CENTRO DE SUPERCOMPUTACIÓN DE GALICIA

Departamento de Aplicaciones y Proyectos

Análisis Técnico de Rendimiento del Finis Terrae

Informe Técnico CESGA-2009-001

Damián Álvarez Mallón[1]
Andrés Gómez Tato[2]
Guillermo López Taboada[3]

[1] dalvarez@cesga.es
[2] agomez@cesga.es
[3] taboada@udc.es

Resumen

El Centro de Supercomputación de Galicia cuenta en su haber con el supercomputador Finis Terrae desde Noviembre de 2007, fecha en la cual se situó en el puesto 100 del top500. Este sistema supone una valiosa herramienta para la comunidad científica, y maximizar su rendimiento es una prioridad alcanzable mediante el conocimiento de su comportamiento en diversas situaciones. El Finis Terrae, al haber sido instalado recientemente, no ha sido evaluado en profundidad, raíz de este trabajo.

Por su arquitectura basada en celdas, en el Finis Terrae tiene influencia en las comunicaciones entre nodos el procesador en el que se está ejecutando la aplicación. Para cuantificar esta influencia se han realizado pruebas con varias configuraciones, resultando que se puede obtener una mejora de rendimiento del 5 % ajustando la afinidad de los procesos.

También se ha evaluado la escalabilidad de operaciones colectivas, involucrando hasta a 1024 procesos, de forma que ahora se conoce la degradación que producirá el uso de estas operaciones en aplicaciones con un gran número de procesos.

Cabe salientar en este punto que las pruebas se han realizado con HP-MPI 2.2.5 e Intel MPI 3.1. Recientemente se han instalado en el Finis Terrae las versiones 2.3 de HP-MPI y 3.1.038 de Intel MPI, por lo que los números mostrados en este informe pueden variar ligeramente respecto al rendimiento actual.

Con un gran número de núcleos por nodo, la programación híbrida MPI+OpenMP puede resultar una opción atractiva para obtener el máximo rendimiento. Los resultados obtenidos usando benchmarks de programación híbrida muestran que es habitual que el uso de únicamente procesos MPI sea el que ofrezca mejor rendimiento. Sin embargo, en problemas de gran tamaño y con muchos procesos se puede dar la situación en la que prime el uso de OpenMP. Con estos datos se pueden estimar las configuraciones óptimas de las aplicaciones.

UPC se perfila como un lenguaje interesante para la programación de este tipo de plataformas, y tiene gran interés para el CESGA, por lo que la cuantificación de las comunicaciones viene motivada también para aplicar el conocimiento adquirido a UPC. Los resultados alcanzados muestran que su rendimiento con el entorno evaluado es inferior al de MPI y OpenMP. No obstante, dada su riqueza semántica para la programación paralela, es una opción interesante para la programación del Finis Terrae. También se ha evaluado Java, que ha mostrado un pobre rendimiento pero una buena escalabilidad para tamaños de problema grandes.

En base a los resultados observados se recomienda el uso de afinidad a la celda 0 (procesadores 8-15), por ser la que dispone de la tarjeta Infiniband, cuando se vaya

a hacer uso de esta red y **se disponga del nodo de computación en exclusiva**, así como elegir la biblioteca de comunicaciones en base al perfil de la aplicación y las observaciones acerca del rendimiento de cada biblioteca realizadas anteriormente. Así mismo, el uso de programación híbrida queda condicionado al perfil de la aplicación. Por otra parte también resulta recomendable el uso de UPC, por su proyección futura en esta plataforma y su facilidad de programación.

El presente trabajo ha constituído parte del proyecto de Fin de Máster presentado en Septiembre de 2008 en la Facultad de Informática de la Universidad de Coruña titulado "Análisis Técnica de Rendimiento de Sistemas HPC", que ha obtenido la calificación de Matrícula de Honor.

Palabras clave

Finis Terrae, UPC, MPI, OpenMP, supercomputación, Infiniband, clúster, paralelismo, rendimiento, benchmark, multi-core, thread, Java.

Índice general

Índice de figuras

1 Introducción

Recientemente ha sido instalado en el Centro de Supercomputación de Galicia (CESGA) [1] el supercomputador Finis Terrae [2], compuesto por 2400 núcleos y 19584 GB de memoria. Dicho supercomputador es el de mayor memoria por núcleo de Europa, y en el momento de su instalación se situó en el puesto 100 de los supercomputadores más potentes del mundo según la lista de top500 de Noviembre de 2007[3, 4].

Dadas las exigencias técnicas de la computación de altas prestaciones es necesario obtener el mayor partido posible de estos sistemas. Para ello es necesario conocer detenidamente su comportamiento ante diversas situaciones, así como los motivos que se esconden detrás de las respuestas observadas.

El Finis Terrae cuenta con 142 nodos, cada uno de ellos con una potencia considerable al estar compuestos por 16 núcleos Itanium 2 y 128 Gigabytes de RAM, interconectados por una red Infiniband de altas prestaciones [5], de hasta 16 Gbps y 5,88 μs de latencia. Además cuenta con un nodo *Superdome* con 128 nucleos y 1024 Gigabytes de RAM, y en fechas próximas se integrarán los 2 nodos *Superdome* que se encontraban en el CESGA antes de la instalación del Finis Terrae. La peculiar arquitectura del Finis Terrae, con nodos muy potentes, se adapta perfectamente a un modelo de programación híbrido, usando paralelización en memoria compartida con OpenMP [6] y paralelización con paso de mensajes con MPI[1] [7]. Resulta de sumo interés medir empíricamente qué combinaciones de estos dos paradigmas resultan más eficientes.

Otro modelo de programación en auge que se adapta a la arquitectura del Finis Terrae es el de los lenguajes PGAS[2], y por tanto también resulta de interés una evaluación del rendimiento de las aplicaciones programadas con dicho paradigma, en concreto en el lenguaje UPC[3] [8].

Por último, también resulta interesante la evaluación de Java, al ser este un lenguaje en boga en la computación de altas prestaciones, con soporte intrínseco multihilo y bibliotecas de comunicaciones de calidad.

Sentadas estas bases se propone una evaluación del rendimiento ante diferentes cargas de trabajo, para observar la respuesta del sistema, especialmente de sus comunicaciones, y tratar de optimizar su uso, prestando especial atención a los tiempos de acceso a la red de interconexión desde diferentes procesadores y a las configuraciones de programación híbrida que obtienen mejor rendimiento.

[1] MPI: Message Passing Interface
[2] PGAS: Partitioned Global Address Space
[3] UPC: Unified Parallel C

2 Evaluaciones Previas de Rendimiento en Supercomputadores

La evaluación de rendimiento se torna en una herramienta imprescindible para la optimización del rendimiento en un centro de supercomputación. Las características de cada supercomputador se pueden consultar en la documentación del fabricante, pero el comportamiento real de cada máquina varía según la carga de trabajo que tenga que soportar, por lo que es necesario un análisis concienzudo. Además, diversas aplicaciones deberán parametrizarse correctamente para aprovechar la arquitectura de los computadores del centro, y estos parámetros sólo se podrán usar cuando se tenga un profundo conocimiento del comportamiento de los diversos subsistemas.

En todos los centros de supercomputación se realiza una evaluación técnica de rendimiento para los supercomputadores instalados. Estos análisis se integran en la documentación interna del centro, constituyendo una valiosa fuente de información del rendimiento de los computadores instalados. Sin embargo, no todos los centros de supercomputación hacen públicos sus análisis. En este ámbito destacan los análisis del *Oak Ridge National Laboratory* [9] –de ahora en adelante ORNL– , un laboratorio dependiente del departamento de energía de los Estados Unidos, disponibles públicamente y cuyo número, calidad y profundidad constituyen referente. También son salientables los análisis de la *NASA Advanced Supercomputing Division* [10], con sede en el *NASA Ames Research Center*, cuyos ingenieros son autores de la suite de benchmarking paralela por excelencia, los *NAS Parallel Benchmarks* [11, 12].

En el CESGA también se han realizado diversas evaluaciones del rendimiento de los supercomputadores que han pasado por su haber. De entre ellos destaca el análisis efectuado sobre el SVG[1] [13] analizando el comportamiento de varias arquitecturas multi-core integradas en dicho sistema.

En las siguientes secciones y subsecciones se analizarán las principales evaluaciones realizadas en el ORNL, en la NAS Division, y en el CESGA.

2.1. Oak Ridge National Laboratory

El ORNL es un centro de investigación dependiente del departamento de energía del gobierno de los Estados Unidos. Sus áreas de investigación incluyen sistemas biológicos, nuevos materiales, energía y supercomputación. Tiene una gran trayectoria y una

[1] SVG: Supercomputador Virtual Gallego

dilatada experiencia en el uso de supercomputadores de diferentes vendedores, arquitecturas y topologías de red. Cuenta en su haber con varios sistemas Cray, SGI, NEC e IBM. Los análisis de rendimiento de los supercomputadores del ORNL son un referente para la comunidad de evaluadores a nivel mundial, por su profundidad, calidad y facilidad de acceso, ya que están disponibles públicamente. A continuación se expondrán brevemente sus análisis más recientes.

2.1.1. Cray X1

La evaluación del Cray X1 en el ORNL [14], sistema que ha sido actualizado a un Cray X1E y que ocupa el puesto 175 del top500 de Junio de 2008 [15], representa el análisis técnico de rendimiento disponible públicamente más extenso hasta la fecha. En él han participado 29 investigadores de los siguientes centros:

- Oak Ridge National Laboratory.
- Pacific Northwest National Laboratory.
- Argonne National Laboratory.
- Auburn University.
- University of Tennessee.
- Lawrence Berkeley National Laboratory.
- Cray Inc.

Este estudio se ha estructurado en seis secciones, a lo largo de 37 páginas:

1. Implementación y plan de evaluación.
2. Visión general de la evaluación.
3. Descripción del Cray X1.
4. Componentes de la evaluación
5. Actividades relacionadas.
6. Publicaciones resultantes.

De entre estas seis secciones resulta especialmente interesante la sección “Componentes de la evaluación”. En ella tiene lugar la evaluación propiamente dicha, estructurada en distintas subsecciones fruto de la evaluación jerárquica descrita en la metodología a seguir, explicada en el punto 2 de dicho estudio.

La evaluación comienza por una serie de microbenchmarks destinados a la evaluación de subsistemas del computador muy concretos. Los subsistemas evaluados son:

1. Rendimiento de las unidades de cálculo (escalares y vectoriales).

2. Rendimiento de la jerarquía de memoria.
3. Rendimiento de operaciones con threads.
4. Rendimiento de primitivas de paso de mensajes.
5. Rendimiento del sistema y de la entrada/salida.

Para el microbenchmarking se han usado códigos propios, así como los benchmarks STREAM[2] [16], para medir el ancho de banda de acceso a memoria, y HALO [17], para medir la latencia de las comunicaciones.

El siguiente paso lógico es la evaluación de kernels computacionales. En esta evaluación se han usado diferentes benchmarks estándar –el kernel MG de los NAS Parallel Benchmarks [11], ParkBench [18], EuroBen [19]–, así como kernels extraídos de aplicaciones propias, principalmente el PSTSWM[3], de un modelo de simulación climática global, y kernels de la suite HPC Challenge [20, 21] destinados a comprobar el comportamiento del Cray X1 en operaciones con matrices densas y dispersas usando las rutinas BLAS[4] [22] de la biblioteca de cálculo científico de Cray (Cray Scientific Library). Se ha evaluado el rendimiento del Cray X1 frente a otros supercomputadores, y se han probado varios paradigmas de programación (MPI, memoria compartida y Co-Array Fortran) y distintas configuraciones hardware y software. Además, se ha manifestado el deseo de hacer una evaluación de OpenMP, UPC y paralelismo multinivel.

El último paso de la evaluación ha sido el análisis del rendimiento de las aplicaciones. Dada la variedad de la investigación realizada en el ORNL, se han evaluado un gran número de aplicaciones de distintos campos:

Simulación del clima: Para la simulación del clima en todo el globo se usa el *Community Climate System Model* (CCSM) [23], un software altamente paralelo dividido en cuatro componentes básicos: el *Community Atmosphere Model* (CAM), el *Community Land Model* (CLM), el *Parallel Ocean Program* (POP), el *Los Alamos Sea Ice Model* (CICE) y un quinto componente llamado *Coupler* (CPL) que permite interactuar a los cuatro componentes básicos.

Simulación de fusión nuclear: En la investigación acerca de la fusión nuclear se usa principalmente GYRO [24], una aplicación para la simulación numérica de microturbulencias del plasma en un tokamak. Otra aplicación para una simulación de fusión más general es NIMROD[5] [25]. También se ha evaluado AORSA3D[6], una aplicación para la simulación del calentamiento de plasma usando ondas de radio frecuencia.

Investigación en materiales: En este campo se usa modelos DCA-QMC[7] para la simulación de superconductores. También se usa software que implementa el méto-

[2]STREAM: Sustainable Memory Bandwidth in High Performance Computers
[3]PSTSWM: Parallel Spectral Transform Shallow-Water Model
[4]BLAS: Basic Linear Algebra Subprograms
[5]NIMROD: Non-Ideal Magnetohydrodynamics with Rotation, Open Discussion
[6]AORSA3D: All Orders Spectral Algorithm 3D
[7]DCA-QMC: Dynamic Cluster Approximation Quantum-Monte-Carlo

do LSMS[8] [26].

Química computacional: Para la investigación sobre reacciones químicas se utiliza NWChem [27], un paquete software que se apoya sobre el toolkit *Global Arrays* [28], que a su vez se apoya sobre la biblioteca de acceso a memoria remota ARMCI[9] [29]. Otra aplicación muy usada en este campo es GAMESS[10] [30], y por tanto también se ha usado en esta evaluación.

Astrofísica: En investigación astrofísica se usa AGILE-BOLTZTRAN, un código de simulación que modela el colapso de supernovas.

Biología: En el ámbito de la investigación biológica se usan dos paquetes software de modelización de sistemas biológicos a nivel atómico usando simulaciones de dinámica molecular. Son AMBER[11] [31] y NAMD[12] [32].

Dinámica de fluidos: Para la investigación en dinámica de fluidos se ha usado el benchmark sPPM [33], benchmark que modela el comportamiento de un problema de dinámica de gases en un espacio 3D usando el *Piecewise Parabolic Method*.

Además, también se han evaluado otros benchmarks de la suite HPC Challenge [20, 21] y códigos propios de física atómica y molecular.

Cabe resaltar que muchos de los resultados de este estudio se encuentran publicados en distintos artículos [34, 35], siendo éste un resumen.

2.1.2. Cray XD1

El ORNL cuenta en su haber con un supercomputador Cray XD1, que ha sido evaluado convenientemente [36]. Dicha evaluación sigue una estructura lógica similar a la descrita en el apartado anterior, comenzando con una descripción del sistema Cray XD1, siguiendo con una explicación de la metodología a seguir y las máquinas a utilizar comparativamente, para posteriormente exponer los resultados alcanzados en el benchmarking y sus correspondientes conclusiones. El benchmarking, al igual que en la evaluación del Cray X1, se ha organizado de forma jerárquica, partiendo de un benchmarking a bajo nivel, para seguir con kernels computacionales y posteriormente con aplicaciones.

Así, para el microbenchmarking se ha evaluado el rendimiento de kernels de la suite HPC Challenge [20, 21] que realizan multiplicación de matrices con la biblioteca BLAS [22] de AMD (ACML[13] [37]), se ha usado el benchmark STREAM [16], para comprobar la capacidad de acceso a memoria, y el benchmark CacheBench [38], para analizar el rendimiento de la jerarquía de memoria. Para el microbenchmarking de comunicaciones se ha optado por HALO [17] y los PMB[14] –ahora conocidos como IMB[15]– [39].

[8] LSMS: Locally Self-Consistent Multiple Scattering
[9] ARMCI: Aggregate Remote Memory Copy Interface
[10] GAMESS: General Atomic & Molecular Electronic Structure System
[11] AMBER: Assisted Model Building with Energy Refinement
[12] NAMD: NAnoscale Molecular Dynamics
[13] ACML: AMD Core Math Library
[14] PMB: Pallas MPI Benchmarks
[15] IMB: Intel MPI Benchmarks

Los kernels evaluados han sido dos: PSTSWM y SMG2000[16] [40], un código usado en modelos de difusión de radiación y flujo en medios porosos.

En esta evaluación se ha puesto especial énfasis en el análisis del rendimiento de aplicaciones. Así, las elegidas han sido POP y CAM, extraídas ambas del CCSM [23], GYRO [24], sPPM [33] y VASP[17] [41], un software de simulación de dinámica molecular usando principios básicos de mecánica cuántica.

2.1.3. Cray XT3

Otro de los sistemas evaluados en el ORNL es el Cray XT3 [42], sistema que ocupa el puesto número 5 en la lista del top500 de Junio de 2008 [43], después de una ampliación usando nodos Cray XT4. Esta evaluación sigue el mismo esquema que la anterior, comenzando con una descripción del sistema, una visión general de la evaluación, microbenchmarking y benchmarking de aplicaciones.

Así, se han utilizado STREAM [16] y CacheBench [38] para evaluar la jerarquía de memoria, la biblioteca matemática del fabricante (ACML [37]) para evaluar la capacidad de cómputo aritmético frente a otros sistemas usando kernels de la suite HPC Challenge [20, 21], y los *Intel MPI Benchmarks* para la evaluación de las comunicaciones. También se ha utilizado el benchmark FFTW [44] en esta evaluación.

Las aplicaciones elegidas han sido CAM y POP, del CCSM [23], GYRO [24], AMBER [31], AORSA y tres aplicaciones no vistas hasta este punto:

S3D: Un código destinado a calcular interacciones químicas y turbulencias durante la combustión de diversas sustancias.

LAMMPS[18]: Simulación de dinámica molecular en sistemas biológicos [45].

PFLOTRAN[19]: Prototipo para la simulación de corrientes en fluidos [46].

Este estudio se complementa con un estudio anterior [47] en el que se evaluó, además de algunos puntos vistos en este análisis, el rendimiento de una biblioteca BLAS alternativa para procesadores AMD Opteron [48], los benchmarks EuroBen [19], los kernels PSTSWM y SMG2000 [40], y la aplicación sPPM [33].

2.1.4. Otros

El ORNL también ha realizado evaluaciones de sistemas Cray XT4 [49] (puesto 5 del top500 al fusionarse con el sistema Cray XT3 [43]), IBM Blue Gene/P [50] (puesto 74 del top500 [51]) y SGI Altix 3700 [52], pero no se detallarán en este trabajo, ya que desde el punto de vista metodológico no aportan nada nuevo respecto a las evaluaciones aquí mencionadas. Además, el ORNL ha evaluado comparativamente las comunicaciones de los sistemas Cray X1, Cray XD1 y Cray XT3 [53]

[16] SMG2000: Semicoarsening MultiGrid 2000
[17] VASP: Vienna Ab-initio Simulation Package
[18] LAMMPS: Large-scale Atomic/Molecular Massively Parallel Simulator
[19] PFLOTRAN: Parallel FLOw and TRANsport

2.2. NASA Advanced Supercomputing Division

La NASA Advanced Supercomputing Division es un departamento de la NASA dedicado a la investigación en supercomputación. Su sede es el *Ames Research Center* (ARC), que constituye uno de los centros de referencia en lo que a supercomputación se trata. Sus análisis de rendimiento disponibles públicamente no son tan numerosos como los del ORNL, pero los desarrolladores de la suite de benchmarking paralela por excelencia, los NAS Parallel Benchmarks [54, 11], forman parte de dicha división, hecho que abala por si solo su conocimiento de la computación de altas prestaciones. En las siguientes subsecciones se examinarán algunas de sus evaluaciones.

2.2.1. Cray X1

La evaluación del sistema Cray X1 por parte del ARC [55] es similar a los análisis vistos del ORNL. De esta forma, se dividido el análisis en una explicación de la arquitectura del Cray X1 y del sistema instalado en el ARC, y se han expuesto los resultados divididos en microbenchmarking, kernels y aplicaciones.

Para el microbenchmarking y la evaluación de kernels se han usado los HPC Challenge [20, 21] y los NAS Parallel Benchmarks [54, 11] respectivamente. En la evaluación de los NAS Parallel Benchmarks se han usado sólo los códigos más relevantes para el ARC, que son tanto kernels como pseudoaplicaciones, a pesar de que en este estudio se le de el tratamiento genérico de kernels:

MG: El *kernel* MG —*MultiGrid*— resuelve una ecuación diferencial parcial 3D de Poisson. El problema se simplificó añadiendo coeficientes constantes en vez de variables. Proporciona una buena medida del rendimiento de comunicaciones estructuradas.

FT: En el *kernel* FT —*Fourier Transform*— se resuelve una ecuación diferencial parcial 3D usando FFTs —*Fast Fourier Transforms*—. Es un buen test para medir el rendimiento de la comunicación entre nodos no vecinos.

LU: La pseudoaplicación LU —*Lower-Upper diagonal*—, al igual que SP y BT, resuelve una versión discretizada de ecuaciones de Navier-Stokes en un espacio tridimensional. Emplea un método de sobre-relajación y divide la matriz en dos mitades, usando la diagonal como divisor. Además anteriormente existía otra versión llamada LU-HP —*Lower-Upper HyperPlane*— que fue eliminada recientemente, a pesar de haber sido usada en este estudio.

SP: La pseudoaplicación SP —*Scalar Pentadiagonal*—, al igual que la anterior, resuelve una versión discretizada de las ecuaciones de Navier-Stokes en tres dimensiones. La diferencia con LU y BT es la forma de dividir el problema, que en este caso usa el algoritmo *Beam and Warming*.

BT: La pseudoaplicación BT —*Block Tridiagonal*— comparte características con las dos anteriores, factorizando la matriz como producto de tres operandos, uno por cada dimensión, a efectos de resolver el problema.

Las aplicaciones usadas en este análisis son las siguientes:

OVERFLOW: Es una aplicación de dinámica de fluidos destinada a computar el flujo aerodinámico alrededor de cuerpos complejos [56].

ROTOR: Software de simulación del comportamiento de un flujo inestable de gas en una turbina.

INS3D: Código que resuelve las ecuaciones de Navier-Stokes para fluidos en estado estable e inestable.

GCEM3D[20]: Aplicación dedicada a modelar el comportamiento de nubes.

2.2.2. Columbia (SGI Altix 3700/4700)

El ARC también ha evaluado su supercomputador Columbia (posición número 25 del top500 en la lista de Junio de 2008 [57]), pero haciendo una distinción entre sistemas SGI Altix 3700 y SGI Altix 4700 [58].

Así, han empleado los benchmark HPC Challenge [20, 21] y las siguientes aplicaciones científicas:

OVERFLOW-2: Es, como su nombre indica, una versión actualizada de la aplicación OVERFLOW [56].

Cart3D: Aplicación para el diseño aerodinámico preliminar de estructuras en medios no viscosos [59].

USM3D: Código encargado de resolver las ecuaciones de dinámica de fluidos de Euler y Navier-Stokes en un volumen finito [60].

ECCO[21]: Es un software de simulación oceánica que usa una aproximación hidroestática [61].

Además, también se ha evaluado la aplicación NAMD que ya se ha visto anteriormente [32].

2.2.3. Otros

La NAS Division del ARC tiene en su haber más evaluaciones interesantes como un estudio del rendimiento de las redes de interconexión en diversos supercomputadores [62], un estudio de escalabilidad del Columbia [63], una evaluación del rendimiento de combinaciones de distintos niveles de paralelismo usando una pseudo aplicación de los NPB [64], o una evaluación de distintos sistemas usando la suite HPC Challenge y los Intel MPI Benchmarks [65]. Dado que, a pesar de tener distintos enfoques, no aportan nada desde el punto de visto metodológico, no se examinarán en profundidad.

[20] GCEM: Goddard Cumulus Ensemble Model
[21] ECCO: Estimating the Circulation and Climate of the Ocean

2.3. CESGA

El Centro de Supercomputación de Galicia constituye una Instalación Científico-Tecnológica Singular del Estado. En su seno se encuentran instalados varios supercomputadores, entre los cuales destaca el Finis Terrae, un supercomputador formado por 2400 núcleos Itanium 2 y 19 Terabytes de memoria RAM. Este supercomputador figura en el puesto 209 [4] de los 500 supercomputadores más potentes del mundo [3], y es el supercomputador con más memoria por núcleo de Europa, característica que le otorga una especial relevancia más allá de la capacidad bruta de cálculo. Se habla en estos términos de *capability computing*, ya que este supercomputador es capaz de afrontar problemas que otros supercomputadores con mayor número de núcleos –pero menor memoria por núcleo– afrontarían con mayores dificultades. El CESGA cuenta con numerosas evaluaciones de rendimiento de sus sistemas. Sin embargo, carece de un análisis del Finis Terrae, análisis que se torna imperativo. A continuación se analizará brevemente la evaluación de nodos *single*, *dual* y *quadcore* del CESGA, nodos que se integran actualmente en el SVG.

2.3.1. Arquitecturas single, dual y quadcore

La evaluación de rendimiento más reciente realizada en el CESGA es una evaluación acerca del rendimiento de distintos nodos *single*, *dual* y *quadcore* [66]. Dicha evaluación sigue un esquema ligeramente distinto al mostrado por las evaluaciones anteriores, ya que parte de una explicación de los benchmarks usados, para dar lugar a una explicación de los sistemas y posteriormente mostrar los resultados y las conclusiones.

Los benchmarks han sido divididos en benchmarks sintéticos, equivalentes a los microbenchmarks y a kernels de las evaluaciones del ORNL y el ARC vistas, y no sintéticos o aplicaciones. Los benchmarks sintéticos elegidos han sido los HPC Challenge y dos más que no han sido vistos hasta ahora:

LINPACK: A pesar de que existe una versión del benchmark LINPACK en los benchmarks HPC Challenge [20, 21], se han realizado pruebas con una versión ajustado por Intel para una ejecución óptima en sus procesadores Xeon [67]. Dicho benchmark computa la resolución de sistemas de ecuaciones, y su resultado es el baremo utilizado en la clasificación de la lista del top500.

SparseBench: Benchmark que evalúa la resolución de sistemas de ecuaciones que dan lugar a matrices dispersas [68], usando diferentes métodos numéricos y de almacenamiento de matrices.

Por otro lado, los benchmarks no sintéticos elegidos han sido:

Montecarlo: Código desarrollado en la Universidad de Vigo que aplica métodos de simulación de variables aleatorias (de Montecarlo) a cadenas moleculares para calcular propiedades derivadas de hidrocarburos de alta presión.

Gaussian03: Software que permite el cálculo de propiedades de moléculas [69].

GROMACS[22]: Paquete software de simulación de dinámica molecular [70].

También se ha utilizado AMBER. En una evaluación anterior del sistema SVG [71] se han utilizado los benchmarks LINPACK, Pallas MPI Benchmarks (ahora conocidos como Intel MPI Benchmarks), y la aplicación de dinámica molecular CPMD [72].

2.4. Consideraciones de las Evaluaciones Previas

A lo largo de este apartado se ha visto como en las diversas evaluaciones se han usado distintos benchmarks adaptados a las necesidades del centro que ha realizado la evaluación. Esta situación está especialmente marcada a la hora de la elección de las aplicaciones a evaluar, ya que tienen una variación considerable de un centro a otro, pero que sin embargo en diferentes evaluaciones realizadas por el mismo centro se mantienen casi constante. Esto se resume en la idea de evaluar los benchmarks importantes para el centro y la arquitectura de la máquina.

[22] GROMACS: GROningen MAchine for Chemical Simulations

3 Análisis de Necesidades

Como se ha visto al final del apartado anterior, las evaluaciones realizadas sobre los supercomputadores responden a necesidades específicas del centro evaluador y a las características arquitecturales de la máquina. En este apartado se expondrá la arquitectura del Finis Terrae desde una perspectiva general para, posteriormente, analizar con detalle las características de los nodos de computación. Una vez explicada la arquitectura del Finis Terrae se expondrán las necesidades específicas de evaluación derivadas de ésta.

3.1. Arquitectura del Finis Terrae

El Finis Terrae es un cluster formado por 142 nodos HP Integrity rx7640 y 1 nodo HP Integrity Superdome –que no debe ser confundido con HP Superdome, un sistema similar pero con procesadores PA-RISC–. Tiene además 2 nodos de login HP Proliant DL580 G4, que actúan a su vez como servidores de ficheros, 2 nodos de desarrollo y 1 nodo de gestión HP Integrity rx3600, y 2 robots de cintas HP StorageWorks ESL 712e, cada uno con una capacidad máxima de 1 Petabyte y una tasa de transferencia que alcanza los 6,91 Terabytes/hora. También dispone de 1 sistema de almacenamiento paralelo de alto rendimiento HP SFS[1] accedido a través de la red Infiniband y basado en Lustre [73], una arquitectura de almacenamiento paralelo. La infraestructura SFS está formada por 18 servidores de ficheros, 1 servidor de metadatos y 1 servidor de administración, todos ellos HP Proliant DL380 G5. Cabe notar también que próximamente se integrará en el Finis Terrae los 2 nodos HP Integrity Superdome que se encontraban en el CESGA anteriormente.

