

El software de E/S ha pasado de ser una librería secuencial monolítica a una *stack* de software en tres niveles: librerías de E/S de alto nivel, middleware o librería de bajo nivel y los sistemas de archivos paralelos. Estas capas se muestan en la figura 3.9. Cada una de estas capas proporciona un conjunto de funcionalidades; y para comprender el rol del sistema de archivo paralelo en este sistema multicapa, primero se debe ver *stack* de software como un todo [3].

3.3.1. Librerías de E/S de Alto Nivel

Es la encargada de aplicar estructura a los archivos para mantener una *self-describing*, un contenedor de dato portable, y presentar una abstracción de los datos al programador que este más

cerca al modelo de datos usado en la aplicación. Ejemplo de estas librerías son PnetCDF, HDF5; que se describieron en el capítulo 2.

3.3.2. Middleware de E/S

Este componente es responsable de proporcionar la base para que las librerías de alto nivel puedan ser construidas, pero también se las puede usar para hacer las operaciones de E/S pero requiere un nivel más de experiencia para hacer las operaciones de E/S. Esta capa proporciona un mapping desde interfaces relativamente simples de los sistemas de ficheros paralelos en interfaces que aprovechan conceptos de modelos de programación, tales como comunicadores, tipos de datos del modelo de programación MPI. Uno de los mejores ejemplos es la interfaz MPI-IO.

3.3.3. Sistema de Archivos Paralelo

El sistema de archivos paralelo es el responsable de gestionar todos los componentes hardware de almacenamiento. Este presenta una simple vista lógica de este hardware que puede ser aprovechado por otras capas de software. También hace cumplir un modelo de consistencia para que los resultados de los accesos concurrentes sean correctos. El sistema de archivos paralelo debe escalar para un gran número de clientes, manejar muchas operaciones concurrentes y aparentemente independientes. Debe presentar una interfaz en la cual los componentes de las implementaciones de las interfaces de alto nivel puedan ser construidas. Destacamos tres sistemas de archivos paralelos que actualmente están siendo desarrollados y desplegados en HPC. El Sistema de Archivo Paralelo General (GPFS) de IBM que surgió de el sistema de archivo multimedia *Tiger Shark* [26] y que ha sido ampliamente usado en plataformas AIX. El sistema de archivo Lustre [27] que es uno de los sitemas más usados entre los supercomputadores del TOP 500. El sistema de ficheros paralelo PVFS2 que es le segundo sistema de archivo desarrollado por los autores de [28]. Este difiere de las versiones anteriores porque es una versión libre, abierta a la comunidad para construir sistemas de ficheros paralelos para HPC y sirve no solo para una producción de un sistema de archivo paralelo sino también para investigación en conceptos de E/S paralela y esta disponible para cluster Linux.

Capítulo 4

Estado del Arte

En este capítulo se presenta el estudio realizado sobre E/S paralela, las lineas de investigación actuales, la historia de la E/S en los últimos 20 años y se presentan las tendencias de la investigación para los próximos años para hacer frente a las exigencias de la era de la exaescala. Y en cada punto tratamos de enmarcar el trabajo.

4.1. Lineas de Investigación en E/S Paralela

Un estudio realizado por Thomas Ludwig en [29] divide a la investigación de E/S en 4 categorías:

- Prestaciones y Disponibilidad: Alto paralelismo y redundancia de componentes
- Usabilidad: Middleware común para el manejo y sintonización de las Librerías de E/S
- Administración: Gestión y Configuración de la Arquitectura de E/S Paralela
- Herramientas: Análisis de las prestaciones de la E/S por medio de Benchmarks de E/S, Simuladores de la E/S, etc.

Incrementar las prestaciones es por supuesto la meta de cualquier sistema paralelo, y en este ítem se concentran la mayoría de las cuestiones actuales de E/S paralela: el análisis y predicción de los patrones de acceso es analizado para aprender como la E/S de las aplicaciones acceden a objetos y funciones; el mapeo entre el diseño lógico y físico debe centrarse en una localidad con mayor acceso a disco y una reducción del tráfico de red; el acceso no contiguo, que es frecuente en programas paralelos, debe ser mapeado en *chunk* orientado al acceso a disco; las operaciones colectivas en el nivel de usuario permite empaquetar muchas solicitudes independientes en una solicitud simple;

con el uso de los metadatos se tiene que hacer frente al problema del conocimiento distribuido y como mantener eficientemente la información consistente.
Varios aspectos técnicos han sido desarrollados: RAIDs, SANs (storage area networks), NAS (network attached storage) y NASD entre las propuestas más populares. Para acceder a los datos eficientemente se usan abstracciones de alto nivel como sistemas de archivos paralelos, MPI-IO, HDF5, NetCDF [29].

4.2. Trabajos Relacionados

Este trabajo se encuadra en la linea de investigación **Gestióny Configuración de la Arquitectura de E/S Paralela**. Sin embargo, esto no está aislado de las otras áreas porque las prestaciones y disponibilidad son parte del objetivo planteado, conocer la librerías de E/S paralela y las herramientas de medición es fundamental para lograr validar las propuestas. La usabilidad debe ser un factor a considerar al momento de diseñar e implementar cualquier herramienta.
Los trabajos muestran cómo se puede mejorar los sistemas de E/S para aumentar las prestaciones de E/S y garantizar la disponibilidad. Algunos autores [30] consideran que primero se debe analizar las demandas de E/S de las aplicaciones para poder acortar la brecha entre la velocidad de cómputo y la E/S.
Entre los primeras ideas se realizo una clasificación de las aplicaciones científicas según su patrón de E/S, Miller and Kantz [5] las dividen en tres categorías: E/S requerida, *checkpoint* y *staging data*. Las de E/S requerida incluyen lecturas de entrada y escritura final de datos. Existen diversos estudios realizados para determinar los patrones de E/S de las aplicaciones científicas. En el artículo “Learning to Classify Parallel Input/Output Access Patterns” de [31] donde clasifican a los patrones de acuerdo a si son de lectura, escritura o lectura/escritura, también de acuerdo al tamaño de las solicitudes de E/S: uniforme pequeño, uniforme medio, uniforme grande, variable pequeño, variable medio, variable grande; y por último por la secuencialidad: secuencial, 1-d strided, 2-d strided, sin variación y strided variable. Si bien existen varios estudios realizados esta clasificación se sigue utilizando en la actualidad. Investigadores de Cray vienen trabajando en el problema de la E/S. Han presentado varios trabajos [30] [32] [33] [34] donde se puede ver las investigaciones realizadas sobre la E/S de las aplicaciones del supercomputador Jaguar y las herramientas desarrolladas para lograr unas mayores prestaciones en su sistema de E/S. Otros trabajos también realizados fueron los de [35] que trabajan sobre el sistema Red Storm que muestra un estudio de las prestaciones de su sistema de E/S, la gente de Sandia National Laboratories también realizo

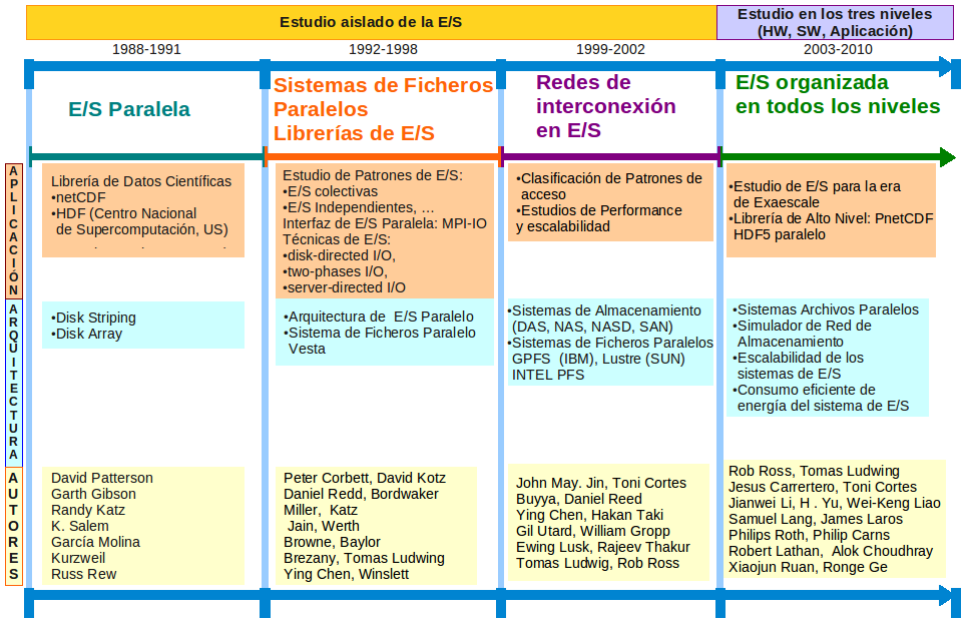


Figura 4.1: Estado del arte de la E/S Paralela

estudios de la E/S en las aplicaciones de su computador paralelo. Otros que trataron el tema [36] trabajaron en la evaluación de las prestaciones de la E/S del supercomputador Blue Gene/L.

Una herramienta necesaria para evaluar y cuantificar el cuello de botella de la E/S son las herramientas de simulación. El artículo "Performance evaluation of a massively parallel E/S subsystem"por S. J. Baylor, C. Benveniste and Y. Hsu" [37] introduce una herramienta de simulación que permite modelar y usar una simulación orientado por trazas para evaluar las prestaciones del subsistema interno de E/S paralelo de la arquitectura de Vulcan MPP, que les permitió hacer una estimación cuantitativa de ratio nodo de computo – nodo de E/S y la potencial escalabilidad de la arquitectura. En el paper "SIMCAN: A SIMulator Framework for Computer Architectures and Storage Networks" de [38] que permitirá estudiar el comportamiento de ambientes distribuidos complejos para varios propósitos, como la detección de los cuellos de botellas del sistema, cálculo del grado de escalabilidad del sistema y probar las prestaciones de aplicaciones, sin usar el sistema real.

4.3. El Sistema de E/S Paralela en los últimos 20 años

La evolución de la E/S Paralela se puede dividir en 4 partes y se destacan los principales de cada época. En la figura 4.1 se muestra en una línea de tiempo los avances en E/S Paralela.

4.3.1. Origen de la E/S Paralela - Finales de los 80s inicios de los 90s

A finales de los 80s y principio de los 90s aparecen los primeros papers que hablan sobre E/S paralela con la idea de **Disk Striping** [39] **Multi-disk management algorithms** [40], **A Case for Redundant Arrays of Inexpensive Disks (RAID)** de [41].

