Capítulo 4: Experimentación

En este capítulo se presentará inicialmente todo el contexto de experimentación con los algoritmos de cómputo paralelo para el cálculo de multiplicaciones de matrices. Se explica detalladamente el hardware (tres redes locales) y los tamaños de matrices con los cuales se llevaron a cabo los experimentos. También se dan los valores de velocidades relativas de las computadoras y con estos valores más una estimación (optimista) *a piori* del rendimiento de las redes de interconexión se estima el máximo valor de speedup posible en cada una de las redes locales con las máquinas disponibles.

La primera aproximación a la implementación de los algoritmos se basa en la biblioteca PVM y se muestran todos los resultados de la experimentación. El rendimiento que se obtiene es muy distante del óptimo y no es aceptable. Por esta razón se muestran algunos detalles de la ejecución paralela que dan a conocer que el problema no son los algoritmos sino específicamente la implementación de los mensajes broadcast de la biblioteca PVM.

Finalmente, se propone e implementa un mensaje broadcast basado directamente en el protocolo de comunicaciones UDP, se vuelven a realizar los experimentos y se muestran los resultados obtenidos con esta implementación de los mensajes broadcast. En este contexto se llega a que: o bien los algoritmos obtienen un rendimiento aceptable o se puede detectar de manera automática las computadoras que generan problemas de rendimiento y por lo tanto se las puede aislar para obtener rendimiento optimizado.

4.1 Características de las Redes Locales Utilizadas

Cada una de las redes locales utilizadas en la experimentación son preexistentes al presente trabajo y no se cambiaron ni se adaptaron para obtener mejores resultados en cuanto a rendimiento. En las subsecciones que siguen se describen sintéticamente cada una de las redes, identificando las características más importantes en cuanto a las computadoras que la componen y la topología de la red de interconexión. Básicamente, las redes locales son:

- CeTAD: perteneciente al Centro de Técnicas Analógico-Digitales, Departamento de Electrotecnia, Facultad de Ingeniería, Universidad Nacional de La Plata, La Plata, Argentina. Es la que está instalada desde hace más tiempo y las computadoras que la componen son utilizadas con múltiples propósitos.
- LQT: perteneciente al Laboratorio de Química Teórica, CEQUINOR, Departamento de Química, Facultad de Ciencias Exactas, Universidad Nacional de La Plata, La Plata, Argentina. Es una red destinada a la resolución de problemas numéricos, fue instalada hace varios años y se ejecutan trabajos secuenciales y paralelos desarrollados con PVM y Linda.
- LIDI: perteneciente al Laboratorio de Investigación y Desarrollo en Informática, Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina. Está dedicada a enseñanza de programación paralela e investigación. Puede considerarse directamente una instalación más del tipo Beowulf, aunque no de las más costosas en cuanto a cantidad de máquinas y red de interconexión.

En términos generales, el software paralelo se desarrolló utilizando PVM (Parallel Virtual Machine). En el caso particular de las PCs utilizadas, el sistema operativo elegido para la ejecución fue Linux [44] [PVM].

Las razones para la elección de PVM (además de su libre disponibilidad), son básicamente dos:

- Existe un único grupo de desarrollo y fuente del software. Esto puede ser una desventaja, pero simplifica el análisis de los resultados obtenidos en cuanto a rendimiento dado que no hay posibilidad de distintas implementaciones. Esto no es posible de asegurar en el caso de MPI (Message Passing Interface) [88] [92] [107] que tiene múltiples implementaciones y potencialmente distintas características en cuanto a rendimiento.
- Es ampliamente utilizado, tiene varios años de evolución y sus características son muy bien conocidas, lo que simplifica también la interpretación de los resultados obtenidos en cuanto a rendimiento de las aplicaciones paralelas que lo utilizan.

En la mayoría de las redes locales, las PCs ya contaban con Linux instalado, principalmente de la distribución RedHat [LinuxRH]. Siempre que fue posible, se intentó la instalación de Linux en una partición de disco separada y la distribución utilizada en tal caso es la de RedHat.

4.1.1 Red Local del CeTAD

Tal como se mencionó anteriormente, la red local del CeTAD tiene dos características

generales que implican una gran variedad en cuanto a la heterogeneidad de las máquinas que la integran:

1. Tiene más de diez años de instalación, con los cambios, agregados y actualizaciones que esto implica.
2. Las máquinas tienen múltiples propósitos, que abarcan trabajo administrativo, prototipación de algoritmos de procesamiento de señales y diseño de circuitos integrados de propósito específico.

La Tabla 4.1 muestra las características más importantes de las computadoras que integran la red local y que se utilizan en la experimentación. En el Apéndice A se dan mayores detalles de cada una de las máquinas. Excepto las computadoras **cetadfomec1** y **cetadfomec2** identificadas con el "tipo" IBM PC, todas las PCs son construidas por partes, lo cual suele ser el caso general de las PCs.

	Nombre	**Tipo**	**CPU**	**Frec. Reloj**	**Mem.**
1)	purmamarca	PC	Pentium II	400 MHz	64 MB
2)	cetadfomec1	IBM PC	Celeron	300 MHz	32 MB
3)	cetadfomec2	IBM PC	Celeron	300 MHz	32 MB
4)	sofia	IBM RS6000	IBM PPC604e	200 MHz	64 MB
5)	fourier	PC	Pentium MMX	200 MHz	32 MB
6)	Josrap	PC	AMD K6-2	450 MHz	62 MB
7)	tilcara	PC	Pentium	133 MHz	32 MB
8)	paris	SPARCstation 4	MicroSPARC-II	110 MHz	96 MB
9)	cetad	SPARCstation 5	MicroSPARC-II	85 MHz	96 MB
10)	prited	SPARCstation 2	CY7C601	40 MHz	32 MB

Tabla 4.1: Computadoras del CeTAD.

Una vez más se debe aclarar que no se ha cambiado nada de lo que se tenía ya instalado, solamente se agregaron las herramientas de software necesarias para el desarrollo, implementación y ejecución de programas paralelos en el caso de las computadoras que no lo tenían antes de la experimentación. El mayor costo de instalación al respecto se dio con las PCs que no tenían instalado Linux ni las herramientas de software necesarias para cómputo paralelo (bibliotecas tales como PVM [44] [PVM]):

- **purmamarca**, se particionó el disco rígido y se instaló Linux (RedHat) y PVM.
- **fourier** y **Josrap**, donde se instaló la distribución Winlinux [WinLinux] dado que era muy difícil (o riesgoso) particionar el disco rígido y se suponía que la distribución Winlinux era la más sencilla de instalar en tales condiciones.

La red de interconexión de las máquinas es Ethernet de 10 Mb/s y el cableado se muestra en la Figura 4.1, donde

1. Se utilizan Hubs en cascada, y la interconexión lógica de las computadoras sigue siendo la de un bus.
2. "Trans." indica *Transceiver*, necesario porque la placa de red de la computadora con nombre **prited** tiene salida con conexión BNC (para cable coaxial) únicamente.

3. "**cf1**" y "**cf2**" se utilizan como abreviaciones de los nombres **cetadfomec1** y **cetadfomec2** respectivamente (y tales abreviaciones se seguirán utilizando por razones de espacio).

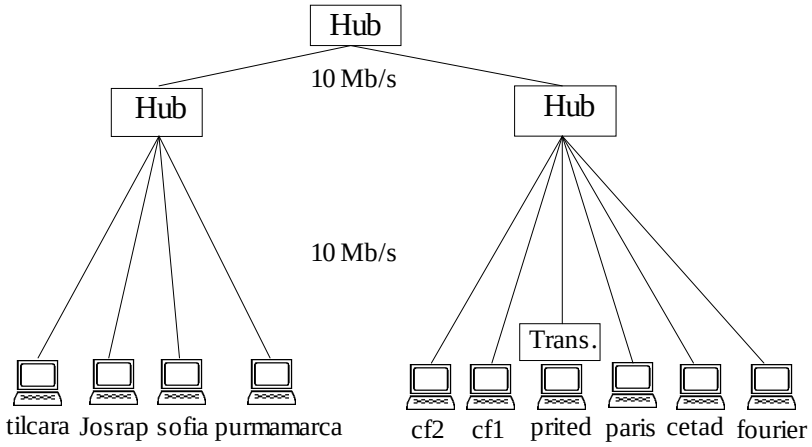


Figura 4.1: Cableado de la Red Local del CeTAD.

4.1.2 Red Local del LQT

A diferencia de la red local del CeTAD, la del LQT está dedicada a resolver problemas numéricos. De hecho, solamente tienen instalado lo mínimo necesario para tal fin, sin herramientas de oficina consideradas *clásicas* como editores/formateadores de texto o planillas de cálculo. Sin embargo, a semejanza de la red local del CeTAD, está instalada y en uso desde hace varios años y consecuentemente ha sido actualizada (y *aumentada*) varias veces.

La Tabla 4.2 muestra las características más importantes de las computadoras que integran la red local y que se utilizan en la experimentación. En el Apéndice A se dan mayores detalles de cada una de las máquinas.

	Nombre	**Tipo**	**CPU**	**Frec. Reloj**	**Mem.**
1)	lqt_07	PC	Pentium III	1 GHz	512 MB
2)	lqt_06	PC	Pentium III	1 GHz	512 MB
3)	lqt_02	PC	Celeron	700 MHz	512 MB
4)	lqt_01	PC	Pentium III	550 MHz	512 MB
5)	lqt_03	PC	Pentium II	400 MHz	512 MB
6)	lqt_04	PC	Pentium II	400 MHz	512 MB

Tabla 4.2: Computadoras del LQT.

A diferencia de la red local del CeTAD, todas las computadoras disponibles en esta red local son PCs construidas por partes y tienen una capacidad bastante mayor en cuanto a cálculo (procesadores y frecuencias de reloj de operación) y almacenamiento (memoria

principal instalada).

La red de interconexión de las máquinas es Ethernet de 10 Mb/s y el cableado se muestra en la Figura 4.2, donde se puede notar que la interconexión es de las más sencillas posibles.

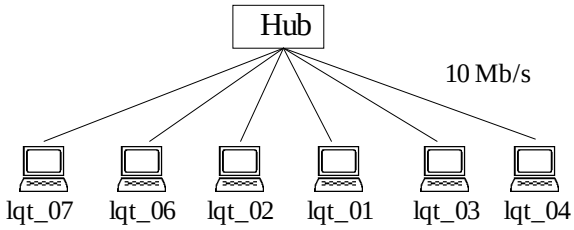


Figura 4.2: Cableado de la Red Local del LQT.

4.1.3 Red Local del LIDI

A diferencia de las dos redes locales anteriores:
- La red local del LIDI fue planificada y construida exclusivamente para cómputo paralelo y por esta razón también coincide con una instalación Beowulf.
- No llega a tener un año de instalada y por lo tanto no ha tenido ningún cambio desde su instalación, manteniéndose homogénea.

La Tabla 4.3 muestra las características más importantes de las computadoras que integran la red local y que se utilizan en la experimentación. En el Apéndice A se dan mayores detalles de cada una de las máquinas. Se muestran las computadoras en una tabla solamente para utilizar el mismo formato que en el caso de las redes anteriores, pero dado que las computadoras son iguales alcanza con la descripción de una de ellas.

	Nombre	**Tipo**	**CPU**	**Frec. Reloj**	**Mem.**
1)	lidipar14	PC	Pentium III	700 MHz	64 MB
2)	lidipar13	PC	Pentium III	700 MHz	64 MB
3)	lidipar12	PC	Pentium III	700 MHz	64 MB
4)	lidipar9	PC	Pentium III	700 MHz	64 MB
5)	lidipar8	PC	Pentium III	700 MHz	64 MB
6)	lidipar7	PC	Pentium III	700 MHz	64 MB
7)	lidipar6	PC	Pentium III	700 MHz	64 MB
8)	lidipar5	PC	Pentium III	700 MHz	64 MB

Tabla 4.3: Computadoras del LIDI.

Como lo muestra la Figura 4.3, el cableado de la red del LIDI coincide en cuanto a simplicidad con el de la red del LQT, pero no en cuanto a costo y rendimiento ya que
- Las placas de red instaladas son Ethernet de 10/100 Mb/s, lo cual implica que la velocidad de comunicación depende del hub o switch que las interconecta.
- En vez de utilizar un hub se utiliza un switch que al igual que las placas de red de las

computadoras también es de 10/100 Mb/s.
- Toda la red entonces tiene la capacidad de llevar a cabo varias (hasta cuatro) comunicaciones punto a punto simultáneas de 100 Mb/s.

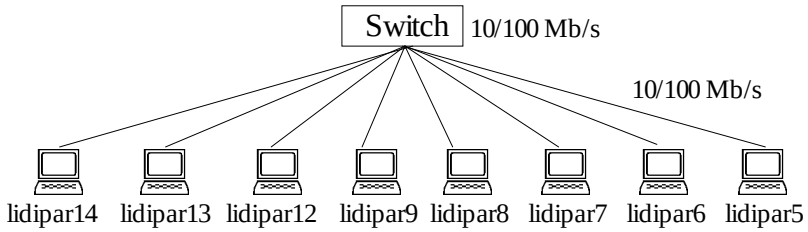


Figura 4.3: Cableado de la Red Local del LIDI.

4.2 Rendimiento Secuencial de las Computadoras

El cálculo del rendimiento secuencial de las computadoras tiene principalmente dos razones:
1. Cálculo del speedup obtenido al utilizar procesamiento paralelo. Para conocer la ganancia de utilizar procesamiento paralelo necesariamente se debe conocer como mínimo el rendimiento de la computadora más rápida disponible en cada red local.
2. Tal como se adelantó en el capítulo anterior, se calcula la velocidad relativa de las computadoras de acuerdo con la capacidad de cálculo secuencial de cada una de ellas para resolver una multiplicación de matrices.

Por estas dos razones se incluyen en esta sección los valores de rendimiento obtenidos para cada una de las máquinas en cada una de las redes locales. Además, en el Apéndice B se explica detalladamente cómo se obtuvieron estos valores de acuerdo a los distintos experimentos realizados.

Dada la naturaleza casi exclusivamente numérica de procesamiento de datos, los valores de rendimiento se expresan en Mflop/s (Millones de operaciones de punto flotante por segundo). La representación de los datos numéricos es la de punto flotante de precisión simple (norma IEEE 754 [72]) en todas las computadoras. La opción de punto flotante de precisión doble, aunque recomendada en general [11] se descarta por dos razones:
- Las computadoras utilizadas tienen la misma o similar capacidad de procesamiento de números en punto flotante de precisión simple que de precisión doble, dadas las características de sus unidades de punto flotante.
- Al utilizar números representados en precisión simple se pueden tener mayores tamaños de matrices en memoria y por lo tanto mayores requerimientos en cuanto a cantidad de operaciones de punto flotante necesarias para resolver una multiplicación de matrices.

4.2.1 Tamaños de Matrices Utilizados

Es indudable que el rendimiento de las computadoras en cuanto a procesamiento de datos en general y de números en punto flotante en particular depende de las relaciones entre:

- Cantidad de datos.
- Jerarquía de memoria (niveles y tamaños de memoria cache).
- Patrón de acceso a los datos.

Y es por esto que los valores de rendimiento se muestran en función de tamaños de matrices considerados significativos. Estas relaciones y los detalles más significativos de los tamaños elegidos en particular se explican con mayor precisión en el Apéndice B.

Por un lado, es importante tener una referencia de la capacidad de procesamiento de las computadoras cuando todos o una buena proporción de los datos a procesar se pueden contener en los niveles de memoria cache más cercanos al procesador (niveles 1 y 2, por ejemplo). En este contexto, se tomaron como referencia varios tamaños de matrices que son relativamente pequeños con respecto al tamaño de memoria principal: matrices de orden n = 100, 200, 400. Teniendo en cuenta que los datos numéricos se representan con números en punto flotante de precisión simple, para n = 100, la cantidad de datos necesaria para almacenar una matriz será de $100^2 \times 4$ bytes, un poco menos de 40 KB de datos.

La utilización de las computadoras al límite de su capacidad (al menos en cuanto a memoria principal) ha sido una constante y, de alguna manera, es lo que se refleja en la "revisión" a la Ley de Amdhal [6] [60]. Dos valores se tomaron como representativos de los tamaños de matrices que se pueden manejar en memoria principal de 32 MB: n = 800 y n = 1600. Con matrices de 800×800 datos, la cantidad de memoria necesaria para contener las tres matrices que intervienen en una multiplicación ($C = A \times B$) es de aproximadamente 7.3 MB de datos (22.8% del total de 32 MB de memoria principal aproximadamente). En el caso de matrices de 1600×1600 datos, la cantidad de memoria requerida es de aproximadamente 29.3 MB, lo que representa el 91.6% del total de 32 MB de memoria principal.

Por lo tanto, en todas las computadoras se realizaron los experimentos con matrices cuadradas de orden n = 100, 200, 400, 800 y 1600. En el caso de las computadoras con memoria principal de 64 MB o 512 MB, y para tener valores de referencia para ser utilizados en el cálculo de speedup, se llevaron a cabo experimentos con matrices mayores. Además, dado que siempre se tiende a la utilización de las computadoras en el límite de su capacidad, también se llevaron a cabo experimentos con el máximo posible en cuanto a tamaño de las matrices. Como es de suponer, esto depende no solamente del tamaño de la memoria principal instalada sino también del espacio de swap configurado en el sistema.

Para las computadoras de 64 MB de memoria principal, los tamaños considerados representativos de los problemas que requieren una buena parte o toda la memoria principal corresponden a valores de n = 1900, 2000, 2200 y 2400. Estos tamaños de matrices implican los porcentajes aproximados de requerimientos de memoria (asumiendo 64 MB en total) de: 65%, 72%, 87%, y 103% respectivamente. Se debe recordar que es posible experimentar con los valores cercanos y superiores al 100% de requerimientos de memoria principal dependiendo del tamaño de memoria swap configurada.

Si bien cuando los datos ocupan toda o la mayor parte de la memoria principal se puede considerar que se está utilizando *al máximo* una computadora (al menos en cuanto a datos en memoria principal), el caso extremo se da cuando se consideran los tamaños que exceden la capacidad de memoria principal y se recurre al espacio de memoria swap. En las

computadoras de 64 MB de memoria principal, el tamaño máximo con el cual se pudo llevar a cabo la multiplicación de matrices es para n = 3200, y como referencia se hicieron también experimentos con n = 3000. Estos tamaños de matrices implican los porcentajes aproximados de requerimientos de memoria (asumiendo 64 MB en total) de: 183% y 161% respectivamente. Como se puntualizó antes, los tamaños máximos del problema dependen de tres aspectos: a) memoria principal instalada, b) espacio de swap configurado y c) sistema operativo, ya que es éste el que en última instancia decide cuándo cancelar un proceso por falta de memoria. Y estos tres aspectos coinciden al menos en la máquinas más rápidas de la red del CeTAD y de la red del LIDI.

Para las computadoras de 512 MB de memoria principal, los tamaños considerados representativos de los problemas que requieren una buena parte o toda la memoria principal corresponden a valores de n = 4000, 5000, 6000 y 7000. Estos tamaños de matrices implican los porcentajes aproximados de requerimientos de memoria (asumiendo 512 MB en total) de: 36%, 56%, 80%, y 110% respectivamente. Se debe notar que es posible experimentar con los valores cercanos y superiores al 100% de requerimientos de memoria principal dependiendo del tamaño de memoria swap configurada.

En las computadoras de 512 MB de memoria principal, el tamaño máximo con el cual se pudo llevar a cabo la multiplicación de matrices es para n = 9000, y como referencia se hicieron también experimentos con n = 8000. Estos tamaños de matrices implican los porcentajes aproximados de requerimientos de memoria (asumiendo 512 MB en total) de: 181% y 143% respectivamente.

4.2.2 Red Local del CeTAD

Los valores de rendimiento obtenidos para cada una de las computadoras de la red local del CeTAD se muestran en la Figura 4.4.