La red de interconexión para las tareas de computación es Infiniband 4x DDR[2], es decir, usa cuatro enlaces agregados que envían dos símbolos por cada ciclo de reloj. Su estructura se muestra en la figura 3.1. Cada enlace SDR[3] Infiniband envía 2,5 gigabits por segundo, resultando en 20 gigabits por segundo en una configuración Infiniband 4x DDR. Dado que la codificación usada por Infiniband es 8B/10B [74], el ancho de banda efectivo es de 16 Gbps. El switch Infiniband es un switch Voltaire Grid Director ISR 2012.

Aparte de la red de interconexión Infiniband el Finis Terrae tiene una red Fast

[1] SFS: Scalable File System
[2] DDR: Double Data Rate
[3] SDR: Single Data Rate

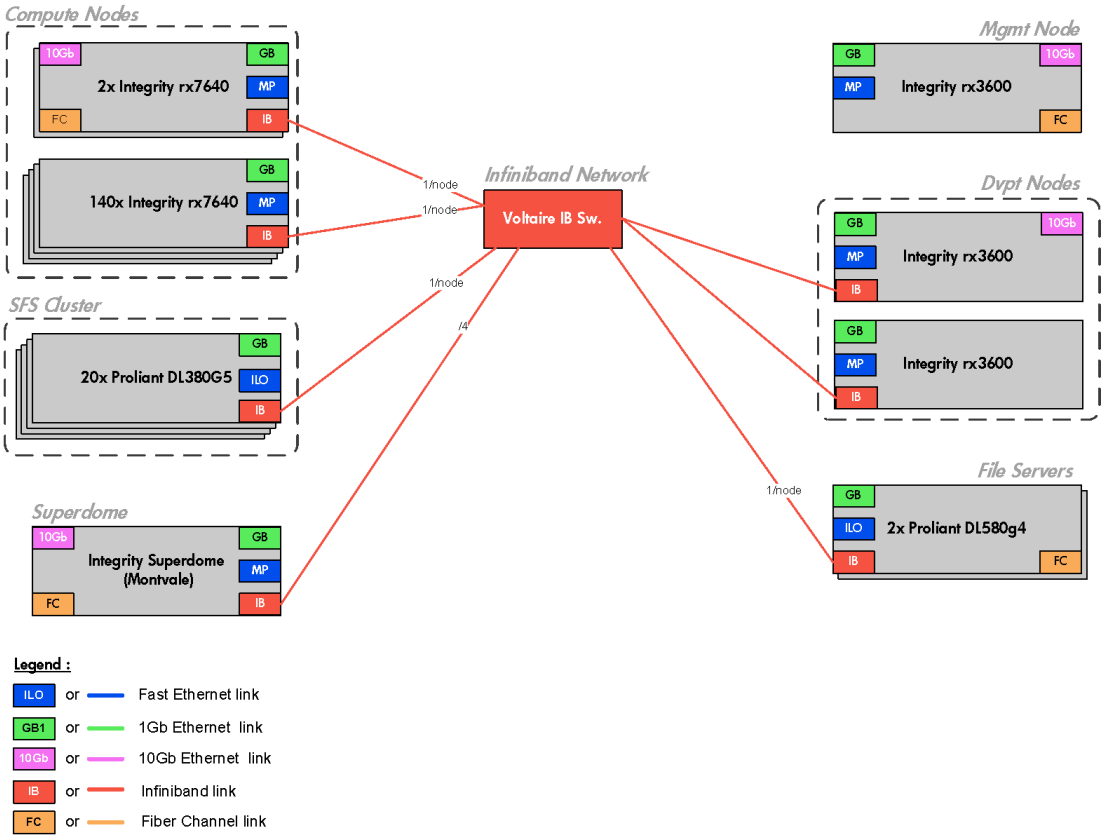


Figura 3.1: Red Infiniband del Finis Terrae

Ethernet para la gestión de los equipos a través del sistema iLO2[4] y tecnologías similares (dependientes del tipo de nodo) de HP, y una red Gigabit Ethernet para la administración. La arquitectura de estas redes sigue una estructura jerárquica en los nodos de cómputo, tal y como se muestra en la figura 3.2.

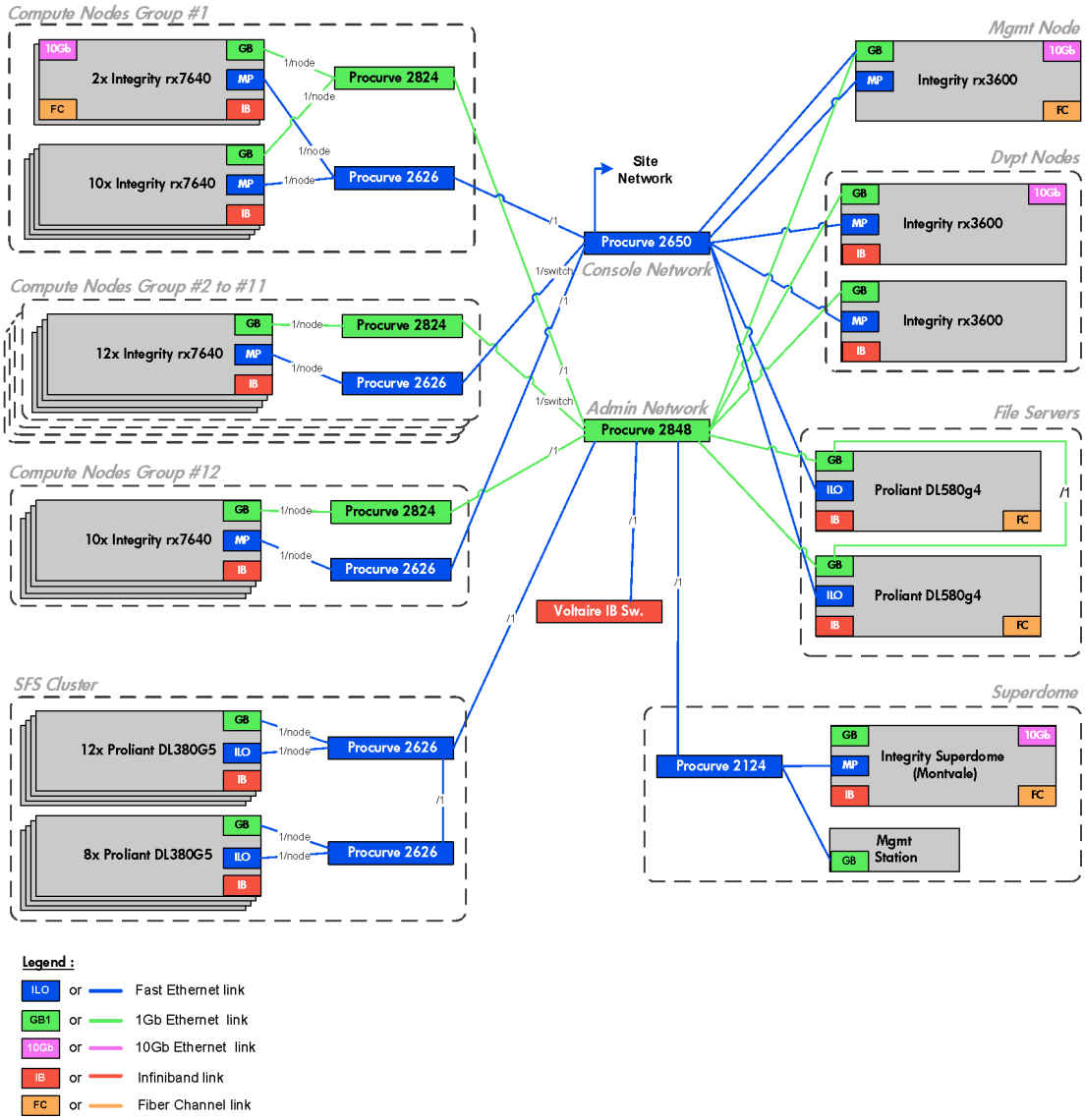


Figura 3.2: Redes de gestión y administración del Finis Terrae

Existe otra red para conectar el Finis Terrae con la red del CESGA. Dicha red tiene enlaces Gigabit Ethernet para los nodos de cómputo, y, además, enlaces 10 Gigabit Ethernet con el nodo Superdome, los dos nodos redundantes de login, un nodo de desarrollo y el nodo de gestión. Esta red tiene todos los enlaces con los nodos de cálculo redundados, así como los switches, y su estructura se muestra en la figura 3.3.

[4]iLO2: Integrated Lights-Out 2

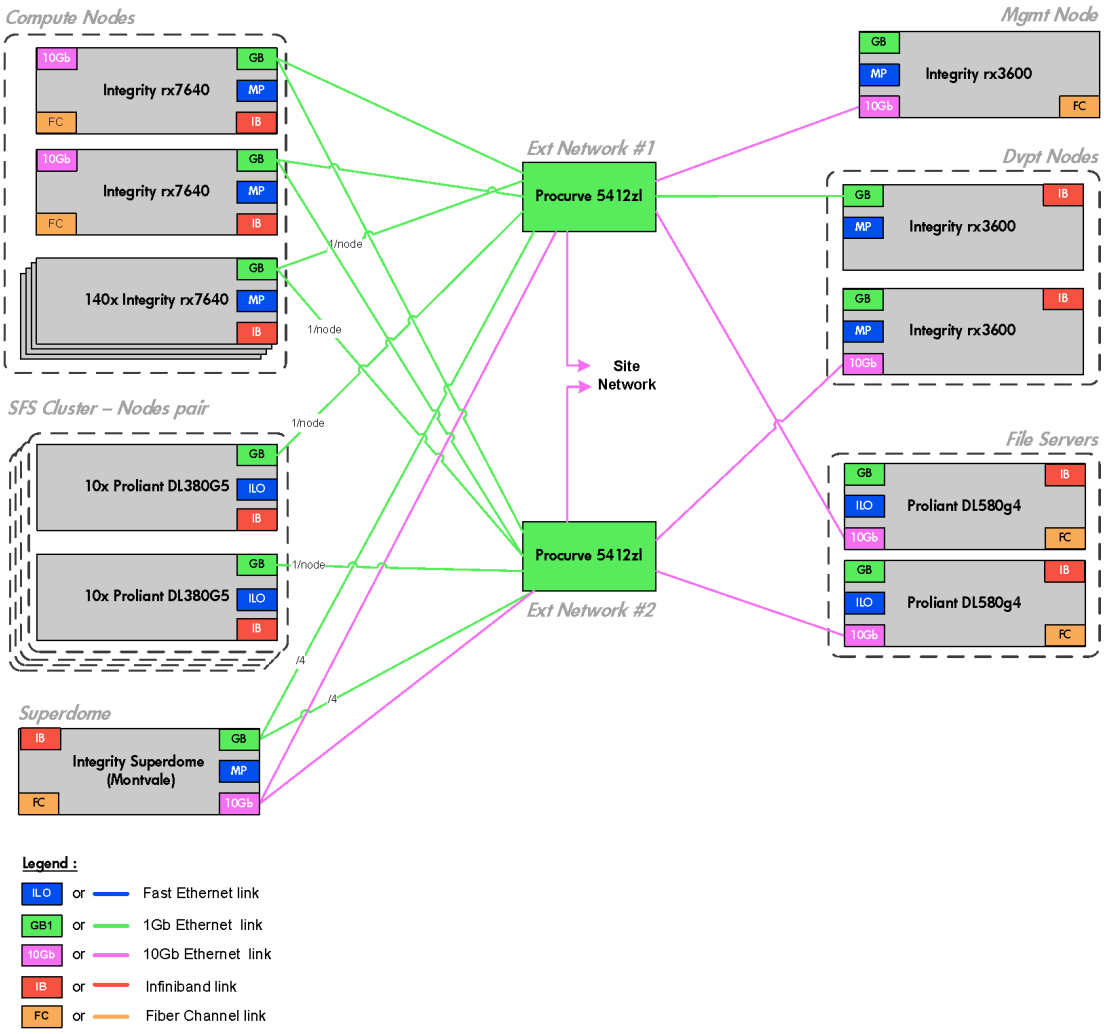


Figura 3.3: Red para la conexión externa del Finis Terrae

Por último, existe una red Fibre Channel a 4 Gigabits por segundo para acceder a la infraestructura SAN[5] del CESGA, en donde residen los directorios *home* de los usuarios y sistemas de ficheros compartidos con otros supercomputadores del CESGA. Dicha red está totalmente redundada y a ella se conectan los nodos de computación destinados para su uso interactivo, el nodo Superdome, el nodo de gestión, los dos nodos redundantes de login y el robot de cintas. Su estructura se muestra en la figura 3.4.

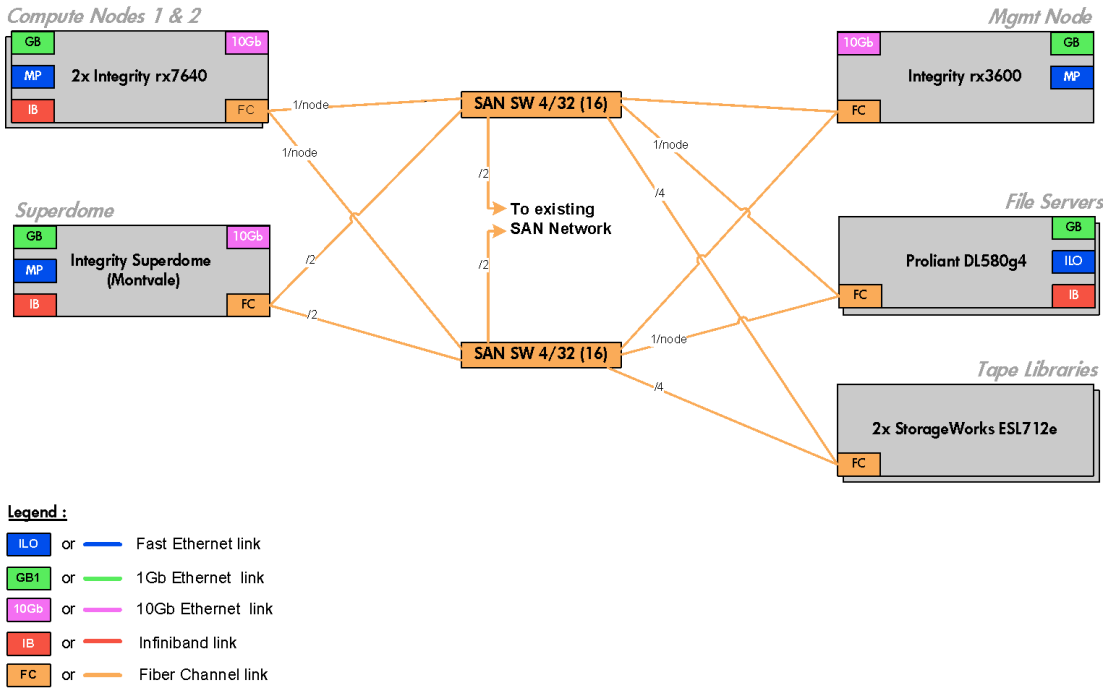


Figura 3.4: Red SAN del Finis Terrae

Para trabajar con el supercomputador se establecieron 2 sistemas:

Interactivo: Destinado a pruebas y pequeños trabajos. Cuenta para ello con 2 nodos dedicados.

Sistema de colas: Planificador que deben usar todos los usuarios para enviar sus trabajos. Se le debe especificar el número de cores por nodo, la cantidad de memoria por nodo, el uso que se hará del scratch local, el tiempo máximo de ejecución y el número de procesos MPI que se lanzarán en el trabajo. El sistema usado es Sun Grid Engine.

El sistema operativo usado en todo el supercomputador es SuSE Linux Enterprise Server 10, con kernel 2.6.16.

[5]SAN: Storage Area Network

3.1.1. Arquitectura de los Nodos HP Integrity rx7640

Los nodos de cálculo HP Integrity rx7640 tienen 8 procesadores Intel Itanium 2 9140N. Dicho procesador funciona a 1,6 GHz y cuenta con dos núcleos, cada uno con 16 KB de caché de nivel 1 para instrucciones y otros 16 KB para datos, 1 MB de caché de nivel 2 para instrucciones y 256 KB para datos, y 18 MB de memoria caché de nivel 3 unificada y compartida entre los 2 núcleos. También poseen 128 GB de memoria RAM DDR2-533 MHz ECC, formando por tanto nodos de 16 cores. Estos recursos hardware se encuentran repartidos a partes iguales en dos celdas conectadas entre sí a través de 3 canales full duplex a 11,5 Gigabytes por segundo cada uno.

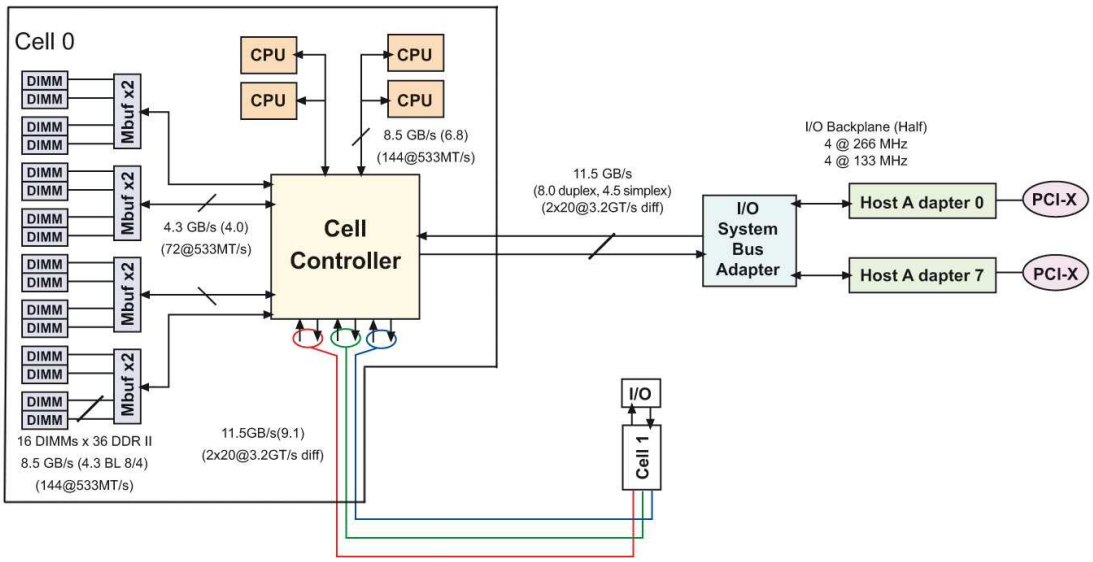


Figura 3.5: Arquitectura de los nodos HP Integrity rx7640 del Finis Terrae

Cuentan con una tarjeta HP Smart Array 6402 con 128 MB de caché y con un bus PCI-X 133 MHz, conectada a 2 discos Ultra320 SCSI de 73 GB y 15000 rpm[6]. Dichos discos alojan el sistema de ficheros del sistema operativo. Para el scratch local (sistema de almacenamiento local de altas prestaciones y alojamiento temporal) tienen una tarjeta HP Smart Array P800 con una memoria caché de 512 MB y bus PCI-Express x8 conectada a una cabina de discos HP StorageWorks MSA50 con 8 o 4 discos SAS[7] (existen nodos con 4 y con 8 discos de scratch, empleados según el tamaño de scratch requerido) de 72 GB y 10000 rpm, configurados como un RAID de nivel 5.

Cada celda tiene una tarjeta Gigabit Ethernet Intel 82546GB integrada. Cada nodo tiene además una tarjeta HP Server Gigabit Ethernet con dos puertos y dos chips Intel 82546GB y bus PCI-X 133 MHz. Para la gestión remota tiene un puerto ethernet específico Fast Ethernet. La tarjeta Infiniband es una tarjeta 4x DDR Voltaire HCA 400EX-D con chip Mellanox Infinihost III Ex, 2 puertos, 128 MB de caché y PCI-Express x8. Los 2 nodos destinados a su uso en interactivo cuentan además con 1

[6]rpm: revolutions per minute
[7]SAS: Serial ATA SCSI

tarjeta 10 Gigabit Ethernet Neterion Xframe en un zócalo PCI-X 133 MHz y con 2 tarjetas Fibre Channel QLogic QLA2462 4Gb con dos puertos y bus PCI-X 266 MHz, a pesar de encontrarse colocadas en zócalos PCI-X 133 MHz.

En la figura 3.5 se muestra la arquitectura de los nodos de computación. Cabe notar que la figura mostrada contempla una configuración con slots PCI-X exclusivamente. En una configuración con slots PCI-X y PCI-Express, los puertos a 266 MHz cuentan con puentes a PCI-Express x8, por lo que la configuración es de 7 conectores PCI-X 133 MHz (mas 1 extra usado por dispositivos integrados) y 8 conectores PCI-Express x8.

3.1.2. Arquitectura del Nodo HP Integrity Superdome

La arquitectura del sistema HP Integrity Superdome se apoya en la de los nodos HP Integrity rx7640. Contiene 64 procesadores Intel Itanium 2 9140N, cada uno de ellos con las mismas características que las detalladas en los nodos rx7640, haciendo un total de 128 cores. La memoria RAM instalada es de 1024 GB DDR2-533 MHz ECC. El sistema se divide en 16 celdas, con una estructura básica muy similar a la de los nodos de computación estándar en el Finis Terrae. Sin embargo, para poder conectar todas las celdas entre sí, se hace uso de 6 *crossbars*, y las celdas tienen cada uno de sus 3 canales a 11,5 Gigabytes por segundo full duplex conectado a un *crossbar* distinto. La figura 3.6 muestra la arquitectura de este sistema con más detalle.

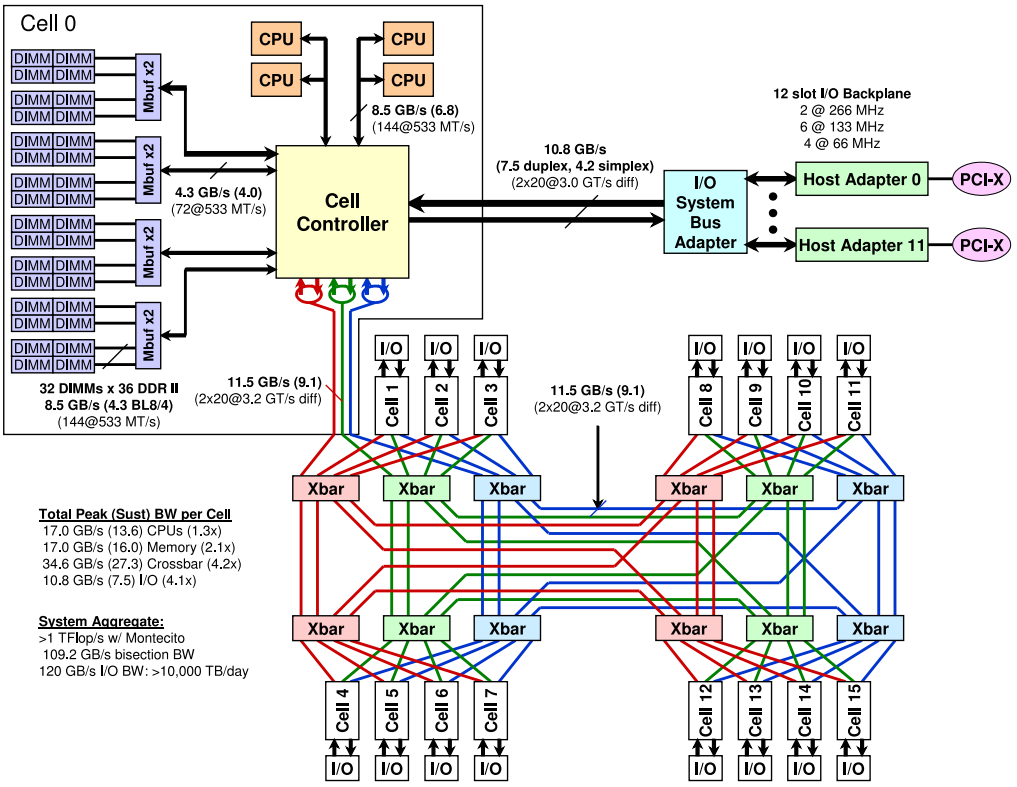


Figura 3.6: Arquitectura del nodo HP Integrity Superdome del Finis Terrae

El Superdome cuenta con 8 tarjetas Smart Array P800 con 512 MB de memoria caché y bus PCI-Express x8. Cada una de ellas está conectada a 2 cabinas de discos MSA50 con 8 discos SAS de 72 GB y 10000 rpm en RAID 5. Estos discos están destinados al scratch local y al sistema operativo, y forman un total de 9,2 Terabytes.

Al igual que los nodos de computación rx7640 cada celda tiene una tarjeta Gigabit Ethernet Intel 82546GB integrada. Tiene 2 tarjetas 10 Gigabit Ethernet Neterion Xframe en zócalos PCI-X 133 MHz. También cuenta con interfaces Fast Ethernet para la gestión remota. Para la red Infiniband cuenta con 4 tarjetas 4x DDR Voltaire HCA 400EX-D con chip Mellanox Infinihost III Ex, 2 puertos, 128 MB de caché y PCI-Express x8. Por último, cuenta con 2 tarjetas Fibre Channel QLA2462 4Gb con dos puertos y bus PCI-X 266 MHz, conectada en bus PCI-X 133 MHz.

3.1.3. Arquitectura del SFS

El Finis Terrae cuenta con un sistema de almacenamiento paralelo de alto rendimiento. Dicho sistema es un HP SFS, solución basada en la arquitectura y sistema de ficheros Lustre. En dicha arquitectura se establecen MDSs[8] que contienen información acerca de la localización y demás atributos de los ficheros, OSTs[9] que es en donde se localizan los ficheros y OSSs[10], encargados de manejar los OSTs.

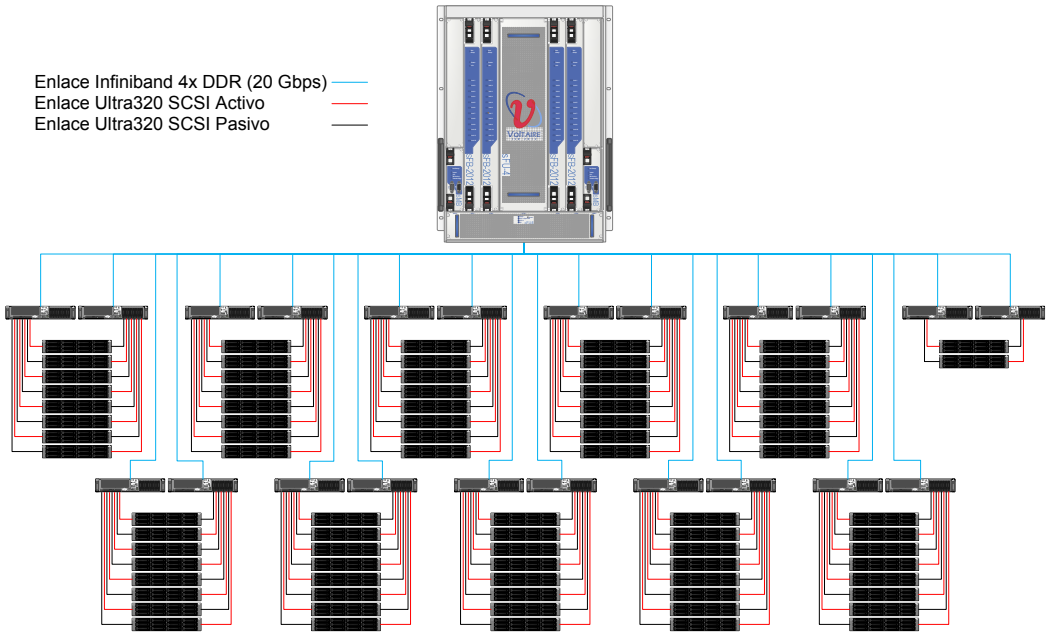


Figura 3.7: Arquitectura del sistema SFS del Finis Terrae

El sistema SFS del Finis Terrae cuenta con 2 HP Proliant DL380 G5 en el papel de MDS y administración, 18 HP Proliant DL380 G5 en el papel de OSSs y 72 cabinas

[8]MDS: Meta-Data Server
[9]OST: Object Storage Target
[10]OSS: Object Storage Server

de discos SFS20, que son una versión de las cabinas de discos MSA20 con dos puertos Ultra320 SCSI.