4.3.2. Sistema de Ficheros y Librerías de E/S - Mediados de los 90s

A mediados de los 90s empezaron a surgir los supercomputadores basados en clúster a gran escala, cientos o un poco de miles de nodos. Para poder proporcionar un ancho de banda de almacenamiento necesario para estos supercomputadores se emplearon cientos de dispositivos RAID con cientos de discos. NFS no fue el sistema de fichero porque no podía proporcionar un ancho de banda escalable y gestionable. Sistemas como GPFS (General Propose File System) de IBM, Meiko PFS (sistema de fichero Paralelo de Meiko), y PFS de Intel, son sistemas de Archivos paralelos que surgieron en esos años. Además fueron creadas para manejar el paralelismo a gran escala, librerías de E/S como MPI-IO.
Proyectos para desarrollar SAN basada en sistema de archivos. Al inicio los SAN estaban pensado para una pequeña cantidad de nodos (decenos o pocos cientos) dado la complejidad y costo de construir una SAN de fibre channel. Otra idea que volvió fue la incorporación de servidores de metadatos. Algunos sistemas de archivos desplegaron este concepto a través del uso de servidores de metadatos y otros compartian la responsabilidad entre los sistemas de archivos clientes a través del uso de mecanismos de bloqueos.
Entre los principales autores de esta epoca se destacan: Peter Corbett, David Kotz, Daniel Redd, Bordwaker, Miller, Katz, Jain, Werth, Browne, Baylor, Brezany, Tomas Ludwing, Ying Chen, Winslett.

4.3.3. Redes de Interconexión para E/S Paralela

A finales de los 90s los cluster de computadores empezaron a tener miles o incluso decenas de miles de nodos. Dado que los discos no mejoran en velocidad como los procesadores, la cantidad de discos necesitan ser coordinados para lograr alcanzar el ancho de banda de estos supercomputadores. Soluciones para sistemas de archivos globales fueron necesarias. En esta etapa surge una nueva estructura para arquitectura de E/S como por NASD. Entre los principales autores de esta época se destacan: John May. Jin, Cortes, Buyya, Daniel Reed, Ying Chen, Hakan Taki, Gil Utard, William Gropp, Ewing Lusk, Rajeev Thakur, Tomas Ludwig, Rob Ross.

4.3.4. La E/S Paralela organizada en todos los niveles

Se ha trabajado a lo largo de lo últimos años en lograr mejorar la E/S Paralela, se han desarrollado librerías de E/S como HDF5, PnetCDF y MPIO. y los estudios mostrado en el area E/S apuntan a un estudio integrado de todos los niveles. Esto se puede observar en los trabajos relacionados y las lineas futuras de investigación. Entra los principales autores de esta época se destacan:Rob Ross, Ludwing T, Carrertero, Toni Cortes, Jianwei Li, H . Yu, Wei-Keng Liao, Samuel Lang, James Laros, Philips Roth, Philip Carns, Robert Lathan, Alok Choudhray

4.4. Los desafíos de la Era de Exascala para E/S Paralela

Hay muchas tecnologías y exigencias de las aplicaciones científicas para la E/S [3] que van desde alternativas de arquitectura para sistemas de E/S, requisitos de aplicación o propósito para hacer E/S, la *stack* de software de E/S, la capacidad esperada de los dispositivos y la tolerancia a fallos. La gestión de los datos, el ciclo de vida, su futuro uso y disponibilidad también tienen influencia en como el sistema software de E/S debe ser diseñado. Dado el estado actual de la E/S y de los sistemas de almacenamiento en los sistemas de petaescala, soluciones incrementales son muy poco probable que puedan proporcionar la capacidades requeridas por los sistemas de exaescala. Las arquitecturas de E/S diseñadas como componentes separados e independientes de la infraestructura del computador no están preparadas para escalar como se les exige. Esto es, tradicionalmente, la E/S ha sido considerada como una actividad separada que es realizada antes o después de la simulación o del análisis principal del cómputo, o periódicamente para actividades como checkpointing, incluso se lo toma como un *overhead* separado. Esta mentalidad en el diseño de las arquitecturas, software y las aplicaciones debe cambiar si se quiere explotar la potencialidad de los sistemas de exaescala.

Los sistemas de ficheros, que han sido principalmente adaptaciones de sistemas de ficheros heredados, con demasiadas restricciones semánticas que no son escalables. Las interfaces tradicionales en los sistemas de archivos y sistemas de almacenamiento, o incluso en algunos casos, las librerías de datos de alto nivel, son diseñados para manejar escenarios en el peor de los casos para conflictos, sincornización, coherencia; en su mayoría ignorando la E/S para una aplicación, la cual es una fuente importante de información para escalar las prestaciones de la E/S para millones de core accediendo simutaneamente al sistema de E/S.

Los dispositivos de almacenamiento emergentes tales como los discos de estado solido (SSDs), las memorias de clase de almacenamiento (SCM) tiene el potencial para alterar significativamente las

arquitecturas de E/S, sistemas, prestaciones y sistema software para explotarlo. Las tecnologías emergentes tienen también el potencial de optimizar el consumo de energía. La fiabilidad de una aplicación bajo fallas en una sistema de exaescala dependerá principalmente del sistema de E/S, su capacidades, su capacidad y prestaciones ya que guardar el estado del sistema en forma de checkpointing es muy probable que siga siendo una de las propuestas.

4.4.1. Alternativas de Estrategias en Investigación y Desarrollo

Hay muchas estrategías de investigación y desarrollo para sistemas de E/S en diferentes niveles de la arquitectura y *stack* de software que pueden potencialmente ser útiles en aspectos de la tecnología y para los sistemas de exaescala [42]. Las métricas de los sistemas de E/S son las prestaciones, escalabilidad, adaptabilidad de aplicaciones, programabilidad y tolerancia a fallos.

- Delegación y Personalización en el Middleware de E/S: el mejor lugar para la optimización y la escalabilidad de la E/S es el middleware en el espacio de usuario ya que es aquí donde hay más información disponible de la semántica, distribución de datos, uso de los datos, patrones de acceso. El middleware no está únicamente para un simple espacio de usuario sino también está cooperando con otras actividades de E/S de los archivos de usuario en la máquina, así una optimización que se extienda al sistema podría ser realizada. El concepto de delegación en el midleware de E/S implica el uso de una pequeña fracción del sistema, en el cuál el midleware existe y se ejecuta en el espacio de usuario para realizar las funciones en relación a la E/S y a la optimización en nombre de la aplicación.

  Usando los requisitos de la aplicación, ésta puede realizar caching proactivo e inteligente, optimizaciones, nivelar los accesos de E/S a ráfagas para ordenar patrones. Esta propuesta puede proporcionar servicios para las aplicaciones de forma tal que la aplicación pueda personalizar los recursos usados basado en sus requerimientos. La delegación y personalización dan la oportunidad para realizar varias funciones sobre los datos mientras estos están siendo producidos o están en preprocesamiento antes de ser consumidos. La disponibilidad de nodos multicore dan la oportunidad de usar uno o más core en cada nodo para realizar servicios de E/S usando un conjunto exclusivo de nodos selectos, proporcionando un rango de opciones de personalizaciones inlcuyendo mejoras en la localidad.

- Almacenamiento activo y análisis online: el concepto de de almacenamiento activo esta basado en la premisa de que las arquitecturas modernas de almacenamiento podrían incluir recursos de procesamietno útiles en los nodos de almacenamiento, que pueden ser explota-

dos para realizar varias tareas importantes incluyendo análisis de datos, organización, redistribución, etc. Este concepto tiene un potencial significativo para ayudar a mejorar las prestaciones y descripción de conocimiento por la explotación del poder de procesamiento en el caching, nodo delegados, o en el sistema de almacenamiento.

- Capas de Software de E/S dirigida por propósitos: las interfaces de E/S tradicionales no explotan explícitamente el propósito de las operaciones de E/S. Un checkpointing de una operación de E/S es diferente de cualquier actividad de E/S, éste almacena datos para un análisis futuro usando algún otro patrón de acceso. Un ejemplo de este último es el uso de datos analizando un subconjunto de variables a lo largo del tiempo. Las optimizaciones en las dos actividades pueden requerir diferentes propuestas para las capas de software. Las capas de software desde los sistemas de archivos, middleware y librerías de alto nivel podrían ser modificadas con la incorporación de estas capacidades para explotar el proposito de la E/S.

- Sistemas Software para la integración de dispositivos de almacenamiento emergente: la incorporación de esta tecnología puede mejorar las prestaciones, reducir el consumo de energía, mejorar el caching, y puede potencialmente reducir/eliminar las actividades de E/S explícitas y el tráfico de los discos tradicionales si ellos son incorporados de forma transparente en la capa de software de E/S.

- Extender los sistemas de ficheros actuales: en esta propuesta, los esfuerzos pueden extender los sistemas de ficheros actuales para cubrir la necesidades de prestaciones o paralelismo. Sin embargo, debido a las actuales capacidades y prestaciones de estos sistemas, que derivan de diseños proactivos y conservativos con semánticas secuenciales estrictas, las oportundades de éxito es esta área estan limitadas.

- Desarrollo de nuevas propuestas para sistemas de archivos escalables: nuevos modelos, interfaces y propuestam que no estén limitadas por la semántica secuencial y los modelos de consistencia, que incorporen nuevas técnicas y metadatos altamente escalables, que puedan explotar la información disponible desde los niveles más altos y del usuario y que puedan incorporar nuevos dispositivos de almacenamiento.

- Incorporación de la E/S en los modelos y lenguajes de programación: proporcionar capacidades a los modelos de programación y características a los lenguajes que permitan al usuario proporcionar requisitos de E/S, patrones de acceso y otra información de alto

nivel a los modelos y lenguajes de programación. Esta ifnormación puede ser usada por compiladores para optimizar la E/S, los *pipeline* de E/S y planificar inteligentemente la E/S para maximizar el solapamiento con el cómputo; y en las cuales las arquitecturas multicore pueden ser explotadas para utilizar cores de E/S para mejorar las prestaciones de E/S; específicamente funciones de análisis online y sistemas delegados de almacenamiento activo son área de invetigación muy importantes.

- E/S de área amplia e integración de sistemas de almacenamiento externos: se necesitan técnicas escalables en las cuales el paralelismo en el acceso a los dispositivos de almacenamiento este integrado con el flujo de red. También la integración del flujo paralelo de datos sobre la red.

Capítulo 5

Metodología para la Configuración de la E/S

5.1. Descripción de la Metodología

Para definir la configuración de E/S de un sistema paralelo se debe elegir la Arquitectura de E/S y las librerías de E/S. Pero las prestaciones que esta configuración logre entregar dependerá de la aplicación que se ejecute en el sistema paralelo. Esto quiere decir que para realizar la configuración del sistema se debe considerar: Aplicación, Librerias de E/S y Arquitectura de E/S. En al figura 5.1 se presentan los factores de E/S considerados. De estos factores de E/S se debe realizar un análisis de la influencia que tienen en las prestaciones y disponibilidad. Debido a la cantidad de factores para cada componente del sistema de E/S, el estudio de los factores de E/S es una tarea compleja. Por esta razón, se decidío diseñar una metodología que permita identificar los factores claves de E/S y evaluar la influencia que tienen en las prestaciones y disponibilidad.