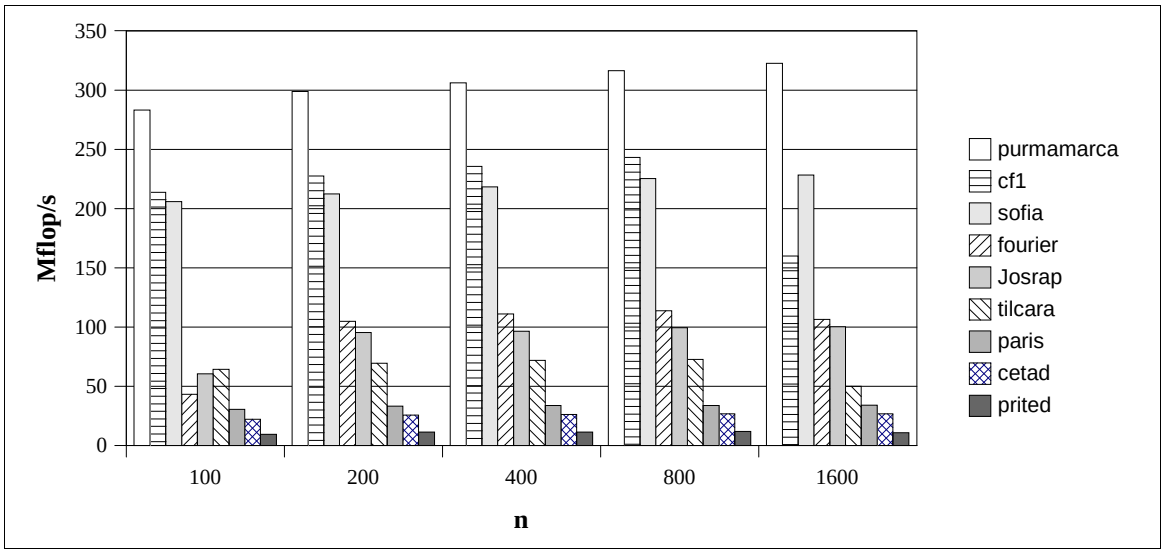


Figura 4.4: Rendimiento de las Computadoras del CeTAD.

En la Figura 4.4 se puede notar que:

- **purmamarca** es la computadora con mayor velocidad relativa y es aproximadamente 30 veces más rápida en procesamiento que **prited**, que es la que tiene menor capacidad.
- Solamente aparece **cf1**, en referencia a **cetadfomec1,** dado que el rendimiento de **cetadfomec2** es el mismo.
- Se puede comprobar con los valores de la Tabla 4.1 que la frecuencia de reloj a las cuales operan las computadoras no necesariamente determina la capacidad de procesamiento (al menos en operaciones de punto flotante).

Las características de rendimiento y/o los valores obtenidos se explican con mayor nivel de detalle en el Apéndice B.

En la computadora con mayor capacidad de procesamiento, **purmamarca**, se llevaron a cabo experimentos con matrices mayores y los resultados obtenidos se muestran en la Figura 4.5.

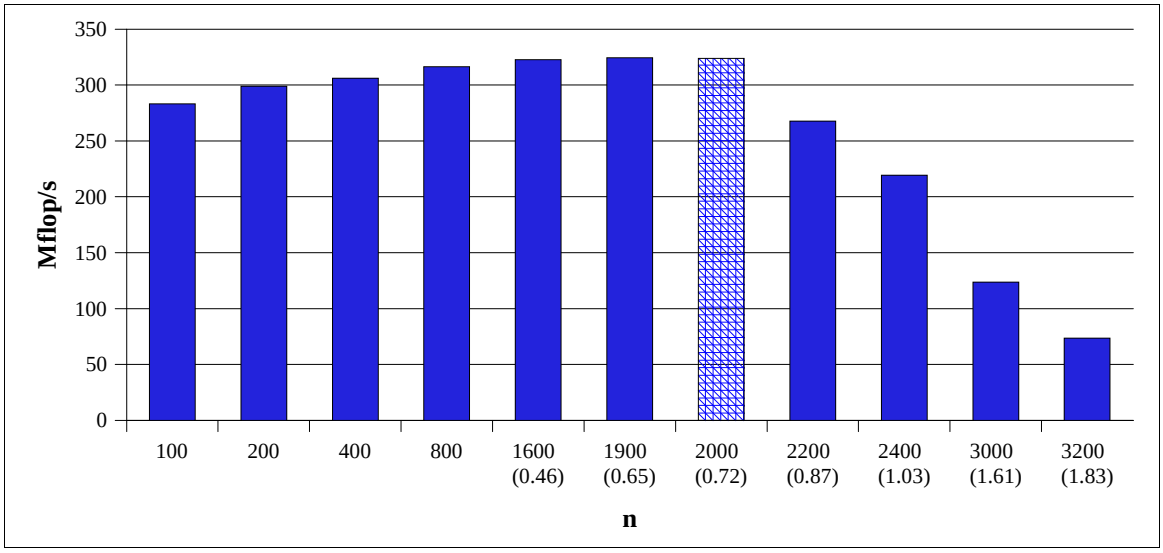


Figura 4.5: Rendimiento de **purmamarca** para Multiplicación de Matrices.

donde:

- Se indica entre paréntesis la proporción de memoria necesaria para contener las tres matrices. Para n = 1900, por ejemplo, se necesita el 65% de la memoria para almacenamiento de datos.
- Se muestra resaltado (con el relleno distinto de la barra correspondiente) el mayor tamaño de matrices para el cual no se utiliza espacio de swap. Se debe notar que cuando el espacio para contener los datos de las matrices excede el 72% de la memoria ya se comienza a utilizar el espacio de swap y por lo tanto el rendimiento disminuye.

4.2.3 Red Local del LQT

Los valores de rendimiento obtenidos para las computadoras de la red local del LQT se muestran en la Figura 4.6, donde

- Las computadoras **lqt_07** y **lqt_06** son las más veloces y las diferencias relativas no son

tan grandes como en el caso de las computadoras del CeTAD.

- Dado que no hay problema de espacio para la representación de las barras, se incluyen todas las máquinas aunque **lqt_06** es igual a **lqt_07** y **lqt_03** es igual a **lqt_04**. Debe notarse que en el contexto de las PCs construidas por partes, aunque las computadoras sean "iguales" en cuanto a procesador, frecuencia de reloj del sistema y cantidad de memoria instalada, aún es posible que tengan distinto rendimiento porque, por ejemplo, tienen distinta velocidad de acceso a memoria.
- Una vez más la frecuencia de reloj de operación no necesariamente determina la velocidad relativa de las máquinas (ver Tabla 4.2).

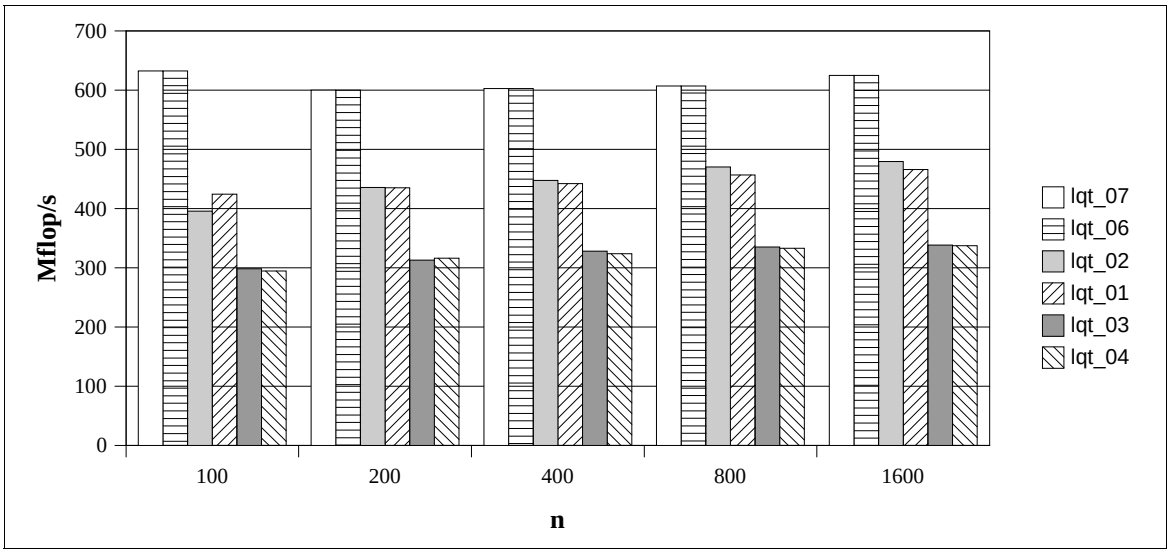


Figura 4.6: Rendimiento de las Computadoras del LQT.

En la computadora con mayor capacidad de procesamiento, **lqt_07**, se llevaron a cabo experimentos con matrices mayores y los resultados obtenidos se muestran en la Figura 4.7, donde se puede notar que:

- El mayor tamaño de matrices en el cual no se utiliza espacio de swap durante el procesamiento es n = 5000. La cantidad de memoria que se debe destinar al almacenamiento de los datos de las matrices de 5000×5000 elementos representa aproximadamente el 56% del total de memoria.
- Nuevamente se reduce el rendimiento a medida que es necesario utilizar mayor espacio de memoria swap aunque esta caída no es tan abrupta en relación a la que se produce en **purmamarca** (Figura 4.5).
- Muy probablemente, para matrices de 100x100 elementos la mayoría de los datos puedan ser alojados en la/s memoria/s cache/s del procesador y por lo tanto se tiene mejor rendimiento que para las matrices de, por ejemplo 200x200 y 400x400 elementos. Este efecto se *diluye* a medida que las matrices son mayores y por lo tanto siempre se logra optimizar el acceso a los datos en cache (reutilizarlos tantas veces como es posible) por el procesamiento por bloques de datos que utiliza el código optimizado.

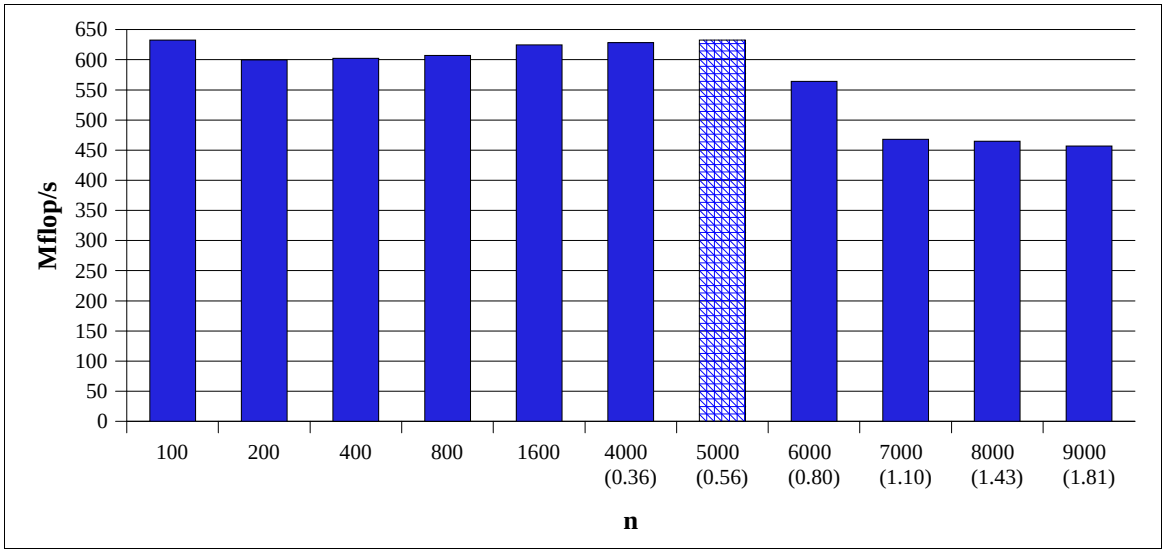


Figura 4.7: Rendimiento de **lqt_07** para Multiplicación de Matrices.

4.2.4 Red Local del LIDI

Dada la homogeneidad de las máquinas del LIDI, se muestran los resultados obtenidos en una de las máquinas, **lidipar_14**, para todos los tamaños de matrices en la Figura 4.8.

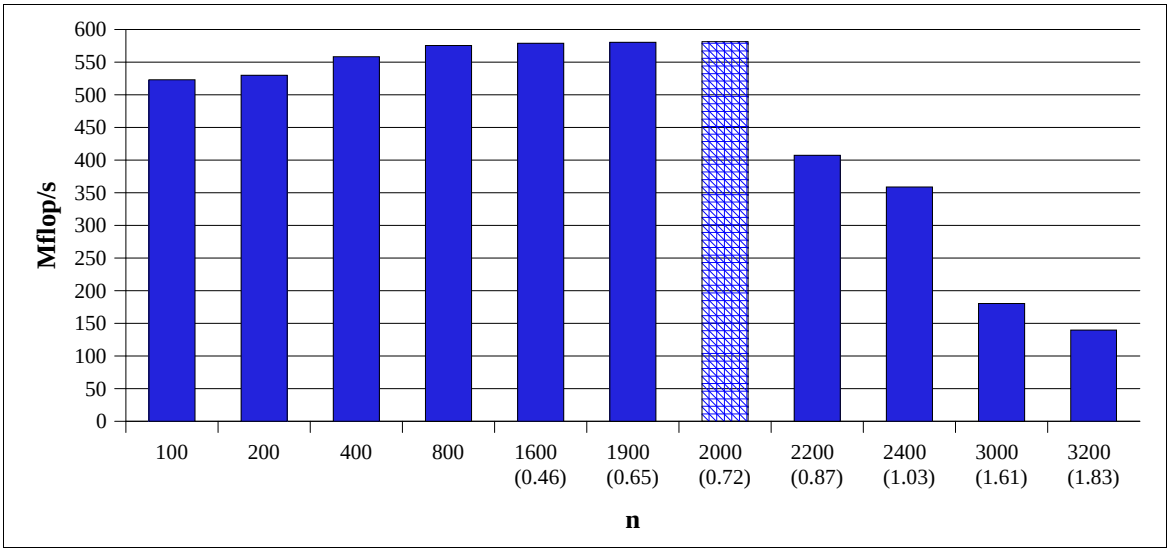


Figura 4.8: Rendimiento de **lidipar14** para Multiplicación de Matrices.

El *comportamiento* del rendimiento de las computadoras de la red del LIDI es similar al de **purmamarca** de la red local del CeTAD dado que:

- El mayor tamaño de matrices para el cual no se utiliza espacio de memoria swap durante el procesamiento de la multiplicación de matrices es n = 2000, lo cual es consistente dado que tienen la misma memoria principal instalada: 64 MB.

- A partir de que se comienza a utilizar espacio de memoria swap durante el procesamiento de la multiplicación de matrices el rendimiento cae abruptamente.

A diferencia de **purmamarca**, las computadoras de la red local del LIDI son bastante más veloces, dado que llegan a procesar a razón de más de 580 Mflop/s mientras que **purmamarca** no llega a 350 Mflop/s.

Dado que las computadoras de la red local del LIDI son todas iguales, se las puede utilizar, y de hecho se las utiliza como referencia del algoritmo de multiplicación de matrices en redes homogéneas. También esta red es la más apropiada para cómputo paralelo ya que la la red de interconexión es de 100 Mb/s (las otras dos son de 10 Mb/s) y además en el cableado se utiliza un switch en vez de uno o más hubs.

4.3 Análisis de Rendimiento Paralelo de las Redes Locales

Siempre se ha buscado definir y calcular analíticamente el rendimiento óptimo o posible de las computadoras paralelas y también el rendimiento que se puede obtener con un algoritmo paralelo sobre una computadora paralela en particular. Las razones más importantes para calcular analíticamente el mejor rendimiento posible de una computadora paralela son:

- Estimar de antemano si la computadora paralela es capaz de proporcionar un resultado en un tiempo determinado. No sería útil, por ejemplo, tener en dos semanas la predicción del estado meteorológico de un día de la semana siguiente.
- Evaluar y comparar máquinas paralelas para, por ejemplo, calcular la relación entre el costo y el beneficio (o costo/rendimiento) de cada una de ellas.

Por otro lado, estimar analíticamente el rendimiento de un algoritmo paralelo es útil para determinar si el algoritmo diseñado puede obtener el máximo rendimiento de la computadora para la cual se implementa y donde se ejecuta para resolver el problema un dado.

Dado que ya se dispone de

- Las características de rendimiento de cada una de las máquinas de todas las redes locales.
- Las características de rendimiento de la red de interconexión de cada una de las redes locales.
- Las principales características de los algoritmos de cómputo paralelo para el cálculo de la multiplicación de matrices (con y sin mensajes solapados, por ejemplo).

Se puede calcular analíticamente el mejor valor posible para el índice de speedup y utilizarlo luego como referencia en la evaluación de los resultados de experimentación. En cierta forma, el cálculo analítico del mejor valor de speedup (o speedup óptimo) trata de predecir el rendimiento de las redes de computadoras utilizadas como máquinas paralelas.

4.3.1 Cálculo del Speedup *Real*

La idea básica del índice de speedup de una computadora paralela es determinar cuántas veces más capacidad tiene una computadora paralela con respecto a un procesador, o a una computadora secuencial. La definición *clásica* del speedup es

Tiempo de ejecución del mejor algoritmo secuencial / Tiempo de ejecución paralelo

Por lo tanto, habría que determinar el *Tiempo de ejecución del mejor algoritmo secuencial* y también el *Tiempo de ejecución paralelo*. En el ambiente heterogéneo de las redes de computadoras instaladas el

Tiempo de ejecución del mejor algoritmo secuencial

se "transforma" en [147]

Tiempo de ejecución del mejor algoritmo secuencial en la computadora más rápida

o, de forma resumida,

Tiempo de ejecución en la computadora más rápida

asumiendo directamente que siempre se utilizará el mejor algoritmo. Básicamente se trata del mejor tiempo de ejecución secuencial posible en la red de estaciones de computadoras, es decir utilizando

- la computadora con mayor capacidad de cálculo y
- el mejor algoritmo secuencial.

En el caso de las tres redes locales que se presentaron esta tarea ya está resuelta, dado que:

- En la red local del CeTAD, **purmamarca** es la de mayor velocidad relativa y la experimentación ya ha determinado su capacidad en Mflop/s que a su vez determina unívocamente el tiempo de cómputo.
- En la red local del LQT, **lqt_07** es la de mayor velocidad relativa y la experimentación ya ha determinado su capacidad en Mflop/s que a su vez determina unívocamente el tiempo de cómputo.
- En la red local del LIDI son todas las computadoras iguales y la experimentación ya ha determinado la capacidad en Mflop/s de **lidipar14** que a su vez determina unívocamente el tiempo de cómputo.

De la misma manera, el *tiempo de ejecución paralelo* se determina vía experimentación, utilizando las máquinas disponibles de cada una de las redes locales.