Cada servidor HP Proliant DL380 G5 cuenta con 2 procesadores Intel Xeon 5160 dual-core a 3 GHz con 4 MB de caché de nivel 2. Los servidores MDS y de administración poseen 4 GB de memoria RAM Fully Buffered DDR2-667 MHz ECC, mientras que los servidores OSS tienen 2 GB. Cuentan con 2 tarjetas Gigabit Ethernet HP NC371i integradas. También tienen interfaces Fast Ethernet para gestión remota iLO2. Las tarjetas Infiniband son 4x DDR Voltaire HCA 410EX-D con chip Mellanox Infinihost III Lx, 128 MB de caché y PCI-Express x8 para el caso de los servidores MDS, y 4x DDR Voltaire HCA 400EX-D con chip Mellanox Infinihost III Ex, 2 puertos, 128 MB de caché y PCI-Express x8 para el caso de los servidores OSS.

Para el almacenamiento cuentan con 1 tarjeta HP Smart Array P400 con 256 MB de memoria caché y bus PCI-Express x8 que se encarga de gestionar 1 disco SAS de 72 GB y 10000 rpm, 2 discos en el caso de los servidores MDS y de administración. Para gestionar las cabinas de discos cuentan con 1 tarjeta HP Smart Array 6404 con 256 MB de memoria caché y 4 canales Ultra320 SCSI, con un bus PCI-X 133 MHz. En el caso de los servidores OSS el número de estas tarjetas es 2, de forma que pueden controlar un máximo de 8 cabinas. 4 de estas cabinas serán controladas de forma activa, mientras que las otras 4 estarán controladas de forma pasiva, de forma que se pasa a modo activo si el OSS que estaba controlando dicha cabina sufre algún problema el servicio no es interrumpido. Esto mismo es aplicable a los servidores MDS y de administración, pero en una escala reducida, ya que cada uno controla 1 cabina. En la figura 3.7 puede verse como está organizado el cluster SFS.

Las cabinas de discos SFS20 contienen 12 discos SATA de 250 GB y 7200 rpm. Cada cabina tiene 128 MB de caché y una configuración de RAID 5.

3.2. Necesidades de Evaluación

Conocida la arquitectura del Finis Terrae se pueden establecer las necesidades de evaluación más urgentes, y se pueden extraer que pruebas resultan más interesantes. En las siguiente subsecciones se analizarán dichas necesidades.

3.2.1. Afinidad a Celdas

En el estudio de la arquitectura de los nodos se ha visto que cada uno está formado por 2 celdas. La tarjeta Infiniband se encuentra en una celda. Partiendo de esto se hace imperativo saber la penalización en el rendimiento que se añade cuando se accede a la red Infiniband desde la celda que no tiene la tarjeta.

3.2.2. Escalabilidad de Operaciones Colectivas

La escalabilidad de operaciones colectivas es un problema en clusters de gran magnitud. En el Finis Terrae cobra especial importancia debido a la cantidad de cores por nodo. Con un número significativo de cores es fácil que se de la situación de varios procesos MPI (con paso de mensajes entre nodos) por nodo, pudiendo alcanzar esta cifra 16 procesos por nodo. En operaciones colectivas se tendrían hasta 16 procesos por

nodo enviando y recibiendo datos por la misma interfaz Infiniband, lo que podría dar lugar a contención en el acceso a la red, que es importante determinar.

3.2.3. Configuraciones Híbridas

Como ya se ha dicho, el Finis Terrae es un cluster con un número considerable de cores por nodo. Esta característica arquitectural se presta a un modelo de programación híbrida OpenMP+MPI, en donde cada proceso MPI puede generar varios threads OpenMP. Esta aproximación parece la más adecuada para la programación del Finis Terrae, por aprovechar un modelo de programación que es, a priori, más eficiente en paralelización con memoria compartida (OpenMP), y minimizar las comunicaciones cuando la paralelización es con memoria distribuida (MPI), ya que existirán menos procesos MPI por nodo que en una aproximación puramente MPI. Con este escenario es necesario evaluar que combinaciones de procesos MPI y OpenMP son los que obtienen mayor rendimiento. Recientemente el grupo de investigación HEMCUVE++, compuesto por investigadores de la Universidad de Vigo y la Universidad de Extremadura, han superado un reto computacional en el CESGA, en el que se ha resuelto un problema de electromagnetismo con 500 millones de incógnitas, con lo que se ha batido el record mundial, usando un modelo de paralelización híbrido [75]. Este hecho pone de relevancia el interés en la programación híbrida en el Finis Terrae.

3.2.4. Bibliotecas de Comunicaciones

En el Finis Terrae se encuentran instaladas dos bibliotecas de comunicaciones MPI: HP MPI 2.2.5 e Intel MPI 3.1, ambas con un gran surtido de opciones y capacidades. No obstante, su diseño e implementación varían, y la diferencia entre estas dos bibliotecas puede ser significativa. Su peso en el rendimiento de las aplicaciones se incrementa a medida que el problema tiene mayor número de comunicaciones, por lo que se hace necesario evaluar la eficiencia de estas dos bibliotecas.

3.2.5. Eficiencia de Lenguajes

Los investigadores que desarrollan sus trabajos con los supercomputadores del CESGA programan principalmente en C y Fortran. Ambos lenguajes disponen de soporte OpenMP y MPI, con lo que la eficiencia de las aplicaciones desarrolladas puede ser muy alta en el Finis Terrae. No obstante, a medida que aumenta el número de cores por procesador, cada vez se hace más importante el uso de lenguajes que permitan una paralelización eficiente con la menor penalización posible en la productividad. Los lenguajes PGAS poseen potencial para convertirse en los lenguajes indicados para este tipo de sistemas, ya que proveen una paralelización con construcciones semánticas sencillas, a la vez que permiten indicar la privacidad de las variables, permitiendo una programación híbrida explícita en el propio lenguaje. Entre estos lenguajes destaca UPC, y evaluar el rendimiento de las aplicaciones programadas con él es un requisito deseable de esta evaluación. Asi mismo, la evaluación de Java también es deseable, debido a que es un lenguaje que está ganando presencia en entornos HPC, posee bibliotecas de comunicaciones de calidad, y el soporte multihilo es parte fundamental del lenguaje.

4 Diseño del Benchmarking

Explicada la arquitectura del Finis Terrae y las necesidades de evaluación más urgentes, se está en disposición de realizar una elección de los benchmarks a realizar para cubrir las distintas necesidades. Además, con el conocimiento adquirido del estudio de trabajos previos, el abanico de oportunidades para cubrir los requisitos de evaluación es amplio y variado.

La elección de benchmarks viene condicionada no sólo por las necesidades de evaluación, sino que también depende de los benchmarks ejecutados anteriormente para la aceptación del Finis Terrae. En dicha evaluación se utilizaron los siguientes benchmarks:

- LINPACK.
- HPC Challenge.
- IOZone [76]. Un benchmark destinado a medir el rendimiento del sistema de ficheros.
- STREAM.
- Gaussian.
- Intel MPI Benchmarks. Sólo el test PingPong, sin invalidación de caché.
- Iperf [77]. Un benchmark del NLANR[1] destinado a medir el ancho de banda de una red TCP o UDP.

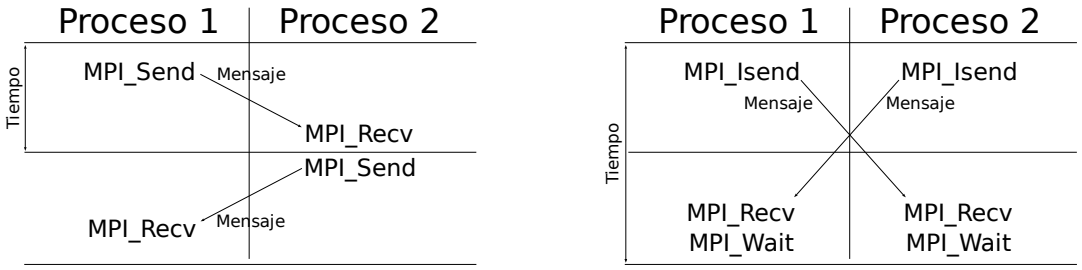
Con estas premisas se procederá a la elección del benchmarking.

4.1. Benchmarks para Definir la Influencia de la Localidad Respecto a las Celdas

Para definir y delimitar la influencia de la arquitectura NUMA[2] (en celdas) de los nodos rx7640 al acceder a recursos no locales a la propia celda los test PingPong y PingPing de los Intel MPI Benchmarks son una poderosa herramienta. Ambos test realizan comunicaciones punto a punto, por lo que son necesarios dos procesos. El test PingPong envía un mensaje desde el proceso 0 al proceso 1, que lo recibe y lo

[1] NLANR: National Laboratory for Applied Network Research
[2] NUMA: Non Uniform Memory Access

devuelve. Es equivalente a una comunicación *"echo"*. Realiza una serie de envíos de mensajes de distintos tamaños, cada uno repetido un determinado número de veces. La información que proporciona es el ancho de banda y la latencia promedio para cada tamaño de mensaje. Es importante notar que el ancho de banda y la latencia medidas se corresponden al envío y recepción del mensaje original, dejando fuera de la medición el envío y recepción del mensaje de vuelta. El test PingPing es similar al PingPong, pero en este caso ambos procesos envían un mensaje al otro a la vez. Los mensajes no son devueltos. Los patrones de comunicación de ambos test se puede consultar en la figura 4.1.



(a) PingPong (b) PingPing

Figura 4.1: Patrones de comunicación de PingPong y PingPing

Para medir la influencia del acceso a otra celda se usará PingPong y PingPing en diferentes configuraciones, que incluyen realizar ejecuciones con los dos procesos en la misma celda, y en diferentes celdas dentro de un mismo nodo. También se medirán la influencia existente en las comunicaciones entre nodos, al acceder a la tarjeta Infiniband desde la misma celda en donde se encuentra ésta, o desde una celda diferente, resultando en tres configuraciones distintas, los dos procesos en la celda 0, los dos procesos en la celda 1, y ambos procesos en diferentes celdas.

Los Intel MPI Benchmarks tienen una opción para invalidar los datos de la caché de los procesadores, eliminando su influencia y resultando en ejecuciones en el peor caso posible. Dado lo limitado del tiempo para realizar las pruebas, y la complicación añadida de que el Finis Terrae es un recurso compartido por toda la comunidad científica gallega, las pruebas realizadas con los Intel MPI Benchmarks serán realizadas con invalidación de caché, para obtener el peor escenario posible y ver situaciones que de otra forma no sería posible.

4.2. Benchmarks para Definir la Escalabilidad de Operaciones Colectivas

Para medir la escalabilidad de operaciones colectivas los Intel MPI Benchmarks cuentan con tests para numerosas funciones MPI. Los patrones de comunicación de las funciones colectivas se pueden agrupar en dos grandes grupos: comunicaciones uno a todos, y comunicaciones todos a todos. Existen funciones con un patrón todos a uno, pero en términos de acceso a la red Infiniband son muy similares a las funciones con patrones

uno a muchos, y por tanto no se contemplan. Se han elegido los test correspondientes a las llamadas `MPI_Bcast(...)` (Broadcast, uno a muchos) y `MPI_Alltoall(...)` (muchos a muchos). Para la ejecución de estas pruebas es necesario aprovechar una parada del supercomputador, ya que al ser un recurso compartido entre muchos grupos investigadores no sería posible reservar un número suficiente de nodos de otra forma.

Al igual que en las comunicaciones punto a punto se usará invalidación de caché. Además, se aprovecharán estas pruebas para realizar test de afinidad a celdas, para completar los resultados de las comunicaciones punto a punto.

4.3. Benchmarks para Definir Configuraciones Híbridas Óptimas

Estudiar la configuración optima de procesos MPI y threads OpenMP requiere unos benchmarks más complejos que los Intel MPI Benchmarks. Al ser ésta una aproximaximación relativamente novel, y con poca fuerza hasta la aparición de procesadores multi-core, no son muchos los benchmarks destinados a evaluar el rendimiento de configuraciones híbridas. Los principales son los NAS Parallel Benchmarks Multi-Zone [11], algunos benchmarks de la suite Sequoia del LLNL[3] [78] y algunos benchmarks del CCSM[4] [23].

Los benchmarks utilizados serán los NAS Parallel Benchmarks Multi-Zone, ya que, además de ser los más relevantes, permiten una evaluación con sus contrapartidas en MPI puro y OpenMP puro.

4.4. Benchmarks para Definir la Eficiencia de las Bibliotecas de Comunicaciones

La eficiencia de las bibliotecas de comunicaciones se ve mejor reflejada en test sintéticos que evalúen sólo las comunicaciones, como los test de comunicaciones punto a punto y de comunicaciones colectivas empleados para definir la influencia de la afinidad a celdas o la escalabilidad de operaciones colectivas. Así pues, se aprovechará el benchmarking realizado para cubrir esas áreas, y se realizarán ejecuciones con HP MPI y con Intel MPI.

4.5. Benchmarks para Definir la Eficiencia de Lenguajes

El número de lenguajes a cubrir por este estudio limitan en gran medida los benchmarks a utilizar. A los lenguajes usados profusamente en HPC como C y Fortran se une el lenguaje UPC, con una presencia limitada en el momento actual, por lo que el número de benchmarks disponibles para UPC es relativamente pequeño. Además, también resulta interesante la evaluación de Java, por lo que las posibilidades se ven reducidas aún más. A día de hoy, los únicos benchmarks implementados en C/Fortran, UPC y

[3]LLNL: Lawrence Livermore National Laboratory
[4]CCSM: Community Climate System Model

Java son los NAS Parallel Benchmarks, por lo que, una vez más, serán los benchmarks utilizados.

5 Análisis de Resultados

Se ha evaluado exahustivamente el Finis Terrae con los benchmarks seleccionados que han sido presentados en el apartado anterior. El presente apartado muestra los resultados obtenidos, así como un análisis pormenorizado de cada resultado. Los análisis presentados se dividen en:

- Operaciones punto a punto.
- Operaciones colectivas.
- Configuraciones híbridas.
- Rendimiento de lenguajes.

Dichas secciones se corresponden con las necesidades de evaluación, excepto en el caso de la evaluación de las bibliotecas de comunicaciones, que está incluído dentro de los epígrafes de operaciones punto a punto y operaciones colectivas.

Cabe notar además que el benchmarking preliminar sobre comunicaciones punto a punto ha arrojado algunos resultados dignos de estudio. En concreto existe una caída importante de rendimiento cuando el tamaño del mensaje enviado rebasa los 16 KB. Fruto de esta situación se ha decidido realizar un análisis más fino en el rango próximo a los 16 KB, que coincide con el tamaño de la caché de nivel 1. Además se ha analizado finamente el rango de tamaño de mensaje próximo al tamaño de la caché de nivel 2 (256 KB) y de nivel 3 (18 MB).

5.1. Operaciones Punto a Punto

El análisis de las operaciones punto a punto se hará a varios niveles. En primer lugar se examinarán la latencia y el ancho de banda obtenidos. La latencia es vital para tamaños de mensaje pequeños, mientras que el ancho de banda lo es para tamaños de mensaje grandes, por lo que las gráficas de cada uno serán representadas en el rango más relevante. Así, las gráficas de latencia se representarán de 1 byte a 4 KB, mientras que las gráficas de ancho de banda son representadas desde 4 KB hasta 32 MB. Además las gráficas correspondientes a los test PingPong y PingPing se representarán por separado para mayor claridad. Se incluyen en esta batería de análisis los resultados obtenidos con las bibliotecas de comunicaciones HP MPI e Intel MPI.

Dado que el objetivo principal de estas pruebas es comprobar y medir la influencia del acceso a recursos de otra celda, se representarán también unas gráficas con los

rendimientos porcentuales de las comunicaciones en el peor escenario (usando diferentes celdas en el caso de memoria compartida, y celdas sin tarjeta Infiniband en el caso de comunicaciones entre nodos) respecto a la comunicación en el mejor escenario (usando la misma celda en el caso de memoria compartida, y celdas con tarjeta Infiniband en el caso de comunicaciones entre nodos), para obtener una mejor perspectiva de la magnitud del impacto en el rendimiento.

5.1.1. Operaciones Punto a Punto con Memoria Compartida

La figura 5.1 muestra el ancho de banda conseguido con el benchmark PingPong. En él se aprecia como el rendimiento sufre una pequeña degradación entorno a los 16 KB, pero sin embargo se recupera y prosigue su línea ascendente. En el límite de los 8 MB se produce una degradación mucho más pronunciada. Este fenómeno tiene lugar con ambas bibliotecas, y con comunicaciones dentro de la misma celda y entre celdas, pero se atenúa en las configuraciones con menor rendimiento. Además, la línea de rendimiento vuelve al valor esperado en torno a los 16 MB. Las diferencias de rendimiento entre comunicaciones intracelda e intercelda son notables, alcanzando valores de hasta 190 MB/s (un 17 % del ancho de banda). Otra carecterística notable es que la diferencia entre ambas configuraciones es prácticamente idéntica en ambas bibliotecas, a pesar de que Intel MPI obtiene un rendimiento muy inferior al de HP MPI. El ancho de banda máximo para éste es de 1375 MB/s.

En la figura 5.2 se ven unas tendencias similares a las observadas en el ancho de banda. Las diferencias entre comunicaciones intra e intercelda son prácticamente iguales para ambas bibliotecas, a pesar de que, nuevamente, Intel MPI obtiene un rendimiento peor que HP MPI, especialmente destacable para tamaños de mensaje pequeños. La latencia mínima de HP MPI es de 1,29 μs.

Las figuras 5.3 y 5.4 muestran el ancho de banda y la latencia obtenidos con el test PingPing. Sus resultados son similares en variaciones y diferencias a los obtenidos con el test PingPong. Sin embargo el rendimiento es aproximadamente la mitad, con lo que el ancho de banda máximo es prácticamente la mitad del obtenido en PingPong, y la latencia el doble para los mensajes más pequeños. Dado que los resultados son aplicables a ambas bibliotecas y con comunicaciones intra e intercelda, la causa apunta a un problema hardware. Recordando la figura 3.5 se ve que el único elemento común es el *Cell Controller*. Además cabe notar que PingPing mide el ancho de banda de un único mensaje, por lo que el ancho de banda global permanece casi constante, de forma que la bajada de rendimiento viene justificada por la gestión de comunicaciones bidireccionales por parte del *Cell Controller*.

Las figuras 5.5 y 5.6 muestran cómo afecta la localidad (la ubicación concreta de un proceso en un core de procesador determinado dentro de un nodo multi-core) de los procesos a la latencia y ancho de banda en los test PingPong y PingPing ejecutados con ambas bibliotecas, en términos porcentuales, en donde un porcentaje alto es deseable para el ancho de banda, y un porcentaje bajo es el deseable para la latencia. Hasta el límite de los 16 KB Intel MPI parece verse menos afectado por las comunicaciones intercelda que HP MPI, cambiando esta tendencia para tamaños superiores a 16 KB. La tendencia de HP MPI es a verse menos afectado por la localidad de los procesos a medida

que aumenta el tamaño de mensaje, lo que refleja una implementación más eficiente para esta máquina, ya que minimiza la influencia negativa del acceso a otra celda. El rendimiento de las comunicaciones intercelda rara vez supera el 90 % del rendimiento de las comunicaciones intracelda, llegando a tener un rendimiento tan bajo como del 50 % para tamaños de mensaje pequeños.

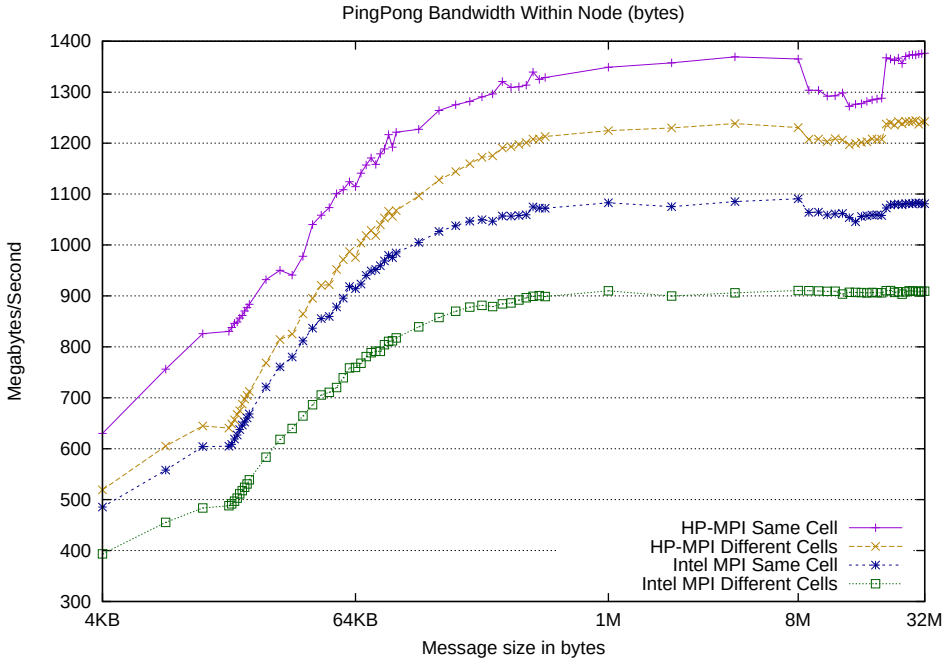


Figura 5.1: Ancho de Banda en PingPong con Memoria Compartida

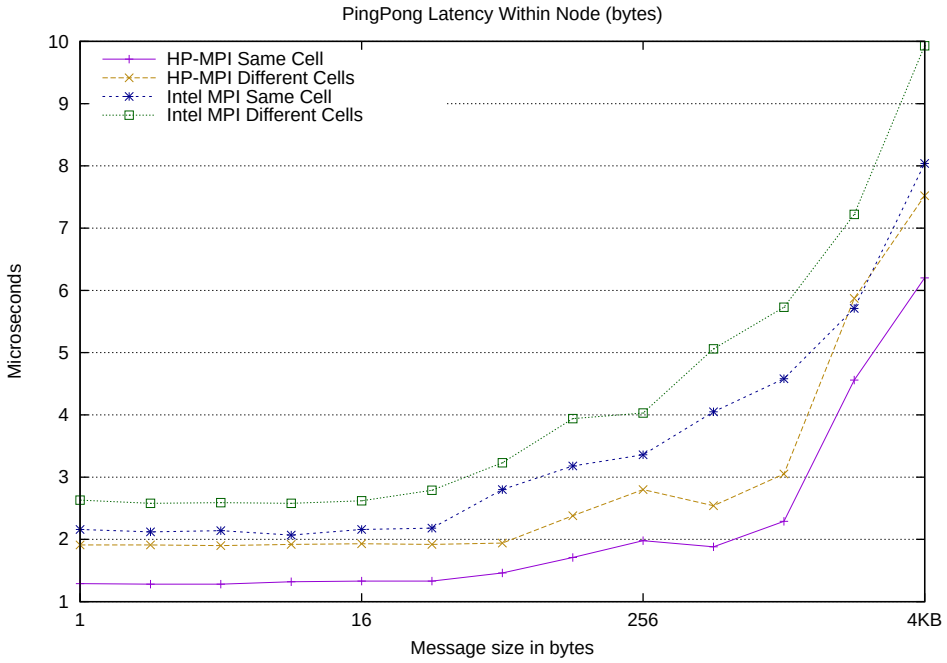


Figura 5.2: Latencia en PingPong con Memoria Compartida

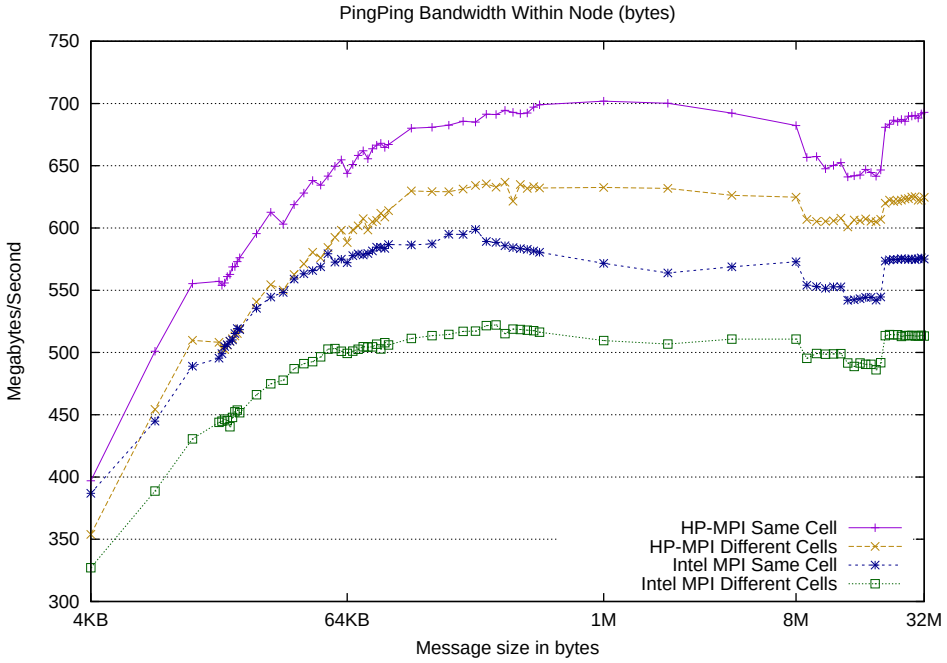


Figura 5.3: Ancho de Banda en PingPing con Memoria Compartida

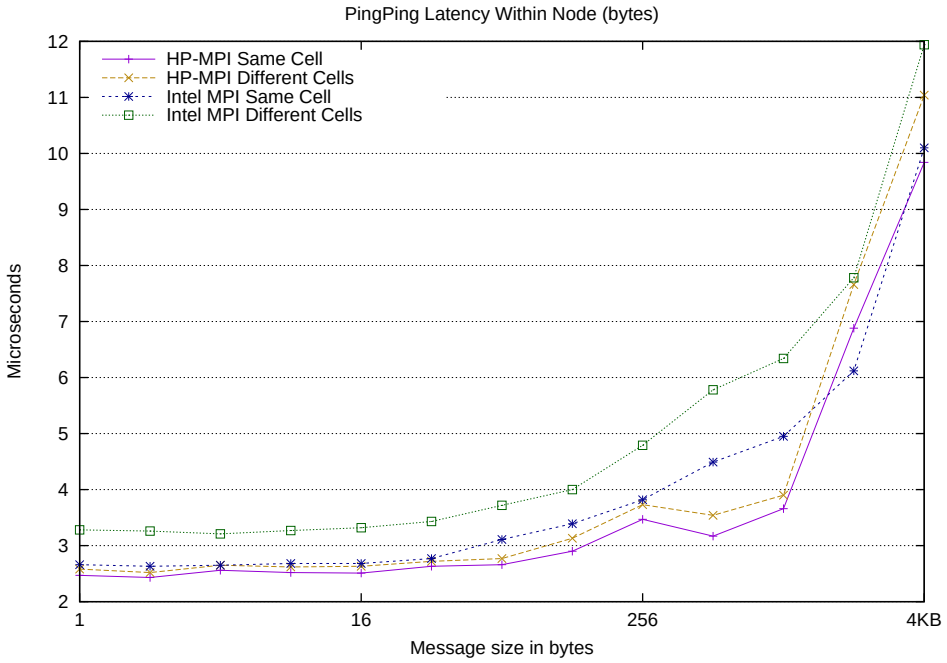


Figura 5.4: Latencia en PingPing con Memoria Compartida

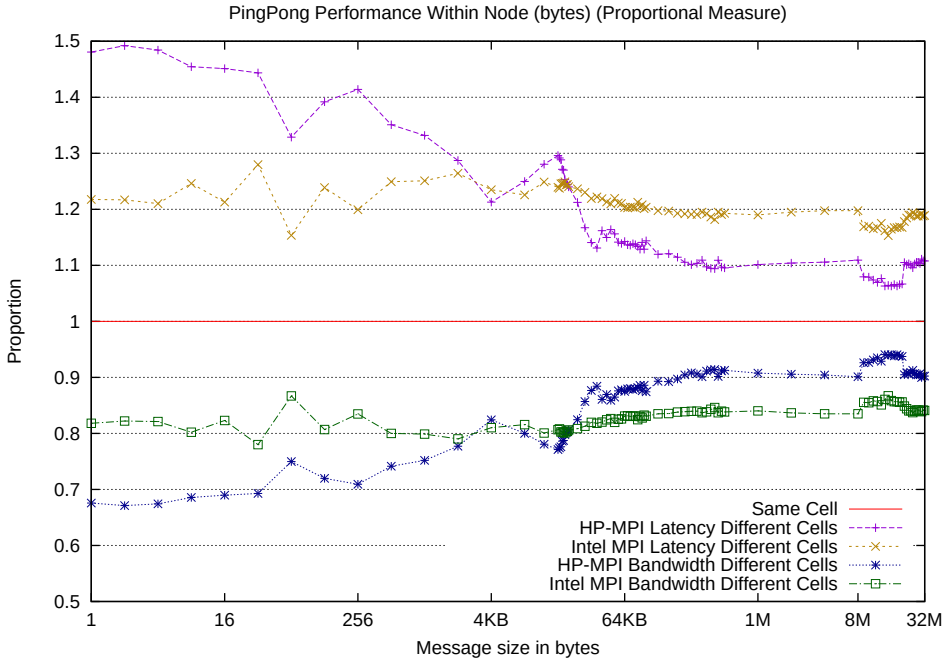


Figura 5.5: Rendimiento Relativo con PingPong y Memoria Compartida

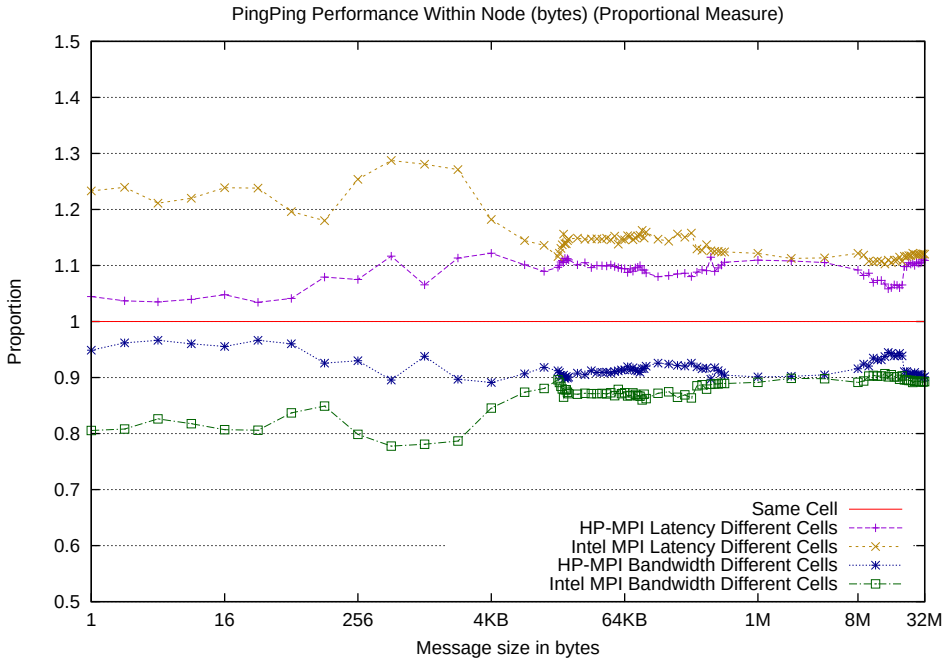


Figura 5.6: Rendimiento Relativo con PingPing y Memoria Compartida

5.1.2. Operaciones Punto a Punto con Infiniband

Las figuras 5.7 y 5.8 muestran los resultados de anchos de banda y latencias para MPI PingPong, respectivamente. El análisis de la figura 5.7 revela un resultado no esperado. En el límite de los 16 KB para el tamaño de mensaje se produce una notable caída en el rendimiento, para todas las configuraciones, siendo más acentuada en HP MPI al pasar de un ancho de banda de 426 MB/s a 52 MB/s. Dicha caída se recupera progresivamente hasta, al llegar a un tamaño de mensaje de 256 KB recuperar el ancho de banda esperado. La causa de este fenómeno se estudiará en la subsección "Efectos de los Algoritmos de Comunicaciones". En el tramo comprendido entre 16 KB y 8 MB, el rendimiento de Intel MPI es notablemente superior al de HP MPI, llegando incluso a diferencias de 250 MB/s. Sin embargo, fuera de ese rango es superior HP MPI. Esto contrasta con la situación de memoria compartida, en donde el rendimiento de HP MPI es superior a Intel MPI. Las diferencias entre la ejecución en la celda con la tarjeta Infiniband (celda 0) y la ejecución en la celda sin tarjeta Infiniband (celda 1) son apreciables, máxime en el rango 16 KB – 8 MB. El ancho de banda máximo lo ha alcanzado HP MPI y corresponde a 1240 MB/s.