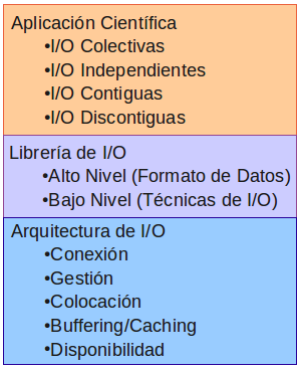


Figura 5.1: Sistema de E/S Paralela. Factores para la configuración de E/S

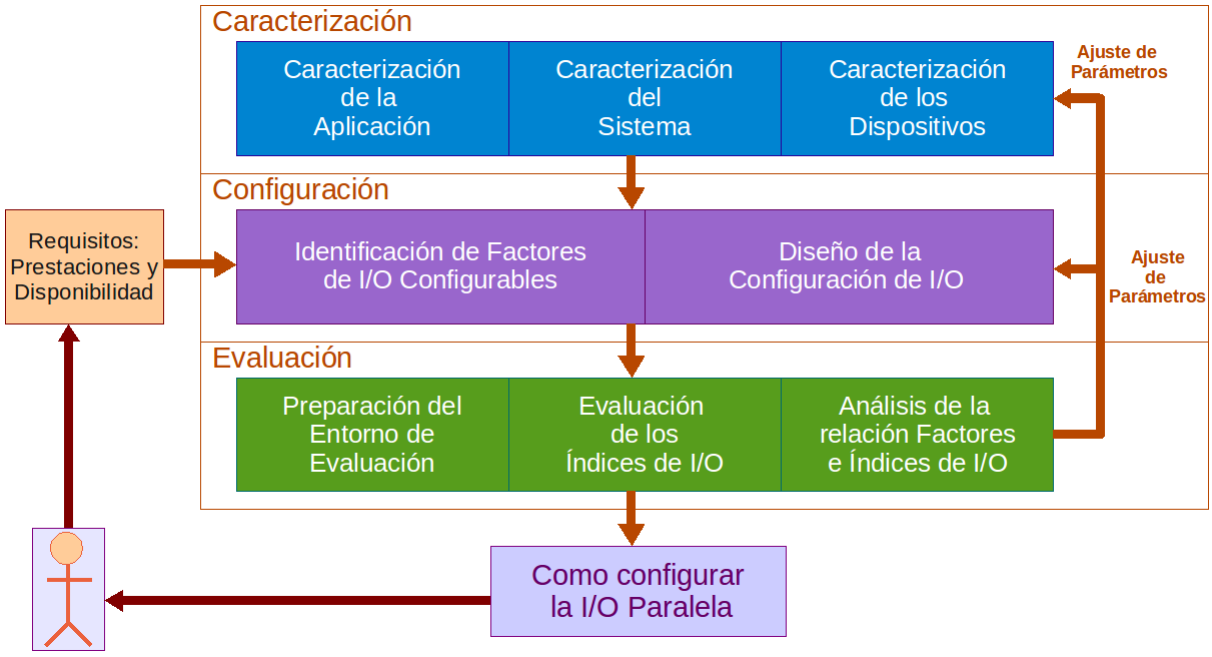


Figura 5.2: Metodología para Configurar la E/S Paralela

Esta metodología de configuración de la E/S (figura 5.2) incluye los pasos, índices y herramientas que deberían considerarse al momento de decidir que configuración de E/S es la más apropiada para un sistema paralelo.

La metodología propuesta consta de 3 fases: Caracterización, Configuración y Evaluación. Un factor fundamental a considerar son los requisitos del usuario, dado que el usuario proporcionará los recursos que esta dispuesto a adquirir para configurar el sistema de E/S para conseguir mayores prestaciones y el nivel de disponibilidad que esta dispuesto a pagar.

5.2. Caracterización

La fase de caracterización esta destinada a obtener los factores de la Aplicación, Sistema de Paralelo y Dispositivos de E/S que brinden la información necesaria para poder realizar la fase de configuración. En la figura 5.3 se muestra ¿Qué? y ¿Cómo? caracterizar.

5.2.1. Aplicación Científica

La información del comportamiento de la aplicación se puede obtener de la especificación que puede acompañar al código fuente. Si esta información no es suficiente se debe hacer un análi-

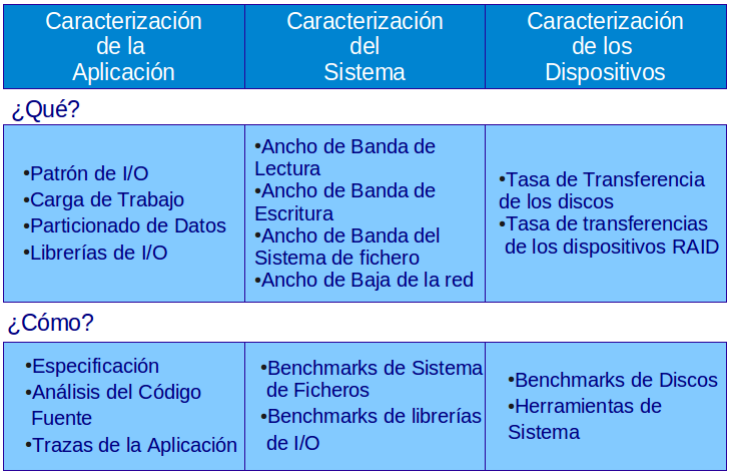


Figura 5.3: Fase de Caracterización - Metodología de Configuración de E/S

sis de las estructuras de datos que maneja el programa, para poder determinar el particionado de datos.

Para ver el patrón de E/S es conveniente realizar un ejecución y obtener las trazas de la aplicación. Esta será útil para ver si la aplicación es intensiva de E/S y el tipo de patrón de acceso. Puede darse el caso que la aplicación presente más de un patrón de E/S, en ese caso sería conveniente centrarse en el patrón que domina la aplicación.

Un herramienta útil y que puede hacer las trazas de una aplicación con simples agregados a lineas de compilación es JUMPSHOT que viene incorporado a MPE. La librería MPE se encarga de hacer las trazas en formato que JUMPSHOT puede visualizar. Jumpshot muestra todas las llamadas MPI y proporciona la cantidad de veces que la operación fue invocada, esta herramienta es aplicable a programas MPI.

La carga de trabajo de la aplicación se debe conocer para poder determinar la cantidad de información que la aplicación puede producir en el momento de ejecutarse, por lo menos tener claro que recursos debería tener reservado el sistema de E/S para que la aplicación funcione correctamente. Algunas aplicaciones incrementan su carga de trabajo cuando se incrementa el número de procesadores. Por otro lado, hay aplicaciones que a pesar de aumentar la cantidad de procesos el trabajo sigue siendo el mismo. Esta información es importante al momento de configurar la E/S para ese tipo aplicaciones.

El particionado de datos nos sirve para determinar que tipo de ubicación y colocación conviene utilizar. Además esto es muy útil para determinar el tamaño de cache y el stripping de archivo. Debido a que las aplicaciones científicas utilizan varias librerías de E/S, es conveniente tener identificadas que librerías usa la aplicación. Se debe tener cuidado con las librerías de E/S de alto nivel

porque estas pueden ser tanto secuenciales como paralelas y muchas veces en las especificaciones de las aplicaciones no se explicita que tipo usa la aplicación. Por ejemplo: pueden poner como requisito tener instalada la librería HDF5, pero HDF5 tiene una parte paralela que también debe ser instalada si los procesos de la aplicación accederán a los archivos en forma paralela.

5.2.2. Sistema Paralelo

Desde el punto de vista de la E/S lo que interesa caracterizar del sistema paralelo es la velocidad con la que hace las operaciones de E/S. La velocidad de las operaciones depende del dispositivo de almacenamiento pero esta varia dependiendo del sistema de fichero. Debido a que se tener tres niveles de sistema de ficheros (local, paralelo, distribuido). La evaluación del ancho de banda de las lecturas y escrituras se debe realizar en los distintos niveles. Existen en la comunidad de E/S dos benchamrks muy utilizados Bonnie++ y Iozone; estos son usados para realizar comparaciones de sistemas de ficheros así como de dispositivos de almancenamiento. Estos dos benchmarks se explicaron en el capítulo 2. En el caso de Iozone también permite evaluar las prestaciones de sistemas ficheros distribuidos y paralelos.

Otro aspecto a tener en cuenta es la red de interconexión; el benchmark *ttcp* permite evaluar la velocidad de transferencia de la red para diferentes tamaños de paquetes.

Si bien la operaciones de lectura y escritura se realizan en los dispositivos de almancenamiento, en los supercomputadores los datos deben ser transferidos por la red de interconexión (sea de E/S o compartida con los datos) y además se van afectados por el sistema de fichero a nivel de sistema paralelo y después a nivel de disco por el sistema de fichero local.

Por otro lado, las librería de E/S también realizan las operaciones de E/S, o sea que también se debe evaluar la velocidad con la que las librerías realizan las operaciones de E/S. Una de las librerás más usadas es MPI-IO, y se han desarrollado varios benchmarks que permiten medir la velocidad con la que hacen las operaciones de E/S para diferentes patrones de acceso, una de los más usado es b_eff_io (explicado en el capítulo 2). Este benchmark realiza pruebas sobre varios patrones de E/S y para distintos tamaños de archivo, probando todas las operaciones de MPI-IO sobre los patrones de E/S.

5.2.3. Dispositivos de Almacenamiento

Al igual que en el caso del sistema paralelo para evaluar la velocidad con la transfieren los dispositivos se pueden usar herramientas de sistemas(hdparm) que dan información de la tasa transferencia de los discos y esto se puede aplicar a los sistemas de multiples dispositivos como

por ejemplo los sistemas RAID. Este valor es una medida que sirve como límite máximo, es la mayor tasa de transferencia que se podría conseguir en una operación de lectura o escritura. Pero debido a la gran influencia que tiene el sistema de fichero local (o sea el sistema en el que se le dió formato al disco) se usan además los benchmark de sistemas de ficheros pero a nivel local. Con estas dos herramientas se puede obtener valores que realmente muestren el rendimiento de los discos en situaciones normales.

5.3. Configuración

En esta fase se identifican los factores de E/S que es posible configurar y para diseñar configuraciones de E/S. A continuación se presentan la identificación de factores de la E/S y la subfase de diseño de la configuración de E/S.

5.3.1. Identificación de Factores de E/S Configurables

Si bien existen varias formas de ver el sistema de E/S, en este trabajo se eligió las estructura presentada en el capítulo 3. En la figura 5.4 se muestran los factores identificados a nivel de Aplicación, Librería de E/S y Arquitectura de E/S. Dentro de esta estructura se identificaron los siguientes factores y se los presenta de acuerdo a la posición en la estructura del sistema de E/S.

Aplicación Científica

La técnica de E/S que se puede aplicar a una aplicación depende de sus patrones de E/S. Éste es un factor que tiene mucha influencia en el rendimiento de la aplicación en una determinada configuración de E/S. En este caso si se puede indicar que la aplicación se ejecute con algunas de las técnicas de E/S habilitada o deshabilitada, es un factor que debe ser considerado para configurar y variar para analizar el comportamiento de la aplicación bajo determinada técnica de E/S. El programador puede realizar una implementación de la aplicación que maneja las operaciones de E/S de manera eficiente pero generalmente esta implementación sólo es eficiente para algunos patrones.
Sería útil, de acuerdo a los patrones de E/S, sugerir al usuario que teécnica de E/S es la que más se ajusta al comportamiento. Si la aplicación ya incluye una implementación en la que se pueda probar las técnicas de E/S. En este caso se configura a nivel ejecución o compilación, generalmente usando *hints* este permite pasar a un programa parámetros para hacer la operaciones de E/S, incluso se puede elegir el tamaño de buffer para hacer operaciones de E/S colectivas.