4.3.2 Cálculo del Speedup *Optimo*

Desde el punto de vista teórico, lo mejor que puede suceder en una máquina paralela es que todos los procesadores se utilicen todo el tiempo o que todos los procesadores se utilicen a

su máxima capacidad de cómputo. Esto induce a asumir que la capacidad de cálculo de la computadora paralela es igual a la suma de las capacidades de cálculo de cada uno de los procesadores que son parte de la misma. En el contexto de las máquinas paralelas con procesadores homogéneos esto significa que utilizar un procesador más indica reducir proporcionalmente el tiempo de ejecución paralelo. Es decir que si se utilizan *P* procesadores, el mejor tiempo de ejecución paralelo está dado por

Tiempo de ejecución paralelo = Tiempo de ejecución del mejor algoritmo secuencial / P

Y de hecho, no es más que asumir que la potencia de cálculo de la máquina paralela con *P* procesadores es *P* veces mayor que la potencia de cálculo de la máquina secuencial (un procesador). Puesto de otra forma, el speedup consiste en identificar la relación entre la potencia de una máquina con un procesador y una máquina paralela con *P* procesadores. De esta manera se llega a que el speedup óptimo en las computadoras paralelas *clásicas* homogéneas es igual a la cantidad de procesadores que se utilizan. Esto es equivalente a definir la "potencia de cálculo relativa de la máquina paralela con respecto a un procesador" o directamente el valor del speedup óptimo como

$$SpeedupOptimo = \sum_{i=0}^{P-1} rpw(proc_i) \tag{4.1}$$

donde $proc_0$, $proc_1$, ..., $proc_{P-1}$, son los *P* procesadores de la máquina paralela y $pw(proc_i)$ es la potencia de cálculo relativa de $proc_i$ con respecto a los demás o a cualquiera de los demás procesadores. En el contexto de las aplicaciones numéricas se puede calcular utilizando la capacidad de cálculo en Mflop/s como en el capítulo anterior, pero asumiendo que los procesadores son iguales se cumple que

$$rpw(proc_i) = 1; \quad \forall\ i = 0, ..., P\text{-}1 \tag{4.2}$$

y por lo tanto

$$SpeedupOptimo = P \tag{4.3}$$

En el contexto de las aplicaciones numéricas, esto es equivalente al cálculo del speedup óptimo utilizando directamente las potencias de cálculo de la máquina secuencial y la máquina paralela en Mflop/s, es decir como

$$SpeedupOptimo = \frac{\sum_{i=0}^{P-1} Mflop/s(proc_i)}{Mflop/s(proc_0)} \tag{4.4}$$

En el contexto de las máquinas paralelas con procesadores heterogéneos no es posible afirmar lo expresado en la Ecuación (4.2) ya que los procesadores tienen o pueden tener distinta velocidad relativa. Por lo tanto, se debe calcular *SpeedupOptimo* de acuerdo con la

Ecuación (4.1), es decir utilizando el cálculo de cada $rpw(proc_i)$ o, como se denomina de manera equivalente en el capítulo anterior, $pw(ws_i)$, dada por

$$pw(ws_i) = \frac{Mflop/s(ws_i)}{\underset{j=0..P-1}{máx}\ (Mflop/s(ws_j))} \tag{4.5}$$

Y de manera análoga, la Ecuación (4.3) se debe adaptar al ambiente heterogéneo como lo muestra la Ecuación (4.6), que determina como referencia al procesador con mayor capacidad de cálculo entre todos los utilizados.

$$SpeedupOptimo = \frac{\sum_{i=0}^{P-1} Mflop/s(ws_i)}{\underset{j=0..P-1}{máx}\ (Mflop/s(ws_j))} \tag{4.6}$$

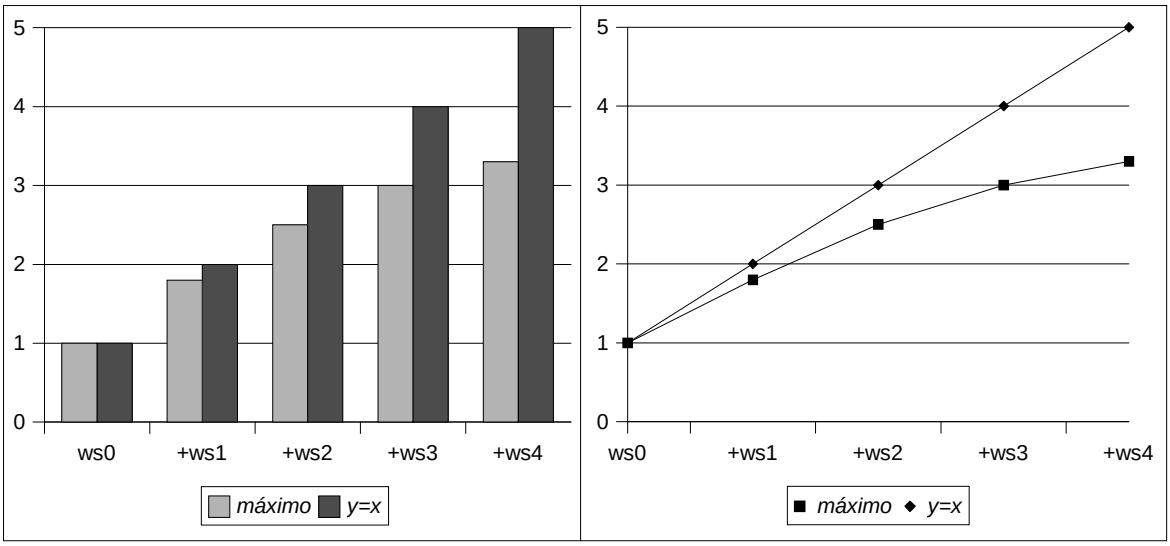
De esta manera, también se *pierden* dos ideas subyacentes en la interpretación de los gráficos de speedup que provienen de las máquinas paralelas homogéneas:

- Ya no se puede afirmar que el máximo teórico está dado por la recta $y = x$, o que para x cantidad de procesadores el máximo teórico del speedup está dado por x. En el ambiente heterogéneo ya no es posible relacionar la cantidad de procesadores con la potencia de cálculo de la máquina paralela completa. Más específicamente, agregar un procesador implica agregar potencia de cálculo pero no necesariamente relacionada con la cantidad total de procesadores sino con la suma de las potencias de cálculo de los procesadores.
- Ya no se puede afirmar que como mínimo debería lograrse speedup lineal, o que aunque el speedup no sea exactamente igual a la cantidad de procesadores, debería ser directamente proporcional a la cantidad de procesadores. Ya no es posible mantener esta idea por la misma razón dada anteriormente: no es posible relacionar o cuantificar la relación entre la cantidad de procesadores (o máquinas) con la potencia de cálculo de la máquina paralela.

La Figura 4.9 muestra los valores de speedup máximos en una red de cinco computadoras ws0, ..., ws4, cada una con su potencia de cálculo relativa dada por

pw(ws0) = 1.0 pw(ws1) = 0.8 pw(ws2) = 0.7
pw(ws3) = 0.5 pw(ws4) = 0.3

En la Figura 4.9-a) los valores de speedup se muestran con barras y en la Figura 4.9-b) los valores se unen con líneas, mostrando un poco más claramente cómo el máximo speedup posible para estas cinco computadoras se “aleja” de la recta $y = x$ a medida que se agregan computadoras. También la Figura 4.9 muestra la tendencia a incorporar las computadoras más rápidas de las que están disponibles. En el caso de utilizar las computadoras ws0 y ws1, la que más probablemente se incorpore es ws2 ya que es la que tiene mayor capacidad de cálculo de las tres disponibles: ws2, ws3 y ws4, es por eso que en los gráficos de speedup se incorporan las máquinas de “mayor a menor” de acuerdo a su capacidad de cálculo, a menos que explícitamente se determine otro criterio.



a) b)

Figura 4.9: Speedup de Cinco Computadoras Heterogéneas.

Es importante notar que esta forma de calcular el máximo valor de speedup asume que *todas* las computadoras y en particular la más veloz tienen *siempre* la misma capacidad de cálculo. Cuando las computadoras se utilizan a su máxima capacidad configurada en cuanto a espacio de swap es posible y de hecho muy probable que se puedan resolver tamaños de problemas mayores de los que permite la memoria principal disponible y se utilice el espacio de swap. Esto a su vez ocasiona dos inconvenientes desde el punto de vista del rendimiento que son bien conocidos:

1. Mientras se lleva a cabo cálculo hay mayor actividad del sistema operativo por el manejo de los datos que deben transferirse desde y hacia el espacio de swap (normalmente en disco).
2. Es posible y de hecho muy probable que el procesador deba esperar por datos que están en espacio de swap (disco) hasta que se transfieran a memoria principal desde donde el procesador puede utilizarlos para operar con ellos.

Y de hecho, el rendimiento se degrada notablemente, aunque la cuantificación de esta degradación depende de la computadora (velocidad de disco, subsistema de entrada/salida, etc.) y también del problema (patrón de acceso a los datos, cantidad de datos en espacio de swap, etc.)

Por lo tanto, si el problema resuelto en la computadora secuencial o en la computadora con mayor capacidad de cálculo en el caso de las redes locales, implica la utilización de espacio de swap se tendrá como referencia un tiempo de ejecución afectado por la utilización del espacio de swap. Cuando se considera la posibilidad de ejecutar en paralelo en las computadoras de una red local, de hecho se está implícitamente asumiendo la distribución de los datos del problema y por lo tanto es posible que el mismo tamaño de problema se pueda resolver de forma tal que en cada máquina solamente se utiliza la memoria principal y no se recurre al espacio de memoria swap con la degradación de rendimiento que eso implica. Por lo tanto, dependiendo del tamaño del problema y de las máquinas utilizadas, el valor de speedup que se puede obtener puede ser mayor que el óptimo calculado de

acuerdo con la Ecuación (4.1). Puesto de otra manera, el valor óptimo de speedup calculado según la Ecuación (4.1) asume que la computadora más veloz tiene *siempre* la capacidad de cálculo determinada con la ejecución de parte del problema en espacio de swap, o lo que es similar, es más lenta de lo que *realmente* es (cuando todo el problema puede manejarse en memoria principal sin recurrir al espacio de swap).

Dado que se realizan experimentos secuenciales que implican la utilización del espacio de swap, se debería proporcionar un valor óptimo para el speedup que no sea "desviado" por esta razón. En el contexto de los problemas numéricos, conviene recurrir una vez más a la idea de capacidad de cálculo dada en cantidad de operaciones de punto flotante por segundo, o Mflop/s. Dado que se utilizan directamente los Mflop/s para esta "nueva" forma de cálculo del speedup óptimo también se deben tener en cuenta la cantidad de operaciones que se tienen que realizar. Retomando el ejemplo de las cinco computadoras ws0, ..., ws4, anterior, se debería utilizar ahora la capacidad de cada una de ellas en términos de Mflop/s que pueden ser, por ejemplo

Mflop/s(ws0) = 1000
Mflop/s(ws1) = 800
Mflop/s(ws2) = 700
Mflop/s(ws3) = 500
Mflop/s(ws4) = 300

Y también ahora es necesario conocer la cantidad de operaciones de punto flotante que se necesitan llevar a cabo para resolver el problema, que pueden ser, por ejemplo, 10^9. Por lo tanto, si la computadora con mayor capacidad, ws0, puede resolver el problema sin recurrir al espacio de swap, entonces resuelve los cálculos a su máxima capacidad, es decir a razón de 1000×10^6 operaciones por segundo. En este caso, el máximo speedup "coincide" con el calculado utilizando la Ecuación (4.1), es decir:

$$SpeedupOptimo = \sum_{i=0}^{4} pw(proc_i) = \sum_{i=0}^{4} \frac{Mflop/s(ws_i)}{1000} = 3.3$$

Si, por el contrario, la computadora con mayor capacidad, ws0, utiliza el espacio de swap durante la resolución del problema, ya no realiza los cálculos a su máxima velocidad. Suponiendo que la degradación por la utilización del espacio de swap durante los cálculos es del 30%, esto implica que realiza los cálculos a razón de 700×10^6 operaciones por segundo, por lo tanto, asumiendo que durante la ejecución paralela todas las computadoras operan a su máxima capacidad, utilizando la Ecuación (4.3) se tiene que el speedup óptimo sería

$$SpeedupOptimo = \frac{\sum_{i=0}^{4} Mflop/s(ws_i)}{700} = 4.71$$

que evidentemente es mayor al calculado según los valores *pw*(ws0), ..., *pw*(ws4). En todo caso, se tendrían dos puntos de vista para el cálculo del speedup óptimo. El calculado

según según la Ecuación (4.1), que a su vez depende de las potencias de cálculo relativas, $pw(ws_i)$, calculadas según la Ecuación (4.5), que asume que las computadoras tienen siempre la misma capacidad de cálculo, y que se podría llamar "speedup óptimo de cómputo según las velocidades relativas", o Comp(rsf).

$$\text{Comp}(\text{rsf}) = \sum_{i=0}^{P-1} pw(proc_i) \tag{4.7}$$

La idea subyacente sobre la que se apoya el cálculo de Comp(rsf) es básicamente: si se agrega una máquina, se agrega su potencia de cálculo relativa a la de mayor capacidad. Siguiendo el ejemplo dado con ws0, ..., ws4, esto significa que si en vez de utilizar solamente ws0 se utilizan ws0 y ws1 entonces se debería tener una computadora (paralela) que tiene 1.8 veces la capacidad de ws0.

El speedup óptimo calculado según la Ecuación (4.3) asume que todas las computadoras ejecutan siempre al máximo de su capacidad, independientemente de que sea necesaria la utilización del espacio de swap durante el procesamiento o no. Por lo tanto, se podría llamar "speedup óptimo de cómputo según las capacidades de cálculo de cada computadora dadas en Mflop/s", o Comp(Mf).

$$\text{Comp}(\text{Mf}) = \frac{\sum_{i=0}^{P-1} Mflop/s(ws_i)}{\underset{j=0..P-1}{máx}\ (Mflop/s(ws_j))} \tag{4.8}$$

La idea subyacente sobre la que se apoya el cálculo de Comp(Mf) es básicamente: si se agrega una máquina, se agrega directamente su potencia de cálculo en Mflop/s. Siguiendo el ejemplo dado con ws0, ..., ws4, esto significa que si en vez de utilizar solamente ws0 se utilizan ws0 y ws1 entonces se debería tener una computadora (paralela) que tiene una capacidad de cálculo de 1000+800 Mflop/s = 1800 Mflop/s, lo que significa que la computadora paralela es 1800/700 ≅ 2.57 veces la capacidad de ws0, dado que el valor de referencia en cuanto a Mflop/s de ws0 es 700 Mflop/s, que se tiene utilizando el espacio de swap.

Evidentemente el cálculo del speedup óptimo será el mismo, Comp(rsf) = Comp(Mf) si se toman como referencia los valores de Mflop/s máximos, es decir sin que se utilice el espacio de swap de cada máquina. Comp(rsf) es derivado directamente de la forma clásica del cálculo de speedup y Comp(Mf) sería el que hay que observar con cierta atención cuando el tiempo de ejecución secuencial está afectado por la utilización del espacio de swap. De alguna manera, cuando durante la ejecución secuencial se utiliza el espacio de memoria swap, Comp(Mf) podría entenderse como lo que se ha llamado ocasionalmente "speedup superlineal". Siguiendo con el ejemplo desde este punto de vista:

- Al resolver un problema en ws0 se tiene el tiempo de ejecución derivado de la capacidad de ws0 al utilizar espacio de swap, es decir directamente proporcional a 700 Mflop/s.
- Al resolver el mismo problema con cómputo paralelo y utilizar ws1, es "esperable" que

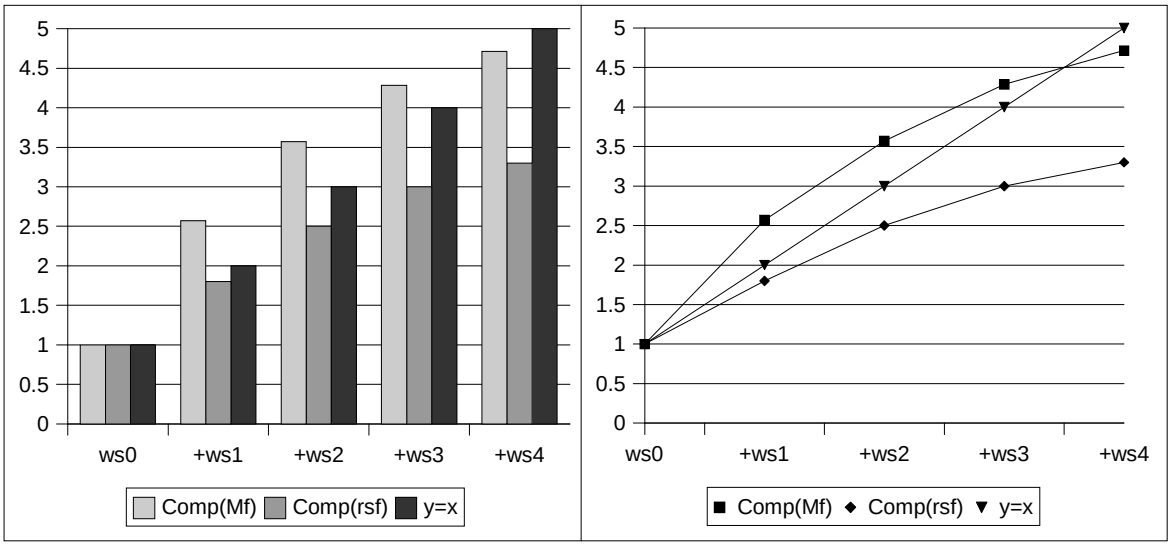
se resuelva el problema considerando una computadora con una capacidad de (700 + 0.8×700) Mflop/s = 1260 Mflop/s dado que ws1 tiene una capacidad de cálculo de 0.8 veces la capacidad de ws0.

- Si la distribución de todo el problema entre ws0 y ws1 hace innecesaria la utilización del espacio de swap, se tendrá que ambas computadoras resuelven los cálculos a su máxima capacidad y por lo tanto el tiempo de ejecución paralelo será proporcional a (1000+800) Mflop/s = 1800 Mflop/s y por lo tanto el tiempo de ejecución será menor que el "esperable" considerando solamente las velocidades relativas.

La Figura 4.10 muestra los diferentes valores de speedup óptimos, Comp(Mf) y Comp(rsf), considerando las cinco computadoras del ejemplo, cuyas características de rendimiento están resumidas en la siguiente tabla

Computadora	***Mflop/s (Máximo)***	***pw***
ws0	1000	1
ws1	800	0.8
ws2	700	0.7
ws3	500	0.5
ws4	300	0.3

y además considerando que la computadora con mayor capacidad de cálculo utiliza el espacio de swap para resolver el problema dado con su consiguiente penalización en rendimiento del 30%.



a) b)

Figura 4.10: Cálculos de Speedup Optimos para Cinco Computadoras.

En la Figura 4.10-a) los valores se muestran con barras y en la Figura 4.10-b) los valores se unen con líneas, donde se puede ver claramente que:

- Las diferentes formas de cálculo de speedup óptimo proporcionan distintos valores en el

caso de que la solución secuencial implica la utilización del espacio de swap.
- Se confirma que la recta $y = x$ no proporciona información significativa en el contexto de procesadores heterogéneos.

Por último, se debe hacer notar que en el cálculo del speedup óptimo no hay ninguna consideración respecto de las comunicaciones, siempre se tiene en cuenta *solamente* la capacidad de cálculo de todos los procesadores (computadoras) utilizadas.

4.4 Análisis de Rendimiento de los Algoritmos

En general, el cálculo analítico del rendimiento de los algoritmos paralelos tiene dos propósitos generales:
- Determinar si el algoritmo es capaz de aprovechar el rendimiento de la computadora paralela sobre la cual puede implementarse.
- Comparar y evaluar distintos algoritmos diseñados para una misma tarea.

El cálculo analítico de rendimiento de los algoritmos tiene (o debe tener) en cuenta no solamente las características de cómputo de las computadoras sino que también incorpora como mínimo otro factor que afecta el rendimiento: las comunicaciones. Aunque se conozca la arquitectura de una computadora paralela con todos los detalles necesarios, es muy difícil cuantificar el impacto que tienen las comunicaciones sobre el rendimiento de las aplicaciones que se resuelven. Sin embargo, la situación cambia al diseñar un algoritmo paralelo específico dado que éste determina con claridad todo lo necesario relacionado con las comunicaciones y la sincronización entre los procesos. En general, se consideran los puntos de sincronización entre procesos como una clase de comunicación en particular.