La figura 5.8 muestra los resultados hasta 64 KB, para ver el efecto en la latencia del fenómeno que provoca la caída en el ancho de banda. Cabe destacar que la escala vertical es logarítmica. El fenómeno de la degradación de rendimiento al llegar al límite de los 16 KB tiene grandes implicaciones en la latencia, ya que ésta crece un orden de magnitud, de 40 a 400 μs. Al igual que en ancho de banda, en latencia el rendimiento de Intel MPI es superior al de HP MPI sólo a partir de los 16 KB. La localización de los procesos respecto de las interfaces Infiniband tiene un efecto más notable en Intel MPI que en HP MPI, siendo diferencias típicas de 1 μs para tamaños de mensaje pequeños. Existe un pico de latencia con Intel MPI en la ejecución en las celdas 1 con un tamaño de mensaje de 16 bytes, achacable a algún fenómeno puntual como una interrupción hardware, ya que no se ha repetido en repeticiones de esta prueba. La latencia mínima la alcanzó HP MPI, con 5,94 μs.

Las figuras 5.9 y 5.10 muestran los resultados del test PingPing, con un efecto similar al visto en el test PingPong. En el límite de los 16 KB se produce una bajada de rendimiento, que es mucho más apreciable en HP MPI que en Intel MPI. Este test, al igual que en memoria compartida, obtiene un rendimiento de aproximadamente la mitad que el obtenido con el test PingPong, exhibiendo una gestión susceptible de grandes mejoras cuando las comunicaciones son bidireccionales y simultáneas. Nuevamente las diferencias entre las ejecuciones cercanas y lejanas a la tarjeta Infiniband son notables, pero ligeramente inferiores a las diferencias observadas en el test PingPong. También destaca la tendencia de Intel MPI en la celda 1 (sin tarjeta Infiniband) entre los 64 KB y los 8 MB, que obtiene mejor rendimiento que Intel MPI en la celda 0. Examinando resultados previos se puede concluir que es debido a algún fenómeno temporal. Otro efecto destacable es que el rendimiento de Intel MPI es superior al de HP MPI en prácticamente todas las situaciones.

La figura 5.11 refleja que Intel MPI sufre más los efectos de la distancia a la tarjeta Infiniband en el test PingPong. Sin embargo, en la figura 5.12 la situación es ligeramente la inversa, y ambas bibliotecas se ven afectadas prácticamente en igual medida.

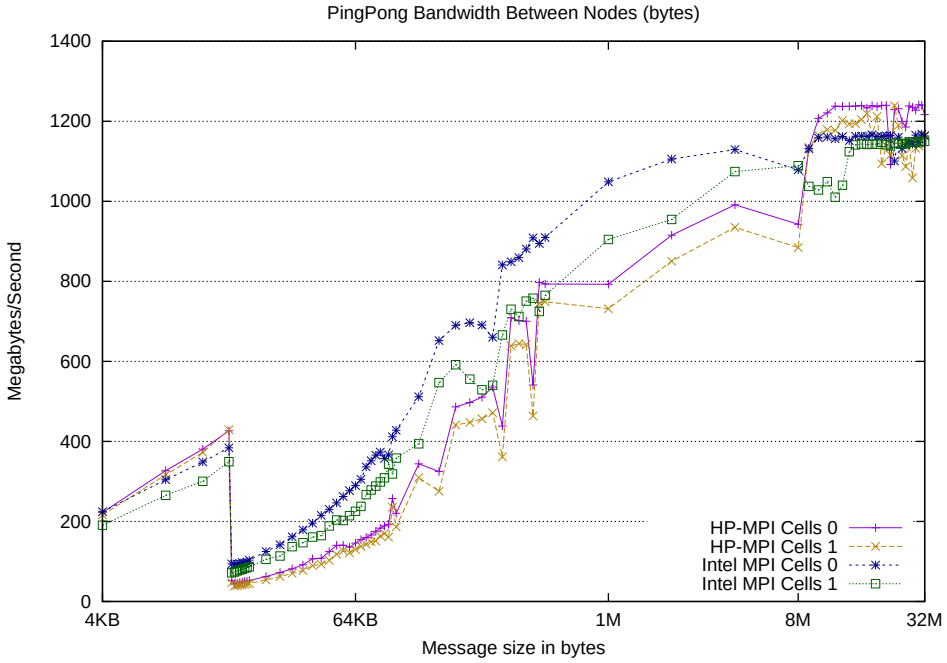


Figura 5.7: Ancho de Banda en PingPong con Infiniband

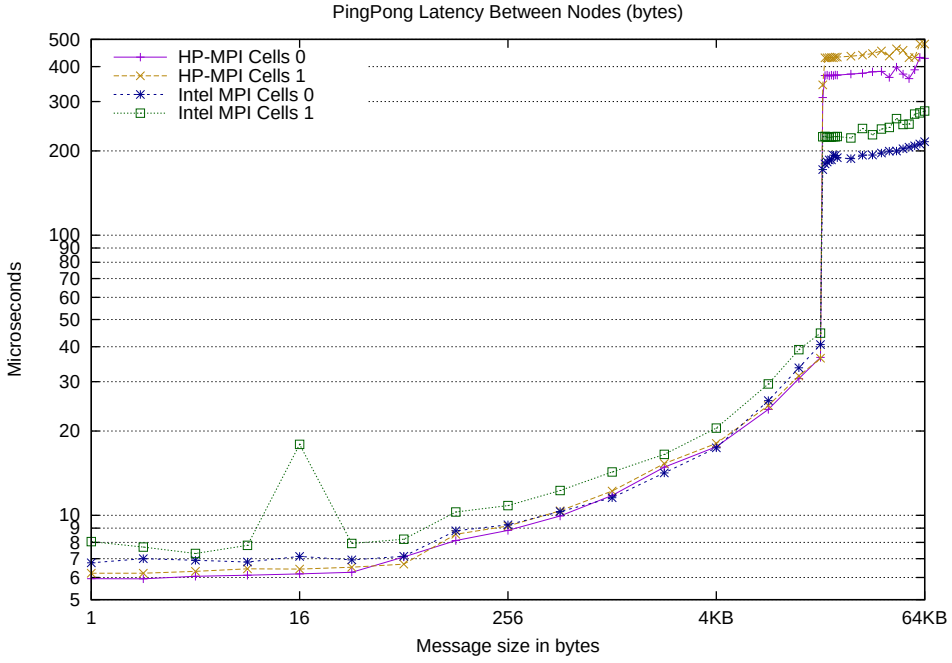


Figura 5.8: Latencia en PingPong con Infiniband

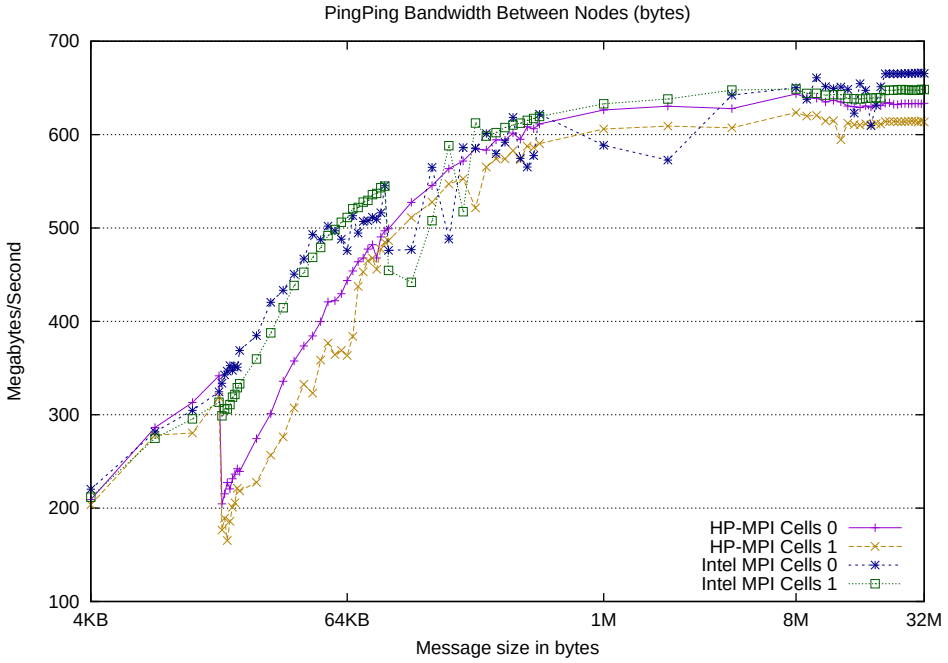


Figura 5.9: Ancho de Banda en PingPing con Infiniband

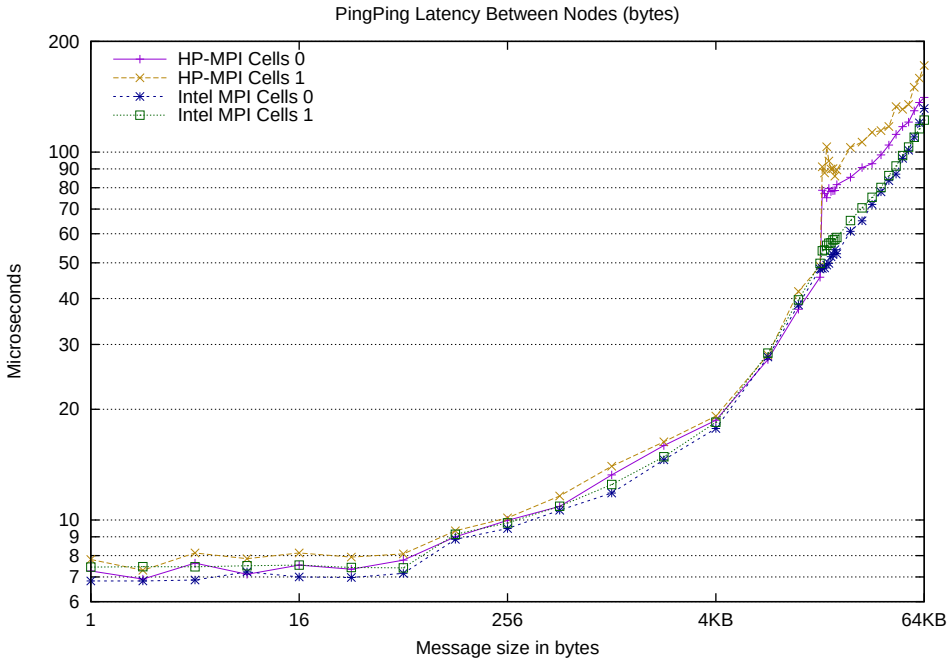


Figura 5.10: Latencia en PingPing con Infiniband

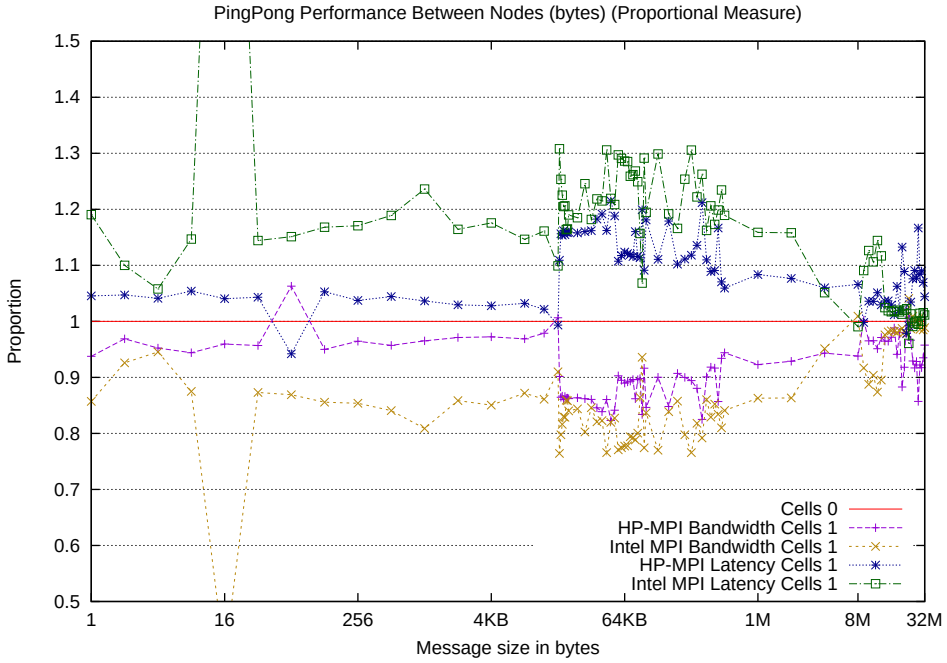


Figura 5.11: Rendimiento Relativo con PingPong e Infiniband

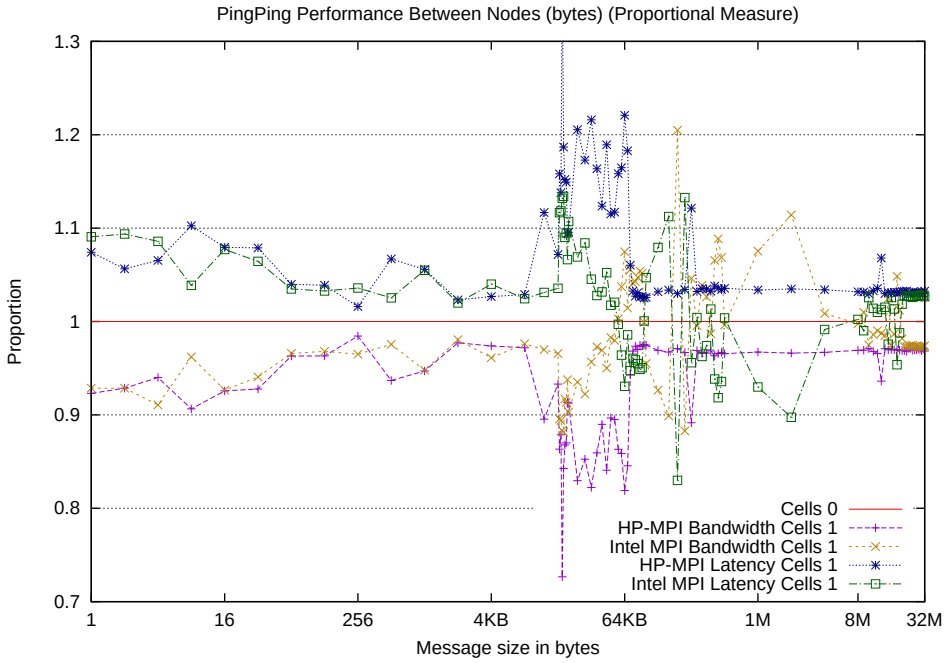


Figura 5.12: Rendimiento Relativo con PingPing e Infiniband

Efectos de los Algoritmos de Comunicaciones

La caída de rendimiento observada en los benchmarks de comunicaciones punto a punto es debida a los algoritmos de comunicaciones empleados por las bibliotecas. Las bibliotecas de comunicaciones MPI usan diferentes algoritmos según el tamaño del mensaje enviado. Así, para tamaños de mensaje pequeños se utiliza una aproximación *impaciente* (eager) para que el envío del mensaje se realice en una única comunicación. Para tamaños de mensaje grandes, debido a la posible falta de espacio en los buffers del proceso receptor y para evitar copias innecesarias de datos, es necesario un *handshaking*, por lo que la latencia aumenta al ser necesario un intercambio de mensajes, pero para tamaños de mensaje grandes lo que prima es el ancho de banda, y éste tiende a ser mejor. Este algoritmo se conoce con el nombre de rendezvous. En algunas arquitecturas este algoritmo permite además determinar dinámicamente la ruta de los paquetes, evitando los cuellos de botella ocasionales de la red [79]. Se han realizado anteriormente evaluaciones de diferentes algoritmos y optimizaciones [80].

El umbral para discernir entre mensajes de tamaño pequeño y tamaño grande se puede fijar manualmente, tanto en HP MPI como en Intel MPI, siendo el umbral por defecto de 16 KB. Los resultados mostrados a continuación fueron obtenidos con afinidad a la celda 1, para ver el rendimiento en la peor situación posible.

En las figuras 5.13 y 5.14 se aprecia el comportamiento de los distintos algoritmos de HP MPI e Intel MPI en el test PingPong, cuando el uso de dichos algoritmos es forzado manualmente. El comportamiento de los dos algoritmos de Intel MPI es el mismo hasta llegar a los 16 KB, de lo que se deduce que, a pesar de forzar manualmente a la biblioteca a usar el protocolo rendezvous, éste no entra en funcionamiento hasta los 16 KB. Por otro lado, la biblioteca HP MPI presentó diversos fallos de funcionamiento al modificar los umbrales, por lo que no se han podido realizar ejecuciones de menos de 128 bytes. En Intel MPI, el uso del protocolo rendezvous no aporta beneficios en ninguna situación. En HP MPI, el uso del algoritmo rendezvous parece el indicado a partir de 350 KB.

Las figuras 5.15 y 5.16 presentan resultados de anchos de banda y latencias para MPI PingPing, respectivamente, y muestran una situación inversa a la anterior, para Intel MPI, puesto que el uso del protocolo eager obtiene peor resultado siempre. En HP MPI el punto de corte entre el protocolo eager y el protocolo rendezvous se establece en los 64 KB.

Cabe destacar que en diversas pruebas realizadas con HP MPI, el uso de la opción `-dd`, que implica una liberación de buffers en plazos, resultó en una mejora sustancial del protocolo rendezvous. Otro hecho destacable es que el rendimiento de los distintos protocolos se ve muy afectado por la invalidación de caché con la que se ejecutaron estas pruebas. Por este motivo las ejecuciones sin invalidación de caché puede resultar en distintos rendimientos brutos y puntos de corte entre algoritmos.

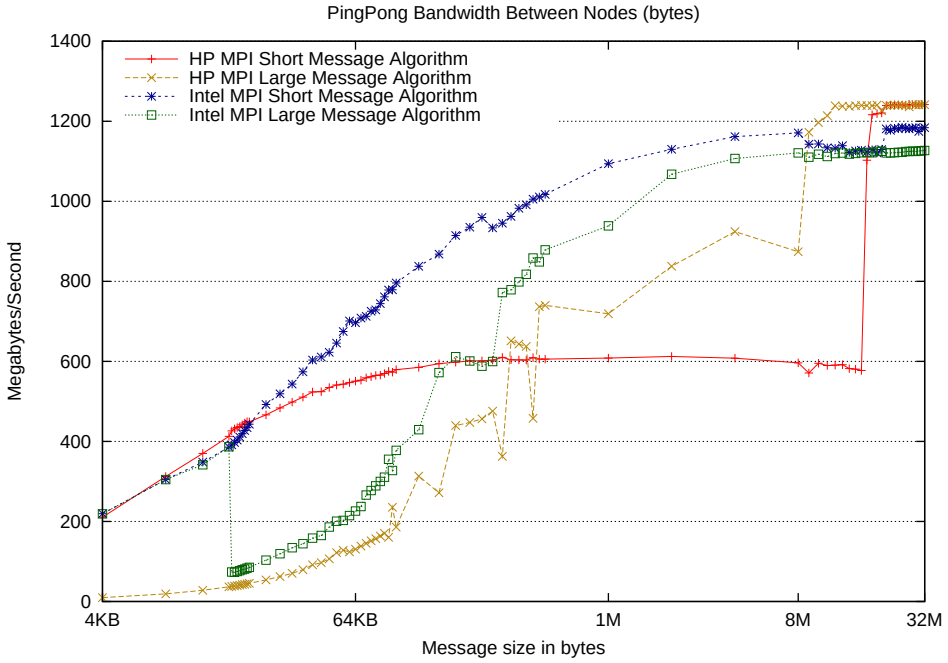


Figura 5.13: Efectos de los Algoritmos en el Ancho de Banda con PingPong

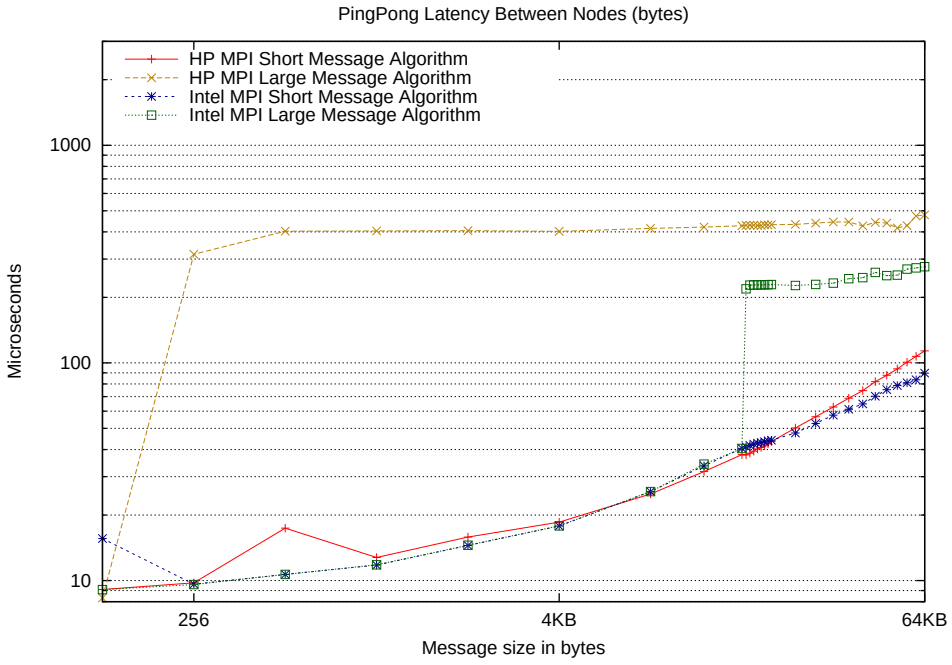


Figura 5.14: Efectos de los Algoritmos en la Latencia con PingPong

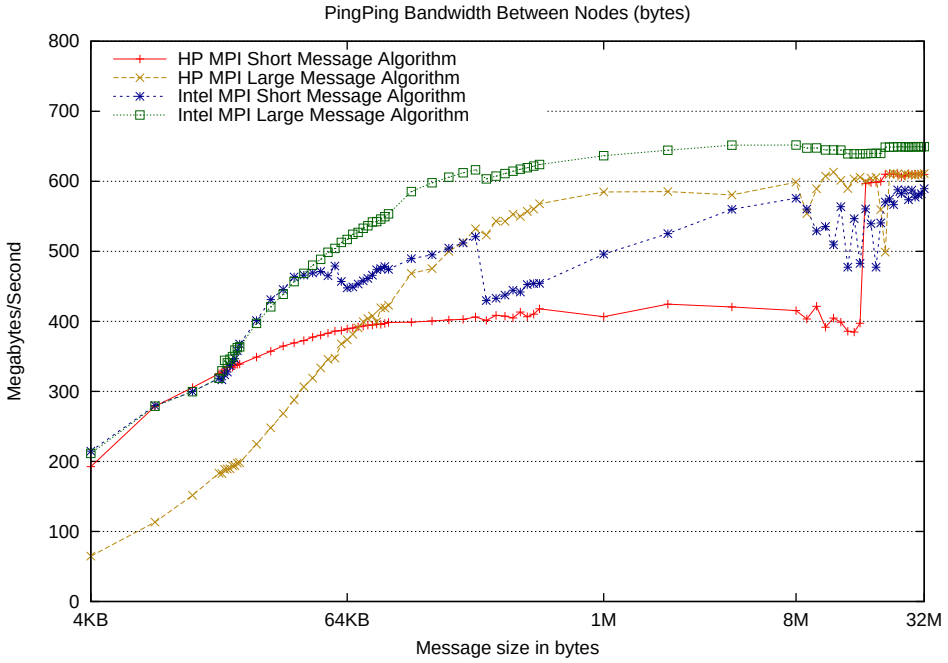


Figura 5.15: Efectos de los Algoritmos en el Ancho de Banda con PingPing

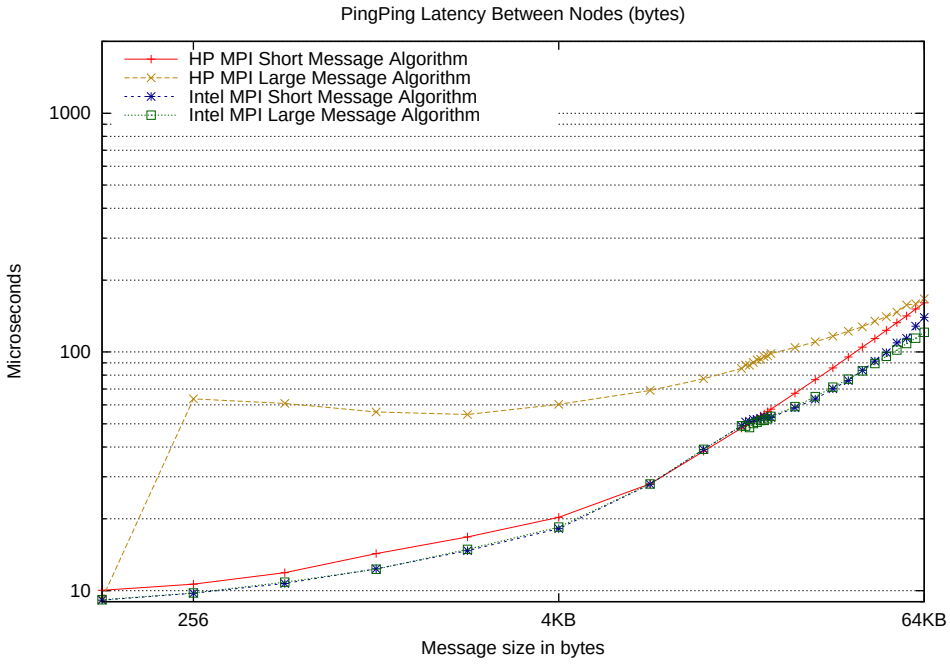


Figura 5.16: Efectos de los Algoritmos en la Latencia con PingPing

5.2. Operaciones Colectivas

A continuación se mostrarán los distintos resultados obtenidos en la evaluación de las operaciones colectivas. Para ello se han usado 64 nodos, repartiendo los procesos lo máximo posible entre los nodos, es decir, usando 1 proceso por nodo hasta 64 procesos, 2 procesos por nodo para 128 procesos, etcétera. Las gráficas muestran resultados de ejecuciones con afinidad a la celda 0 o a la celda 1, excepto en el caso de la ejecución con 1024 procesos, en la que se usan ambas celdas. También se mostrará la diferencia porcentual entre el rendimiento alcanzado por las ejecuciones con afinidad a la celda 0 y la celda 1. Por último, se analizará la escalabilidad de ambas bibliotecas para un tamaño de mensaje fijo, incrementando el número de procesos.