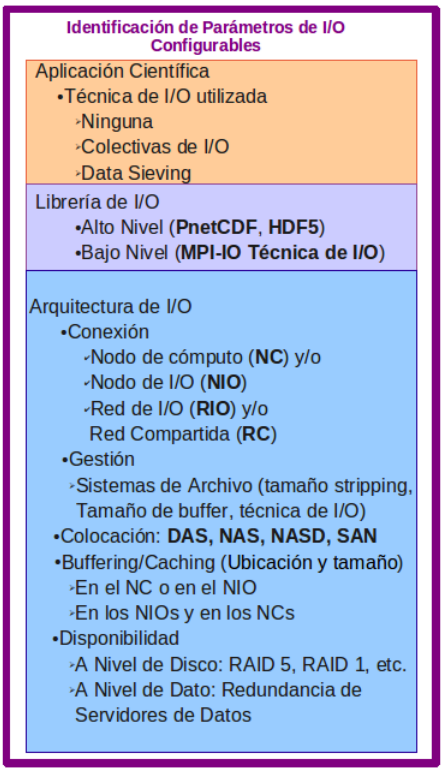


Figura 5.4: Identificación de Parámetros de E/S Configurables

Librería de E/S

Otro factor que se puede configurar es la librería de E/S de alto nivel. Las librerías influyen en las prestaciones de la aplicación. Algunas aplicaciones nos permiten realizar la ejecución con diferentes librerías y esto depende de la plataforma, y si existirán accesos paralelos.
Las librerías de bajo nivel permiten emplear las técnicas de E/S de forma explícita. Por lo tanto, para una aplicación usar librerías de E/S de bajo nivel ó no usarla puede llevar a una perdida de rendimiento y no permitir los accesos paralelos a los archivos por multiples procesos.

Arquitectura de E/S

La arquitectura de la E/S tiene varios factores (figura 5.4) que se pueden configurar. Sin embargo, es la más compleja porque los 5 componentes no se pueden ver analizar de forma aislada, un cambio en un componente implica una variación en las prestaciones que el sistema de E/S entrega, este cambio puede ser consecuencia directa del componente que se modifico o un efecto secundario. Además los cambios en la configuración de la arquitectura paralela muchas veces requiere reiniciar el computador paralelo para que se reflejen los cambios.
Es la arquitectura de E/S el punto más complejo en la configuración de la E/S. Es más, en muchos

Aplicación	Librería de I/O	Conexión	Gestión	Colocación	Buffering / Caching	Disponibilidad
Sin Colectivas de I/O	ROMIO	RC+NC	NFS	DAS	NC	
Con colectivas	ROMIO+HDF5	RIO+NC+NIO	Lustre	NAS	NC+NIO	RAI5D5
Sin Colectivas de I/O	ROMIO+PnetCDF	RC+NC+NIO	Lustre	SAN	NIO	RAID5+1 Server de de I/O Replicado
...						

Figura 5.5: Posibles configuraciones de E/S

casos cambiar un componente de la arquitectura puede llevar a parar el sistema paralelo. Si el lector esta interesado en entrar en detalle en cada elemento, una descripción detallada se encuentra en el capítulo 3.

5.3.2. Diseño de la Configuración de la E/S

Una vez identificados los factores de E/S que se pueden configurar y su impacto en las prestaciones, la tarea de diseñar una configuración que debería ser una combinación de los factores de E/S que son configurables. Sin embargo, esto no es así, configurar la E/S requiere una combinación coherente de los componentes. Esta tarea requiere de conocimientos en el área de redes, sistema de ficheros, librerías de E/S, buffering/caching, Dispositivos de almacenamiento y RAID.
De estos parámetros identificados se diseña una configuración de E/S coherente. Por ejemplo si utilizamos un sistema de ficheros Lustre debemos tener en cuenta que se debe usar nodos de E/S porque la estructura de este sistema de fichero es más eficiente si tenemos nodos de E/S separados. En la figura 5.5 se muestran posibles configuraciones.

5.4. Evaluación

Una vez completadas las fases de Caracterización y Configuración se debe probar la aplicación sobre una configuración. De la fase de configuración se pueden obtener más de una configuración inicial y la evaluación se podrá hacer de forma iterativa sobre cada configuración. En la figura 5.6 se muestran los factores que se deben tener en cuenta al momento de hacer la evaluación.

5.4.1. Índices de E/S Paralela

Los indices considerados para la fase de evaluación son:

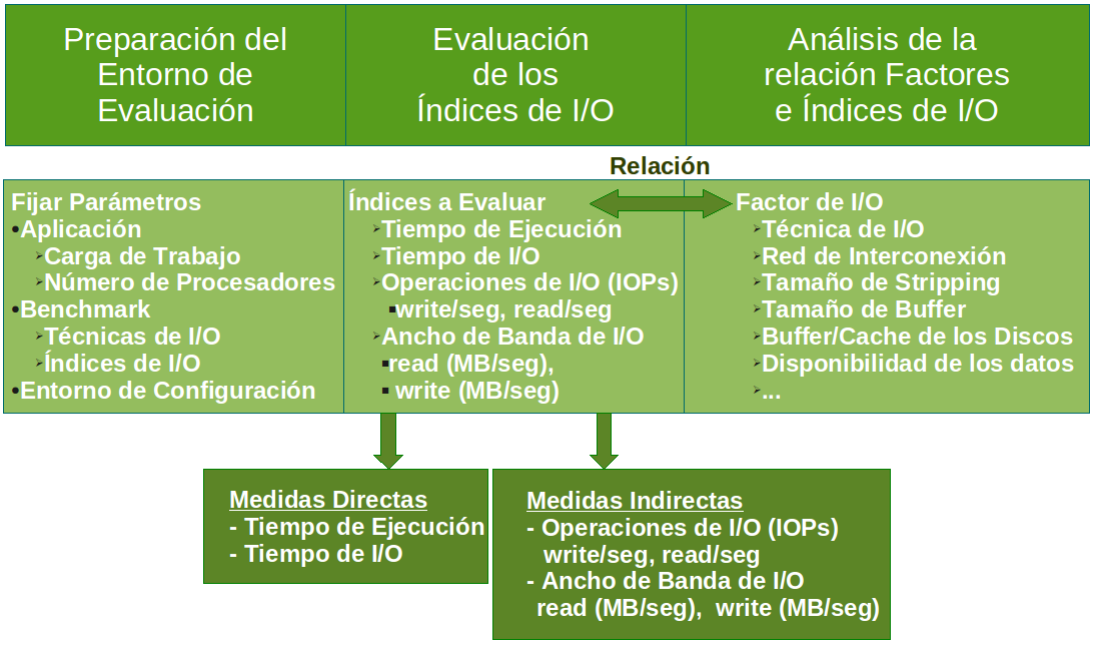


Figura 5.6: Fase de Evaluación de la Metodología de Configuración de E/S

- Tiempo de Ejecución: duración de la ejecución de la aplicación. Unidad de medida *segundos*.

- Tiempo de E/S: es el tiempo que lleva hacer una operación de lectura y de escritura. Éste se mide desde el instante que se invoca la operación de E/S hasta que devuelve los datos leídos o hasta que los datos son escritos. En el caso de que dos procesos iniciaran las operaciones de E/S al mismo tiempo se cuenta el tiempo una vez y la finalización es la del proceso que termina la operación al final. Unidad de medida *segundos*.

- IOPs (Operaciones de E/S por segundo): Cantidad de operaciones de E/S por segundos. Las operaciones que se consideran son: write/seg, read/seg.

- Tasa de transferencia de E/S: cantidad de Megabytes transferidos por segundos en una operación de E/S. Para las operaciones de write y read se mide por operación en forma independientes en MB/seg.

5.4.2. Configuración del Entorno de Evaluación

Una vez caracterizado la aplicación, el sistema, los dispositivos y diseñadas una o varias configuraciones apropiadas para el sistema de E/S. Eo siguiente paso es preparar el entorno de evalu-

ación. La configuración del entorno experimental lo hacemos a nivel de:

- Aplicación Científica: a este se debe establecer la carga de trabajo inicial y aquellas cargas de trabajo que vamos a utilizar durante el proceso de evalución. Fijar los recursos de cómputo iniciales y los necesarios para el procesos de evaluación.
  Generalmente en una evaluación de E/S se trabaja ademas con opciones para manipulación de archivos, paramétros como la librería de E/S de bajo o alto nivel que se usará, opción que se puede definir al momento de compilar la aplicación. También, algunas veces se puede definir si los archivos serán individuales o compartidos por los procesos participantes. Toda estas variantes deben estar registradas al momento de empezar con la evaluación.

- Librerías de E/S: Las librerías de E/S implementan técnicas de E/S para poder hacer la operaciones de forma más eficiente. Por ejemplo,si bien, el estandar para la E/S es MPI-IO se debe tener en cuenta que implementación se va a utilizar en el entorno de evaluación y si la aplicación esta diseñada para esa librería.
  Es conveniente tener instalada en el entorno experimental varías librerías de E/S y usar la que correponda a las necesidades de la aplicación. Ésta es una tarea apropiada en especial cuando una aplicación tiene varias versiones con distintas librerías de E/S. Esta tarea implica que debemos tener registrado con que librerías se esta trabajando y cuales se usarán durante el proceso de evaluación.

- Arquitectura de E/S: El sistema de ficheros paralelo o distribuido a utilizar es una decisión importante que se habrá tomado en la fase de configuración. Este es una de las partes del sistema de E/S que permite ver al sistema paralelo de diferentes formas. A partir del sistema de fichero elegido se decide el rol de los nodos del sistema, que nodos serán de E/S y/o de cómputo y quienes gestionarán los accesos a discos. Además a nivel de sistema de fichero se puede definir el stripping y habilitar buffer/caches.
  La red de interconexión también se la puede definir, a través de archivos de configuración de sistema o de los entornos de paso de mensaje que permiten definir con que direcciones IP se realizará la transferencia de datos. Por lo tanto, en la evaluación se debe registrar que red se está usando para la transferencia de datos y si esta será cambiada durante el procesos de evaluación.

5.4.3. Análisis de la relación factores e índices de E/S

Una vez que la aplicación es ejecutada se debe analizar los resultados obtenidos y relacionarlos con los factores de E/S. Esto es una tarea incremental que debe hacerse por etapas. Elegir factores candidatos para cambiarlos y lanzar nuevamente los experimentos. Analizar si se producen cambios en los tiempos de ejecución de la aplicación. Por ejemplo: una aplicación puede demorar 3hs y cuando se la ejecuta usando colectivas de E/S el tiempo se reduce a la cuarta parte, es claro que las colectivas estan influenciando pero no alcanza con ver solo los tiempos de ejecución, también se debe analizar si el tiempo de E/S se redujo y si la tasa de transferencia e IOPs también muestran que la realmente la técnica de E/S es la que provoco este beneficio. Otro ejemplo es el caso de las redes de E/S, se supone que si tengo una red de E/S dedicada los datos no entrarán en conflicto con los mensajes entre procesos, pero al momento de evaluar los tiempos vemos que los tiempos de ejecución es similar, nuevamente habrá que analizar si a pesar de esto, hay mejoras en los indices de E/S.
Debido a que son muchos los factores que influyen en la prestaciones, al momento de analizar se debe ser muy crítico y concentrarse en aquellos factores de E/S.