Aunque son muy similares, los algoritmos a considerar para analizar son dos, y fueron presentados en el capítulo anterior:
- Con los mensajes explícitamente secuenciales respecto del cómputo. Es decir que en todo instante de tiempo una computadora puede estar haciendo una de dos tareas:
  - Cómputo local, es decir resolviendo un cálculo parcial de la porción de la matriz resultado que debe computar.
  - Resolviendo comunicaciones de datos, más específicamente enviando o recibiendo un mensaje broadcast.
- Con los mensajes solapados, de forma tal que la mayoría de los mensajes se pueden transmitir mientras se lleva a cabo cómputo local (*simultáneamente*). Para que esto ocurra en realidad cada computadora debe ser capaz de realizar cómputo y transmisiones de datos a la vez.

Ambos algoritmos se presentaron en el capítulo anterior, junto con la forma analítica para el cálculo del rendimiento de cada uno de ellos y se denominarán SeqMsg y OverMsg respectivamente. En todos los casos, se asume que los períodos de cómputo se llevan a cabo a la máxima capacidad de cálculo de las computadoras involucradas.

4.4.1 SeqMsg: Cómputo y Comunicaciones Secuenciales

De acuerdo con lo explicado en el capítulo anterior, el tiempo paralelo del algoritmo en el cual los períodos de cómputo y comunicaciones se llevan a cabo de forma secuencial está dado por

$$t_{par\_seqmsg} = P\alpha + \beta n^2 + \frac{2n^3 - n^2}{pw}$$

donde

- P es la cantidad de computadoras que se utilizan.
- n es el orden de las matrices cuadradas que se multiplican.
- α es el tiempo de latencia de la red de comunicaciones.
- $1/\beta$ es el ancho de banda (tasa de transferencia) asintótico de la red de comunicaciones, expresado en términos del tipo de elementos de las matrices que se multiplican.
- pw es la suma de todas las capacidades de cálculo de las computadoras utilizadas expresadas en términos de Mflop/s, es decir

$$pw = \sum_{i=0}^{P-1} Mflop/s(ws_i) \tag{4.9}$$

Aunque no explícitamente definido, se tiende a asumir que

- El tiempo de latencia no es demasiado importante siempre y cuando los mensajes sean suficientemente grandes, o lo que es lo mismo, cuando el problema es suficientemente grande, dado que el tamaño de los mensajes está directamente relacionado con el tamaño del problema [71] [52] [146].
- El ancho de banda asintótico de la red de comunicaciones es independiente de la cantidad de procesos que se comunican con un broadcast o, más específicamente, la cantidad de procesos receptores de cada mensaje broadcast. Esto es posible en las redes Ethernet siempre y cuando las rutinas de comunicaciones aprovechen las características del hardware de comunicaciones.

Por lo tanto, se puede simplificar un poco la forma de cálculo del tiempo de ejecución paralelo eliminando completamente el tiempo de latencia de los mensajes (o, lo que es igual, considerándolo igual a cero) y se llega a

$$t_{par\_seqmsg} = \beta n^2 + \frac{2n^3 - n^2}{pw}$$

Y este tiempo paralelo es el que se utiliza para el cálculo del speedup óptimo que este algoritmo puede obtener en una red de computadoras, dando lugar a lo que se denominará **SeqMsg(Mf)**.

4.4.2 OverMsg: Cómputo y Comunicaciones Solapadas

De acuerdo con lo explicado en el capítulo anterior, el tiempo paralelo del algoritmo en el cual gran parte de los períodos de cómputo y comunicaciones se llevan a cabo de forma solapada (*simultánea*) está dado por

$$t_{par\_overmsg} = t_{bcast} + (P\text{-}1)\ máx(t_{bcast}, t_{cómp}) + t_{cómp}$$

donde

$$t_{bcast} = \alpha + \beta\, n^2/P \qquad \text{y} \qquad t_{cómp} = \frac{2n^3 - n^2}{P\ pw}$$

Y este tiempo paralelo es el que se utiliza para el cálculo del speedup óptimo que este algoritmo puede obtener en una red de computadoras, dando lugar a lo que se denominará **OverMsg(Mf)**.

Con esta forma de calcular analíticamente el tiempo de ejecución paralelo se asume que:
- Todas las computadoras son capaces de solapar cómputo con comunicaciones.
- El solapamiento de las comunicaciones no afecta el tiempo de cómputo local ni el tiempo de comunicaciones de los mensajes broadcast.

Es de destacar que ambas suposiciones son muy difíciles de verificar al menos en las computadoras estándares de las redes locales instaladas.

4.5 Redes Locales y Algoritmos

Dado que ya se dispone de:
1. rendimiento secuencial de todas las computadoras de todas las redes locales: Mflop/s(ws_i)
2. forma analítica de calcular el rendimiento de cada una de las redes locales: Comp(Mf) y Comp(rsf).
3. forma analítica de los dos algoritmos propuestos: t_{par_seqmsg} y $t_{par\_overmsg}$.
4. estimación, a menos a nivel de hardware, del ancho de banda asintótico de cada una de las redes locales: Mb/s de las redes Ethernet.

ya es posible estimar el rendimiento tanto de cada una de las redes como de los algoritmos sobre estas redes, al menos en lo que se refiere al máximo *teórico*.

4.5.1 Red Local del CeTAD

La Figura 4.11 muestra los máximos valores de speedup a considerar en la red local del CeTAD cuando todos los datos del problema se pueden contener en memoria principal, de forma tal que no es necesaria la utilización de memoria swap durante los cálculos. En este caso, la computadora de referencia (la de mayor capacidad de cálculo) es **purmamarca** y el

tamaño de matrices es n = 2000. Además, dado que se utiliza una red Ethernet de 10 Mb/s, se asume que por la degradación producida por todas las capas del sistema operativo se pueden transferir datos entre procesos de usuario a razón de 2^{20} bytes (1 MB) por segundo. Podría considerarse una suposición optimista, pero aceptable dado que se estiman valores máximos.

En la Figura 4.11 se puede notar que:

- La capacidad de cómputo relativa de las computadoras proporciona un orden razonable para ser utilizadas en paralelo. Cuando se usa una cantidad determinada de computadoras para cómputo paralelo se tiende a incluir las de mayor capacidad de cálculo entre las disponibles.
- Las computadoras **cetadfomec1** y **cetadfomec2** se muestran con los nombres **cf1** y **cf2** respectivamente.
- Como se esperaba, Comp(Mf) y Comp(rsf) coinciden, dado que la capacidad de cómputo de referencia de **purmamarca** es la máxima (aproximadamente 324 Mflop/s, Figura 4.5), porque no utiliza el espacio de swap durante la ejecución secuencial.
- El speedup calculado para el algoritmo con los mensajes solapados OverMsg(Mf) es similar al de cómputo, Comp(Mf), hasta utilizar la computadora **sofia** inclusive, pero agregando más computadoras casi no hay mejora en cuanto a rendimiento. Expresado de otra manera, a partir de la incorporación de **fourier**, el tiempo de comunicaciones es mayor que el de cómputo y por lo tanto casi no hay mejora en cuanto a tiempo de ejecución paralelo por la incorporación de más computadoras.
- El peso relativo del tiempo de comunicación con respecto al tiempo de cómputo se evidencia para el algoritmo con los mensajes y cómputo secuenciales. El speedup calculado para este algoritmo, SeqMsg(Mf) así lo muestra por la diferencia en los valores con respecto a los demás cálculos de speedup, incluyendo el del algoritmo con los mensajes solapados, que tiene en cuenta al menos una parte del tiempo total de comunicaciones.
- El total de la capacidad de cálculo de las diez computadoras es poco menos que 4.5 veces la capacidad de **purmamarca**.

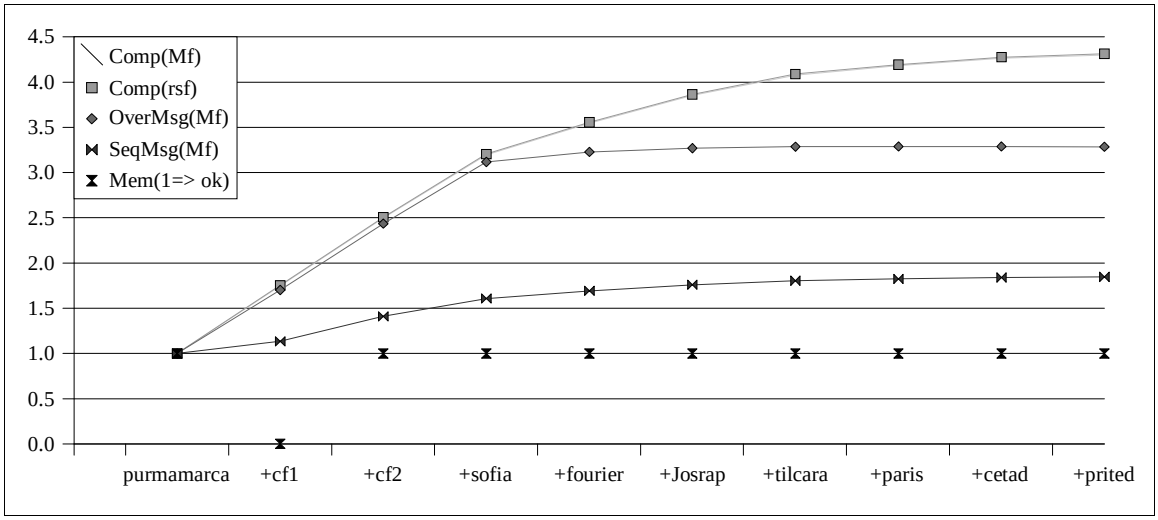


Figura 4.11: Análisis de Speedup de la Red del CeTAD para n = 2000.

También en la Figura 4.11 se muestra una estimación "empírica" de los requerimientos de memoria que se señala en el gráfico como Mem(1 => ok). Cuando se muestra con valor igual a 0 implica que es muy probable que en una o más computadoras se necesite recurrir al espacio de swap durante la ejecución. Cuando se muestra con valor igual a 1 indica que es muy probable que no sea necesario recurrir al espacio de swap en *ninguna* de las computadoras durante la ejecución del algoritmo. Nótese que en el gráfico se muestra con valor igual a 0 solamente cuando se utilizan dos computadoras en paralelo: **purmamarca** y **cf1**. Esto es así porque:

- Cuando se utiliza solamente **purmamarca**, que tiene 64 MB de memoria principal (Tabla 4.1), ya se conoce (de la experimentación misma) que no es necesaria la utilización del espacio de swap.
- Al incorporar **cf1**, que tiene 32 MB de memoria, es muy probable que **purmamarca** no necesite recurrir al espacio de swap pero **cf1** sí.
- Al incorporar **cf2**, ya son tres las máquinas entre las que se distribuyen los datos y a partir de aquí es muy probable que no haya inconvenientes por la memoria.

Aunque esta estimación de memoria no sea muy precisa (y de hecho es muy difícil hacer una que lo sea), siempre es útil tener al menos una idea de referencia dado que, como se ha aclarado antes, la capacidad de cálculo puede ser muy afectada.

La Figura 4.12 muestra los máximos valores de speedup a considerar en la red local del CeTAD para el máximo tamaño de problema que puede resolver la computadora de mayor capacidad de cálculo (recurriendo a la memoria swap). La computadora de referencia sigue siendo **purmamarca** y el tamaño de matrices es n = 3200. Como para la estimación anterior, se asume que se pueden transferir datos entre procesos de usuario a razón de 2^{20} bytes (1 MB) por segundo.

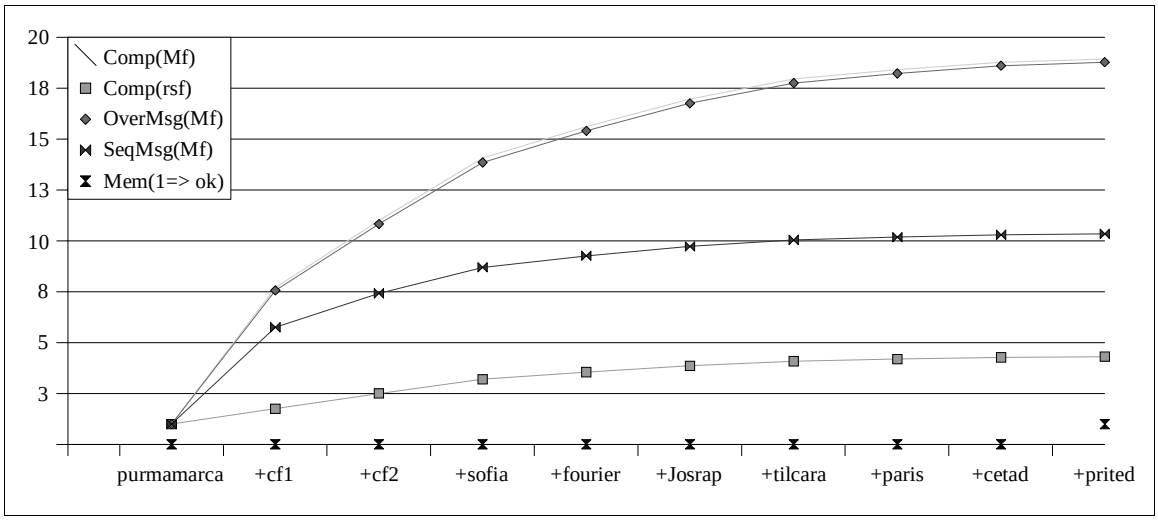


Figura 4.12:Análisis de Speedup de la Red del CeTAD para n = 3200.

Dado que el mayor tamaño de problema (con matrices de 3200×3200 elementos), que puede resolver **purmamarca** implica la utilización del espacio de swap, la capacidad de cálculo de referencia es aproximadamente 74 Mflop/s (Figura 4.5). Por lo tanto, ya no es posible que coincidan Comp(Mf) y Comp(rsf), más específicamente, el valor de speedup óptimo calculado con la capacidad máxima de cálculo (en Mflop/s) necesariamente será

mayor al valor de speedup óptimo teniendo en cuenta las velocidades relativas entre las computadoras. Esto implica que el máximo valor de speedup utilizando todas las máquinas del CeTAD (10 computadoras), da como resultado una computadora paralela que tiene casi 19 veces la capacidad de cálculo de **purmamarca** para matrices de 3200×3200 elementos, es decir cuando **purmamarca** tiene que recurrir al espacio de swap durante la ejecución.

Además, de la Figura 4.12 se desprende que:

- Los valores de Comp(rsf) no cambian con respecto a los de la Figura 4.11, dado que las velocidades relativas se asumen iguales.
- Asumiendo que cada computadora puede resolver efectivamente sus cálculos y comunicaciones de manera simultánea, los valores de OverMsg(Mf) son casi idénticos a los de Comp(Mf).
- El peso del tiempo de las comunicaciones sigue siendo relativamente alto respecto del tiempo de cómputo, y esto se identifica claramente por las diferencias entre los valores de Comp(Mf) y SeqMsg(Mf). Más específicamente, cuando se contabiliza el tiempo de comunicación además del tiempo de cómputo (tal como debe hacerse para el algoritmo con los períodos de transmisión de mensajes y cómputo resueltos de manera secuencial), los valores óptimos de speedup se reducen notablemente con respecto a los que se obtienen con la suma de las capacidades de cálculo.
- Dado que para n = 3200 **purmamarca** se "transforma" en una computadora con mucho menor capacidad de cálculo que para n = 2000, aún con cálculos y comunicaciones secuenciales se espera que la ganancia sea sustancial, y bastante mayor que la que es esperable de acuerdo con las velocidades relativas. Los valores de Comp(rsf) bastante menores que todos los demás así lo muestran.
- La estimación de requerimientos de memoria en cada computadora, Mem en el gráfico, muestra que recién con la utilización de todas las máquinas es poco probable que se requiera la utilización del espacio de memoria swap en todas las computadoras. Esto es debido básicamente a que dos de las computadoras con mayor capacidad de cálculo relativa, **cf1** y **cf2**, tienen poca capacidad de memoria también en términos relativos a la de mayor capacidad de cálculo (**purmamarca**).

4.5.2 Red Local del LQT

La Figura 4.13 muestra los máximos valores de speedup a considerar en la red local del LQT cuando todos los datos del problema se pueden contener en memoria principal, de forma tal que no es necesaria la utilización de memoria swap durante los cálculos. En este caso, la computadora de referencia (la de mayor capacidad de cálculo) es **lqt_07** y el tamaño de matrices es n = 5000. También en este caso, dado que se utiliza una red Ethernet de 10 Mb/s, se asume que por la degradación producida por todas las capas del sistema operativo se pueden transferir datos entre procesos de usuario a razón de 2^{20} bytes (1 MB) por segundo.

La computadora paralela *obtenida* utilizando la totalidad de la máquinas proporciona en el mejor de los casos poco más de 4.5 veces la capacidad de **lqt_07**, aproximadamente 2.9 Gflop/s. Una vez más, el cálculo analítico del máximo speedup posible de obtener con el algoritmo de cómputo y comunicaciones secuenciales muestra el peso relativo del tiempo

de comunicaciones con respecto al tiempo de cómputo, y se mantiene en valores aproximadamente iguales a la mitad de los valores de las demás estimaciones.

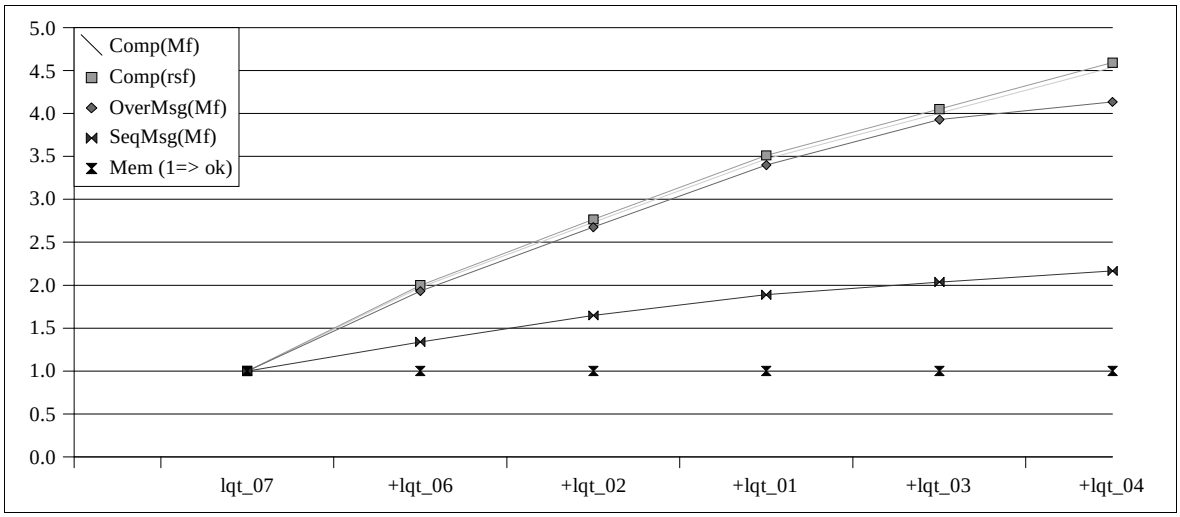


Figura 4.13: Análisis de Speedup de la Red del LQT para n = 5000.

La Figura 4.14 muestra los máximos valores de speedup a considerar en la red local del LQT para el máximo tamaño de problema que puede resolver la computadora de mayor capacidad de cálculo (recurriendo a la memoria swap). La computadora de referencia sigue siendo **lqt_07** y el tamaño de matrices es n = 9000. Como para la estimación anterior, se asume que se pueden transferir datos entre procesos de usuario a razón de 2^{20} bytes (1 MB) por segundo.