5.2.1. Colectiva Alltoall

Las figuras 5.17 y 5.18 muestran los resultados alcanzados con HP MPI. Ambas gráficas son similares, con un rendimiento casi constante hasta llegar a 128 bytes, punto a partir del cual el tiempo de comunicación crece de forma lineal conforme aumenta el tamaño de mensaje. Se aprecia un aumento del tiempo de comunicación que se sale del patrón descrito, al llegar a los 16 KB. Esto es debido al cambio de algoritmo que ya se ha analizado. Al utilizar 64 procesos se aprecia un ligero escalón en el rendimiento al llegar a los 512 bytes. Dicho escalón se vuelve más pronunciado y se produce antes según se aumenta el número de nodos. Así, para 128 procesos el escalón se sitúa en 256 bytes, para 256 procesos en 128 bytes y así sucesivamente. Otro efecto destacable se aprecia al hacer un análisis comparativo entre las figuras 5.17 y 5.18. Al observar el rendimiento de la ejecución con 512 procesos se aprecia una gran diferencia entre la ejecución en la celda 0 y la ejecución en la celda 1, hasta llegar a los 64 bytes, momento en el que se produce el escalón comentado anteriormente. Este hecho destaca especialmente por ser la única diferencia significativa apreciable entre gráficas.

En las figuras 5.19 y 5.20 se muestran los resultados de Intel MPI. A partir de 1 KB los resultados obtenidos con la celda 0 y la celda 1 son muy semejantes. Sin embargo, a diferencia de HP MPI, antes de 1 KB existen irregularidades que resultan en un peor rendimiento de las ejecuciones en la celda 1, especialmente para 64 procesos. También resulta llamativa la diferencia existente para tamaños pequeños, entre las ejecuciones con 4 y 8 procesos, y la diferencia observada en ese mismo tramo para 512 procesos, entre las ejecuciones de distintas celdas, diferencia también observada en HP MPI, lo que indica que no es un problema de la biblioteca. Destacan los picos observados en los mensajes de 512 bytes y un número de procesos superior a 8. A partir de 256 procesos no se han podido obtener resultados superiores a 256 bytes, debido a un fallo de la interfaz DAPL[1]. El rendimiento general de Intel MPI es superior al de HP MPI para tamaños pequeños, llegando a ser el triple para mensajes cortos. Para tamaños grandes los resultados de ambas bibliotecas se homogeneizan, e incluso HP MPI muestra una recuperación más pronta de la caida de rendimiento observada en los 16 KB.

Las figuras 5.21 y 5.22 muestran la diferencia de rendimiento en términos porcentuales, entre ejecutar en la celda 0 y en la celda 1. Las diferencias típicas para HP MPI

[1]DAPL: Direct Access Programming Library

son de entre el 1 % y el 5 %. Para Intel MPI estas diferencias típicas se mantienen en los mismos valores, pero existen unas oscilaciones mucho mayores.

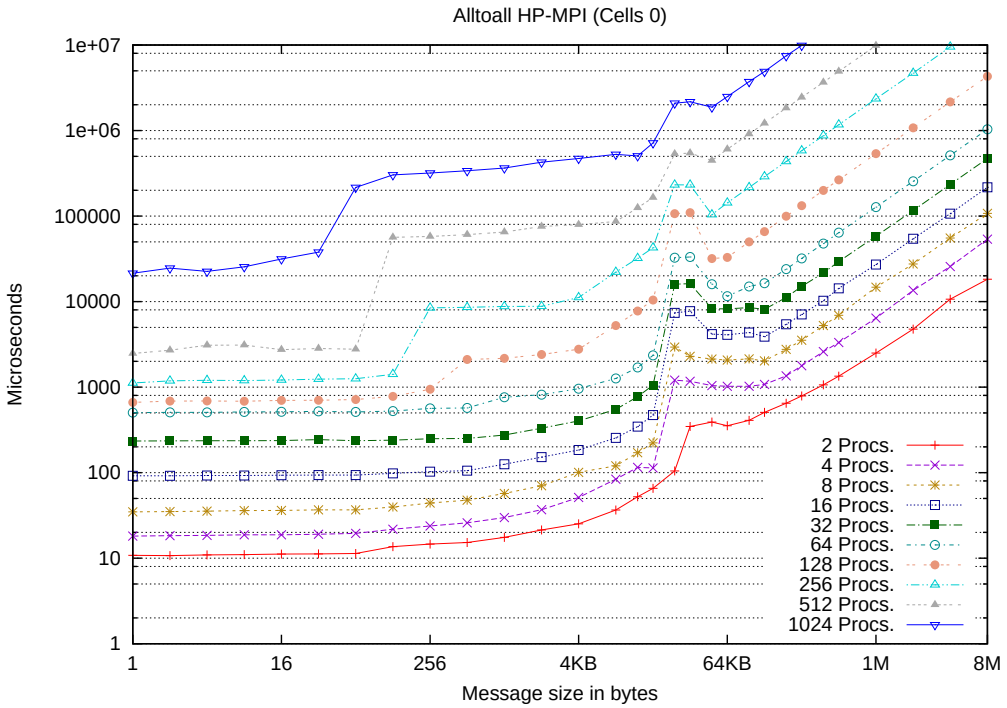


Figura 5.17: Colectiva Alltoall con HP-MPI y afinidad a la Celda 0

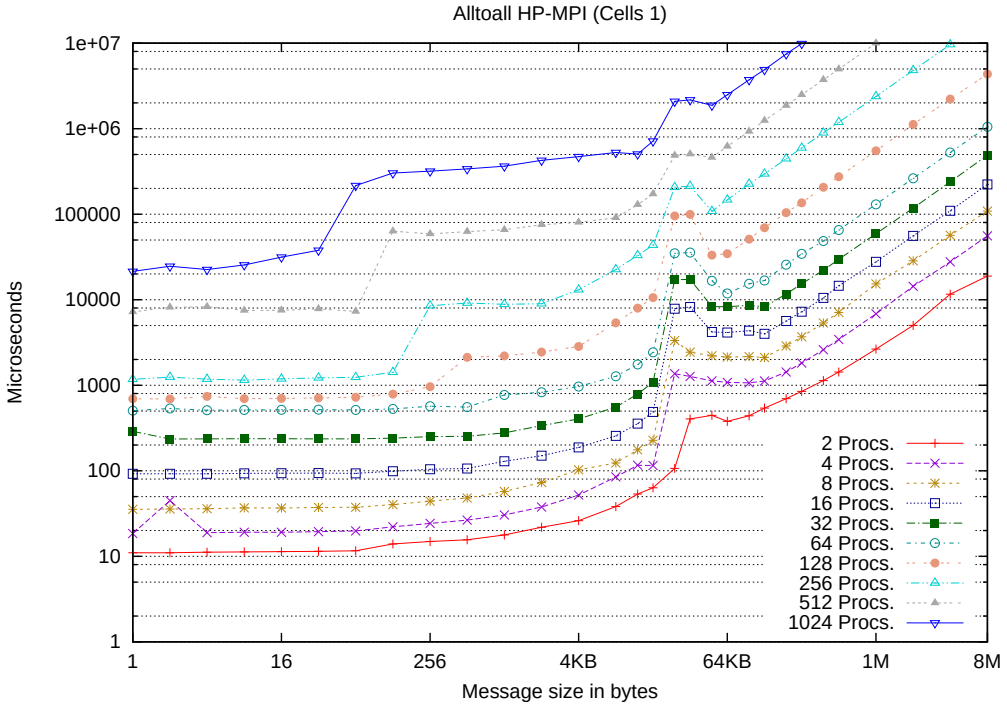


Figura 5.18: Colectiva Alltoall con HP-MPI y afinidad a la Celda 1

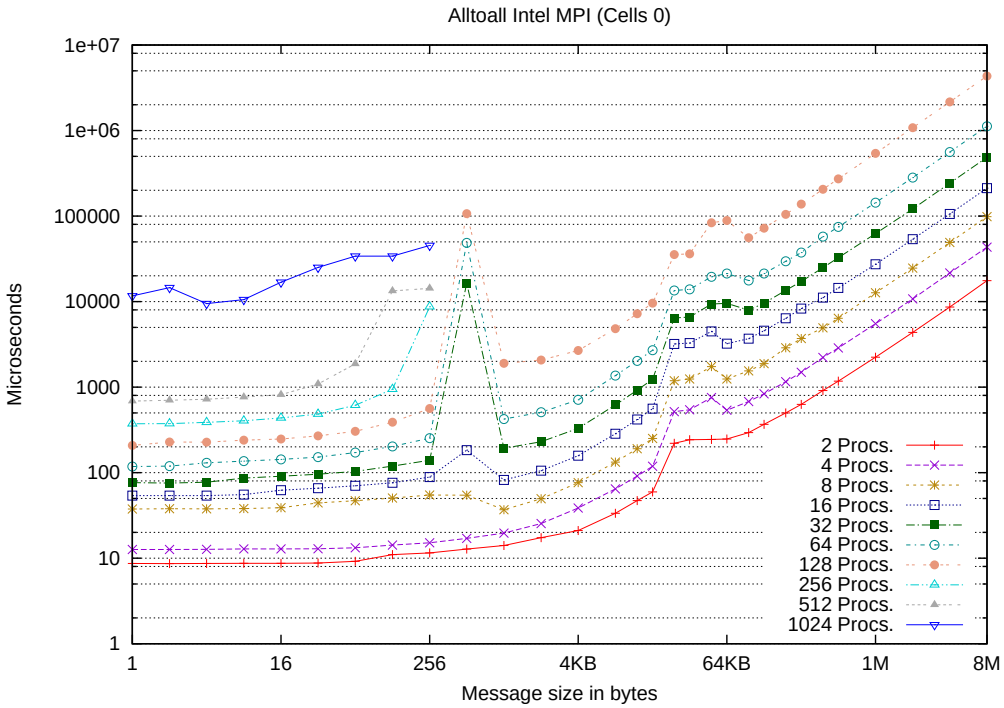


Figura 5.19: Colectiva Alltoall con Intel MPI y afinidad a la Celda 0

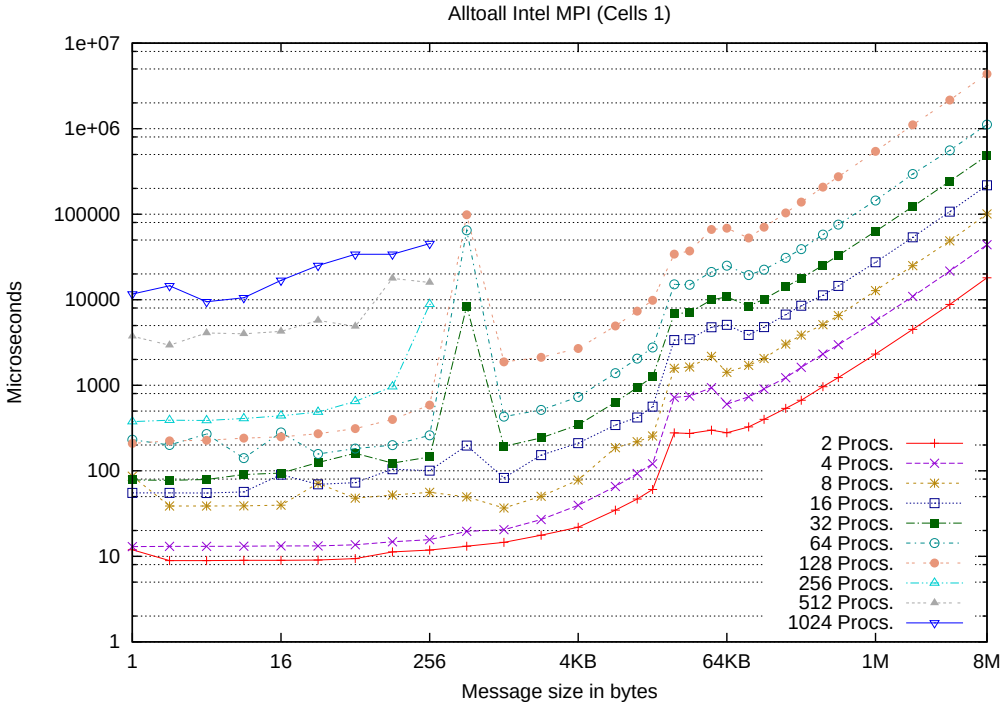


Figura 5.20: Colectiva Alltoall con Intel MPI y afinidad a la Celda 1

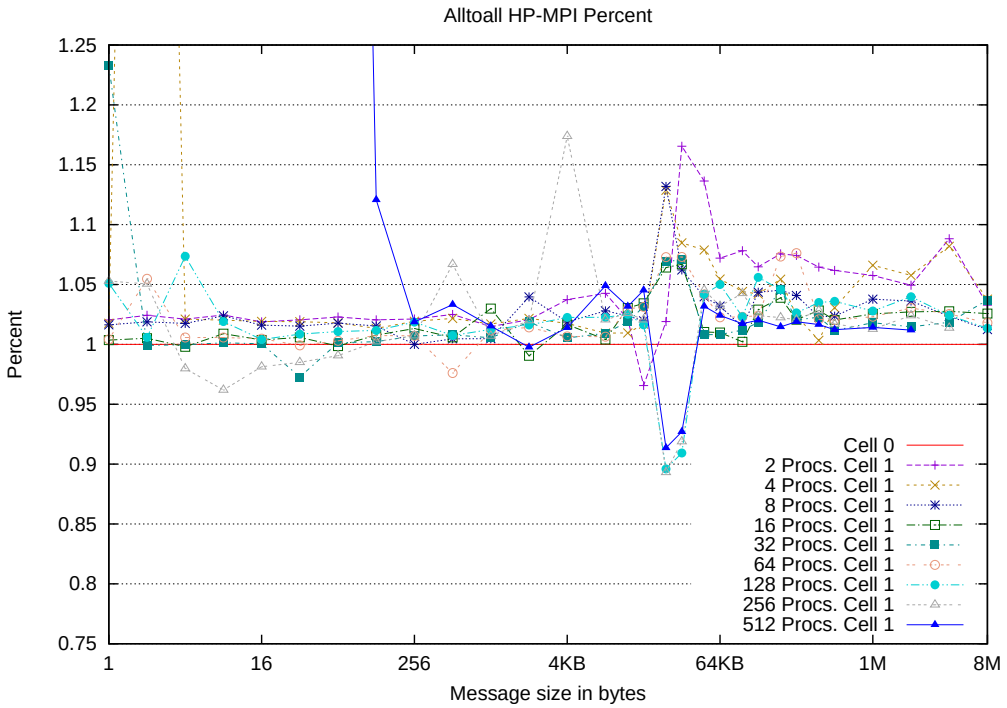


Figura 5.21: Tiempo Relativo de la Colectiva Alltoall con HP-MPI

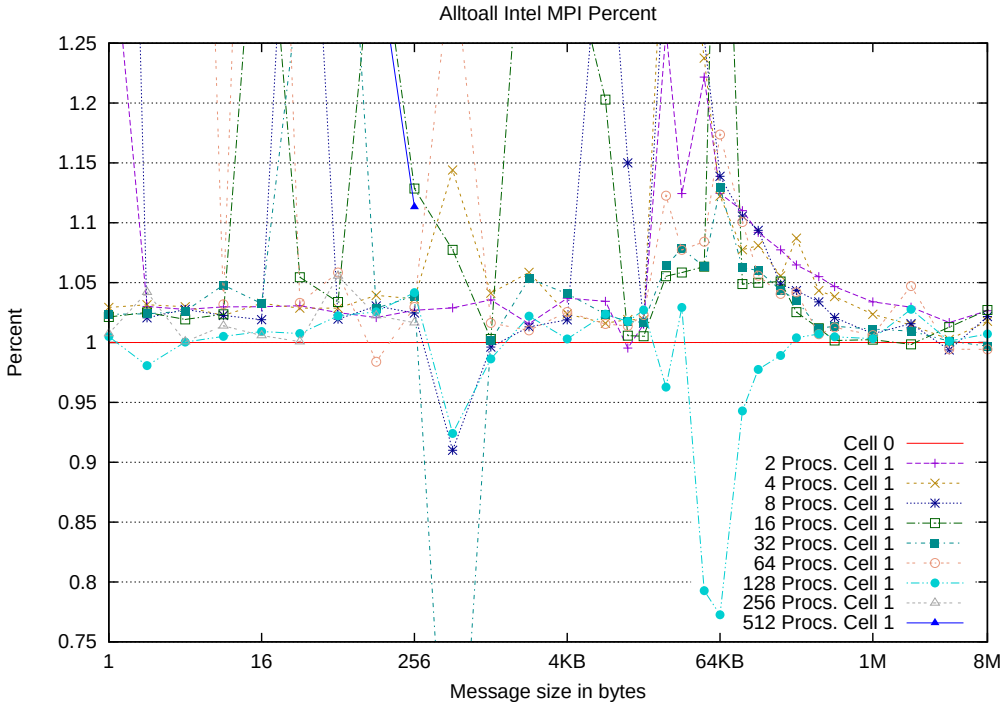


Figura 5.22: Tiempo Relativo de la Colectiva Alltoall con Intel MPI

Escalabilidad de la Colectiva Alltoall

La escalabilidad de las operaciones colectivas en ambas bibliotecas es buena para mensajes de 1 byte, tal y como se ve en la figura 5.23, si bien el comportamiento de Intel MPI es mejor que el de HP MPI, aumentando la diferencia según aumenta el número de procesos. Este dato, puesto en perspectiva con los resultados obtenidos en las pruebas punto a punto en memoria compartida y con red Infiniband, revela que la diferencia entre la eficiencia de ambas bibliotecas en red Infiniband (a favor de Intel MPI), es mayor que la diferencia de eficiencia en memoria compartida (a favor de HP MPI), ya que, a pesar de que a medida que aumenta el número de nodos aumentan las comunicaciones intranodo (lo que beneficia a HP MPI), la diferencia sigue creciendo a favor de Intel MPI. En ambas bibliotecas se aprecia una tendencia casi lineal, de forma que el tiempo con HP MPI aumenta en un orden ligeramente superior a N, siendo N el tamaño del mensaje, mientras que Intel MPI aumenta en un orden ligeramente inferior a N. En cuanto a la diferencia entre ejecutar en la celda 0 y la celda 1, la mayor diferencia se encuentra en la ejecución con 512 procesos, siendo ésta mucho más marcada en Intel MPI.

Para mensajes de 4 KB e Intel MPI sólo se pudieron completar las ejecuciones de hasta 128 procesos. Los resultados son similares en ambas bibliotecas, siendo ligeramente mejores los de Intel MPI. Esto se puede ver en la figura 5.24. La tendencia es lineal en ambos casos, hasta llegar a 64 procesos. A partir de este punto la pendiente se vuelve más empinada, aumentando en un orden de $2N$, y se va aumentando a medida que aumenta el número de procesos.

La figura 5.25 muestra la escalabilidad para mensajes de 64 KB. El análisis es muy similar al anterior, con la salvedad de que Intel MPI obtiene peor rendimiento que HP MPI a partir de 16 procesos, lo que indica que HP MPI es una implementación más potente para grandes requerimientos.

En la figura 5.26 se ve la escalabilidad para mensajes de 8 MB. Tan sólo se han podido realizar ejecuciones hasta 128 procesos, obteniéndose unos resultados muy similares para ambas bibliotecas. El análisis es, nuevamente, similar a los anteriores.

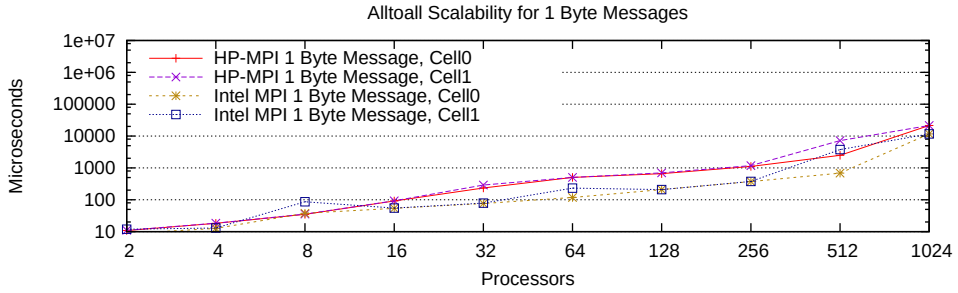


Figura 5.23: Escalabilidad de la Colectiva Alltoall para Mensajes de 1 Byte

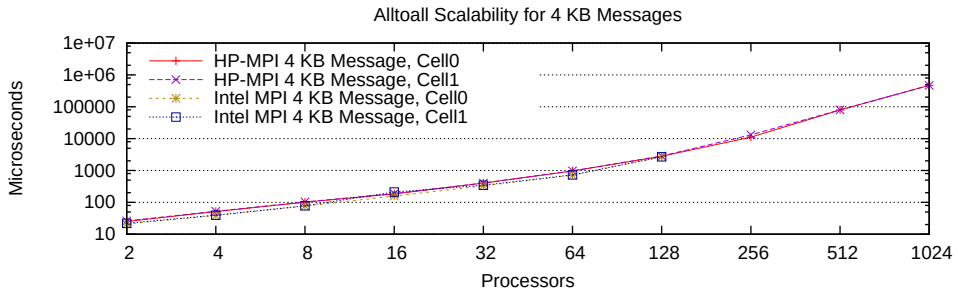


Figura 5.24: Escalabilidad de la Colectiva Alltoall para Mensajes de 4 Kilobytes

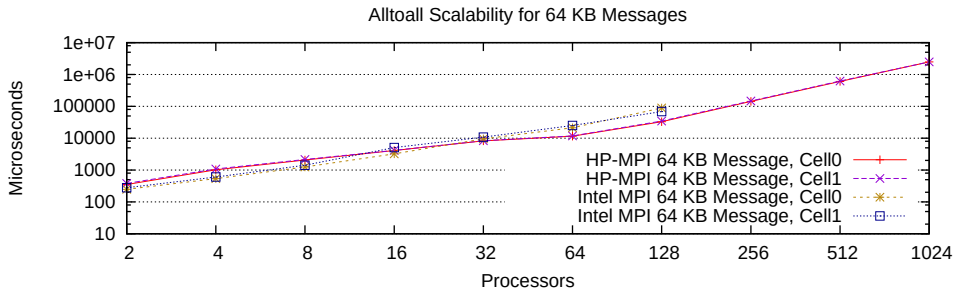


Figura 5.25: Escalabilidad de la Colectiva Alltoall para Mensajes de 64 Kilobytes

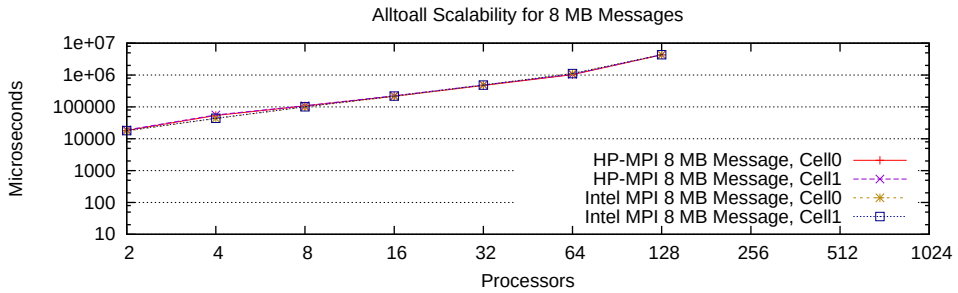


Figura 5.26: Escalabilidad de la Colectiva Alltoall para Mensajes de 8 Megabytes

5.2.2. Colectiva Broadcast

Las figuras 5.27 y 5.28 muestran el comportamiento de la colectiva *broadcast* con la biblioteca HP MPI. En ellas se ve que para mensajes pequeños el tiempo invertido es superior en las ejecuciones de 32 y 64 procesos que en las ejecuciones de 128, 256 y 512 procesos, lo que puede ser debido a las eficientes comunicaciones intranodo de HP MPI. En las ejecuciones en la celda 1 con 512 procesos esta tendencia no existe, y se obtienen rendimientos irregulares. Hasta llegar a 64 bytes el tiempo es casi constante, mientras que a partir de ese punto va aumentando progresivamente hasta aumentar con el mismo orden que el tamaño de mensaje. Otro hecho destacable es la irregularidad de los rendimientos obtenidos con 1024 procesos.

Los resultados obtenidos con Intel MPI se muestran en las figuras 5.29 y 5.30. En ellas no figuran los resultados obtenidos con 1024 procesos ya que no se han podido realizar dichas ejecuciones. En los 16 KB se aprecia el escalón provocado por el cambio de algoritmo, que, como viene siendo habitual, es menos pronunciado que en HP MPI. Su comportamiento general comienza con un duración de la comunicación casi constante para tamaños pequeños, hasta llegar a un punto en donde comienza una tendencia hacia la linearidad con respecto al tamaño de mensaje. Llama la atención el espacio existente entre la ejecución con 64 procesos y la de 128, mucho más pronunciada que el resto. Además, el rendimiento obtenido con las ejecuciones con un número mayor de 64 procesos es constante hasta un tamaño de 2 KB, cuando en las ejecuciones con número menor de procesos sufrían el cambio de tendencia mucho antes, a los 64 bytes. El comportamiento en las ejecuciones en la celda 1 es muy similar al de las ejecuciones en la celda 0, con la diferencia de que las ejecuciones se vuelven ligeramente inestables, especialmente con 512 procesos, que tiene numerosos picos y que sufre mucho el efecto del cambio de algoritmo.

Las diferencias en términos porcentuales se pueden consultar en las figuras 5.31 y 5.32. Tanto en HP MPI como en Intel MPI se ven pronunciadas irregularidades, aunque son más acentuadas en ésta última. La diferencia de rendimiento se sitúa típicamente entre un 1 % y un 5 % en el caso de HP MPI, y entre un 1 % y un 10 % para Intel MPI, para mensajes menores de 16 KB. Para mensajes grandes la diferencia es mayor, entre un 4 % y un 15 % para HP MPI, y entre un 5 % y un 25 % para Intel MPI.