5.5. ¿Cómo Configurar la E/S Paralela?

Decidir cuál es la mejor configuración de E/S es una tarea compleja, pero lo que se puede proporcionar es una sugerencia de las posibles configuraciones que logran mejorar las prestaciones y disponibilidad para aplicaciones cientificas de acuerdo a su patrón de acceso, sobre un sistema de E/S configurado de acuerdo a los requisitos de la aplicación y los recursos que el usuario este dispuesto a pagar. La metodología propuesta trata de dar unos pasos necesarios para que la elección de una configuración sea una tarea más ordenada teniendo en cuenta la gran cantidad de componentes que se pueden configurar. Y de hecho la cantidad de iteraciones que se pueda hacer sobre la metodología depende del usuario, ya que es este el que indica los indices de prestaciones, disponibilidad y recursos que desea para su sistema paralelo.

Capítulo 6

Evaluación Experimental

En este capítulo se presenta los experimentos que se realizaron para poder identificar los factores de E/S que influyen en las prestaciones y disponibilidad del sistema de E/S. Los experimentos se realizaron para el Benchmark NAS-BT-I/O que es un kernel de aplicación científica. Para realizar la experimentación se sigue la Metodología de Configuración de E/S propuesta en el capítulo 5. En primer lugar se presenta la descripción del entorno Experimental, después se desarrollan los experimentos para las fases de Caracterización, Configuración y Evaluación.

6.1. Entorno Experimental

Los experimentos se realizaron en el Cluster Aohyper, disponible para el grupo de Arquitectura Paralelas del Departamento de Arquitectura de Computadoras y Sistemas Operativos de la Univesidad Autónoma de Barcelona. En la figura 6.1 se presenta la estructura del cluster Aohyper. Las descripción técnica se presenta en la tabla 6.1

Cuadro 6.1: Características de Aohyper

Front End	**Nodos de Cómputo**	**Software**
Pentium 4.1.8 GHz	Procesador AMD Athlom(tm) 64x2 Dual	Sistema Operativo: Ubuntu Server 8.04
1GB de RAM	2GB de RAM	Librería de Paso de Mensaje: MPICH
1 Disco de 40GB	2 Discos de 80GB	Librería de E/S: ROMIO, HDF5 Parallel, PnetCDF.
2 Tarjetas de Red Fast Ethernet	2 Tarjetas de Red Gibabit	Sistema de Fichero NFS

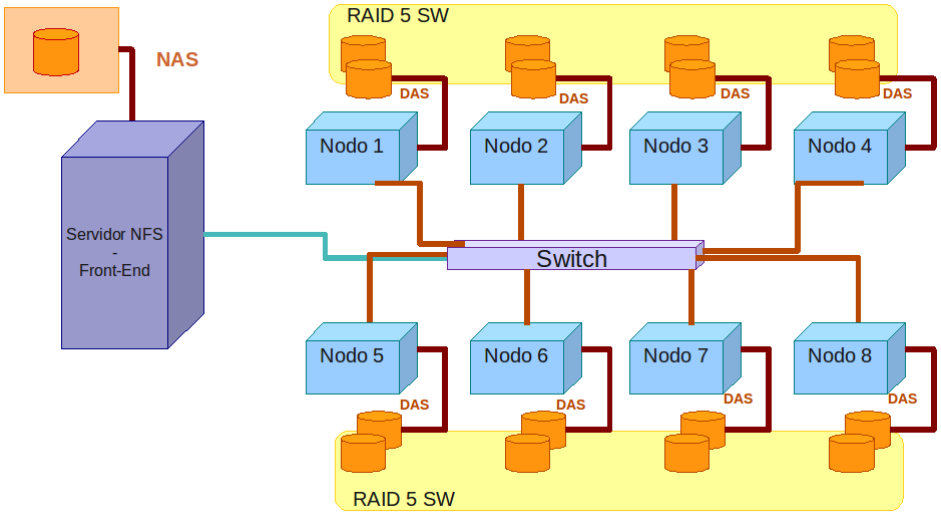


Figura 6.1: Estructura del Cluster Aohyper

6.2. Caracterización

La caracterización se hace a nivel de Aplicación, Sistema Paralelo y Dispositivos de E/S. A continuación se presenta la caracterización del NAS-BT-IO. Después se presentaran la caracterización realizada a nivel de Sistema Paralelo y Dispositivos de E/S.

6.2.1. Caracterización de la Aplicación Científica

Para poder caracterizar a esta aplicación se busco la especificación de particionado de datos y carga de trabajo que proporciona la documentación que acompaña al benchamrk. Pero para ver el patrón de E/S se tomaron trazas del benchamrk. A continuación se presenta los resultados de la caracterización.

NAS-BT-IO

El NAS-BT-IO es un benchamrks de aplicación más usado en el area de E/S para evaluar las prestaciones de un sistema de E/S bajo diferentes configuraciones. En el capítulo 2 se explica con detalle el benchamrk.

En la tabla 6.2 se muestran las características de NAS-BT-IO. En la figura 6.2 se muestra el particionado de datos para 9 procesadores y en la figura 6.3 la traza de la aplicación para 9 procesadores, donde el color violeta indica operaciones de escritura,el verde operaciones de lectura, las lineas de color amarillo muestran la comunicación entre los procesos, el color rojo indica los tiempos de espera de los procesos.

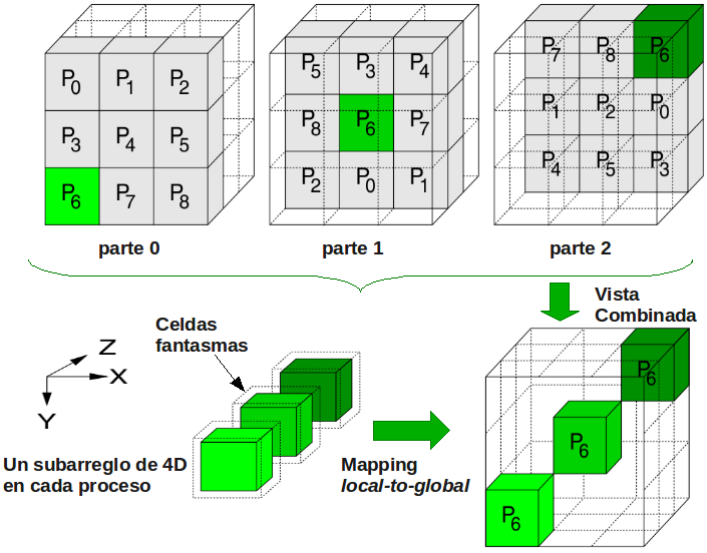


Figura 6.2: Particionado de Datos de NAS BT-E/S

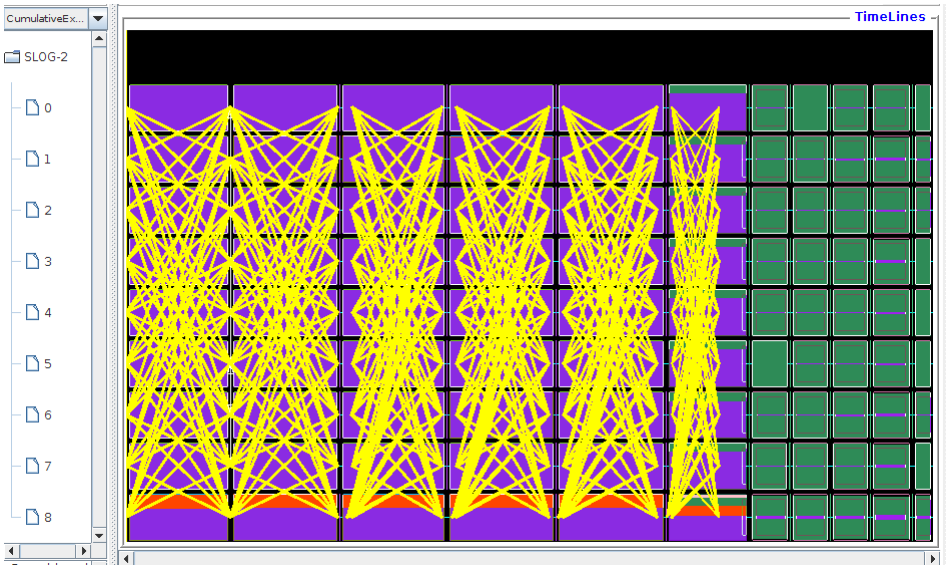


Figura 6.3: Trazas para NAS BT-I/O para 9 procesadores

Cuadro 6.2: Características de NAS BT-I/O

Característica	**Descripción**
Patrón de E/S	Pequeñas solicitudes discontiguas
Carga de Trabajo	Matriz de tres dimensiones de 102x102x102
Librería de E/S	MPI-IO

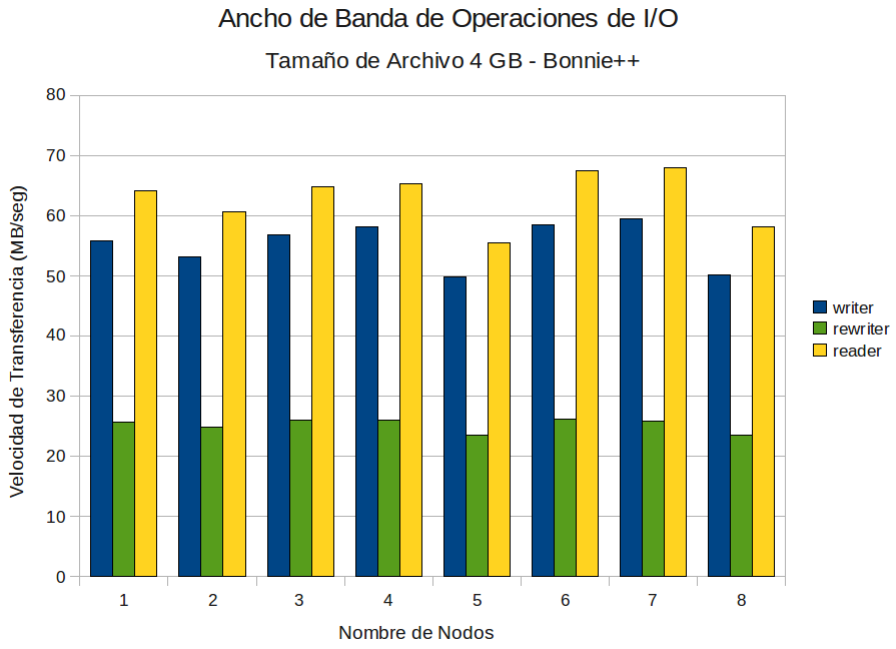


Figura 6.4: Operaciones de E/S medidas con Bonnie++

6.2.2. Sistema Paralelo y Dispositivos de E/S

Se han caracterizado los discos de cada nodo a nivel local con los benchmark Bonnie++ e Iozone (explicados en el capítulo 2). En la figura 6.4 se presentan los valores medios obtenidos con Bonnie++ para un archivo de 4GB de tamaño y en la figura 6.5 los valores obtenidos con iozone. Para evaluar la velocidad de transferencia de los datos por la red junto con el sistema de fichero se realizaron las medidas con los siguientes comandos. Obteniendo los valores medios que se muestran en la tabla 6.3.