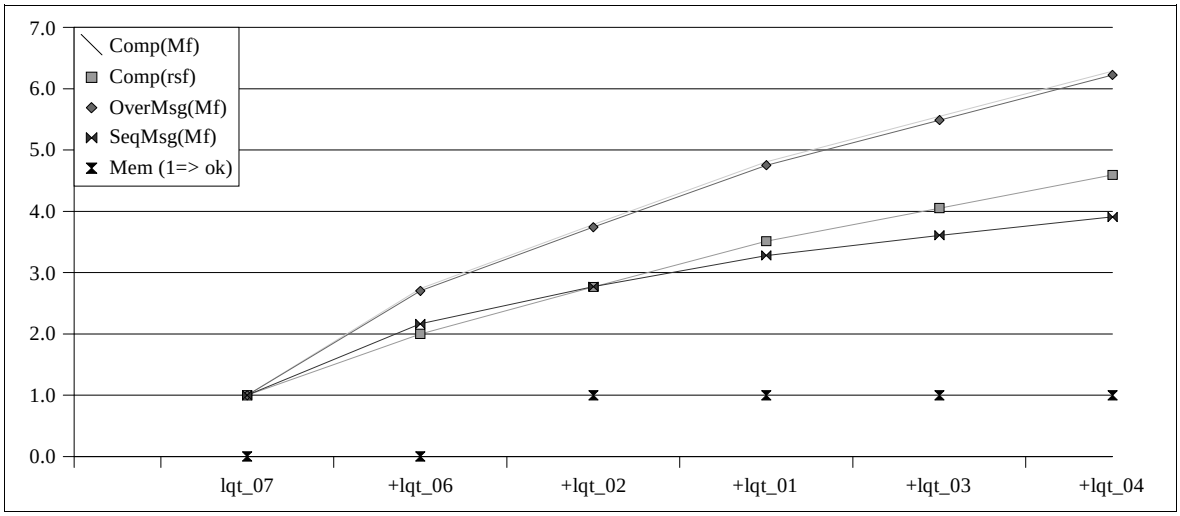


Figura 4.14: Análisis de Speedup de la Red del LQT para n = 9000.

A partir de la Figura 4.14 también se deduce que:

- Al utilizar todas las computadoras para resolver el problema de la multiplicación de matrices de 9000×9000 elementos se *podría* reducir el tiempo de ejecución más de seis veces respecto del tiempo de ejecución de **lqt_07**, aún cuando según las velocidades

relativas la reducción no llegaría a cinco veces.
- Si es posible solapar totalmente los cálculos locales con las comunicaciones, entonces el máximo speedup posible de obtener es similar al máximo absoluto. Expresado de otra manera, el tiempo de cómputo es igual o mayor que el de las comunicaciones.
- Cuando los mensajes y el cómputo local se llevan a cabo de manera secuencial el tiempo de ejecución será mayor que el esperado teniendo en cuenta las velocidades relativas de las computadoras a partir de la inclusión de **lqt_01**, es decir que Comp(rsf) > SeqMsg(Mf) a partir de la inclusión de **lqt_01**.
- Según las estimaciones de memoria no sería necesario recurrir a la utilización del espacio de memoria swap a partir de la inclusión de **lqt_02**.

4.5.3 Red Local del LIDI

La Figura 4.15 muestra los máximos valores de speedup a considerar en la red local del LQT cuando todos los datos del problema se pueden contener en memoria principal, de forma tal que no es necesaria la utilización de memoria swap durante los cálculos. En este caso, la computadora de referencia (la de mayor capacidad de cálculo) es **lidipar14** pero se debe recordar que todas las computadoras son iguales, y el tamaño de matrices es $n = 2000$. A diferencia de las redes locales anteriores se utiliza una red Ethernet de 100 Mb/s, y se asume que por la degradación producida por todas las capas del sistema operativo se pueden transferir datos entre procesos de usuario a razón de 10×2^{20} bytes (10 MB) por segundo. Como en los casos anteriores, podría considerarse una suposición optimista, pero aceptable dado que se estiman valores máximos.

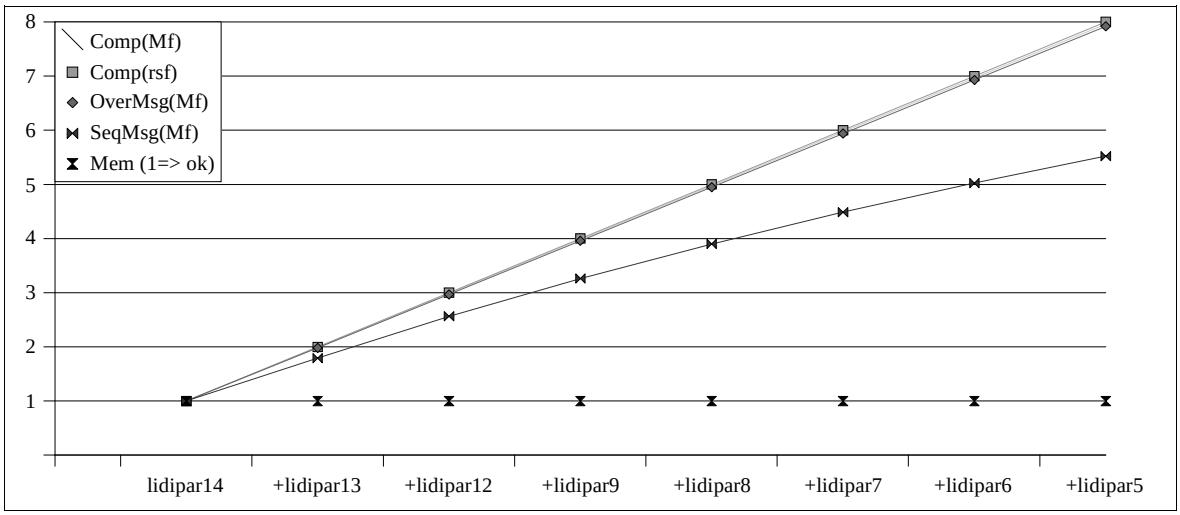


Figura 4.15: Análisis de Speedup de la Red del LIDI para $n = 2000$.

Los valores de speedup que se muestran en la Figura 4.15 son más o menos comunes con los de las computadoras paralelas *clásicas* (homogéneas), dado que:
- Los valores de Comp(rsf) se *corresponden* con la recta $y = x$.
- Al menos hasta la inclusión de **lidipar5** todos los valores máximos estimados para el speedup siguen un patrón de crecimiento lineal respecto de la cantidad de la cantidad de

procesadores (computadoras) utilizados, incluso considerando el algoritmo de cómputo y comunicaciones secuenciales.

También de la Figura 4.15 se desprende que:

- Comparando esta red con las anteriores, al tener una red de interconexión diez veces mejor en cuanto a ancho de banda las comunicaciones no tienen un peso relativo tan grande. De hecho, el algoritmo que lleva a cabo cómputo con comunicaciones de manera solapada tiene speedup óptimo igual al calculado sin tener en cuenta las comunicaciones, OverMsg(Mf) ≅ Comp(Mf).
- Con el algoritmo que lleva a cabo cómputo y comunicaciones secuenciales, al utilizar las ocho computadoras disponibles se obtiene en el mejor de los casos una computadora (paralela) que tiene poco más de 5.5 veces mayor capacidad de cálculo que **lidipar14** (o cualquiera de las demás, dado que son todas iguales).
- Como todas las computadoras son iguales se puede identificar más claramente que en los casos anteriores (CeTAD y LQT) la relación entre el tiempo de comunicaciones y el de cómputo con los valores máximos de speedup del algoritmo con cómputo y comunicaciones secuenciales, SeqMsg(Mf). Dado que a medida que hay mayor cantidad de máquinas se reparte el mismo trabajo entre todas ellas, el tiempo de cómputo total disminuye (hay mayor cantidad de computadoras procesando simultáneamente) pero el tiempo total de comunicaciones se mantiene igual. Por lo tanto, a medida que se agregan máquinas el tiempo de comunicaciones afecta de manera más significativa al tiempo total de ejecución (cómputo más comunicaciones).

Para ejemplificar este último punto se pueden tomar los valores específicos calculados para cuatro y ocho máquinas. Independientemente de la cantidad de computadoras que se utilizan, cuando el tamaño de las matrices es el mismo (en este caso n = 2000) la cantidad de datos que se comunican es la misma ya que siempre deben transmitirse los datos de la matriz B entre las computadoras. El tiempo estimado de transmisión de la matriz B (suma de tiempos de los mensajes broadcast del algoritmo) en una red Ethernet de 100 Mb/s es aproximadamente 1.5 segundos. Cuando se utilizan **lidipar14**, **lidipar13**, **lidipar12**, y **lidipar9**, el tiempo estimado de cómputo es aproximadamente 13.8 segundos. Cuando se agregan a las anteriores las computadoras **lidipar8**, **lidipar7**, **lidipar6**, y **lidipar5**, el tiempo estimado de cómputo es aproximadamente 3.4 segundos. Por lo tanto, cuando los mensajes y las comunicaciones se ejecutan de manera secuencial:

- El tiempo total de ejecución cuando se utilizan cuatro computadoras es (sumando tiempo de cómputo y tiempo de comunicaciones), 1.5 + 13.8 = 15.3 segundos. Esto implica que aproximadamente el 10% del tiempo total de ejecución es utilizado para las comunicaciones.
- El tiempo total de ejecución cuando se utilizan ocho computadoras es (sumando tiempo de cómputo y tiempo de comunicaciones) 1.5 + 3.4 = 4.9 segundos. Esto implica que aproximadamente el 30% del tiempo total de ejecución es utilizado para las comunicaciones.

La Figura 4.16 muestra los máximos valores de speedup a considerar en la red local del LIDI para el máximo tamaño de problema que puede resolver la computadora de mayor capacidad de cálculo (recurriendo a la memoria swap). La computadora de referencia sigue siendo **lidipar14** y las matrices son de 3200×3200 elementos. Como para la estimación anterior, se asume que se pueden transferir datos entre procesos de usuario a razón de

10×2^{20} bytes (10 MB) por segundo. Además de mostrarse que los valores máximos de speedup calculados utilizando las velocidades relativas, Comp(rsf), no cambian respecto de los que se muestran en la Figura 4.15, en la Figura 4.16 también se muestra que:

- El problema de multiplicar matrices de 3200×3200 elementos se *podría* resolver casi 35 veces más rápido en las ocho computadoras que en una de ellas. Esto se confirma no solamente por la potencia de cálculo de las ocho máquinas procesando a su máxima capacidad (sin tener en cuenta las comunicaciones), Comp(Mf), sino también con el algoritmo que ejecuta cálculo solapado con las comunicaciones, OverMsg(Mf).
- Con el algoritmo que resuelve de manera secuencial las comunicaciones respecto de los cálculos, el problema de multiplicar matrices de 3200×3200 elementos se *podría* resolver más de 25 veces más rápido en las ocho computadoras que en una de ellas.
- La penalización en rendimiento por el uso del espacio de swap en **lidipar14** para matrices de 3200×3200 elementos como la tasa de transferencia de datos de la red de interconexión se combinan para tener estos valores de speedup "superlineales".
- Excepto para una y dos máquinas, las estimaciones de requerimientos de memoria no identifican posibles problemas en cuanto a necesidad de utilización del espacio de swap.

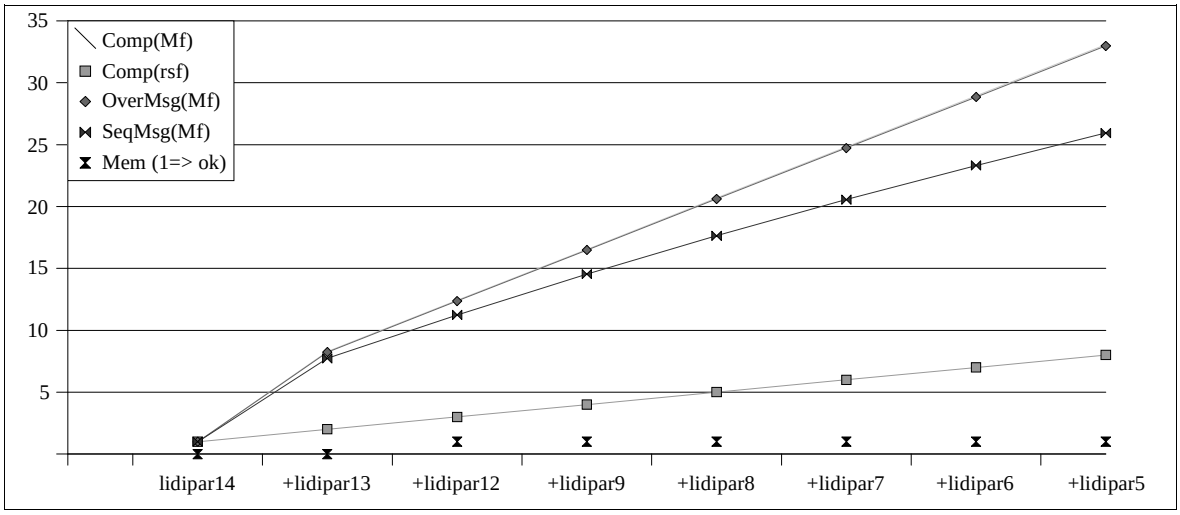


Figura 4.16: Análisis de Speedup de la Red del LIDI para n = 3200.

4.6 Rendimiento Real de las Redes Locales Utilizando PVM

Los algoritmos propuestos en el capítulo anterior se implementaron directamente utilizando la biblioteca PVM (Parallel Virtual Machine) para las rutinas de comunicaciones entre procesos. En cada una de las computadoras se utiliza el mejor código secuencial para los periodos de cálculo local. En cada una de las redes locales (CeTAD, LQT y LIDI) se llevaron a cabo los mismos experimentos, es decir:

- Multiplicación de matrices con el algoritmo de cómputo y mensajes secuenciales (SeqMsg).

- Multiplicación de matrices con el algoritmo de cómputo y mensajes solapados (OverMsg).
- Tamaños de matrices tales que en la máquina con mayor capacidad de cálculo
  - No se utiliza espacio de swap.
  - Más grande posible.

  En cada red local esto se corresponde con matrices de orden
  - $n = 2000$ y $n = 3200$ respectivamente en las redes locales del CeTAD y del LIDI.
  - $n = 5000$ y $n = 9000$ respectivamente en la red local del LQT.

4.6.1 Red Local del CeTAD

La Figura 4.17 muestra los valores de speedup obtenidos en la red local del CeTAD por los algoritmos de cómputo y comunicación secuenciales, y cómputo solapado con comunicaciones, implementados utilizando la biblioteca de pasaje de mensajes PVM, SeqMsg(PVM) y OverMsg(PVM) respectivamente, para $n = 2000$, junto con los que se mostraron anteriormente en la Figura 4.11.

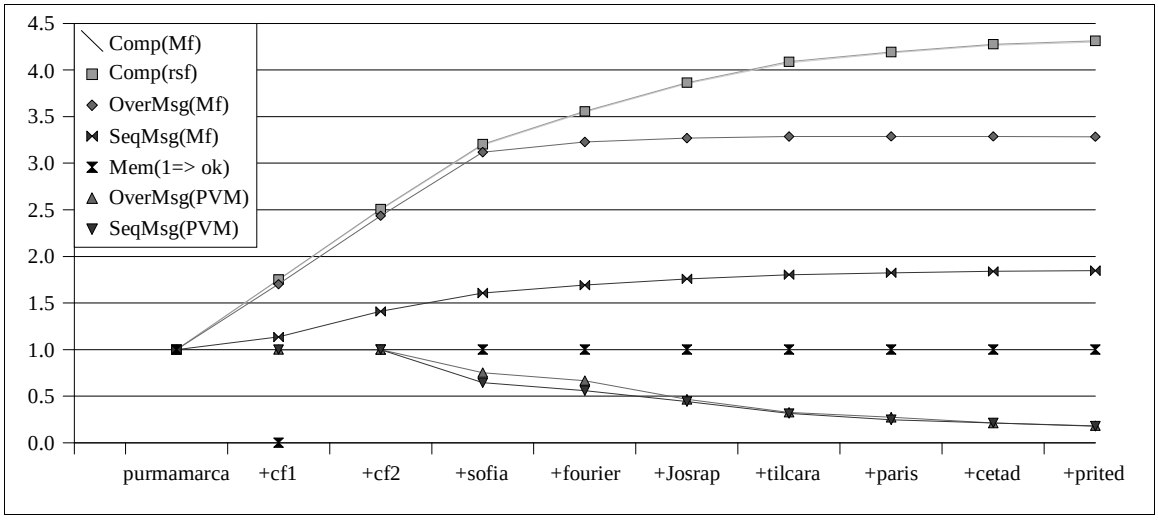


Figura 4.17: Speedup de los Algoritmos con PVM en la Red del CeTAD para $n = 2000$.

Claramente, los resultados están lejos de ser satisfactorios. De hecho, las dos conclusiones más desalentadoras son

- Ninguno de los tiempos de ejecución, es decir independientemente de la cantidad de computadoras utilizadas, fue mejor que el tiempo de la ejecución secuencial, con el problema resuelto en **purmamarca**.
- A medida que se utilizan más computadoras, el tiempo de ejecución aumenta en vez de disminuir.

Más aún, los valores de speedup de los algoritmos que se muestran iguales a uno, es decir para los casos en que se utilizan **purmamarca** y **cf1** y **purmamarca**, **cf1** y **cf2** respectivamente ni siquiera son *reales*. En los dos casos, la ejecución de los procesos del programa paralelo en **cf1** y/o **cf2** se cancela por falta de memoria disponible. Por lo tanto,

se muestran iguales a uno porque de hecho se considera que la única posibilidad de solución para la multiplicación de matrices en ese contexto es la ejecución secuencial (utilizando **purmamarca** solamente). A pesar de que se tiene una aproximación respecto de los requerimientos de memoria (Mem, en los gráficos), evidentemente al menos con PVM esa aproximación no es acertada.

Aún descartando el problema de memoria, el problema de rendimiento es evidente. En principio, las alternativas en cuanto a las causas del bajo rendimiento obtenido pueden ser:

- Bajo rendimiento de cómputo, al resolver cada uno de los cálculo intermedios. Este problema está básicamente relacionado con el rendimiento en cuanto a cómputo de cada una de las computadoras.
- Bajo rendimiento de las comunicaciones, al enviar y recibir los mensajes broadcast. Este problema está básicamente relacionado con PVM y con la red de interconexión. En este sentido, se tienen dos posibilidades:
  - PVM no implementa de manera óptima los mensajes broadcast o
  - la comunicación entre procesos de usuario de distintas computadoras sean muy penalizados en rendimiento con respecto a la capacidad de la red de interconexión.

La Figura 4.18 muestra los valores de speedup obtenidos en la red local del CeTAD por los algoritmos de cómputo y comunicación secuenciales, y cómputo solapado con comunicaciones, implementados utilizando la biblioteca de pasaje de mensajes PVM, SeqMsg(PVM) y OverMsg(PVM) respectivamente, para n = 3200, junto con los que se mostraron en la Figura 4.12.