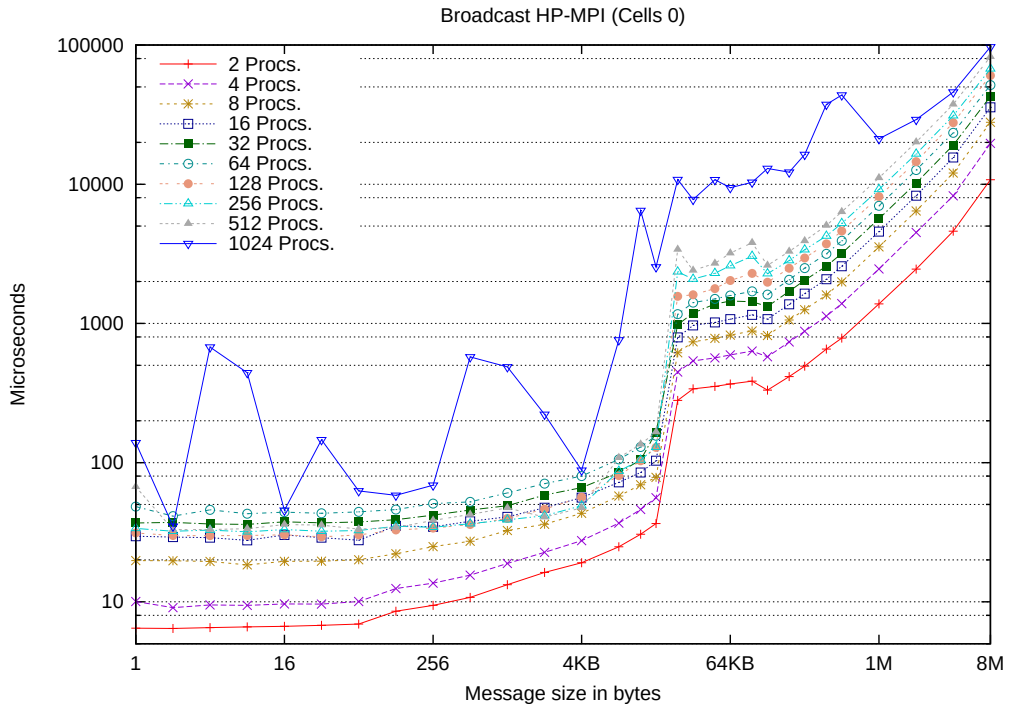


Figura 5.27: Colectiva Broadcast con HP-MPI y afinidad a la Celda 0

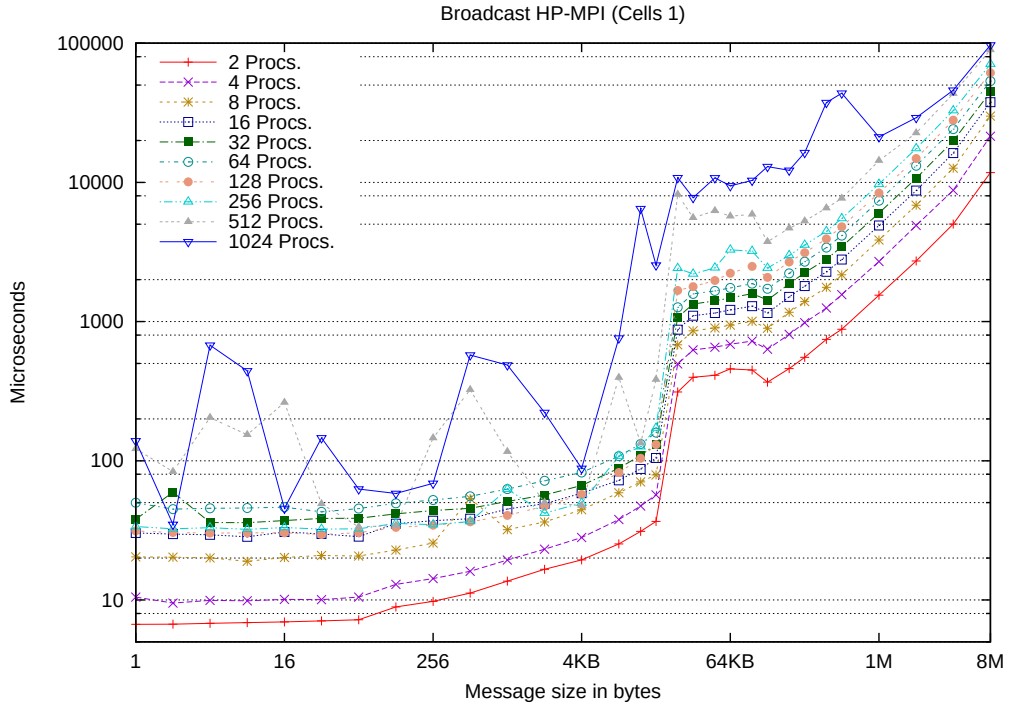


Figura 5.28: Colectiva Broadcast con HP-MPI y afinidad a la Celda 1

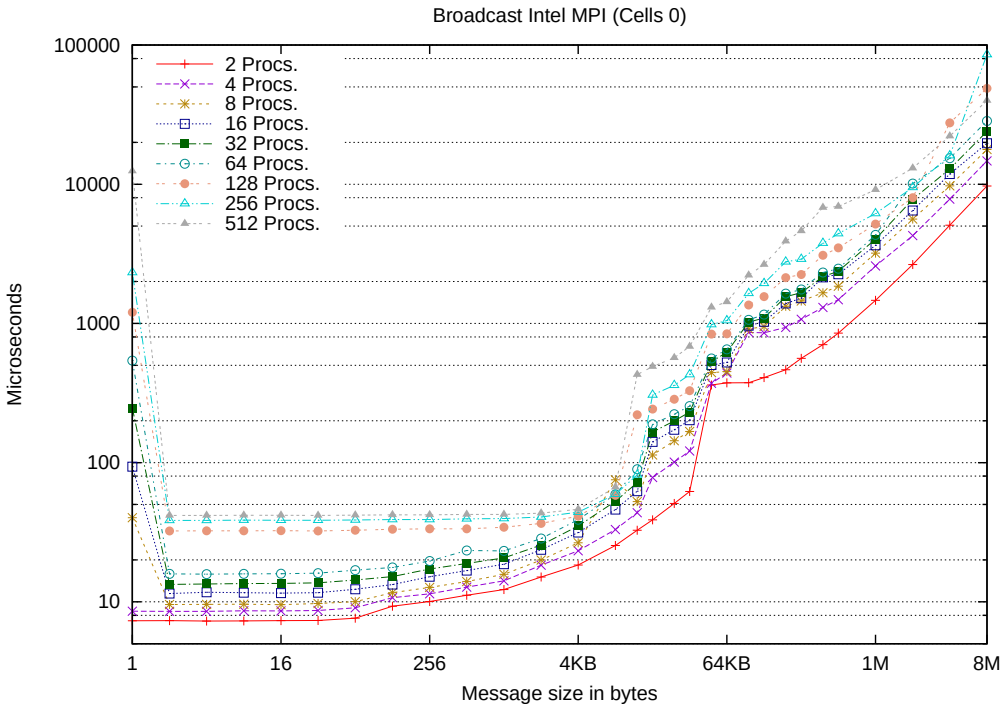


Figura 5.29: Colectiva Broadcast con Intel MPI y afinidad a la Celda 0

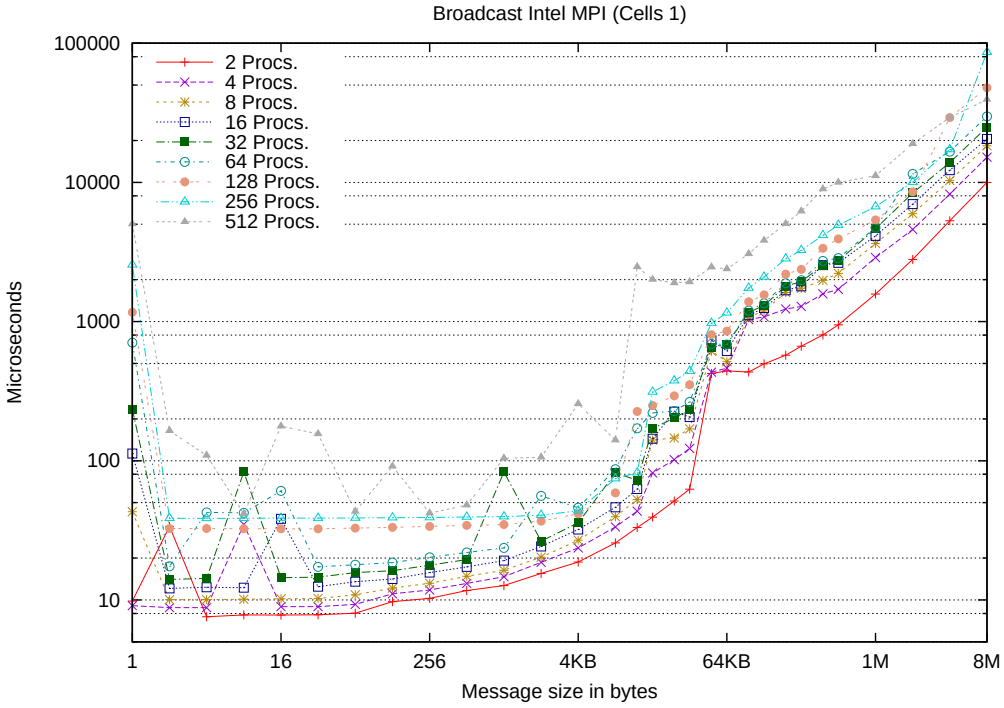


Figura 5.30: Colectiva Broadcast con Intel MPI y afinidad a la Celda 1

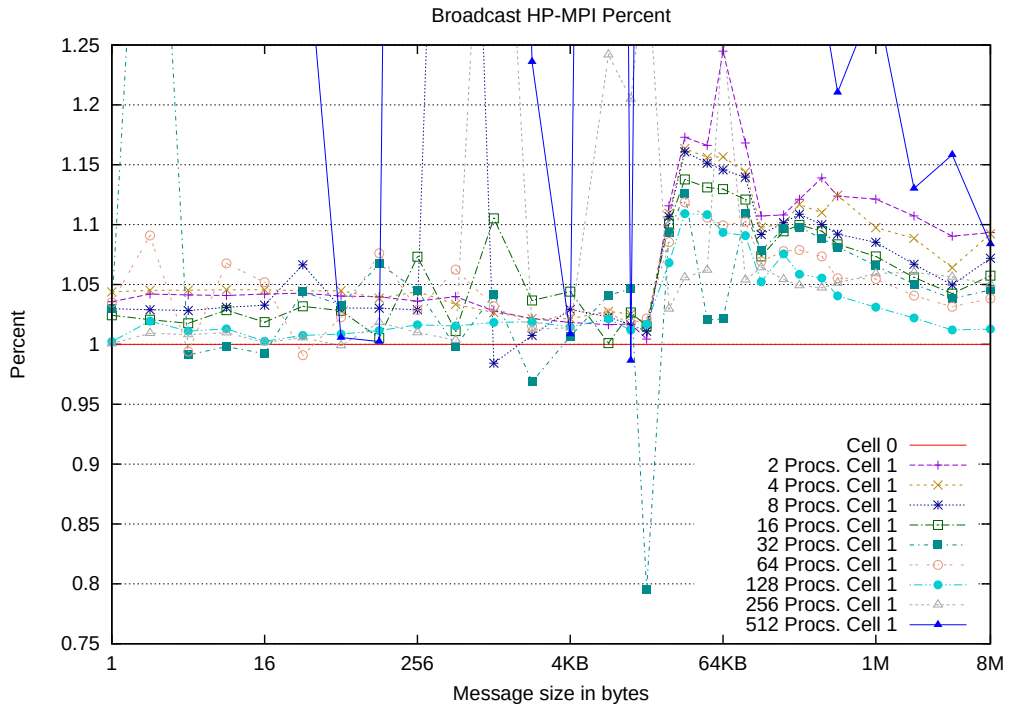


Figura 5.31: Tiempo Relativo de la Colectiva Broadcast con HP-MPI

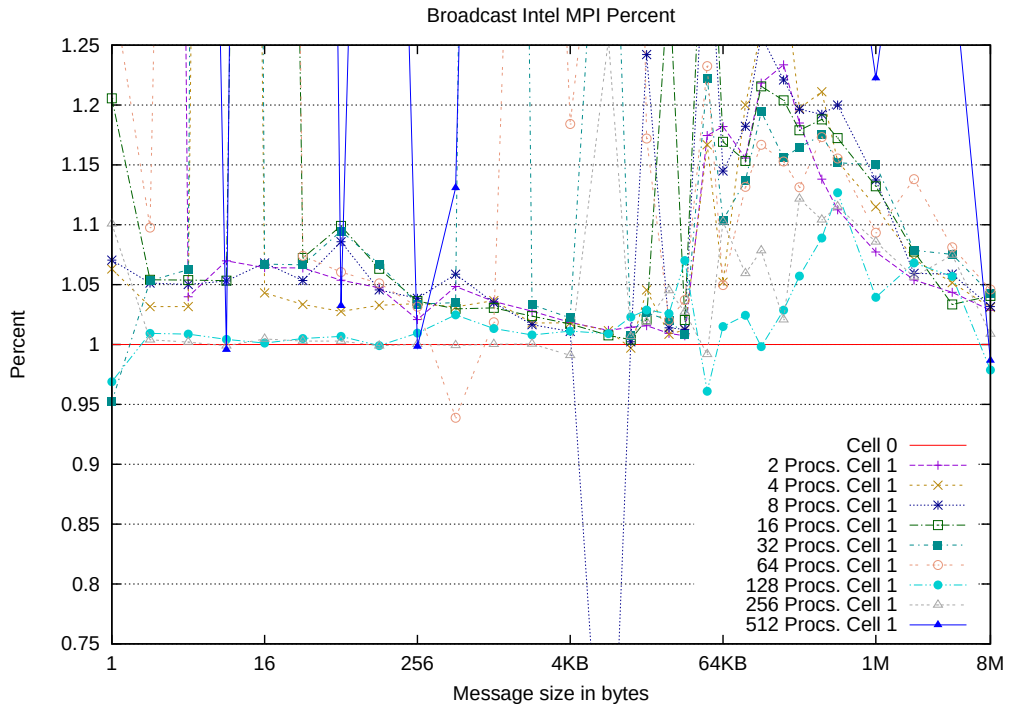


Figura 5.32: Tiempo Relativo de la Colectiva Broadcast con Intel MPI

Escalabilidad de la Colectiva Broadcast

La figura 5.33 muestra la escalabilidad de la colectiva broadcast con mensajes de 1 byte en las bibliotecas HP MPI e Intel MPI. Los resultados obtenidos muestran una variabilidad pequeña entre ejecuciones en la celda 0 y la celda 1, pero sin embargo la diferencia observada entre HP MPI e Intel MPI es muy grande, llegando a ser de dos órdenes de magnitud, a favor de HP MPI. El aumento del tiempo invertido para las comunicaciones es del orden de $2N$ para Intel MPI, mientras que para HP MPI tiene una tendencia logarítmica.

Los resultados obtenidos para ejecuciones con 4KB son más parejos que los vistos anteriormente, tal y como se puede observar en la figura 5.34. Estos datos son más congruentes con los obtenidos hasta ahora, ya que Intel MPI rinde ligeramente mejor que HP MPI. La escalabilidad de ambos es buena, ya que no se aumenta el tiempo ni en un orden de magnitud, entre la ejecución con 2 procesos y la ejecución con 1024 procesos. Destaca el repentino pico en la duración de las comunicaciones con Intel MPI y celda 1.

Para 64 KB (figura 5.35) Intel MPI sigue obteniendo mejores resultados que HP MPI. El tiempo de comunicaciones en general ha aumentado, y la pendiente que describe la escalabilidad de ambas bibliotecas se ha acentuado ligeramente, pasando de 367 μs para 2 procesos a 9439 μs para 1024 procesos, en el peor de los casos.

Por último, para 8 MB (figura 5.36) Intel MPI sigue rindiendo mejor que HP MPI, que pasa de unos 10750 μs para 2 procesos a 96925 μs para 1024 procesos. Un efecto observable para las ejecuciones de Intel MPI en ambas celdas es que el rendimiento con 512 procesos es mayor que con 256.

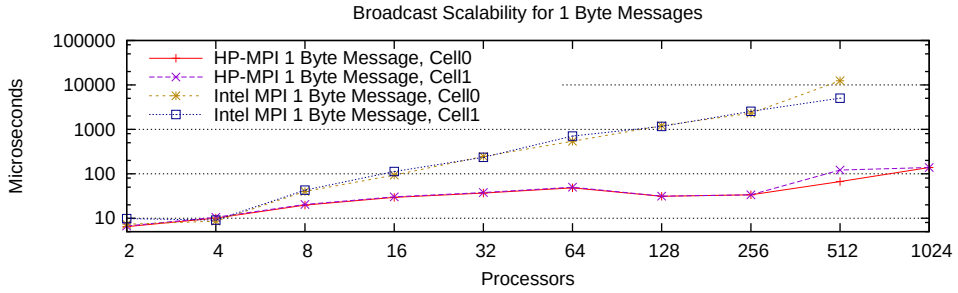


Figura 5.33: Escalabilidad de la Colectiva Broadcast para Mensajes de 1 Byte

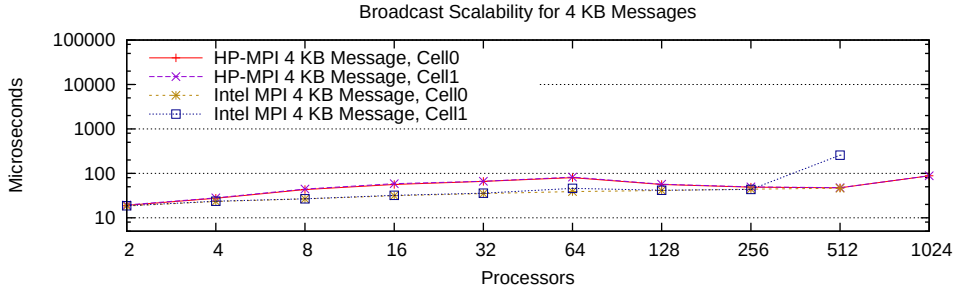


Figura 5.34: Escalabilidad de la Colectiva Broadcast para Mensajes de 4 Kilobytes

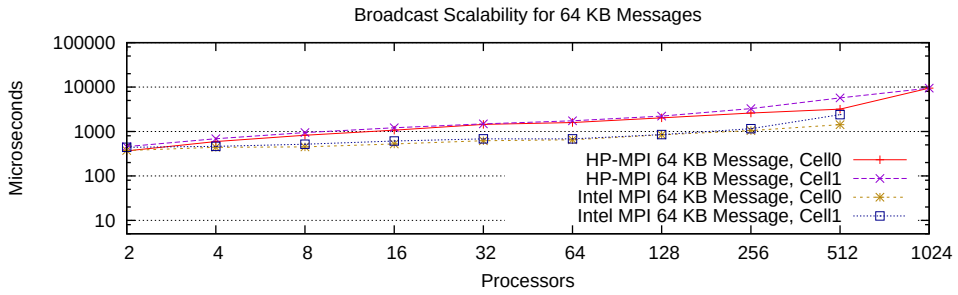


Figura 5.35: Escalabilidad de la Colectiva Broadcast para Mensajes de 64 Kilobytes

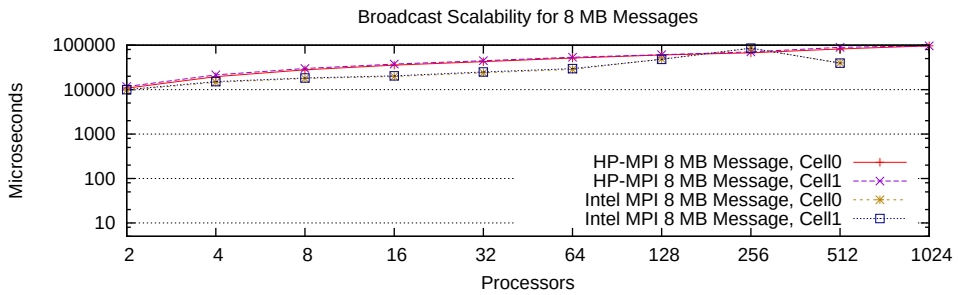


Figura 5.36: Escalabilidad de la Colectiva Broadcast para Mensajes de 8 Megabytes

5.3. Configuraciones Híbridas

En esta sección se analizará la mejor configuración para ejecuciones híbridas, usando los NAS Parallel Benchmarks Multi-Zone, con programación híbrida MPI+OpenMP [64]. Para ello, tomando como base un número total de threads (16, 32, 64 o 128), se han realizado pruebas para evaluar qué configuración obtiene más rendimiento, si una consistente en usar únicamente procesos MPI hasta alcancar el total de threads, una consistente en utilizar un único proceso MPI por nodo y 16 threads OpenMP por nodo, o una aproximación intermedia.

En las gráficas se muestran los resultados con 1, 2, 4, 8 y 16 threads por cada proceso MPI, con HP MPI e Intel MPI. El número máximo de procesos MPI en LU-MZ es 16, por eso no hay datos en las configuraciones que requieren más procesos MPI. Además, no se han podido realizar las ejecuciones de 32 procesos repartidos en 2 nodos con HP-MPI debido a un error software.

Las figuras 5.37, 5.38 y 5.39 muestran los resultados para BT-MZ, LU-MZ y SP-MZ con clase C. En ellas se ve una tendencia general en la que el mayor rendimiento se obtiene usando únicamente procesos MPI. La diferencia de rendimiento es especialmente grande en SP-MZ. Existen excepciones, como LU-MZ con 16 y 32 procesos/threads en total, en los que el mayor rendimiento se obtiene con 4 threads por cada proceso MPI. Otra excepción es BT-MZ con 128 procesos/threads en total, para el cual la mejor combinación es usar 2 threads por cada proceso MPI y 64 procesos MPI. No hay diferencias entre HP MPI e Intel MPI en cuanto a configuraciones óptimas.

Los resultados alcanzados con la clase D se pueden consultar en las figuras 5.40, 5.41 y 5.42. En ellas se observa como la tendencia anterior se suaviza, hasta el punto de que para 128 threads totales el rendimiento máximo se obtiene usando 2 threads por proceso MPI (en total 64 procesos MPI), tanto para BT-MZ como para SP-MZ. Llama la atención de los resultados de LU-MZ, ya que la tendencia se ha invertido totalmente, y la mejor configuración es la que usa un único proceso MPI por nodo y 16 threads OpenMP por proceso MPI.

En este estudio se ha demostrado que la configuración óptima es altamente dependiente del algoritmo de cálculo y del tamaño del problema, y es congruente con estudios anteriores [81]. Además, la tendencia a suavizar las diferencias e incluso a hacer que el modelo híbrido rinda mejor que una aproximación MPI pura está justificada en un estudio realizado sobre el Earth Simulator (el supercomputador más potente del mundo desde Junio de 2002 hasta Junio de 2004) [82], en donde se apunta que a medida que aumenta el número de nodos las comunicaciones se vuelven más importantes, y por tanto minimizarlas usando menos procesos MPI por nodo resulta en un mejor rendimiento.

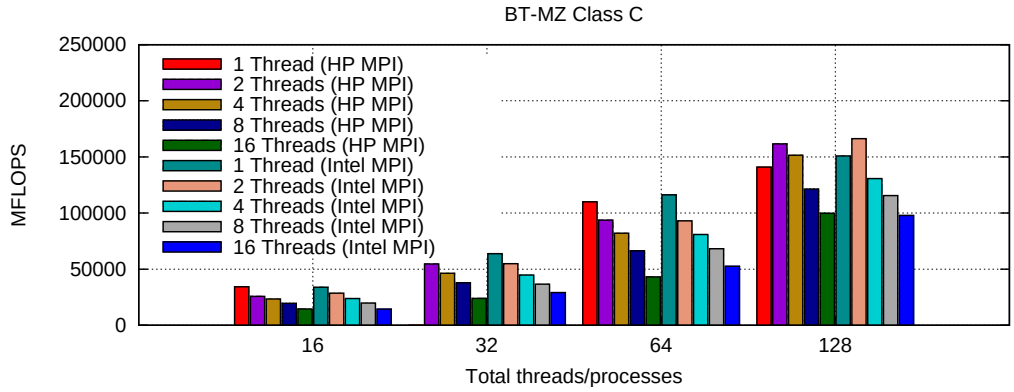


Figura 5.37: Rendimiento de Configuraciones Híbridas en BT con clase C

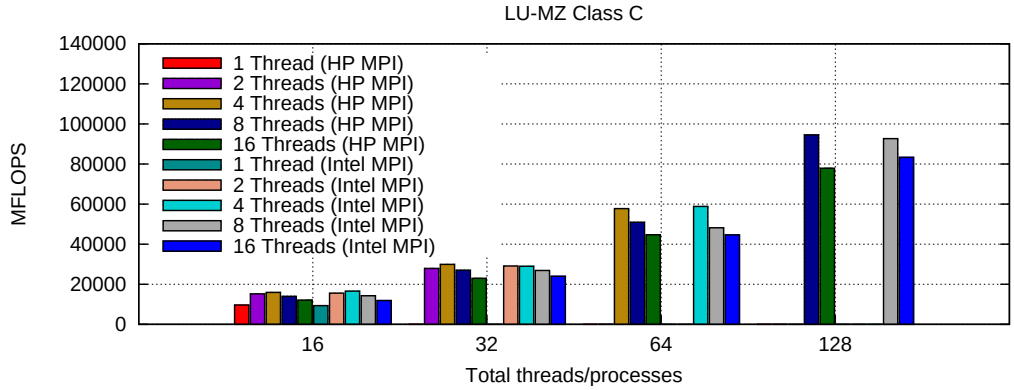


Figura 5.38: Rendimiento de Configuraciones Híbridas en LU con clase C

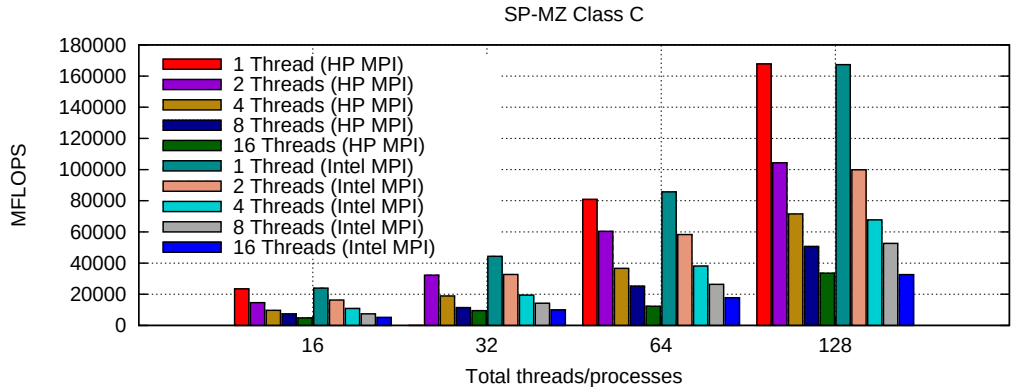


Figura 5.39: Rendimiento de Configuraciones Híbridas en SP con clase C

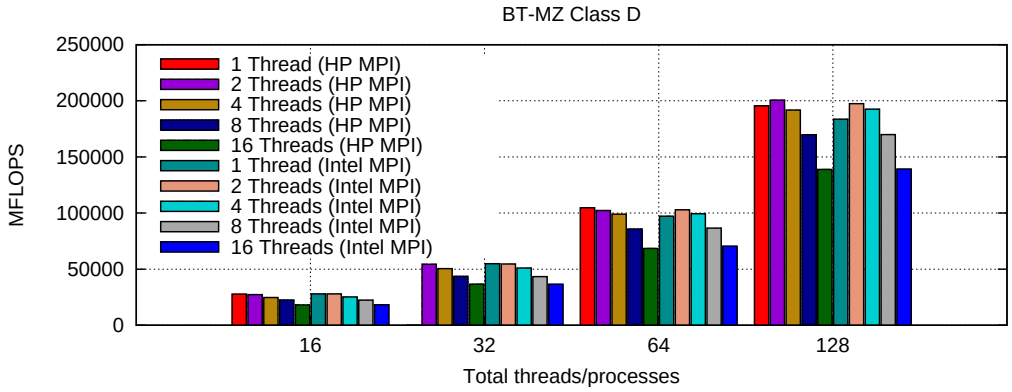


Figura 5.40: Rendimiento de Configuraciones Híbridas en BT con clase D

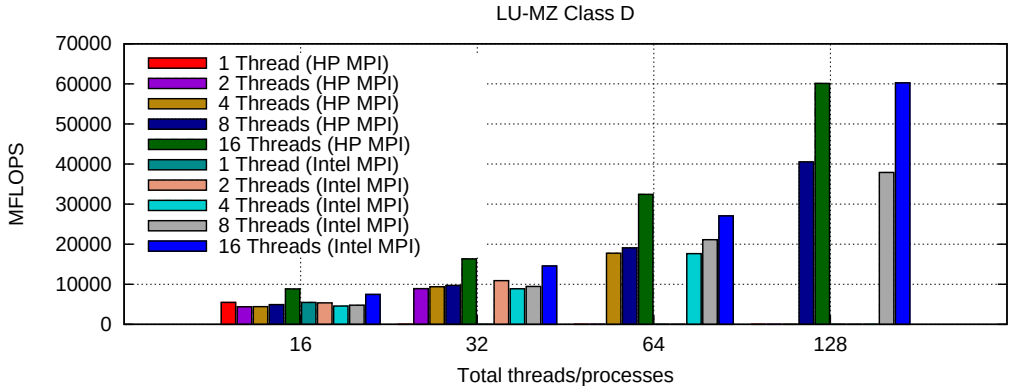


Figura 5.41: Rendimiento de Configuraciones Híbridas en LU con clase D

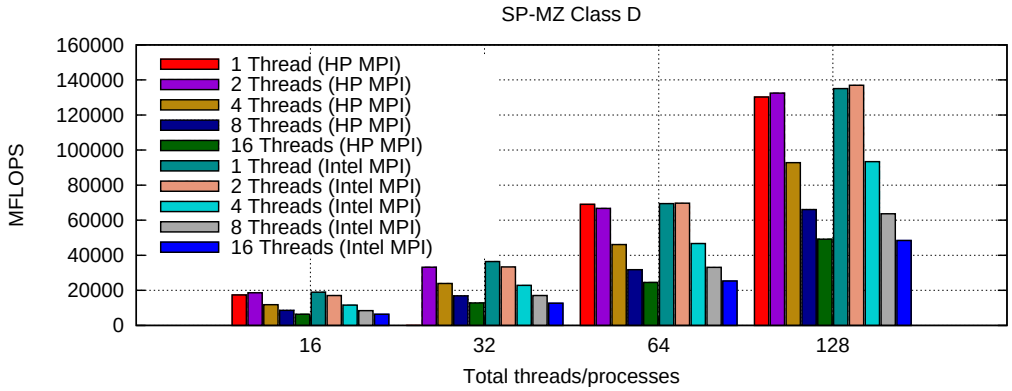


Figura 5.42: Rendimiento de Configuraciones Híbridas en SP con clase D

5.4. Evaluación de Lenguajes

En las próximas subsecciones se analizará la evaluación comparativa realizada acerca de los lenguajes UPC y Java. Dada la escasez de benchmarks implementadas con las 4 aproximaciones (UPC, Java, OpenMP y MPI), esta evaluación se ha realizado usando los kernels de los NAS Parallel Benchmarks. Se han evaluado por separado los resultados de UPC y Java, y cada experimento está sujeto a unas condiciones distintas, que serán detalladas en la correspondiente sección. No obstante, los resultados obtenidos con OpenMP y MPI se han obtenido con el compilador Intel 9.1 y HP MPI 2.2.5.1. El nivel de optimización usado ha sido de `-O3` para los experimentos MPI y, por motivos técnicos, `-O2` para los experimientos OpenMP.