- NFS (front-end) from node5

```
smendez@node5:~$ /usr/bin/time dd if=/dev/zero
of=swapfile1 bs=1024 count=524288
524288+0 records in
524288+0 records out
536870912 bytes (537 MB) copied, 64.6859 s, 8.3 MB/s
0.04user 2.01system 1:04.69elapsed 3%CPU
(0avgtext+0avgdata 0maxresident)k
0inputs+1048608outputs (0major+270minor)pagefaults 0swaps
```

Este comando nos muestra que la tasa de transferencia para un archivo de 512 MBytes es de

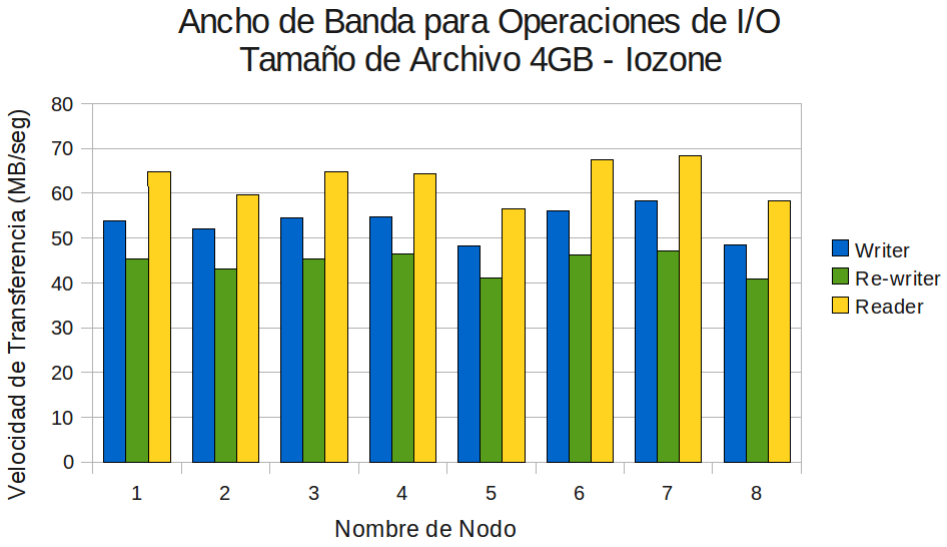


Figura 6.5: Valores para las operaciones de E/S obtenidas por Iozone

8,3MBytes/seg realizada desde el nodo 5 al nodo front-end.

- RAID5+NFS (headnode) from node5

```
smendez@node5:/raid5-nodes1to4/smendez\$ /usr/bin/time dd if=/dev/zero
of=swapfile1 bs=1024 count=524288
524288+0 records in
524288+0 records out
536870912 bytes (537 MB) copied, 115.508 s, 4.6 MB/s
0.10user 2.16system 1:55.55elapsed 1\%CPU
(0avgtext+0avgdata 0maxresident)k
0inputs+1048600outputs (0major+268minor)pagefaults 0swaps
```

Este comando nos muestra que la tasa de transferencia para un archivo de 512 MBytes es de 4,6MBytes/seg realizada desde el nodo 5 al nodo front-end pero en el RAID 5 implementado por software.

- LOCAL at node5

```
smendez@node5:/tmp$ /usr/bin/time dd if=/dev/zero
of=swapfile1 bs=1024 count=524288
524288+0 records in
524288+0 records out
```

Cuadro 6.3: Velocidad de Transferencia en NFS y en la Red de Interconexión

Nodo	**Tasa de Transferencia**
NFS Node Head	7.8 MB/seg
RAID5 + NFS	4.4 MB/seg
RAID 5	9.5 MB/seg

```
536870912 bytes (537 MB) copied, 9.38693 s, 57.2 MB/s
0.06user 4.39system 0:09.38elapsed 47%CPU
(0avgtext+0avgdata 0maxresident)k
32inputs+1048576outputs (0major+270minor)pagefaults 0swaps
```

Este comando nos muestra que la tasa de transferencia para un archivo de 512 MBytes es de 57,2MBytes/seg realizada en forma local.

6.3. Configuración

De acuerdo a los recursos de los que se disponían se definieron las configuraciones de E/S mostradas en la tabla 6.4. Los Factores de E/S configurables para Aohyper identificados son:

- Sistema de Fichero: NFS
- Red de Interconexión: Red de E/S (RIO) y también Red Compartida. Como se disponía de dos placas de Red de 1Gb se realizaron configuraciones con red de E/S exclusiva para datos y otro configuración donde los datos compartían la red de paso de mensajes.
- Disponibilidad: Como se disponían de dos discos rigidos por nodo de procesamiento, se implemento un RAID 5 SW con uno de los discos. De esta forma los datos estaban protegidos frente a fallos de las unidades de disco. El otro disco se dispuso para acceso local.
- Técnica de E/S Además debido a que la aplicación NAS BT-I/O permite configurar el uso de colectivas de E/S para optimizar los accesos a disco. La configuración de E/S incluye el factor librería de E/S.

Las configuraciones que se hicieron usando técnicas de E/S (Colectivas) y sin usar colectivas de E/S por lo tanto se tuvieron 8 configuraciones (6.4).

Cuadro 6.4: Configuraciones de E/S para la Experimentación para un sistema de fichero NFS con DAS

Número	Configuración de E/S
1	RIO+TécnicaDeIO
2	RIO
3	RC+TécnicaDeIO
4	RC
5	RIO+R5SW+TécnicaDeIO
6	RIO+R5SW
7	RC+R5SW+TécnicaDeIO
8	RC+R5SW

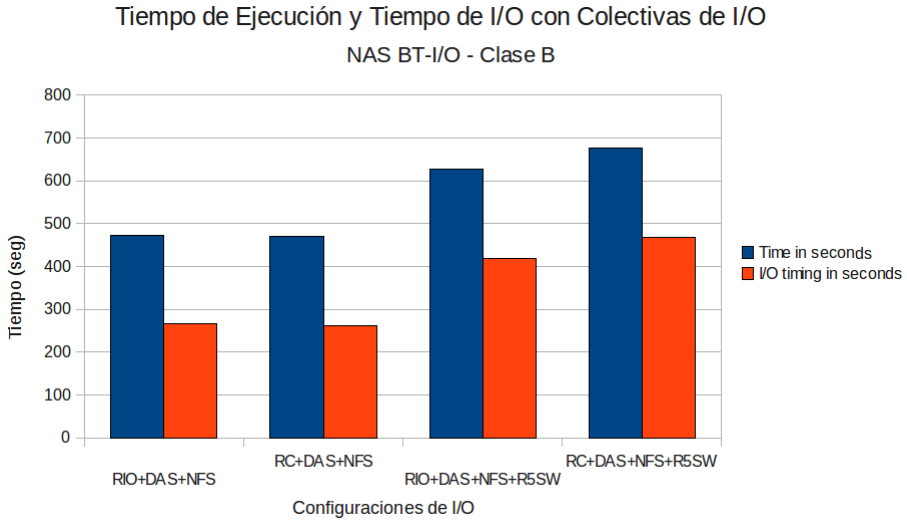


Figura 6.6: Tiempo de Ejecución y Tiempo de IO del NAS BT-I/O utilizando Colectivas de E/S

6.4. Evaluación

En la tabla 6.5 se presentan los valores obtenidos para las diferentes configuraciones para el NAS BT-I/O clase B. En la figura 6.7 se puede claramente como la E/S domina el tiempo de ejecución de la Aplicación. Los resultados muestran que el uso de Colectivas de E/S (Figura 6.6) mejora las prestaciones de forma significativa pasando en los tiempo ejecución para todas las configuraciones del orden de los miles de segundos a cientos de segundos (cerca de los 9000 seg a 200 seg) logrando una reducción del tiempo de E/S del 90 % al 50 %.

También se puede ver que el uso de una red exclusiva de E/S no logra una gran diferencia en las prestaciones pero cuando se incorpora la disponibilidad del nivel 5 de RAID se observa una

Cuadro 6.5: Valores de la medidas realizadas en NAS BT-IO Clase B

Índice	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
I/O timing in seconds	265,28	8869,26	262,41	8740,6	417,81	7269,49	467,52	7551,77
I/O timing percentage	56,12	97,71	55,85	97,67	66,68	97,19	69,22	97,29
Total data written (MB)	1697,93	1697,93	1697,93	1697,93	1697,93	1697,93	1697,93	1697,93
I/O data rate	6,4	0,19	6,47	0,19	4,06	0,23	3,63	0,22
Time in seconds	472,73	9076,79	469,82	8949,01	626,57	7479,36	675,43	7761,79
Write/seg	0,15	23	0,10	23	0,1	28	0,09	28
Read/seg	0,17	1,36	0,16	130	0,16	130	0,16	130
Write	160	832243	160	832305	160	832305	160	832305
Read	159	832299	160	832299	157	832300	157	832299

ganancia en prestaciones con una red de E/S. Pero cuando se utilizan las colectivas de E/S esta ganancia no se logra. Si bien cuando se usan colectivas de E/S la disponibilidad penaliza a las prestaciones estamos protegidos si un componente del almacenamiento fallará.

También se consideró conveniente incluir las medidas realizadas para los IOPs y tasa de transferencia para las operaciones de E/S debido a que esto nos ha permitido entender porque se dan los resultados para las diferentes configuraciones. En el caso de los IOPs (Figura 6.8 y 6.9) cuando no se usan colectivas se realizan muchas operaciones de E/S lo que implica acceder al disco y este acceso penaliza las prestaciones de manera significativa. Sin embargo, con colectivas de E/S, debido a que se juntan las solicitudes de E/S para acceder al disco menos veces, se puede observar se logran reducir los accesos a discos en un 90 % esto es lo que se ve reflejado en los tiempos de ejecución y de E/S además de aumentar la tasa de transferencia (figura 6.10). También analizando estos resultados se puede observar que el sistema de RAID penaliza debido a los bits de paridad que debe generar para poder garantizar la disponibilidad si falla algún componente del RAID, esto se ve en el número mayor de operaciones de escritura que debe hacer.