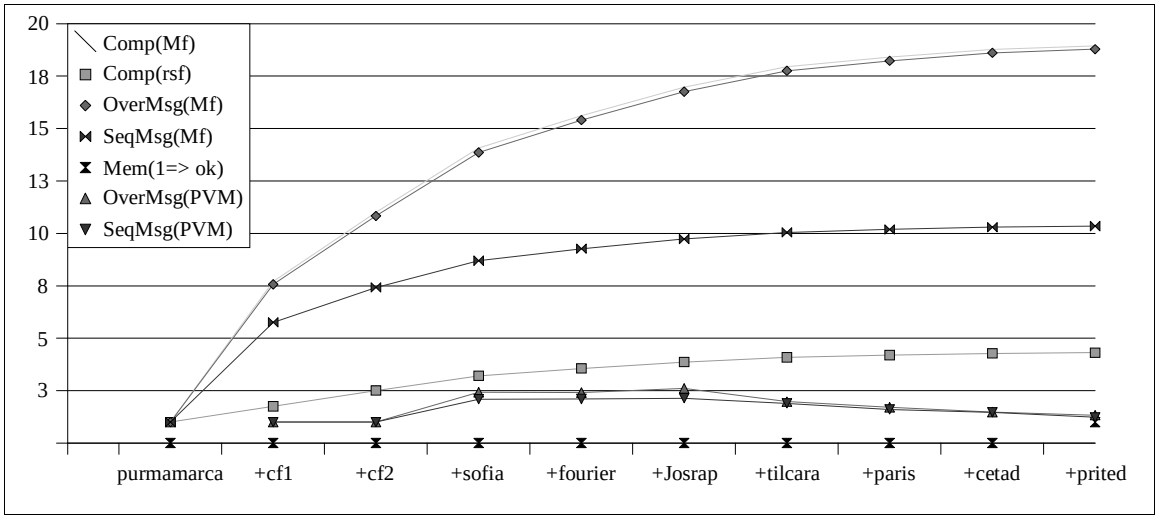


Figura 4.18: Speedup de los Algoritmos con PVM en la Red del CeTAD para n = 3200.

En este caso, es decir tomando como referencia la capacidad de cálculo de **purmamarca** para multiplicar matrices cuadradas de orden n = 3200:

- Una vez más los requerimientos de memoria en **cf1** y **cf2** hacen imposible la ejecución del programa paralelo y los procesos son cancelados por el sistema operativo.
- En el mejor de los casos, se obtienen valores de speedup cercanos a tres cuando todas las estimaciones son superiores para la misma cantidad de máquinas, incluso la que

tiene en cuenta solamente las velocidades relativas: Comp(rsf).
- A partir de la inclusión de **tilcara** en la máquina paralela el tiempo de ejecución paralela empeora, llegando a tener con las diez máquinas valores de speedup de aproximadamente 1.33 con el algoritmo que lleva a cabo cómputo y comunicación solapados y 1.23 con el algoritmo que lleva a cabo cómputo y comunicación secuenciales, OverMsg(PVM) y SeqMsg(PVM) respectivamente.
- Todas las estimaciones de speedup de los algoritmos son muy lejanas de los valores obtenidos.

4.6.2 Red Local del LQT

La Figura 4.19 muestra los valores de speedup obtenidos en la red local del LQT por los algoritmos de cómputo y comunicación secuenciales, y cómputo solapado con comunicaciones, implementados utilizando la biblioteca de pasaje de mensajes PVM, SeqMsg(PVM) y OverMsg(PVM) respectivamente, para n = 5000, junto con los que se mostraron anteriormente en la Figura 4.13.

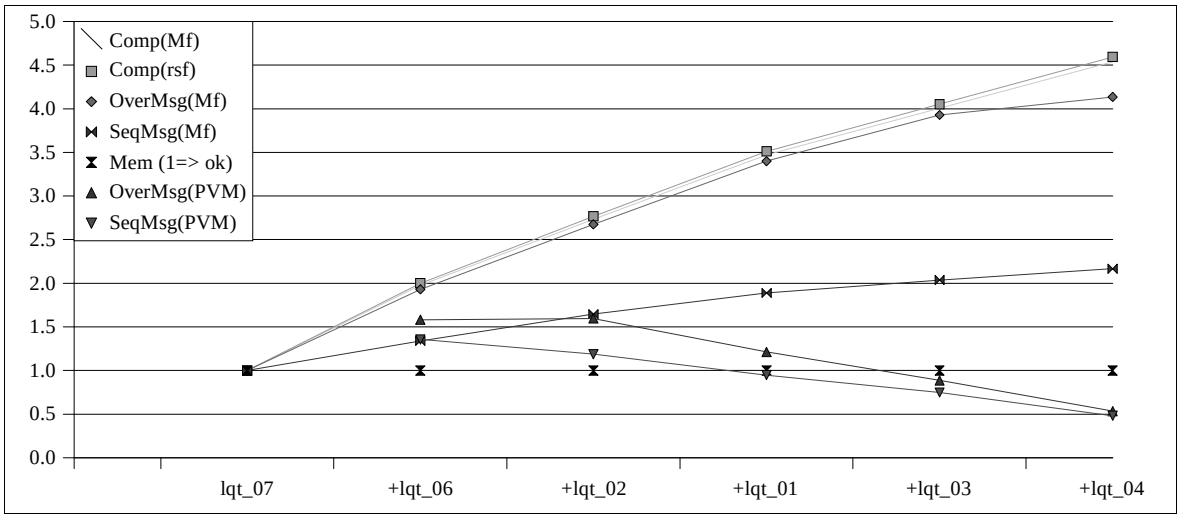


Figura 4.19: Speedup de los Algoritmos con PVM en la Red del LQT para n = 5000.

Comparando estos resultados con los que se corresponden del CeTAD, que se muestran en la Figura 4.17, la similitud es notable. Básicamente las características de los valores de speedup obtenidos es la misma
- Casi no se gana rendimiento por utilizar máquinas procesando en paralelo,
- En la mayoría de los casos, agregar máquinas para procesar en paralelo implica pérdida de rendimiento,

a pesar de que las computadoras y el tamaño del problema son muy diferentes entre sí. Por lo tanto, estos resultados no hacen más que confirmar que hay uno o varios problemas y que el o los problemas no son específicos de la red local del CeTAD ni de la red local del LQT.

La Figura 4.20 muestra los valores de speedup obtenidos en la red local del LQT por los algoritmos de cómputo y comunicación secuenciales, y cómputo solapado con

comunicaciones, implementados utilizando la biblioteca de pasaje de mensajes PVM, SeqMsg(PVM) y OverMsg(PVM) respectivamente, para n = 9000, junto con los que se mostraron en la Figura 4.14. Las similitudes en cuanto a los valores de speedup con respecto al CeTAD en el contexto similar (Figura 4.18), son una vez más bastante evidentes, a pesar de las diferencias entre las máquinas y el tamaño del problema:

- Los requerimientos de memoria que impone el cómputo paralelo con las rutinas de comunicaciones de PVM hacen que cuando se utilizan dos computadoras, **lqt_07** y **lqt_06**, el sistema operativo cancele uno o varios de los procesos involucrados por falta de memoria. Es por esto que en el gráfico se muestra speedup igual a uno para dos computadoras para los dos algoritmos.
- Hasta una determinada cantidad de computadoras, el speedup aumenta. En este caso hasta la inclusión de **lqt_02** para el algoritmo de cómputo y comunicación secuenciales, SeqMsg(PVM) en el gráfico, y hasta la inclusión de **lqt_01** para el algoritmo de comunicaciones solapadas con cómputo OverMsg(PVM) en el gráfico.
- La utilización de todas las máquinas no aporta casi ningún beneficio en cuanto a rendimiento con respecto a la alternativa de solución secuencial. Los valores reales de speedup cuando se utilizan las seis computadoras son 1.3 para OverMsg(PVM) y 1.08 para SeqMsg(PVM).

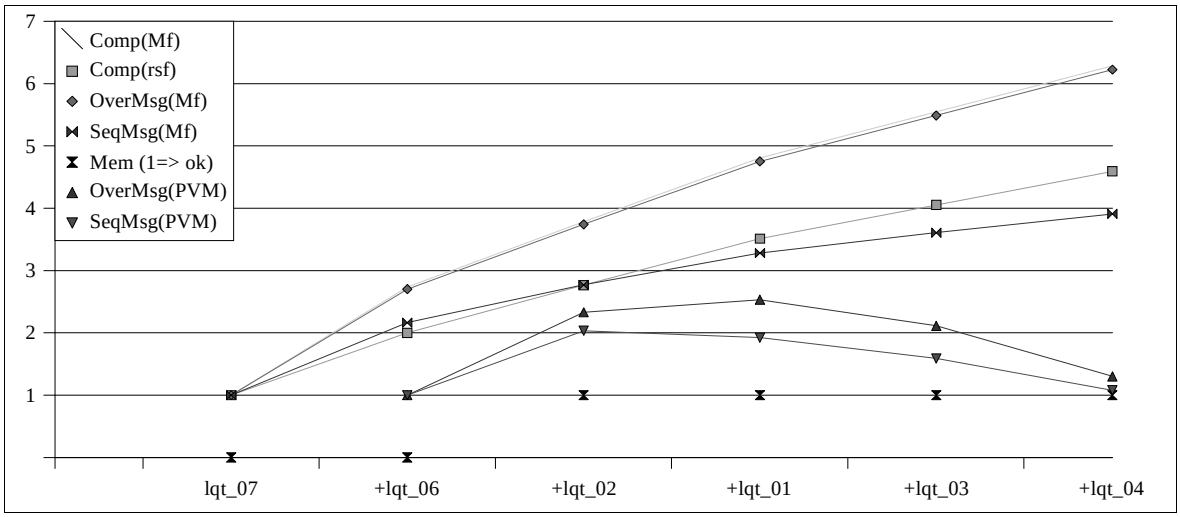


Figura 4.20: Speedup de los Algoritmos con PVM en la Red del LQT para n = 9000.

Comparando los resultados que se muestran en la Figura 4.20 con los de la Figura 4.18, también se pueden encontrar algunas diferencias:

- Algunos valores de speedup obtenidos son bastante más cercanos a los estimados, al menos para tres y cuatro máquinas, es decir cuando se utilizan **lqt_07**, **lqt_06**, **lqt_02**, y **lqt_07**, **lqt_06**, **lqt_02** y **lqt_01** respectivamente.
- El algoritmo de que lleva a cabo cómputo solapado con comunicaciones tiene mejor rendimiento que el que no intenta aprovechar ningún solapamiento. La diferencia llega a ser de poco más del 30% cuando se utilizan **lqt_07**, **lqt_06**, **lqt_02**, **lqt_01** y **lqt_03**.

4.6.3 Red Local del LIDI

La Figura 4.21 muestra los valores de speedup obtenidos en la red local del LIDI por los algoritmos de cómputo y comunicación secuenciales, y cómputo solapado con comunicaciones, implementados utilizando la biblioteca de pasaje de mensajes PVM, SeqMsg(PVM) y OverMsg(PVM) respectivamente, para n = 2000, junto con los que se mostraron anteriormente en la Figura 4.15. Como en las redes del CeTAD y del LQT en el contexto *correspondiente* (Figura 4.17 y Figura 4.19):

- Las estimaciones de speedup son bastante lejanas respecto de los valores obtenidos. A partir de la utilización de cuatro computadoras inclusive la diferencia es cada vez mayor.
- En términos generales, agregar computadoras implica pérdida de rendimiento, las excepciones se dan para dos y tres computadoras, dado que el rendimiento en esos casos aumenta cuando se usan más máquinas.

A diferencia de las redes del CeTAD y del LQT:

- No se llega a tener peor tiempo de cómputo que para la solución secuencial, aunque la tendencia a medida que aumenta la cantidad de computadoras utilizadas indica que se puede llegar a esta situación (con valores de speedup menores que uno).
- La pérdida de rendimiento a medida que aumenta la cantidad de computadoras utilizadas es bastante más gradual.

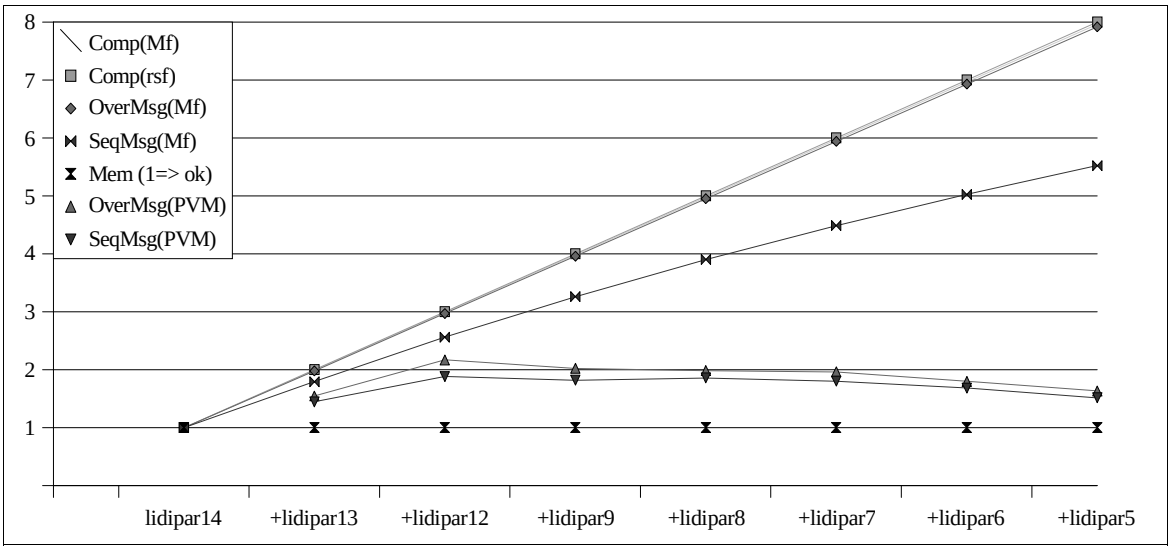


Figura 4.21: Speedup de los Algoritmos con PVM en la Red del LIDI para n = 2000.

La Figura 4.22 muestra los valores de speedup obtenidos en la red local del LIDI por los algoritmos de cómputo y comunicación secuenciales, y cómputo solapado con comunicaciones, implementados utilizando la biblioteca de pasaje de mensajes PVM, SeqMsg(PVM) y OverMsg(PVM) respectivamente, para n = 3200, junto con los que se mostraron anteriormente en la Figura 4.16.

A pesar de que los valores obtenidos de la experimentación no son cercanos a los estimados para los algoritmos, es la primera vez que se tienen valores de speedup mayores

al menos que los valores de speedup calculados según las velocidades relativas. De acuerdo con la Figura 4.22 también se tiene que:

- En el mejor de los casos, que se da al utilizar **lidipar14**, **lidipar13**, **lidipar12**, **lidipar9** y **lidipar8**, se resuelve la multiplicación de matrices un poco más de diez veces más rápidamente que en **lidipar14** (se debe recordar que **lidipar14** utiliza memoria swap para resolver este problema).
- A partir de la inclusión de **lidipar6** el rendimiento disminuye con los dos algoritmos, lo cual implica que se reducen tanto OverMsg(PVM) como SeqMsg(PVM).
- Se tienen valores de speedup "superescalares" (mayores que la cantidad de procesadores homogéneos), dado que el tiempo de referencia de la resolución del problema de manera secuencial en **lidipar14** está penalizado en cuanto a rendimiento por la utilización del espacio de swap durante los cálculos, hecho que se nota claramente en la Figura 4.8.

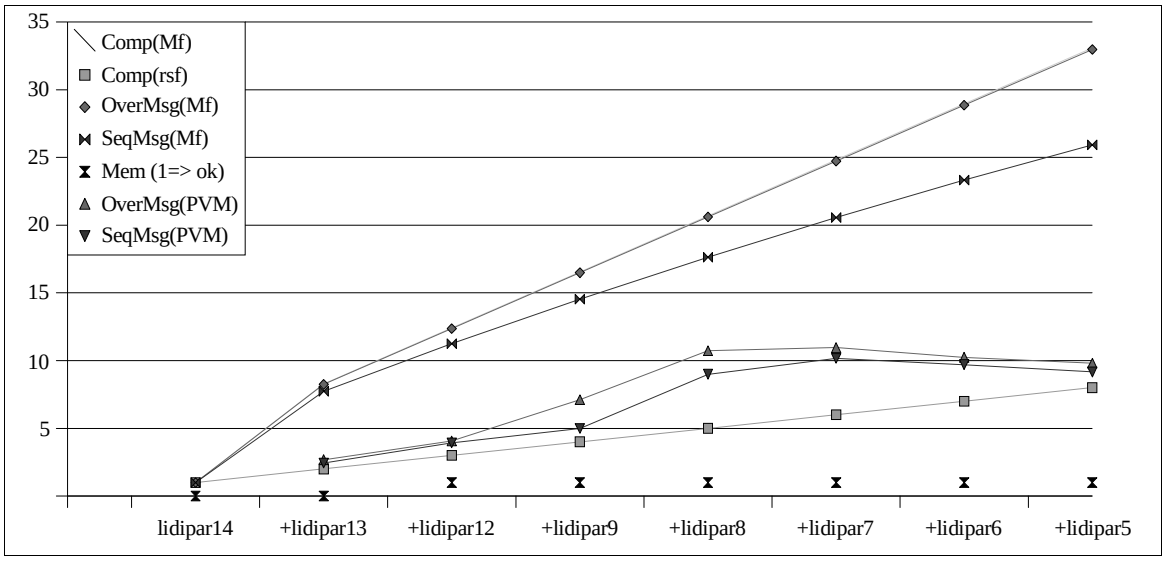


Figura 4.22: Speedup de los Algoritmos con PVM en la Red del LIDI para $n = 3200$.

Los valores obtenidos de la experimentación en la red local del LIDI parecen ser bastante mejores que los que se obtuvieron de la experimentación en las redes locales del CeTAD y del LQT. Más allá de las diferencias entre las máquinas en particular, desde el punto de vista de las computadoras paralelas que se construyen con cada red local, las principales diferencias son:

- La red de interconexión del LIDI es diez veces más rápida que la del CeTAD y del LQT.
- La computadora paralela construida con la red local del LIDI es homogénea, mientras que tanto la del CeTAD como la del LQT son (*muy*) heterogéneas.

Intuitivamente, la razón más importante por la cual se tienen mejores resultados en la red local del LIDI es la capacidad muy superior de la red de interconexión, pero evidentemente se necesitan más datos para dar una explicación mejor fundamentada.

4.7 Perfiles de Ejecución en las Redes Locales Utilizando PVM

Para tener mayor precisión respecto de los tiempos de ejecución paralela y de las razones por las cuales las estimaciones de speedup son tan lejanas respecto del speedup obtenido en la experimentación, se llevaron a cabo los mismos experimentos pero con un mínimo de instrumentación de forma tal que:

- Se identifica claramente qué parte del tiempo total de ejecución se utiliza para cómputo y qué parte del tiempo total se utiliza para comunicaciones. Esta información es muy útil para identificar si el problema son las comunicaciones o no.
- Se identifica gráficamente en qué estado de ejecución está cada proceso (computadora de la red local) durante todo *instante* de tiempo de ejecución. Este tipo de información tiene un poco más de detalle que la anterior, y es útil para identificar si hay alguna computadora en particular que produce un retraso general. Por ejemplo: si por alguna razón local una computadora no envía un mensaje broadcast en el tiempo esperado, todas las demás se verán afectadas porque no lo recibirán.

Dado que los experimentos son muy numerosos como para mostrar el perfil de tiempos de ejecución de cada uno de ellos y además como en su mayoría son similares, se muestran y explican los más significativos en cada una de las redes locales.

4.7.1 Red Local del CeTAD

En la Figura 4.23 se muestra el perfil de ejecución al utilizar las cinco mejores máquinas de la red local del CeTAD para una multiplicación de matrices en paralelo de 2000×2000 elementos, con el algoritmo de cómputo y comunicaciones secuenciales, donde:

- El tiempo se muestra en segundos.
- En todo instante de tiempo, cada computadora puede estar:
  - Ejecutando un cálculo parcial, mostrado como “Cómputo” en el gráfico.
  - Enviando o recibiendo un mensaje broadcast, mostrado como “Bcasts” en el gráfico, y durante el cual los cálculos no se pueden realizar y por lo tanto se “Espera” la terminación del broadcast para seguir con el cómputo.
- Bcasts identifica cada mensaje broadcast, donde un proceso que se ejecuta en una computadora envía datos a todos los demás (en este caso a otros cuatro procesos) que se ejecutan en las demás computadoras (cuatro computadoras).
- Dado que en PVM el envío de *todos* los mensajes es solapado con respecto al cómputo cuando se envía un broadcast simultáneamente se lleva a cabo un período de cálculo parcial de la matriz resultado.