5.4.1. Evaluación de UPC

Para la evaluación de UPC se han realizado experimentos que evalúan comparativamente UPC con OpenMP y MPI. En el momento de la evaluación no estaba disponible el compilador UPC de HP, por lo que se ha utilizado la versión 2.6 del compilador Berkeley UPC [83].

Se han realizado pruebas en memoria compartida, usando el nodo HP Integrity Superdome, para poder ejecutar hasta 128 procesos, y en memoria distribuida, y se han mostrado comparativamente los resultados obtenidos en las dos configuraciones. Los tamaños empleados han sido B y C. El kernel FT no se ha evaluado debido a un fallo en el runtime UPC para este benchmark.

Rendimiento de UPC en Memoria Compartida

Las figuras 5.43 y 5.44 muestran los resultados obtenidos con UPC, MPI y OpenMP en el HP Integrity Superdome. La primera columna indica el número de millones de operaciones por segundo, mientras que la segunda muestra la escalabilidad asociada.

Para la clase B, mostrada en la figura 5.43, se ve que el rendimiento de UPC es inferior a MPI y OpenMP en CG y EP. En MG es superior a OpenMP, pero inferior a MPI. Parte de la justificación de estos resultados puede ser obtenida al analizar el proceso de compilación de UPC con el compilador de Berkeley. Dicho compilador traduce el código UPC a código C con las llamadas a librerías y las directivas adecuadas. Dicho código C es entonces compilado. El lenguaje Fortran es un lenguaje fuertemente tipado y con gran tradición en HPC, gracias a lo cual el código máquina resultante es extremadamente eficiente. Sin embargo, el código C es más flexible y el compilador puede no producir un resultado tan eficiente. Hay estudios que apuntan a que éste es el motivo de esta diferencia [84]. Esto justificaría además el hecho de que UPC tenga un rendimiento muy parejo al de OpenMP en el kernel IS, que obtiene el máximo rendimiento en este benchmark.

Destaca también el hecho de que OpenMP obtenga en muchos casos peor rendimiento que MPI, ya que OpenMP tiene un acceso directo a la memoria, mientras que MPI tiene que realizar copias a través de comunicaciones intranodo. Esto puede venir justificado por problemas de contención en el acceso a memoria y por limitaciones debidas a la coherencia caché propia de sistemas ccNUMA como el HP Integrity Superdome. En

MPI estos problemas no existen ya que cada proceso tiene su conjunto de datos exclusivo, intercambiándolos con paso de mensajes según sea necesario. Estos resultados son coherentes con los obtenidos en otros sistemas ccNUMA [85], pero contrastan con otros en sistemas semejantes [86]. En estos estudios se hacen diferentes pruebas en máquinas distintas, por lo que los motivos pueden estar ligados a la arquitectura de la propia máquina o al tipo de cálculo realizado, así como a la calidad de las optimizaciones OpenMP y MPI y sus compiladores.

Los rendimientos relativos obtenidos con la clase C (mostrados en la figura 5.44) son similares a los de la clase B, a pesar de variar los rendimientos absolutos. El hecho más destacable es el rendimiento de UPC para IS, que con la clase B obtenía un rendimiento parejo con el de OpenMP. En este caso el rendimiento es muy inferior, aunque se mantiene superior al de MPI, que en este benchmark necesita muchas comunicaciones que penalizan su rendimiento cuando se usa un número elevado de procesos.

La escalabilidad observada con ambas clases es similar, teniendo en cuenta que la degradación de la aceleración según se aumenta el número de procesos es menor en la clase de mayor tamaño, ya que con un tamaño mayor la paralelización resulta más provechosa.

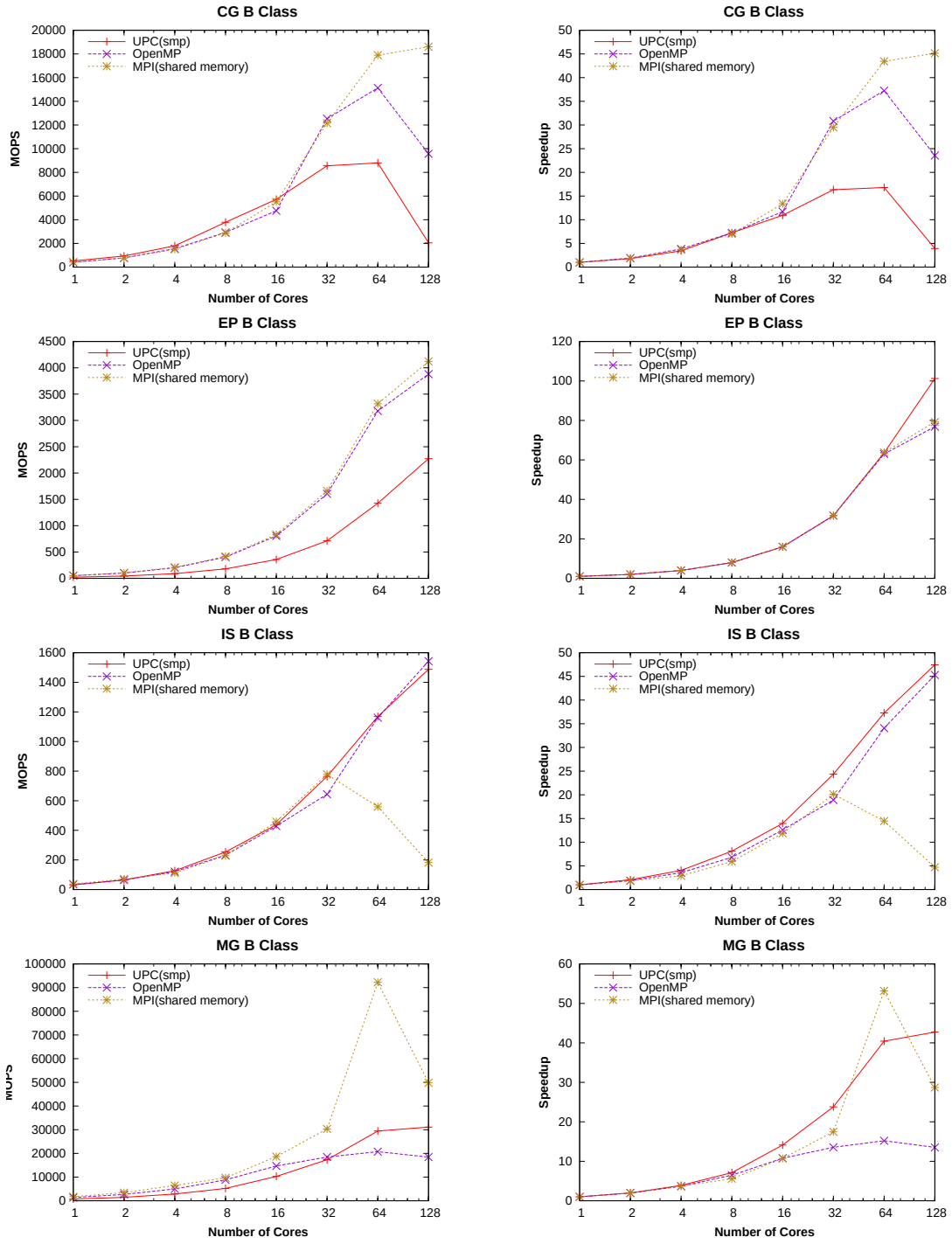


Figura 5.43: Rendimiento de Kernels NPB UPC en memoria compartida (tamaño B)

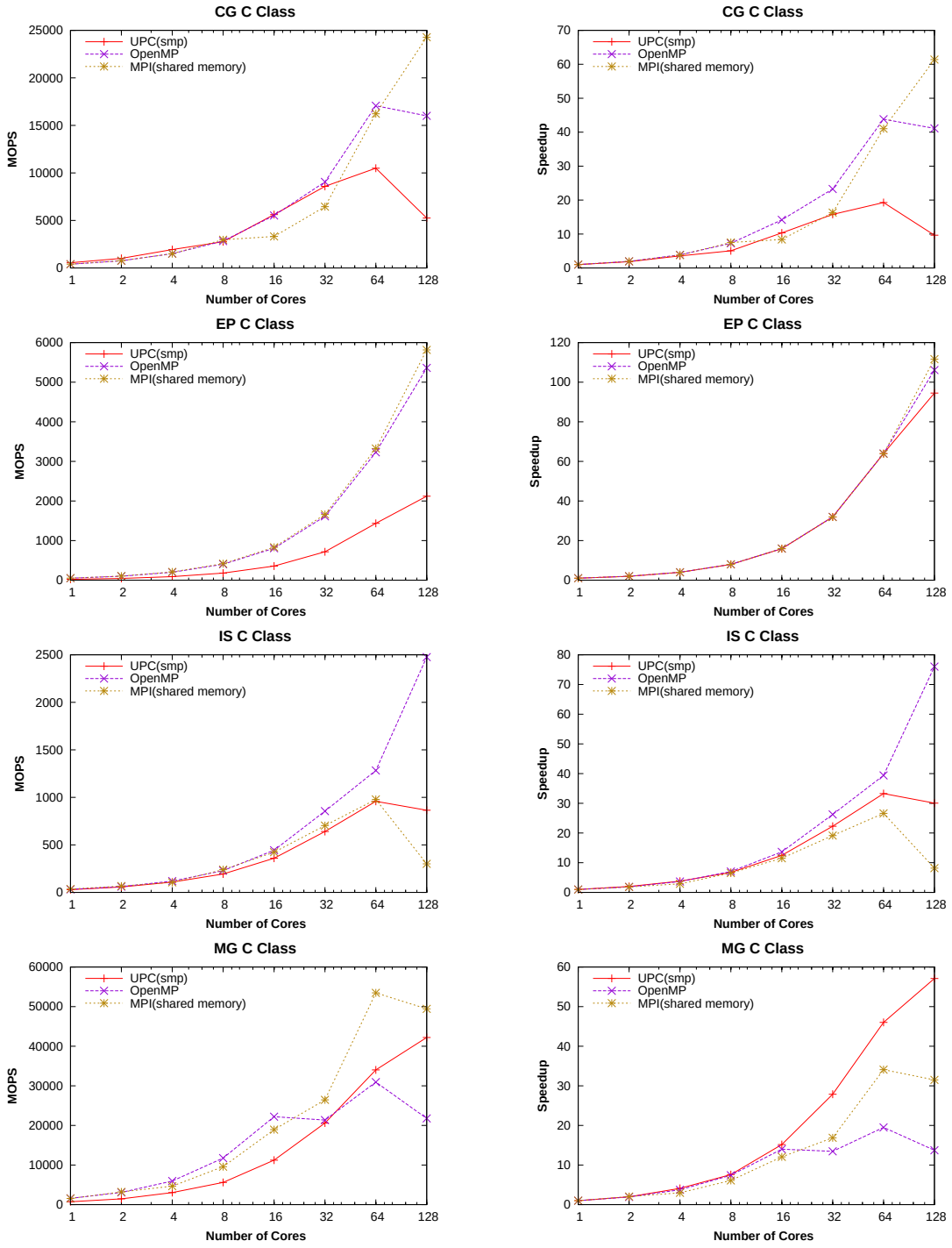


Figura 5.44: Rendimiento de Kernels NPB UPC en memoria compartida (tamaño C)

Rendimiento de UPC en un Escenario Híbrido

Para el análisis del rendimiento de UPC en un escenario híbrido se han realizado todas las ejecuciones posibles para un número dado de procesos, es decir, 1, 2, 4, 8 y 16 procesos por nodo. De entre los resultados obtenidos se ha escogido el de mayor rendimiento, que suele implicar un mayor número de procesos por nodo para UPC que para MPI, aunque no es una regla general. El motivo de esta diferencia es que UPC aprovecha mejor una situación de memoria compartida.

Las figuras 5.45 y 5.46 muestran los resultados alcanzados. El rendimiento de MPI es muy superior a UPC para muchos procesos. Para un número pequeño de procesos la diferencia es más pequeña, llegando incluso a rendir mejor UPC en CG con tamaño B para un número de procesos inferior a 32. Estas diferencias de rendimiento pueden ser achacables a la eficiencia del compilador de C, menos eficiente que el compilador de Fortran, y al runtime de UPC, cuyas operaciones colectivas tienen una sobrecarga significativa.

Comparando con los resultados de memoria compartida se observa que la diferencia entre MPI y UPC se hace más grande, excepto en CG en donde se estrecha, y en EP, en donde permanece casi constante. Es especialmente destacable el hecho de que en IS, en donde UPC obtenía un rendimiento superior a MPI a partir de 32 procesos en memoria compartida, la situación se ha invertido y MPI rinde mejor.

Examinando la escalabilidad de MPI y UPC se ve que los datos son más parejos que los rendimientos brutos, estando ambas escalabilidades razonablemente igualadas, excepto en el caso de CG para ambos tamaños, y MG para el tamaño B, en donde MPI es claramente superior.

Comparativa de Rendimiento de UPC en Memoria Compartida vs. Híbrida

El rendimiento obtenido por UPC en configuraciones de memoria compartida y memoria distribuida se puede consultar en la figura 5.47. Dado que el rendimiento mostrado para un escenario híbrido era el obtenido en la mejor de las combinaciones de número de nodos y número de procesos por nodo, los resultados hasta 16 cores coinciden, ya que la mejor combinación para 16 procesos es ejecutar los 16 en el mismo nodo. Para ejecuciones de más de 16 procesos los resultados obtenidos en memoria compartida obtienen mejor rendimiento, especialmente en IS y MG. Una excepción es el benchmark EP, en donde se obtiene mejor rendimiento en la configuración híbrida. Otro detalle importante es que el rendimiento en memoria compartida en CG se colapsa a partir de 64 procesos, por lo que para 128 procesos el rendimiento de la ejecución híbrida es mayor.

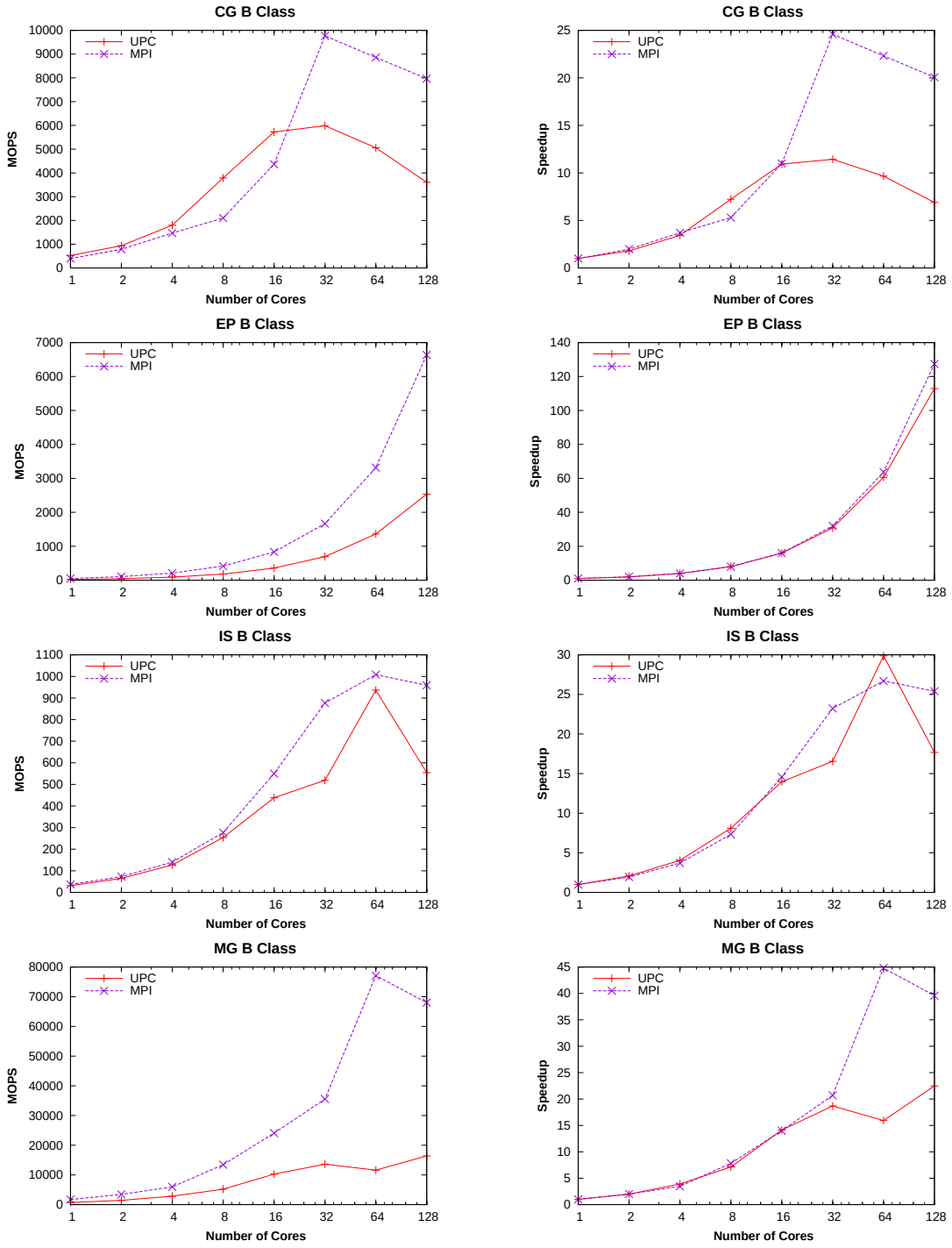


Figura 5.45: Rendimiento de Kernels NPB UPC en un escenario híbrido (tamaño B)

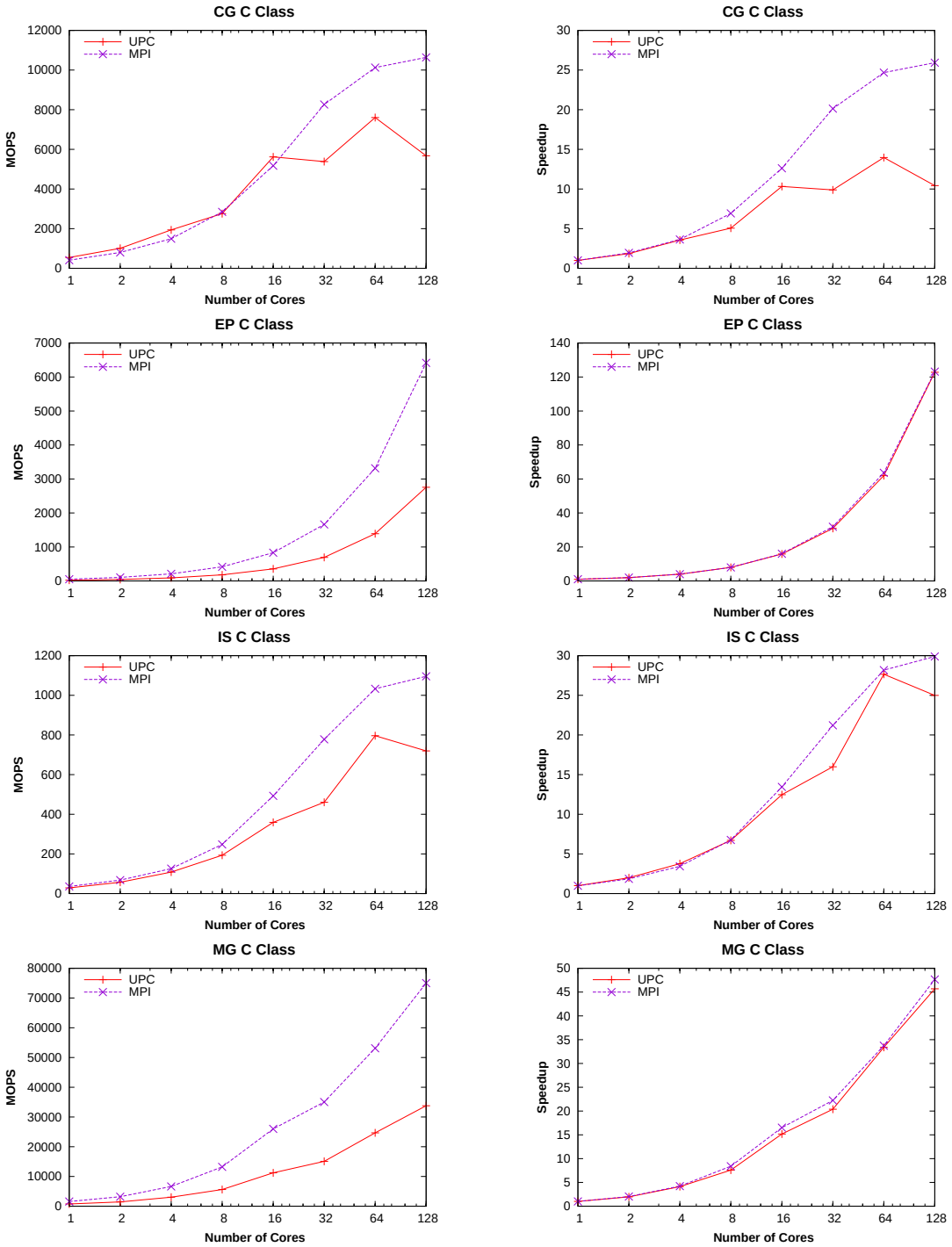


Figura 5.46: Rendimiento de Kernels NPB UPC en un escenario híbrido (tamaño C)

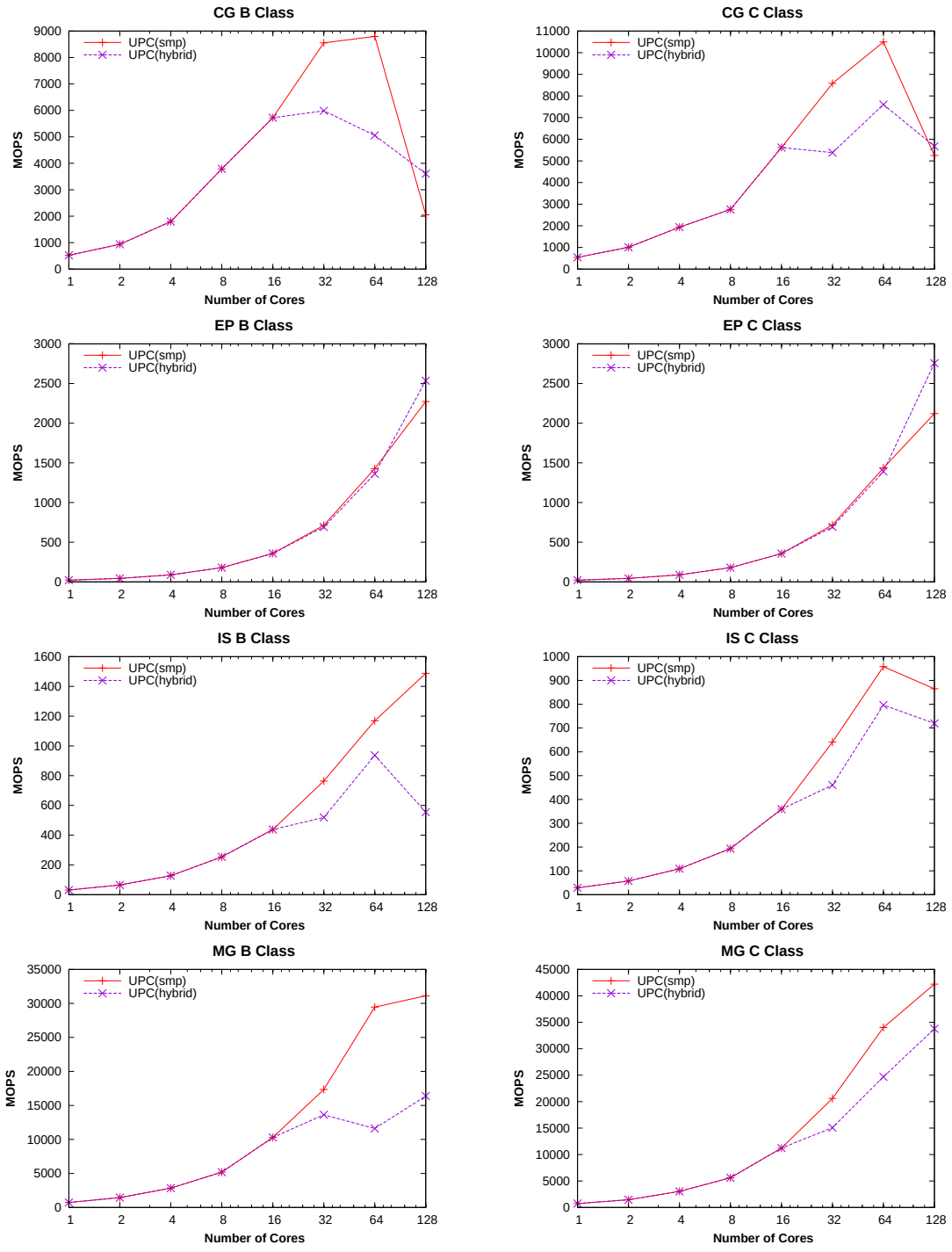


Figura 5.47: Rendimiento Comparativo de Kernels NPB UPC en memoria compartida y memoria distribuida

5.4.2. Evaluación de Java

Para la evaluación de Java se han utilizado NPB-MPJ [87] y las implementaciones oficiales de los NPB en Java multithreading (NPB-JAV), OpenMP (NPB-OMP) y MPI (NPB-MPI). Se ha utilizado el compilador y la máquina virtual JRockit 5.0 (R27.6) de Oracle [88], por ser la única máquina virtual J2SE 5.0 disponible para arquitectura IA64 (Itanium) en el momento de realizar las pruebas. El tener una máquina virtual J2SE 5.0 es un requisito para el correcto funcionamiento de la biblioteca de comunicaciones MPJ Express 0.27 [89, 90] empleada para la evaluación. El rendimiento de dicha máquina virtual está lejos de un rendimiento óptimo, por lo que se esperan resultados mejorables. Las pruebas con MPJ Express se han efectuado con IPoIB[2]. Las pruebas con memoria compartida (NPB-JAV y NPB-OMP) se han efectuado hasta 8 cores.

La figura 5.48 muestra los resultados obtenidos con la clase A. En ella se ve que el rendimiento de Java en la plataforma IA64 está muy por debajo del rendimiento de Java en la plataforma x86 observado en otros estudios [87]. Esto es debido a la máquina virtual empleada, que carece de algunas de las virtudes de la máquina virtual de Sun Microsystems para arquitectura x86.

Sin embargo, analizando la escalabilidad, la situación es distinta. NPB-JAV muestra la mejor escalabilidad en 3 benchmarks, y en el restante obtiene una escalabilidad razonable. El benchmark EP no está disponible en la versión NPB-JAV. NPB-MPJ tiene una escalabilidad limitada debido al uso de IPoIB, que tiene un rendimiento mucho peor que el obtenido con el uso de un stack software Infiniband puro como el usado en las ejecuciones MPI. Esto se comprueba al observar la escalabilidad del benchmark EP, muy pareja a la obtenida en las versiones nativas.