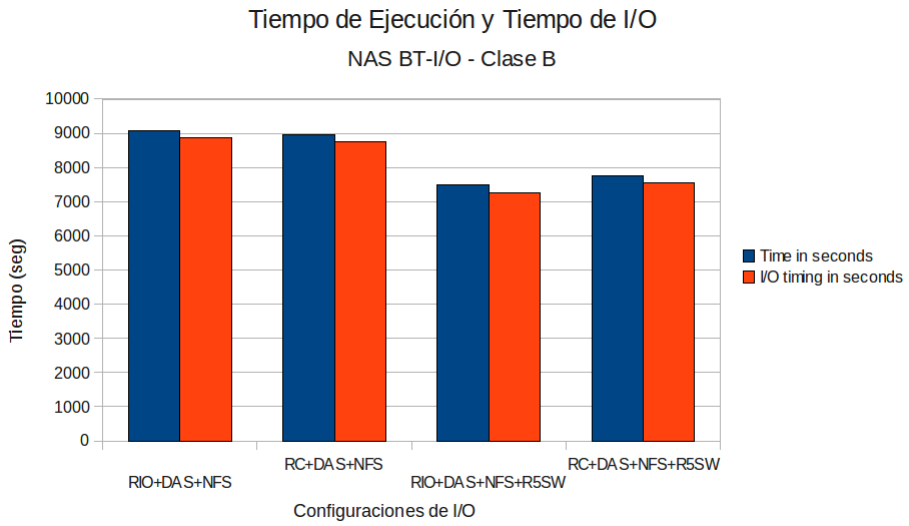


Figura 6.7: Tiempo de Ejecución y Tiempo de E/S para la NAS BT I/O

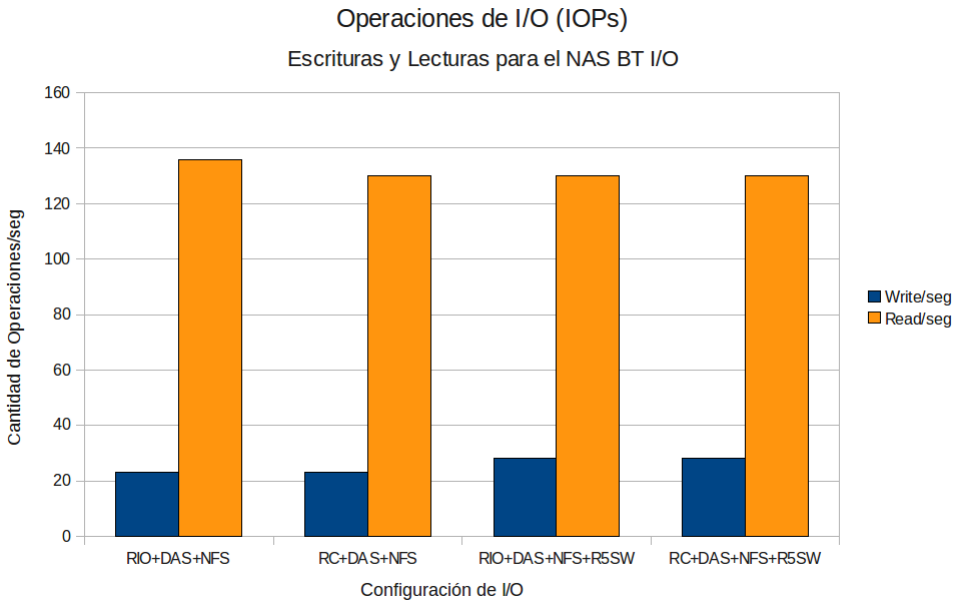


Figura 6.8: Operaciones de E/S para el NAS BT-I/O

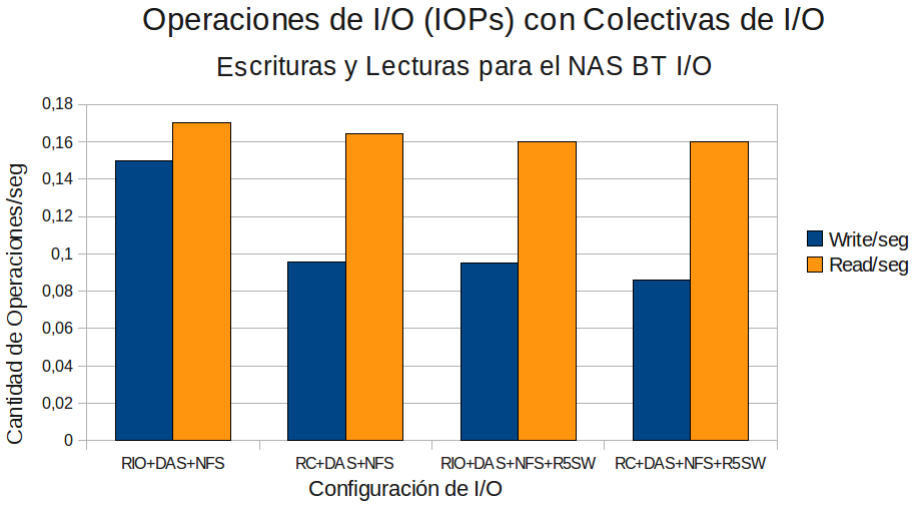


Figura 6.9: Operaciones de E/S para el NAS BT-I/O usando colectivas de E/S

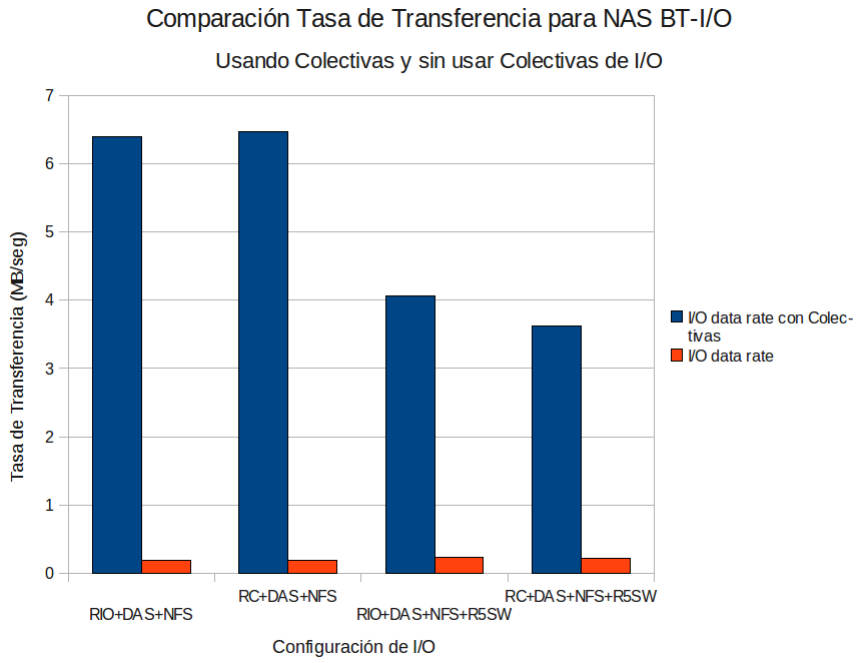


Figura 6.10: Tasa de Transferencia para el NAS BT-I/O con y sin colectivas de E/S

Capítulo 7

Conclusiones y Trabajos Futuros

7.1. Conclusiones

En este trabajo se ha presentado la problemática de la E/S paralela en los sistemas de Cómputo de Altas Prestaciones. Teniendo en mente la investigación creciente en el campo de la E/S y la demandas de los usuarios del HPC se inicia esta nueva línea de investiación en E/S Paralela con este trabajo de iniciación a la investigación. Se estableció como objetivo a corto plazo "Analizar diferentes configuraciones de E/S para identificar los factores de E/S que influyen en la prestaciones y disponibilidad del Sistema de E/S Paralela".

Se ha propuesto una Metodología para la Configuración de E/S Paralela, novel, que ha permitido establecer fases en el análisis de la configuración de E/S paralela y evaluar la influencia de los cambios en los factores de E/S. La misma contempla la caracterización de la aplicación científica, del sistema paralelo (a nivel de operaciones de E/S) y los dispositivos de almacenamiento. La fase de configuración, que brinda una lista amplia de factores de E/S, que se encuentran en la mayoría de las configuraciones de los computadores y que de acuerdo a los recursos con lo que se cuente se podrá elegir que factor variar y configurar. La configuración de la E/S paralela es una tarea compleja que implica la combinación apropiada de la arquitectura de E/S y de las librerías de E/S para una aplicación científica.

Por otro lado, debido a la ausencia de una bibliografía unificada sobre la E/S Paralela, el trabajo se estructuró de manera que permita a los lectores interesados en la E/S Paralela en HPC conocer los conceptos, técnicas, benchmarks y las áreas de mayor interes de investigación en E/S Paralela. En el capítulo 2 se trato de reunir los conceptos más usados en HPC, y en muchos casos se tuvo que optar por determinados trabajos dada la diversidad del material. En el capítulo 3 se presenta una estructura del Sistema de E/S Paralela que se definió en el grupo de investigación y fue en base

a ésta que se planteó todo los estudios de los factores de E/S. Esta estructura es una combinación de varios trabajos en el área de E/S paralela. El capítulo 4 trata de mostrar la evolución de la E/S paralela en los últimos 20 años y encuadrar los objetivos de este trabajo de E/S Paralela.
Si bien existe una tendencia creciente en la investigación de la E/S casí todo se encuentra en papers de diversos congresos tanto específicos de almacenamiento como de HPC en general. Esto hizo difícil la iniciación en este línea de investigación, por eso se considero útil plasmar todo este trabajo inicial en los capítulos 2, 3 y 4 de esta memoria.
La experimentación muestra la aplicación de la metodología y como se logró hacer un análisis ordenado de los factores de E/S. Se eligió uno de los benchmarks más usados en el area de E/S de HPC, NAS BT-I/O, los resultados muestran como las técnicas de E/S a nivel de librería de E/S logran una reducción significativa del tiempo de E/S de la aplicación y como esto reduce el tiempo de ejecución.
Además el uso de una configuración que considere el paralelismo y disponibilidad a nivel de disco, logra unas prestaciones aceptables, si bien la redundancia en la operación de escritura penaliza los tiempos de E/S, este valor es aceptable considerando que si algún disco fallara la información estará disponible.
En cuanto a la red de interconexión, el uso de una red de E/S exclusiva no mostró una ganancia significativa en los tiempos de E/S, pero cuando se usa en una configuración con RAID 5 SW los tiempos son mejores, como un RAID 5 necesita hacer peticiones a todos los discos y generar bits de paridad para lograr la protección y recuperación frenta al fallo de un disco, este tráfico extra aprovecha las dos redes de E/S.
Para la experimentación realizada y bajo la configuración de E/S probadas, los factores que más mostraron influenciar las prestaciones fueron: Técnica de E/S, Red de Interconexión y Servidor de Datos.

Pero es evidente que una variación en la configuración de E/S produce cambios en las prestaciones que percibe el usuario, tales como el tiempo de ejecución, esto se puede ver en la resultados obtenidos, de las ocho configuraciones probadas las prestaciones variaron de configuración a configuración y la variación fue muy notable en algunos casos.

7.2. Trabajos Futuros

Este análisis de diferentes configuraciones de E/S para la identificación de los factores de E/S que influyen en las prestaciones y disponibilidad sirve como base para llegar a largo plazo a definir

un **Modelo de la Configuración de E/S Paralela** de forma que el usuario pueda tener criterios para elegir su configuración de E/S por medio de este modelo.