Más allá de algunos detalles de tiempo de ejecución particulares, se nota claramente que durante la mayor parte del tiempo de ejecución de cada computadora se espera por la conclusión de un mensaje broadcast (más precisamente, por la recepción de un mensaje broadcast desde otra computadora).

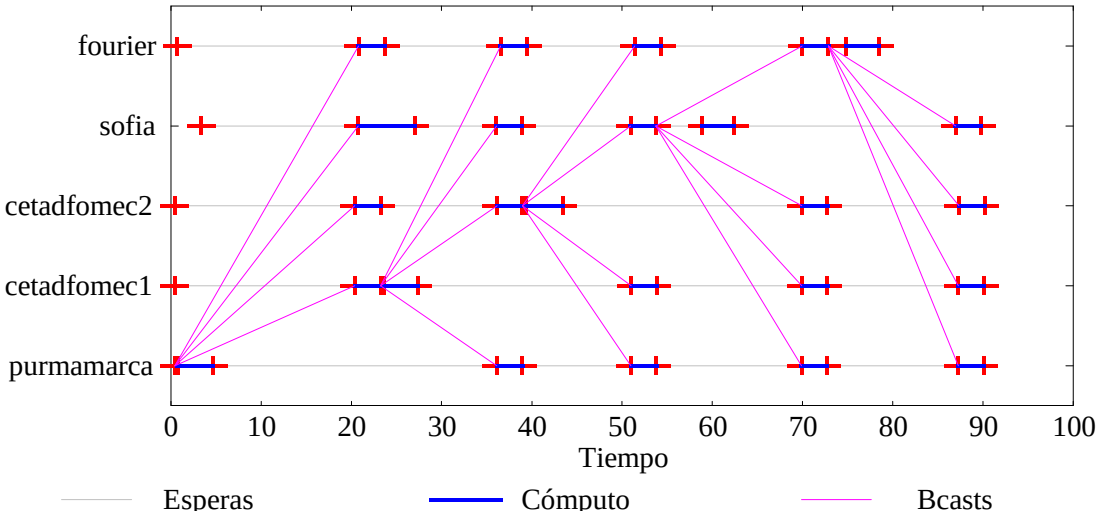


Figura 4.23: Perfil de SeqMsg(PVM) con Cinco Máquinas y *n* = 2000 en el CeTAD.

La Tabla 4.4 muestra la información resumida de la ejecución del programa paralelo que se corresponde con el perfil de ejecución de la Figura 4.23, donde

- ***Nombre*** identifica el nombre de la computadora utilizada.
- ***Filas*** identifica la cantidad de filas de la matriz resultado que fueron asignadas a cada computadora, que es proporcional a la velocidad relativa de cada computadora con respecto a la computadora paralela.
- ***Tot. Cómp.*** identifica la cantidad de tiempo local de ejecución durante el cual se ejecutaron operaciones con números en punto flotante.
- ***Por It.*** identifica la cantidad de tiempo local de ejecución de un paso de cómputo (operaciones con números en punto flotante).
- ***Tot. Msg.*** identifica la cantidad de tiempo local de ejecución utilizado para la espera de la conclusión de los mensajes broadcast (un envío y cuatro recepciones, dado que son cinco computadoras en total).

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	555	15.17	3.03	68.29
cf1	426	15.05	3.01	68.39
cf2	426	15.59	3.12	67.80
sofia	394	17.00	3.40	65.22
fourier	199	15.25	3.05	55.91

Tabla 4.4: Resumen de SeqMsg(PVM) con Cinco Máquinas y *n* = 2000 en el CeTAD.

La Figura 4.24 muestra el perfil de ejecución cuando se utilizan todas las máquinas (diez) disponibles del CeTAD. También a partir de lo que se muestra en el gráfico, queda claro que la mayoría del tiempo de ejecución se emplea para los mensajes.

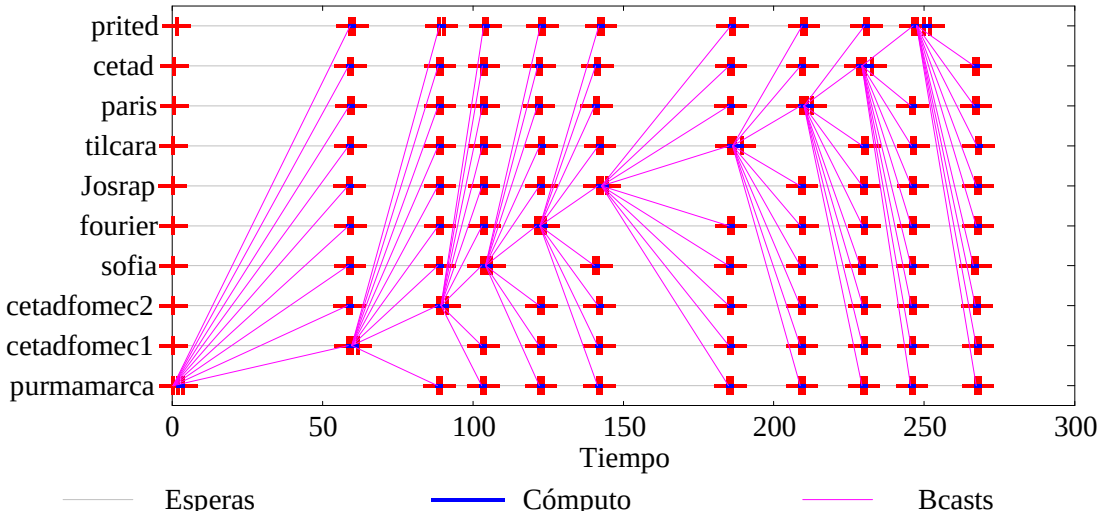


Figura 4.24: Perfil de SeqMsg(PVM) con Diez Máquinas y *n* = 2000 en el CeTAD.

En la Figura 4.24 también se nota bastante claramente que tanto el primer broadcast, enviado desde **purmamarca** como el sexto, enviado desde **Josrap**, utilizan mayor tiempo de transmisión que los demás. Pero aunque estos dos mensajes utilizaran el tiempo promedio de comunicaciones que utilizan los demás, el tiempo total de comunicaciones seguiría siendo dominado por las comunicaciones. Por lo tanto, el primer problema a resolver según estos dos perfiles de ejecución mostrados (Figura 4.23 y Figura 4.24), es el del tiempo de comunicaciones excesivo.

La Tabla 4.5 muestra la información resumida de la ejecución del programa paralelo que se corresponde con el perfil de ejecución de la Figura 4.24, donde se puede cuantificar bastante mejor el peso relativo de las comunicaciones (***Tot. Msg.***) con respecto al cómputo (***Tot. Cómp.***).

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	454	12.35	1.24	255.78
cf1	349	12.44	1.24	255.75
cf2	349	12.48	1.25	255.31
sofia	324	12.21	1.22	255.05
fourier	164	12.85	1.28	255.22
Josrap	142	12.24	1.22	256.03
tilcara	104	13.29	1.33	254.94
paris	48	12.08	1.21	255.01
cetad	38	12.84	1.28	254.16
prited	28	13.68	1.37	236.92

Tabla 4.5: Resumen de SeqMsg(PVM) con Diez Máquinas y *n* = 2000 en el CeTAD.

La situación no es muy diferente cuando se utiliza el algoritmo de comunicaciones solapadas con cómputo, tal como lo muestra la Figura 4.25 para las cinco mejores computadoras. En resumen, la Figura 4.25 muestra el perfil de ejecución al utilizar las cinco mejores máquinas de la red local del CeTAD para una multiplicación de matrices en paralelo de 2000×2000 elementos, con el algoritmo de cómputo y comunicaciones solapados. También en este caso, durante la mayor parte del tiempo de ejecución de cada computadora se espera por la conclusión de un mensaje broadcast (más precisamente, por la recepción de un mensaje broadcast desde otra computadora).

Evaluando la evolución de la ejecución en cada computadora es sencillo identificar que aunque se intentan solapar las comunicaciones con el cómputo local, de todas maneras transcurre más tiempo esperando por la recepción de los datos que se reciben del resto de las computadoras (identificados como “Esperas” en la Figura 4.25) que haciendo los cálculos que corresponden a las distintas iteraciones del algoritmo paralelo.

Comparando los tiempos de ejecución totales de los algoritmos OverMsg(PVM) de la Figura 4.25 y SeqMsg(PVM) de la Figura 4.23, el tiempo de ejecución total de OverMsg(PVM) es menor que el de SeqMsg(PVM) y esto se debe al solapamiento que enmascara en parte el peso de los tiempos de comunicaciones. En este sentido, aunque las computadoras de la red local del CeTAD son muy diferentes entre sí, de todas maneras son capaces de solapar cómputo local y comunicaciones al menos en parte.

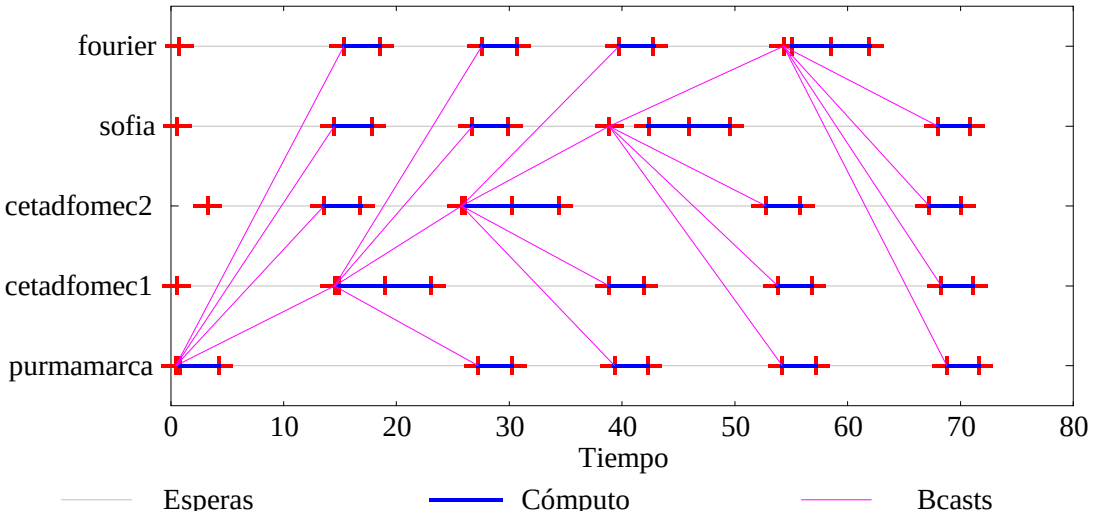


Figura 4.25: Perfil de OverMsg(PVM) con Cinco Máquinas y n = 2000 en el CeTAD.

El resumen de ejecución que se muestra en la Tabla 4.6 es similar al que se muestra en la Tabla 4.4 aunque los tiempos de comunicaciones son menores dado que parte del tiempo de cada mensaje broadcast se *superpone* con cómputo local.

Cuando se toma como referencia el máximo tamaño que se puede resolver en la computadora con mayor capacidad de cálculo del CeTAD, es decir n = 3200, las características en cuanto a perfiles de ejecución y rendimiento siguen siendo las mismas. La Figura 4.26 muestra el perfil de ejecución al utilizar las siete mejores máquinas de la red local del CeTAD para una multiplicación de matrices en paralelo de 3200×3200 elementos,

con el algoritmo de cómputo y comunicaciones solapados.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	555	15.20	3.04	55.97
cf1	426	17.07	3.41	53.54
cf2	426	17.31	3.46	49.44
sofia	394	16.50	3.30	53.78
fourier	199	16.18	3.24	44.97

Tabla 4.6: Resumen de OverMsg(PVM) con Cinco Máquinas y n = 2000 en el CeTAD.

Quizás recién en este momento se nota gráficamente algo que se podía esperar desde la proposición misma del algoritmo pensado para solapar cómputo local con comunicaciones: el tiempo de cómputo local aumenta cuando se llevan a cabo las comunicaciones y es probable que el envío de mensajes sea más *costoso* en términos de tiempo de ejecución que la recepción. Esto se confirma gráficamente en la Figura 4.26 solamente para las computadoras **cetadfomec1** y **cetadfomec2**, para las cuales el tiempo de procesamiento para cálculos parciales (que se muestran como "Cómputo" en la Figura 4.26) mientras envían datos son mayores que el resto de los tiempos de procesamiento para cálculos parciales. Sin embargo, la proporción de aumento de tiempos de cómputo sigue sin ser importante cuando son comparadas con los tiempos de comunicaciones, comparación que se puede cuantificar a partir de los datos en la Tabla 4.7.

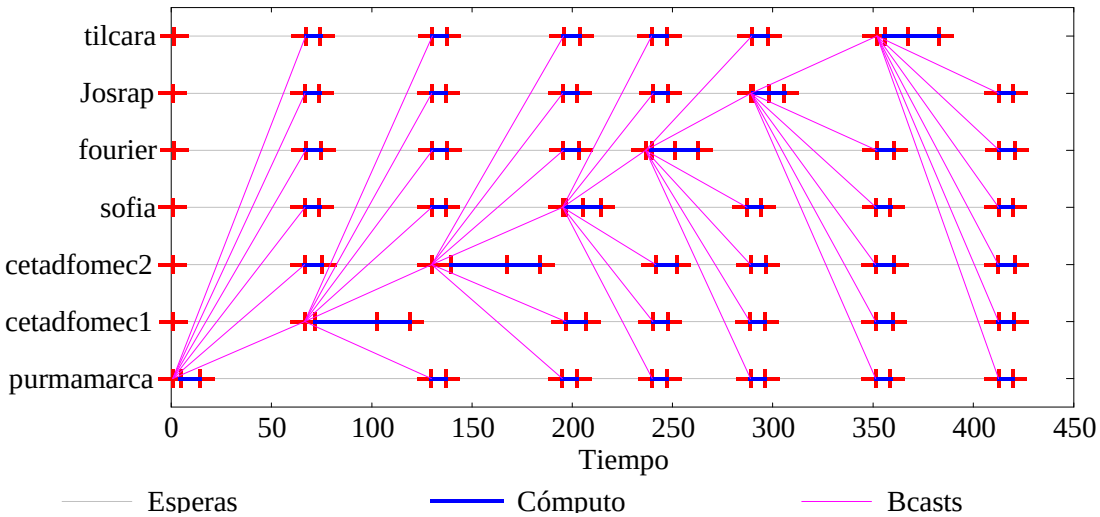


Figura 4.26: Perfil de OverMsg(PVM) con Siete Máquinas y n = 3200 en el CeTAD.

La Tabla 4.7 *completa* la información de la Figura 4.26 con el resumen de la ejecución, mostrando

- las cantidades de filas asignadas (***Filas***),
- los tiempos totales de cómputo y de comunicaciones (***Tot. Cómp*** y ***Tot. Msg.***),
- el tiempo dedicado en cada iteración a cómputo local (***Por It.***),

en cada computadora.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	771	53.36	7.62	365.40
cf1	593	88.55	12.65	330.97
cf2	593	89.01	12.72	330.66
sofia	549	53.26	7.61	365.44
fourier	276	62.05	8.86	356.99
Josrap	242	51.96	7.42	367.13
tilcara	176	65.33	9.33	316.02

Tabla 4.7: Resumen de OverMsg(PVM) con Siete Máquinas y n = 3200 en el CeTAD.

La Figura 4.27 y la Tabla 4.8 muestran todo lo referido a una multiplicación de matrices en paralelo de 3200×3200 elementos, con el algoritmo de cómputo y comunicaciones solapados, utilizando las diez máquinas disponibles del CeTAD.

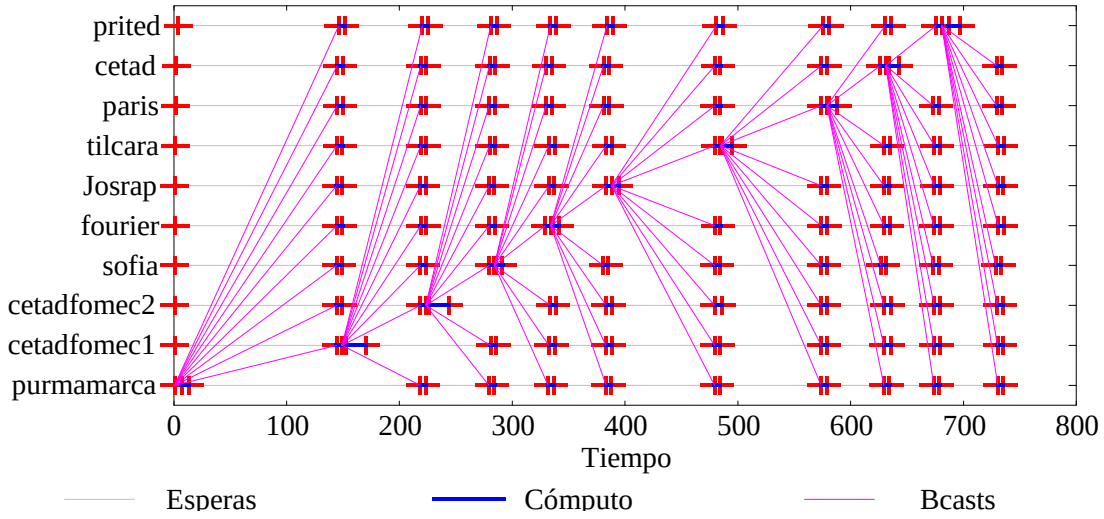


Figura 4.27: Perfil de OverMsg(PVM) con Diez Máquinas y n = 3200 en el CeTAD.

Es muy interesante comparar los totales en cuanto a tiempo de cómputo (***Tot. Cómp.***) y de comunicaciones (***Tot. Msg.***) que se muestran en la Tabla 4.7 y en la Tabla 4.8. El promedio de tiempo total de cómputo en cada computadora es aproximadamente 66.22 segundos cuando se utilizan las siete computadoras con mayor capacidad de cálculo del CeTAD. Cuando se utilizan todas las computadoras, este promedio es aproximadamente 53.5 segundos.

Comparando ahora los tiempos de comunicaciones dados en la Tabla 4.7 con los dados en la Tabla 4.8 deberían ser *iguales* (o, en realidad, muy similares) porque en todos los casos se deben transferir entre las computadoras (vía mensajes broadcast) todos los datos de la matriz B y por lo tanto el tiempo de comunicaciones debería mantenerse más o menos invariante. Esto es asumiendo que la implementación de cada mensaje broadcast entre

procesos aprovecha al máximo la capacidad de broadcast de las redes Ethernet, donde a lo sumo podría haber un pequeño aumento de tiempo por la mayor cantidad de procesos receptores de cada uno de los mensajes que se deben llevar a cabo. Sin embargo, el promedio de tiempo total de comunicaciones cuando se utilizan las siete mejores máquinas del CeTAD es de 347.52 segundos y cuando se utilizan todas las computadoras es de 676.14 segundos, es decir casi el doble de tiempo para transferir los mismos datos.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	726	49.16	4.92	685.10
cf1	559	63.12	6.31	671.01
cf2	559	63.72	6.37	670.37
sofia	518	48.72	4.87	683.82
fourier	261	50.20	5.02	684.24
Josrap	228	48.08	4.81	686.53
tilcara	166	52.02	5.20	682.36
paris	77	50.20	5.02	682.36
cetad	61	52.03	5.20	680.20
prited	45	57.78	5.78	635.42

Tabla 4.8: Resumen de OverMsg(PVM) con Diez Máquinas y n = 3200 en el CeTAD.

Dado que el problema de rendimiento está dado por las comunicaciones, conviene seguir analizando más detalladamente los tiempos de comunicaciones, y en este sentido, los resúmenes de las ejecuciones paralelas pueden proveer más información. Como ya se ha comentado, para un tamaño de matrices dado, la cantidad de datos que se deben comunicar entre las computadoras es el mismo e independiente de la cantidad de computadoras que se utilicen en paralelo. Desde el punto de vista de los mensajes, siempre se deben transferir los datos de la matriz B (C = A×B), entre todas las computadoras.