Para la clase B (mostrada en la figura 5.49) la situación en cuanto a los rendimientos es similar. Sin embargo, al observar las escalabilidades se ve como la escalabilidad de NPB-MPJ está pareja a la de la versión MPI en prácticamente todos los benchmarks, llegando incluso a tener una escalabilidad superior, excepto en IS. Esta observación confirma que la escalabilidad está limitada por el uso de IPoIB, ya que al ser esta una ejecución más pesada, la importancia de las comunicaciones disminuye, aumentando de esta forma la escalabilidad. El comportamiento de NPB-JAV es, comparativamente, inferior al obtenido en la clase A.

[2]IPoIB: IP over InfiniBand

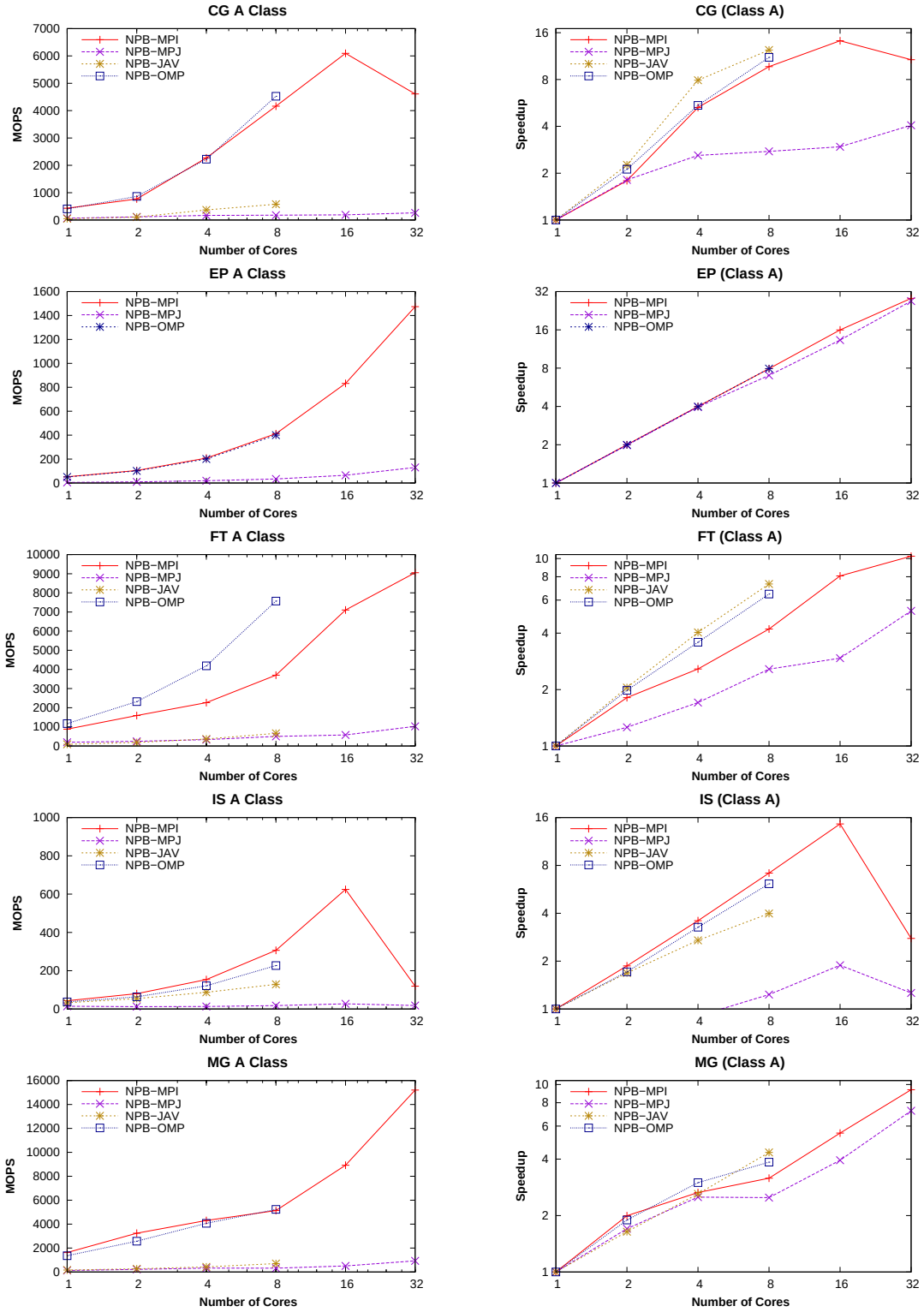


Figura 5.48: Rendimiento de Kernels NPB Java (tamaño A)

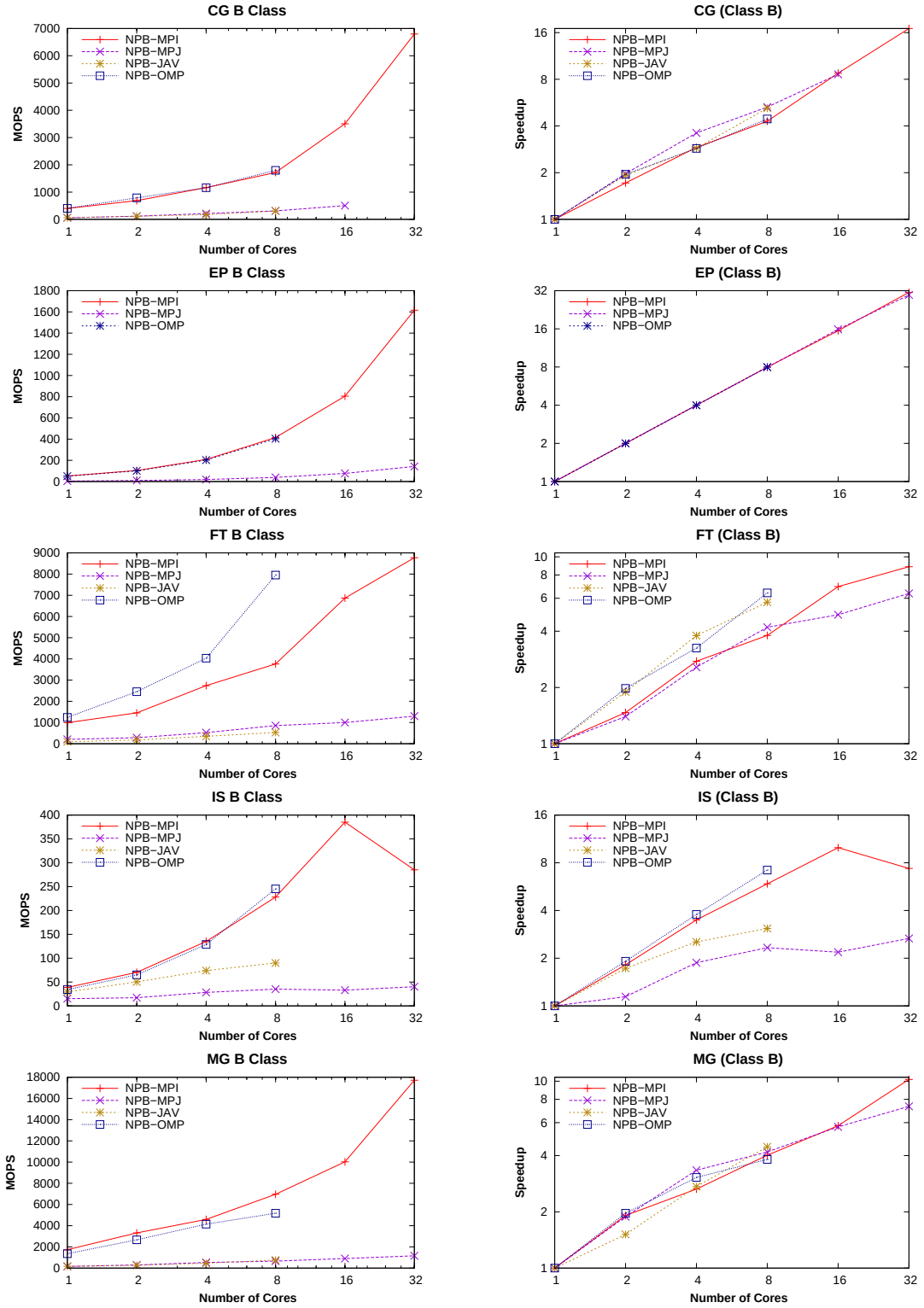


Figura 5.49: Rendimiento de Kernels NPB Java (tamaño B)

6 Principales Aportaciones y Conclusiones

A continuación se detallarán las principales aportaciones y las conclusiones de este trabajo.

6.1. Principales Aportaciones

El Finis Terrae representa una valiosa herramienta para la comunidad científica. Sus 142 nodos de computación con 16 cores y 128 GB de memoria proporcionan una capacidad de cálculo y ratio de memoria por core de procesador (8 GB por core) muy importante. Además su nodo HP Integrity Superdome, con 128 cores y 1 TB de memoria lo sitúa como uno de los supercomputadores con más memoria compartida de Europa. Estas características posibilitan la ejecución de simulaciones que no son posibles en otras máquinas con una capacidad de cálculo mayor, pero con una cantidad de memoria por nodo inferior. Esta afirmación queda respaldada por la reciente superación de un reto computacional en simulación electromagnética con 500 millones de incógnitas, que supone un récord mundial.

La complejidad de este supercomputador hace necesario un conocimiento empírico para conocer su comportamiento ante diversas situaciones. Este trabajo aporta una extensa evaluación del rendimiento del Finis Terrae, a través de diversos microbenchmarks y kernels computacionales.

Se ha evaluado la importancia de la afinidad de procesos a las celdas hardware adecuadas, de forma que se ha verificado y cuantificado la penalización de las comunicaciones a través de los buses de interconexión de celdas, tanto para el acceso a memoria como para el acceso a la interfaz Infiniband. Esto lo ha hecho en numerosas situaciones, por lo que ahora se conoce detalladamente el comportamiento con tráfico bidireccional, que ha resultado en un sorprendente recorte del ancho de banda y latencia esperados, y las virtudes y carencias de las bibliotecas MPI instaladas.

Así mismo, se ha medido el rendimiento de dos operaciones colectivas, a partir del cual se puede inferir el rendimiento de las otras operaciones colectivas. En base a este estudio se puede analizar la escalabilidad de una aplicación, realizando un *profiling* de sus comunicaciones y determinando de esta forma la influencia que tendrían según el número de nodos, el tamaño de los mensajes y el número de comunicaciones.

También se ha evaluado el rendimiento de diversas configuraciones híbridas, probando con todas las configuraciones posibles desde 128 procesos MPI en 8 nodos, hasta

1 proceso MPI y 16 threads OpenMP en un único nodo, pasado por todos los puntos intermedios. Los usuarios del Finis Terrae cuyas aplicaciones tengan patrones computacionales similares al de los benchmarks evaluados podrán inferir una mejor configuración para sus ejecuciones.

Por último, ha sido medido el rendimiento obtenido por dos lenguajes que están adquiriendo importancia en el mundo de la supercomputación: UPC y Java. UPC ha sido evaluado con el compilador de Berkeley, y su rendimiento no suele alcanzar al de aproximaciones OpenMP o MPI, principalmente por estar basado en C, con compiladores ligeramente menos eficientes que los de Fortran, y por la falta de optimización de su runtime. Java, por su parte, tiene un rendimiento muy inferior al de otras aproximaciones debido al pobre rendimiento de la única máquina virtual J2SE 5.0 disponible para Itanium, JRockit 5.0, y al uso de IPoIB, que tiene una sobrecarga importante respecto a un stack software Infiniband puro. Sin embargo, su escalabilidad es buena, igualando a la de las versiones nativas cuando el tamaño del problema crece. Con la evaluación de ambos lenguajes se han apuntado direcciones de investigación y mejora de las soluciones software actuales.

6.2. Conclusiones

El Finis Terrae es una potente máquina capaz de llevar la investigación en Europa a un siguiente nivel. En los pocos meses que lleva en funcionamiento ya ha superado numerosos restos computacionales, entre los que se encuentran varios récords mundiales. A pesar de todo, el conocimiento preciso de las particularidades de la arquitectura y bibliotecas paralelas del Finis Terrae posibilita la obtención de un mayor rendimiento, y una mayor eficiencia en su utilización.

Los análisis de rendimiento efectuados en este trabajo proporcionan una guía de referencia para los investigadores que ejecuten grandes retos computacionales en el Finis Terrae. La aplicabilidad de los análisis de configuraciones híbridas y de los diversos lenguajes es inmediata para el trabajo diario de los investigadores que basan su trabajo en este supercomputador. Sin embargo, los resultados desvelados acerca del posicionamiento de procesos en las celdas con interfaz Infiniband tienen una aplicación limitada para los trabajos diarios y la optimización del sistema de colas. Esto es debido principalmente al hecho de que para acceder al sistema de ficheros SFS no es necesario especificar ningún parámetro. Dado que el acceso a dicho sistema se hace a través de la red Infiniband no es posible discriminar entre trabajos que usen la red Infiniband, para posicionarlos en la celda con interfaz Infiniband, y trabajos que no la usen, para posicionarlos en la celda sin red Infiniband. No obstante, en la ejecución de los grandes retos computacionales en los que se reserva una gran cantidad de nodos para uso exclusivo, la afinidad a celdas tiene una aplicación sencilla e inmediata.

6.3. Trabajo Futuro

Este trabajo ha evaluado el sistema de comunicaciones en la peor situación posible, es decir, cuando los datos a enviar no están en caché. Se plantea como trabajo futuro la evaluación del sistema de comunicaciones en una situación más habitual, con

los datos cacheados. Así mismo se propone una evaluación del rendimiento de diversas aplicaciones, GROMACS, AMBER, LAMMPS, GAMESS y NAMD entre otras, utilizadas comúnmente en el CESGA. También se propone una evaluación de la influencia de distintos parámetros software y de afinidad en distintos benchmarks estándar, como los HPCC o los NPB, y en las aplicaciones evaluadas, así como experimentos con el *scheduling* de OpenMP.

Bibliografía

[1] CESGA website. http://www.cesga.es. [Última visita: Enero 2009].

[2] Finis Terrae website. http://www.cesga.es/content/view/917/115/lang,gl/. [Última visita: Enero 2009].

[3] Top 500. http://www.top500.org. [Última visita: Enero 2009].

[4] Ranking del Finis Terrae en el Top 500. http://www.top500.org/system/9500. [Última visita: Enero 2009].

[5] Infiniband Trade Association website. http://www.infinibandta.org. [Última visita: Enero 2009].

[6] OpenMP website. http://www.openmp.org. [Última visita: Enero 2009].

[7] MPI Forum website. http://www.mpi-forum.org. [Última visita: Enero 2009].

[8] UPC website. http://upc.gwu.edu/. [Última visita: Enero 2009].

[9] Oak Ridge National Laboratory website. http://www.ornl.gov/. [Última visita: Enero 2009].

[10] NASA Advanced Supercomputing Division website. http://www.nas.nasa.gov/. [Última visita: Enero 2009].

[11] NAS Parallel Benchmarks. http://www.nas.nasa.gov/Resources/Software/npb.html. [Última visita: Enero 2009].

[12] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks. Report RNR-94-007, Department of Mathematics and Computer Science, Emory University, March 1994.

[13] SVG website. http://www.cesga.es/content/view/409/42/lang,gl/. [Última visita: Enero 2009].

[14] P. A. Agarwal, R. A. Alexander, E. Apra, S. Balay, A. S. Bland, J. Colgan, E. F. D'Azevedo, J. J. Dongarra, T. H. Dunigan Jr., M. R. Fahey, R. A. Fahey, A. Geist,

M. Gordon, R. J. Harrison, D. Kaushik, M. Krishnakumar, P. Luszczek, A. Mezzacappa, J. A. Nichols, J. Nieplocha, L. Oliker, T. Packwood, M. S. Pindzola, T. C. Schulthess, J. S. Vetter, J. B. White III, T. L. Windus, P. H. Worley, and T. Zacharia. Cray X1 Evaluation Status Report. Technical Report ORNL/TM-2004/13, Oak Ridge National Laboratory, 2004.

[15] Cray X1E del ORNL en el top500. `http://www.top500.org/system/7652`. [Última visita: Enero 2009].

[16] STREAM benchmark website. `http://www.cs.virginia.edu/stream/`. [Última visita: Enero 2009].

[17] HALO benchmark website. `http://www.spscicomp.org/2000/presentations/Wallcraft/`. [Última visita: Enero 2009].

[18] Jack Dongarra and Anthony J. G. Hey. The Parkbench Benchmark Collection. In *Supercomputer 11, 94*, 1995.

[19] Aad J. van der Steen. The benchmark of the EuroBen group. pages 165–175, 1993.

[20] Piotr R. Luszczek, David H. Bailey, Jack J. Dongarra, Jeremy Kepner, Robert F. Lucas, Rolf Rabenseifner, and Daisuke Takahashi. The HPC Challenge (HPCC) benchmark suite. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 213, Tampa, FL, USA, 2006.

[21] HPCC benchmarks website. `http://icl.cs.utk.edu/hpcc/`. [Última visita: Enero 2009].

[22] BLAS website. `http://www.netlib.org/blas/`. [Última visita: Enero 2009].

[23] Community Climate System Model website. `http://www.ccsm.ucar.edu/`. [Última visita: Enero 2009].

[24] GYRO website. `http://fusion.gat.com/theory/Gyro`. [Última visita: Enero 2009].

[25] NIMROD website. `https://nimrodteam.org/`. [Última visita: Enero 2009].

[26] LSMS website. `http://www.psc.edu/general/software/packages/lsms/`. [Última visita: Enero 2009].

[27] NWChem website. `http://www.emsl.pnl.gov/docs/nwchem/nwchem.html`. [Última visita: Enero 2009].

[28] Global Arrays toolkit website. `http://www.emsl.pnl.gov/docs/global/`. [Última visita: Enero 2009].

[29] ARMCI website. `http://www.emsl.pnl.gov/docs/parsoft/armci/`. [Última visita: Enero 2009].

[30] GAMESS website. http://www.msg.chem.iastate.edu/gamess/. [Última visita: Enero 2009].

[31] AMBER website. http://amber.scripps.edu/. [Última visita: Enero 2009].

[32] NAMD website. http://www.ks.uiuc.edu/Research/namd/. [Última visita: Enero 2009].

[33] sPPM benchmark website. https://asc.llnl.gov/computing_resources/purple/archive/benchmarks/sppm/. [Última visita: Enero 2009].

[34] Thomas H. Dunigan Jr., Mark R. Fahey, James B. White III, and Patrick. H. Worley. Early Evaluation of the Cray X1. In *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 18, Phoenix, AZ, USA, 2003.

[35] Thomas H. Dunigan Jr. and Patrick. H. Worley. Early Performance Evaluation of the Cray X1 at Oak Ridge National Laboratory. In *Proceedings of the 45th Cray User Group Conference*, Columbus, OH, USA, 2003.

[36] Mark R. Fahey, Sadaf Alam, Thomas H. Dunigan Jr., Jeffrey S. Vetter, and Patrick. H. Worley. Early Evaluation of the Cray XD1. In *Proceedings of the 47th Cray User Group Conference*, Knoxville, TN, USA, 2005.

[37] AMD Core Math Library website. http://developer.amd.com/cpu/libraries/acml/Pages/default.aspx. [Última visita: Enero 2009].

[38] CacheBench website. http://icl.cs.utk.edu/projects/llcbench/cachebench.html. [Última visita: Enero 2009].

[39] Intel MPI Benchmarks website. http://www.intel.com/cd/software/products/asmo-na/eng/cluster/mpi/219848.htm. [Última visita: Enero 2009].

[40] SMG2000 benchmark website. https://asc.llnl.gov/computing_resources/purple/archive/benchmarks/smg/. [Última visita: Enero 2009].

[41] VASP website. http://cms.mpi.univie.ac.at/vasp/. [Última visita: Enero 2009].

[42] Sadaf R. Alam, Richard F. Barrett, Mark R. Fahey, Jeffery A. Kuehn, O.E. Bronson Messer, Richard T. Mills, Philip C. Roth, Jeffrey S. Vetter, and Patrick H. Worley. An Evaluation of the Oak Ridge National Laboratory Cray XT3. *International Journal of High Performance Computing Applications*, 22(1):52–80, 2008.

[43] Jaguar en el top500. http://www.top500.org/system/9220. [Última visita: Enero 2009].

[44] FFTW website. http://www.fftw.org/. [Última visita: Enero 2009].

[45] LAMMPS website. http://lammps.sandia.gov/. [Última visita: Enero 2009].

[46] PFLOTRAN website. `https://software.lanl.gov/pflotran`. [Última visita: Enero 2009].

[47] Jeffrey S. Vetter, Sadaf R. Alam, Thomas H. Dunigan Jr., Mark R. Fahey, Philip C. Roth, and Patrick H. Worley. Early Performance Evaluation of the Cray XT3 at ORNL. In *Proceedings of the 47th Cray User Group Conference*, Albuquerque, NM, USA, 2005.

[48] GotoBLAS website. `http://www.tacc.utexas.edu/resources/software/#blas`. [Última visita: Enero 2009].

[49] Sadaf R. Alam, Jeffery A. Kuehn, Richard F. Barrett, Jeff M. Larkin, Mark R. Fahey, Ramanan Sankaran, and Patrick H. Worley. Cray XT4: an early evaluation for petascale scientific simulation. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, Reno, NV, USA, 2007.

[50] Patrick H. Worley. Early Evaluation of the IBM BG/P. In *Proceedings of the 9th LCI International Conference on High Performance Clustered Computing*, Urbana, IL, USA, 2008.

[51] IBM Blue Gene/P del ORNL en el top500. `http://www.top500.org/system/8998`. [Última visita: Enero 2009].

[52] Thomas H. Dunigan Jr., Jeffrey S. Vetter, and Patrick H. Worley. Performance Evaluation of the SGI Altix 3700. In *ICPP '05: Proceedings of the 2005 International Conference on Parallel Processing*, pages 231–240, Oslo, Norway, 2005.

[53] Patrick H. Worley, Sadaf R. Alam, Thomas H. Dunigan Jr., Mark R. Fahey, and Jeffrey S. Vetter. Comparative Analysis of Interprocess Communication on the X1, XD1, and XT3. In *Proceedings of the 47th Cray User Group Conference*, Albuquerque, NM, USA, 2005.

[54] Subhash Saini and David H. Bailey. NAS Parallel Benchmark (Version 1.0) Results 11-96. Technical report, NASA Ames Research Center, 1996.

[55] Rupak Biswas, Subhash Saini, Sharad Gavali, Henry Jin, Dennis C. Jespersen, M. Jahed Djomehri, Nateri K. Madavan, and Cetin Kiris. NAS Experience with the Cray X1. Technical Report NAS-05-013, NAS Division, NASA Ames Research Center, 2005.

[56] OVERFLOW website. `http://aaac.larc.nasa.gov/~buning/codes.html#overflow`. [Última visita: Enero 2009].

[57] Columbia en el top500. `http://www.top500.org/system/9232`. [Última visita: Enero 2009].

[58] Subhash Saini, Dennis C. Jespersen, Dale Talcott, M. Jahed Djomehri, and Timothy Sandstrom. Performance Comparison of SGI Altix 4700 and SGI Altix 3700 Bx2. Technical Report NAS-08-001, NAS Division, NASA Ames Research Center, 2008.

[59] Cart3D website. `http://people.nas.nasa.gov/~aftosmis/cart3d/`. [Última visita: Enero 2009].

[60] USM3D website. `http://tetruss.larc.nasa.gov/usm3d/index.html`. [Última visita: Enero 2009].

[61] ECCO website. `http://www.ecco-group.org/`. [Última visita: Enero 2009].

[62] Rod Fatoohi, Subhash Saini, and Robert Ciotti. Interconnect performance evaluation of SGI Altix 3700 BX2, Cray XI, Cray Opteron Cluster, and Dell PowerEdge. In *IPDPS '06: Proceedings of the 20th International Parallel and Distributed Processing Symposium*, Rhodes Island, Greece, 2006.

[63] Subhash Saini, Johnny Chang, Robert Hood, and Haoqiang Jin. A Scalability Study of Columbia using the NAS Parallel Benchmarks. Technical Report NAS-06-011, NAS Division, NASA Ames Research Center, 2006.

[64] Haoqiang Jin, Barbara M. Chapman, Lei Huang, Dieter an Mey, and Thomas Reichstein. Performance Evaluation of a Multi-Zone Application in Different OpenMP Approaches. *International Journal of Parallel Programming*, 36(3):312–325, 2008.

[65] Subhash Saini, Robert Ciotti, Brian T. N. Gunney, Thomas E. Spelce, Alice E. Koniges, Dob Dossa, Panagiotis A. Adamidis, Rolf Rabenseifner, Sunil R. Tiyyagura, and Matthias Müller. Performance evaluation of supercomputers using HPCC and IMB benchmarks. *Journal of Computer and System Sciences*, 74(6), 2007.

[66] J. Carlos Mouriño, Aurelio Rodríguez, and Andrés Gómez. Benchmarks para el Análisis de Nuevas Arquitecturas: Xeon Single, Dual y Quadcore. Informe Técnico CESGA-2007-001, CESGA, 2007.

[67] Intel LINPACK website. `http://www.intel.com/cd/software/products/asmo-na/eng/266857.htm`. [Última visita: Enero 2009].

[68] SparseBench website. `http://www.netlib.org/benchmark/sparsebench/`. [Última visita: Enero 2009].

[69] GAUSSIAN website. `http://www.gaussian.com/`. [Última visita: Enero 2009].

[70] GROMACS website. `http://www.gromacs.org/`. [Última visita: Enero 2009].

[71] Benchmarks Subsistema Beowulf. Informe técnico, CESGA, 2002.

[72] CPMD website. `http://www.cpmd.org/`. [Última visita: Enero 2009].

[73] Lustre website. `http://www.lustre.org/`. [Última visita: Enero 2009].

[74] Mellanox Technologies Inc. *Introduction to InfiniBand™*.

[75] Noticia de la superación del reto computacional con 500 millones de incógnitas en un problema de electromagnetismo. `http://www.cesga.es/content/view/1001/86/lang,es/`. [Última visita: Febrero 2009].

[76] IOZone website. `http://www.iozone.org/`. [Última visita: Enero 2009].

[77] Iperf website. `http://sourceforge.net/projects/iperf`. [Última visita: Enero 2009].

[78] Sequoia suite website. `https://asc.llnl.gov/sequoia/benchmarks/`. [Última visita: Enero 2009].

[79] George Almási, Charles Archer, José G. Castaños, John A. Gunnels, C. Christopher Erway, Philip Heidelberger, Xavier Martorell, José E. Moreira, Kurt Pinnow, Joe Ratterman, Burkhard D. Steinmacher-Burow, William Gropp, and Brian R. Toonen. Design and implementation of message-passing services for the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2-3):393–406, 2005.

[80] Ron Brightwell and Keith D. Underwood. Evaluation of an Eager Protocol Optimization for MPI. In *Proceedings of the10th European PVM/MPI Users' Group Meeting*, pages 327–334, Venice, Italy, 2003.

[81] Franck Cappello and Daniel Etiemble. MPI versus MPI+OpenMP on the IBM SP for the NAS Benchmarks. In *SC '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, Dallas, TX, USA, 2000.

[82] Kengo Nakajima. OpenMP / MPI Hybrid vs. Flat MPI on the Earth Simulator: Parallel Iterative Solvers for Finite Element Method. In *ISHPC '03: Proceedings of the 5th International Symposium on High Performance Computing*, pages 486–499, Tokyo-Odaiba, Japan, 2003.

[83] Berkeley UPC website. `http://upc.lbl.gov/`. [Última visita: Enero 2009].

[84] Tarek El-Ghazawi and François Cantonnet. UPC performance and potential: a NPB experimental study. In *SC '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–26, Baltimore, MD, USA, 2002. IEEE Computer Society Press.

[85] Glenn R. Luecke and Wei-Hua Lin. Scalability and performance of OpenMP and MPI on a 128-processor SGI Origin 2000. *Concurrency and Computation: Practice and Experience*, 13(10):905–928, 2001.

[86] Russell Brown and Ilya Sharapov. High-Scalability Parallelization of a Molecular Modeling Application: Performance and Productivity Comparison Between OpenMP and MPI Implementations. *International Journal of Parallel Programming*, 35(5):441–458, 2007.

[87] Damián A. Mallón, Guillermo L. Taboada, Juan Touriño, and Ramón Doallo. NPB-MPJ: NAS Parallel Benchmarks Implementation for Message-Passing in Java. In *Proc. 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'09), Weimar, Alemania*, 2009.

[88] Oracle JRockit website. `http://edocs.bea.com/jrockit/webdocs/index.html`. [Última visita: Enero 2009].

[89] M. Baker, B. Carpenter, and A. Shafi. MPJ Express: Towards Thread Safe Java HPC. In *Proceedings of the 8th IEEE Intl. Conf. on Cluster Computing (Cluster'06)*, pages 1–10, Barcelona, Spain, 2006.

[90] MPJ Express website. `http://mpj-express.org`. [Última visita: Enero 2009].