Del estudio realizado hasta el momento, se deduce que la elaboración de este modelo es una tarea muy compleja debido a que la E/S paralela depende tanto de hardware como del software de E/S y todas las variantes que esto implica. Además para poder definir el modelo se necesita evaluar diferentes configuraciones que se encuentran en el mundo del HPC, lo que es económicamente inviable, por esta razón que se incluirá una Herramienta de Simulación que nos permita modelar la arquitectura de E/S , la librerías de E/S y los patrones de E/S de la aplicación científica.

Se incluirán aplicaciones con otros patrones de E/S para ver como influye la Configuración frente esta variación en el comportamiento de la aplicación. Además se incorporarán nuevos dispositivos de almacenamiento a los entornos de prueba para probar los factores de buffering/caching, stripping, etc. Hay varios factores que no se han podido evaluar, por no disponer con estos en la configuración del Aohyper. Por otro lado, para identificar y evaluar los factores de E/S asociados con la disponibilidad a nivel de datos, se pretende utilizar la Arquitectura Tolerante a Fallos RADIC (Redundant Array of Distributed Independent Fault Tolerance Controllers) [43] a nivel de servidores de datos. RADIC es una arquitectura desarrollada en la UAB por el grupo de Tolerancia a fallos.

De esta forma, con el Modelo de Configuración de E/S Paralela y la Arquitectura RADIC a nivel de sevidores de datos, se pretende cubrir tanto prestaciones como disponibilidad. Esto es un objetivo a largo plazo que espera concluir en 3 años.

Bibliografía

[1] J. M. May, *Parallel I/O for high performance computing*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[2] R. Jain, J. Werth, and J. C. Browne, eds., *Input/Output in Parallel and Distributed Computer Systems*. USA: Kluwer Academic Publishers, 1996.

[3] R. Ross, R. Thakur, and A. Choudhary, "Achievements and challenges for i/o in computational science," *Journal of Physics: Conference Series*, vol. 16, no. 1, p. 501, 2005.

[4] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 109–116, ACM, 1988.

[5] E. L. Miller and R. H. Katz, "Input/output behavior of supercomputing applications," in *Supercomputing '91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, (New York, NY, USA), pp. 567–576, ACM, 1991.

[6] P. Brezany, ed., *Input/Output Intensive Massively Parallel Computing*. Berlin, Heidelberg, New York: Springer-Verlag, 1997.

[7] S. Byna, Y. Chen, X.-H. Sun, R. Thakur, and W. Gropp, "Parallel i/o prefetching using mpi file caching and i/o signatures," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pp. 1 –12, 15-21 2008.

[8] P. Wong and R. F. V. D. Wijngaart, "Nas parallel benchmarks i/o version 2.4," tech. rep., Computer Sciences Corporation, NASA Advanced Supercomputing (NAS) Division, 2003.

[9] A. Nisar, W.-k. Liao, and A. Choudhary, "Scaling parallel i/o performance through i/o delegate and caching system," in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, (Piscataway, NJ, USA), pp. 1–12, IEEE Press, 2008.

[10] S. Lang, P. Carns, R. Latham, R. Ross, K. Harms, and W. Allcock, "I/o performance challenges at leadership scale," in *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, (New York, NY, USA), pp. 1–12, ACM, 2009.

[11] W.-k. Liao and A. Choudhary, "Dynamically adapting file domain partitioning methods for collective i/o based on underlying parallel file system locking protocols," in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, (Piscataway, NJ, USA), pp. 1–12, IEEE Press, 2008.

[12] J. Borrill, L. Oliker, J. Shalf, and H. Shan, "Investigation of leading hpc i/o performance using a scientific-application derived benchmark," in *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, (New York, NY, USA), pp. 1–12, ACM, 2007.

[13] D. Kotz, "Disk-directed i/o for mimd multiprocessors," *ACM Trans. Comput. Syst.*, vol. 15, no. 1, pp. 41–74, 1997.

[14] J. M. del Rosario, R. Bordawekar, and A. Choudhary, "Improved parallel i/o via a two-phase run-time access strategy," *SIGARCH Comput. Archit. News*, vol. 21, no. 5, pp. 31–38, 1993.

[15] K. Coloma, A. Ching, A. Choudhary, W. keng Liao, R. Ross, R. Thakur, and L. Ward, "A new flexible mpi collective i/o implementation," in *Cluster Computing, 2006 IEEE International Conference on*, pp. 1 –10, 25-28 2006.

[16] K. E. Seamons, Y. Chen, P. Jones, J. Jozwiak, and M. Winslett, "Server-directed collective i/o in panda," in *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, (New York, NY, USA), p. 57, ACM, 1995.

[17] R. Thakur, W. Gropp, and E. Lusk, "Data sieving and collective i/o in romio," in *Frontiers of Massively Parallel Computation, 1999. Frontiers '99. The Seventh Symposium on the*, pp. 182 –189, 21-25 1999.

[18] R. Thakur, A. Choudhary, R. Bordawekar, S. More, and S. Kuditipudi, "Passion: Optimized i/o for parallel applications," *Computer*, vol. 29, pp. 70–78, 1996.

[19] J. Li, W. keng Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel netcdf: A high-performance scientific i/o interface," in *Supercomputing, 2003 ACM/IEEE Conference*, pp. 39 – 39, 15-21 2003.

[20] R. Rew and G. Davis, "Netcdf: an interface for scientific data access," *Computer Graphics and Applications, IEEE*, vol. 10, pp. 76 –82, jul 1990.

[21] T. T. of HDF at NCSA, "Hdf5 wins 2002 r&d 100 awards," 2002.

[22] R. Rabenseifner and A. E. Koniges, "Effective file-i/o bandwidth benchmark," in *Euro-Par '00: Proceedings from the 6th International Euro-Par Conference on Parallel Processing*, (London, UK), pp. 1273–1283, Springer-Verlag, 2000.

[23] R. Ross, "Parallel i/o benchmarking consortium - mpi-tile-io," 2001.

[24] D. G. Feitelson, P. F. Corbett, S. Johnson Baylor, and Y. Hsu, "Parallel i/o subsystems in massively parallel supercomputers," *IEEE Parallel Distrib. Technol.*, vol. 3, no. 3, pp. 33–47, 1995.

[25] D. Kotz, "Introduction to multiprocessor i/o architecture," in *Input/Output in Parallel and Distributed Computer Systems, chapter 4*, pp. 97–123, Kluwer Academic Publishers, 1996.

[26] F. Schmuck and R. Haskin, "Gpfs: A shared-disk file system for large computing clusters," in *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, (Berkeley, CA, USA), p. 19, USENIX Association, 2002.

[27] P. J. Braam and Others, "The Lustre storage architecture," *White Paper, Cluster File Systems, Inc., Oct*, vol. 23, 2003.

[28] R. B. Ross and R. Thakur, "Pvfs: A parallel file system for linux clusters," in *In Proceedings of the 4th Annual Linux Showcase and Conference*, pp. 391–430, MIT Press, 2000.

[29] T. Ludwig, "Research trends in high performance parallel input/output for cluster environments," in *Proceedings of the 4th International Scientific and Practical Conference on Programming UkrPROG2004, Pages 274-281, National Academy of Sciences of*, 2004.

[30] P. C. Roth, "Characterizing the i/o behavior of scientific applications on the cray xt," in *PDSW '07: Proceedings of the 2nd international workshop on Petascale data storage*, (New York, NY, USA), pp. 50–55, ACM, 2007.

[31] T. M. Madhyastha and D. A. Reed, "Learning to classify parallel input/output access patterns," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 8, pp. 802–813, 2002.

[32] W. Yu, S. Oral, J. Vetter, and R. Barrett, "Efficiency evaluation of cray xt parallel io stack," 2007.

[33] W. Yu, H. S. Oral, R. S. Canon, J. S. Vetter, and R. Sankaran, "Empirical analysis of a large-scale hierarchical storage system," in *Euro-Par '08: Proceedings of the 14th international Euro-Par conference on Parallel Processing*, (Berlin, Heidelberg), pp. 130–140, Springer-Verlag, 2008.

[34] M. Fahey, J. Larkin, and J. Adams, "I/o performance on a massively parallel cray xt3/xt4," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pp. 1 –12, 14-18 2008.

[35] J. H. Laros, L. Ward, R. Klundt, S. Kelly, J. L. Tomkins, and B. R. Kellogg, "Red storm io performance analysis," in *CLUSTER '07: Proceedings of the 2007 IEEE International Conference on Cluster Computing*, (Washington, DC, USA), pp. 50–57, IEEE Computer Society, 2007.

[36] H. Yu, R. Sahoo, C. Howson, G. Almasi, J. Castanos, M. Gupta, J. Moreira, J. Parker, T. Engelsiepen, R. Ross, R. Thakur, R. Latham, and W. Gropp, "High performance file i/o for the blue gene/l supercomputer," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pp. 187 – 196, 11-15 2006.

[37] S. J. Baylor, C. Benveniste, and Y. Hsu, "Performance evaluation of a massively parallel i/o subsystem," *SIGARCH Comput. Archit. News*, vol. 22, no. 4, pp. 5–10, 1994.

[38] A. Núnez, J. Fernández, J. D. Garcia, L. Prada, and J. Carretero, "Simcan: a simulator framework for computer architectures and storage networks," in *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, (ICST, Brussels, Belgium, Belgium), pp. 1–8, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[39] K. Salem and H. Garcia-Molina, "Disk striping," in *Proceedings of the Second International Conference on Data Engineering*, (Washington, DC, USA), pp. 336–342, IEEE Computer Society, 1986.

[40] M. Livny, S. Khoshafian, and H. Boral, "Multi-disk management algorithms," *SIGMETRICS Perform. Eval. Rev.*, vol. 15, no. 1, pp. 69–77, 1987.

[41] D. A. Patterson, G. A. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," tech. rep., Berkeley, CA, USA, 1987.

[42] A. Choudhary and Y. Ishikawa, "I/o systems. draft the international exascale software project roadmap," tech. rep., www.exascale.org, 2009.

[43] G. Santos, A. Duarte, Dolores, and E. Luque, "Increasing the performability of computer clusters using radic ii," in *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, (Washington, DC, USA), pp. 653–658, IEEE Computer Society, 2008.

[44] H. Jin, *High Performance Mass Storage and Parallel I/O: Technologies and Applications*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[45] J. Li, L. Yan, Z. Gao, and D. Hei, "A task-pool parallel i/o paradigm for an i/o intensive application," in *Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on*, pp. 679 –684, 10-12 2009.

[46] A. Ching, A. Choudhary, W. keng Liao, L. Ward, and N. Pundit, "Evaluating i/o characteristics and methods for storing structured scientific data," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, p. 10 pp., 25-29 2006.

[47] R. Ross, D. Nurmi, A. Cheng, and M. Zingale, "A case study in application i/o on linux clusters," in *Supercomputing '01: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, (New York, NY, USA), pp. 11–11, ACM, 2001.

[48] J. Piernas, J. Nieplocha, and E. J. Felix, "Evaluation of active storage strategies for the lustre parallel file system," in *Supercomputing, 2007. SC '07. Proceedings of the 2007 ACM/IEEE Conference on*, pp. 1 –10, 10-16 2007.

[49] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. Sclatter Ellis, and M. Best, "File-access characteristics of parallel scientific workloads," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 7, pp. 1075 –1089, oct 1996.