De acuerdo con la Tabla 4.4, con el algoritmo sin solapamiento de cómputo y comunicaciones en cinco computadoras se utilizan en promedio 65.12 segundos en cada computadora para la comunicación de los datos de la matriz B. Según la Tabla 4.5, Cuando se utilizan las diez computadoras (el mismo algoritmo y el mismo tamaño de matrices), el promedio de tiempo es de 253.42 segundos. Aunque los valores absolutos son totalmente distintos, con el algoritmo diseñado para solapar las comunicaciones con cómputo local y para matrices de orden n = 3200, la situación general no parece cambiar demasiado. De acuerdo con la Tabla 4.7, el tiempo promedio de comunicaciones es de 347.52 segundos cuando se utilizan siete computadoras y de acuerdo con la Tabla 4.8 el tiempo promedio de comunicaciones es de 676.14 segundos cuando se utilizan diez computadoras.

La Tabla 4.9 muestra un resumen de lo que sucede con las comunicaciones en cuanto a rendimiento dado en MB/s (2^{20} bytes pos segundos) con los datos de las tablas mencionados anteriormente, donde

- ***n*** es el orden de las matrices que se multiplican (y el orden de la matriz B que se transfiere entre las máquinas.
- ***Cant. Máq.*** es la cantidad de computadoras que se utilizan en paralelo y donde se instrumentó el código para obtener, entre otros, los tiempos de comunicaciones.
- ***Algoritmo*** indica el algoritmo paralelo utilizado.
- ***Tiempo*** es el tiempo local promedio empleado para las comunicaciones.
- ***MB/s*** denota el rendimiento de la red de interconexión dado en Megabytes (2^{20} bytes) por segundo, calculado de acuerdo al tiempo de transferencia de la matriz B (que es la única que se debe transferir entre las computadoras).

n	*Cant. Máq.*	*Algoritmo*	*Tiempo*	*MB/s*
2000	5	SeqMsg	65.12	0.23
2000	10	SeqMsg	253.42	0.06
3200	7	OverMsg	347.52	0.04
3200	10	OverMsg	676.14	0.02

Tabla 4.9: Rendimiento de las Comunicaciones en el CeTAD.

Las dos conclusiones más importantes con respecto al rendimiento de las comunicaciones utilizando los datos de la Tabla 4.9 son:

1. En general, es muy bajo, ya que en el mejor de los casos se utiliza menos del 25% de la capacidad máxima teórica de la red de interconexión.
2. Para una cantidad dada de datos a transferir se cumple que con mayor cantidad de computadoras el rendimiento disminuye.

4.7.2 Red Local del LQT

Tanto los perfiles de ejecución de cada uno de los algoritmos como los resúmenes de tiempos de ejecución de cómputo y de mensajes son muy similares a los que se mostraron del CeTAD. Es claro que no coinciden en términos absolutos por las diferencias en cuanto a computadoras y tamaños de matrices con las que se lleva a cabo el procesamiento. Lo que es significativamente similar es el "comportamiento" en cuanto a rendimiento de las diferentes máquinas paralelas y su consiguiente conclusión en cuanto al problema a resolver: sigue siendo el de las comunicaciones.

La Figura 4.28 muestra el perfil de ejecución que corresponde al mejor tiempo paralelo utilizado para resolver una multiplicación de matrices de 5000×5000 elementos. Este tiempo se obtiene con el algoritmo de cómputo y comunicación solapados utilizando tres computadoras. Es interesante notar dos aspectos bastante relevantes a partir del perfil de ejecución de la Figura 4.28:

- Se aprovecha bastante efectivamente el solapamiento de las comunicaciones con cómputo local. Más específicamente, las computadoras que reciben los mensajes broadcast enviados desde **lqt_06** y **lqt_02** esperan por estos datos muy poco tiempo más del esperado.

- Casi todo el tiempo de comunicaciones que afecta el tiempo total de ejecución es el que corresponde al primer broadcast, por el que las máquinas tienen que esperar inicialmente, es decir el que se envía desde **lqt_07** y que se recibe en **lqt_06** y **lqt_02**.

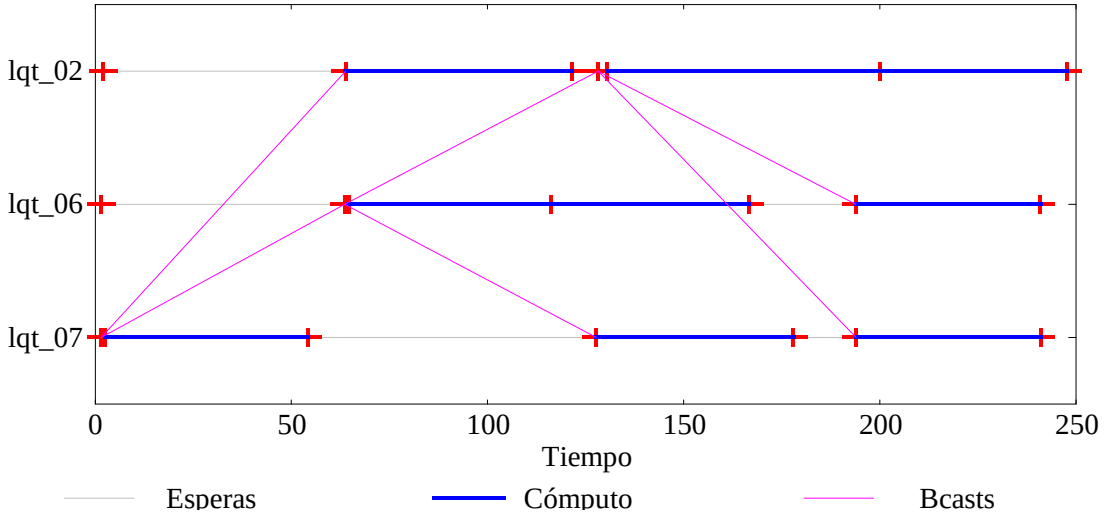


Figura 4.28: Perfil de OverMsg(PVM) con Tres Máquinas y n = 5000 en el LQT.

La Tabla 4.10 muestra el resumen de la ejecución paralela correspondiente al perfil de la Figura 4.28.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
lqt_07	1808	148.87	49.62	90.60
lqt_06	1807	149.06	49.69	90.34
lqt_02	1385	175.20	58.40	70.66

Tabla 4.10: Resumen de OverMsg(PVM) con Tres Máquinas y n = 5000 en el LQT.

Como ya se aclaró con los valores de speedup obtenidos por los algoritmos, el rendimiento empeora cuando se utilizan más máquinas de la red local del LQT. Esto se visualiza claramente en la Figura 4.29, que muestra el perfil de ejecución de una multiplicación de matrices de 5000×5000 elementos utilizando el algoritmo con comunicaciones y cómputo solapados y todas las computadoras disponibles del LQT.

También en la Figura 4.29 es evidente que hay una computadora en particular que se comporta de manera diferente al resto, o por lo menos que el mensaje broadcast enviado desde **lqt_01** utiliza un tiempo de transmisión significativamente mayor que los demás. Una vez más, como cuando esto se identificó la misma situación en la red local del CeTAD (Figura 4.24), se debe aclarar que aunque este broadcast utilizara el tiempo promedio de comunicaciones que utilizan los demás, el tiempo total de comunicaciones seguiría siendo dominado por la suma de los tiempos utilizados para las comunicaciones.

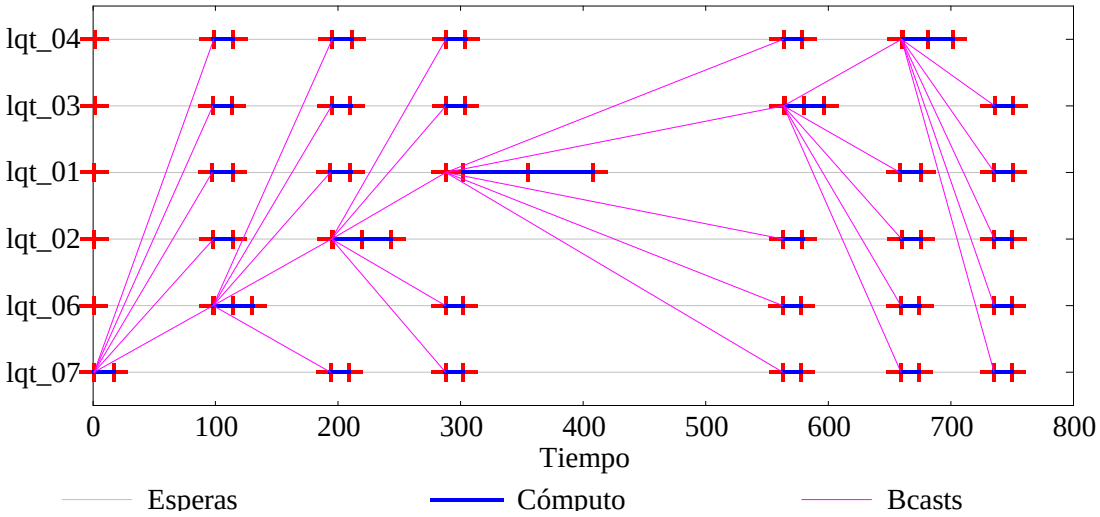


Figura 4.29: Perfil de OverMsg(PVM) con Seis Computadoras y n = 5000 en el LQT.

La Tabla 4.11 muestra el resumen de ejecución de ejecución de la multiplicación de matrices de 5000×5000 elementos utilizando el algoritmo con comunicaciones y cómputo solapados y todas las computadoras disponibles del LQT que *corresponde* a la Figura 4.29. Como ya fue posible identificar a partir de la experimentación con las computadoras del CeTAD, lo que cambia significativamente en cuanto a rendimiento es el tiempo de comunicaciones. Este comportamiento se corrobora en el LQT, comparando la columna que muestra los tiempos utilizados para comunicaciones en cada máquina (***Tot. Msg.***) de la Tabla 4.10 con la de la Tabla 4.11. Cuando se utilizan más máquinas, la misma cantidad de datos (los elementos de la matriz B) se transfieren entre las computadoras en mucho mayor tiempo.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
lqt_07	1089	89.21	14.87	657.71
lqt_06	1089	89.15	14.86	657.57
lqt_02	835	109.13	18.19	637.87
lqt_01	811	174.52	29.09	572.56
lqt_03	589	92.79	15.47	654.50
lqt_04	587	102.52	17.09	592.84

Tabla 4.11: Resumen de OverMsg(PVM) con Seis Máquinas y n = 5000 en el LQT.

En el caso de las matrices de orden n = 9000, la situación en cuanto a rendimiento es similar. La Figura 4.30 muestra el perfil de ejecución que corresponde al mejor tiempo paralelo utilizado para resolver una multiplicación de matrices de 9000×9000 elementos. Este tiempo se obtiene con el algoritmo de cómputo y comunicación solapados utilizando las cuatro computadoras con mayor capacidad de cálculo.

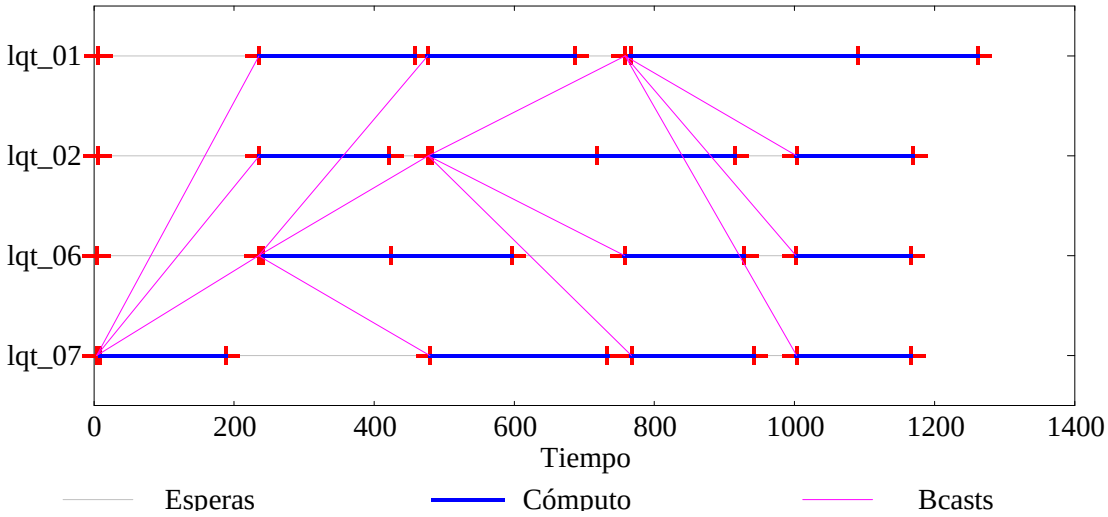


Figura 4.30: Perfil de OverMsg(PVM) con Cuatro Computadoras y n = 9000 en el LQT.

Una vez más, cuando se utilizan más máquinas para resolver el mismo problema (una multiplicación de matrices de 9000×9000 elementos), con el mismo algoritmo (que resuelve las comunicaciones solapadas con cómputo local), el rendimiento empeora y el tiempo total de ejecución es mayor. La Figura 4.31 muestra el perfil de ejecución para resolver una multiplicación de matrices de 9000×9000 elementos en paralelo utilizando todas las computadoras disponibles del LQT y el algoritmo con cómputo y comunicaciones solapados. Los resúmenes de la ejecución no se muestran en este caso, pero no hacen más que confirmar lo que se puede concluir visualmente a partir de la Figura 4.30 y la Figura 4.31: la pérdida de rendimiento se debe al excesivo tiempo de comunicaciones al utilizar mayor cantidad de computadoras.

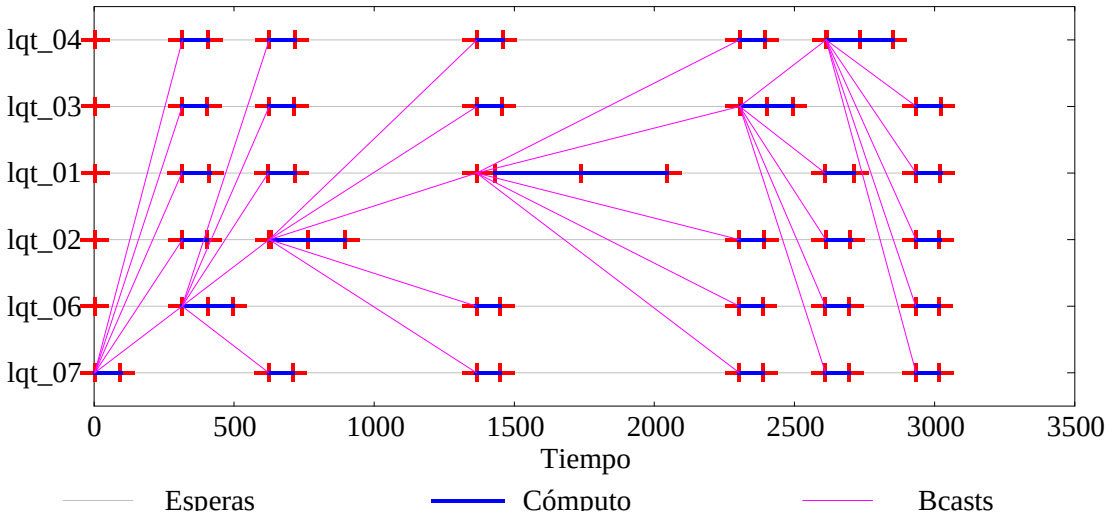


Figura 4.31: Perfil de OverMsg(PVM) con Seis Computadoras y n = 9000 en el LQT.

Específicamente con respecto al rendimiento de las comunicaciones, al analizar los valores de tiempo de ejecución para matrices de orden n = 5000 (dados en la Tabla 4.10 y en la Tabla 4.11), y de orden n = 9000 (que no se muestran explícitamente), sucede lo mismo

que en la red local del CeTAD:
1. En general, es muy bajo comparado con el tiempo de comunicaciones.
2. Para una cantidad dada de datos a transferir se cumple que con mayor cantidad de computadoras el rendimiento disminuye. El tiempo necesario para realizar un broadcast aumenta linealmente con la cantidad de computadoras involucradas.

4.7.3 Red Local del LIDI

En la red local del LIDI se confirman los resultados anteriores pero con la diferencia dada por el mejor rendimiento de la red de interconexión, que es diez veces mejor que el de las redes locales del CeTAD y del LQT. En este sentido, los perfiles de ejecución (y los resúmenes de tiempos de cómputo y comunicaciones) muestran que:
- A diferencia de lo que sucede en las redes locales del CeTAD y del LQT, el tiempo de comunicaciones no tiene un peso tan determinante en el tiempo total de los programas paralelos.
- Tal como sucede en las redes locales del CeTAD y del LQT, el tiempo de comunicaciones aumenta notablemente a medida que se utilizan más computadoras para resolver un mismo problema.

A modo de ejemplo, la Figura 4.32 muestra el perfil de la ejecución del programa paralelo con cómputo y comunicaciones secuenciales utilizando cuatro computadoras para resolver una multiplicación de matrices de orden n = 2000.

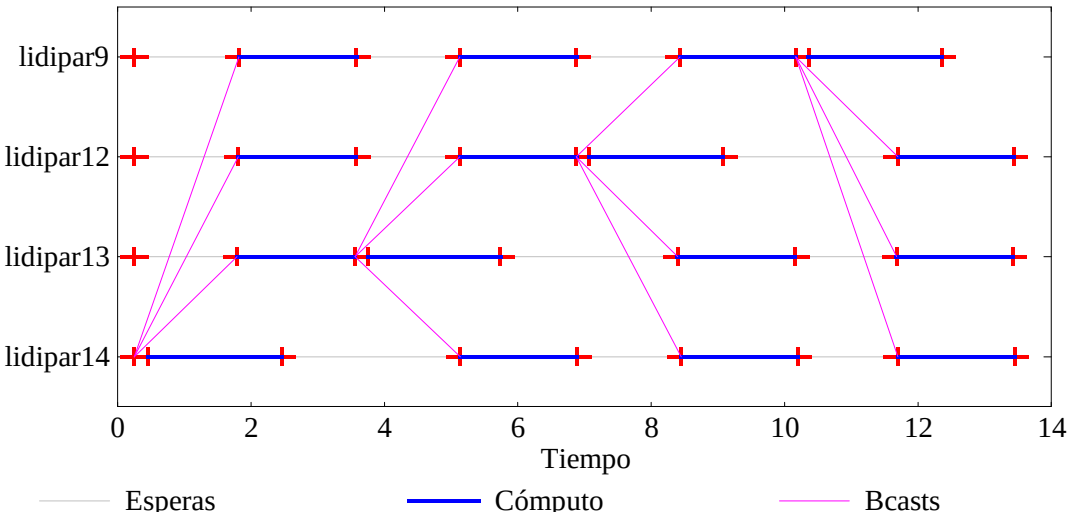


Figura 4.32: Perfil de SeqMsg(PVM) con Cuatro Computadoras y n = 2000 en el LIDI.

Además, Tabla 4.12 muestra el resumen de la ejecución del programa paralelo con cómputo y comunicaciones secuenciales utilizando cuatro computadoras para resolver una multiplicación de matrices de orden n = 2000. Los datos de esta tabla nuevamente completan la visión gráfica de la Figura 4.32 y además permite cuantificar aspectos importantes como la relación entre los tiempos de cómputo y comunicaciones que cada máquina utiliza durante la ejecución del programa.

Nombre	*Filas*	*Tot. Cómp.*	*Por It.*	*Tot. Msg.*
lidipar14	500	7.25	1.81	6.30
lidipar13	500	7.29	1.82	6.16
lidipar12	500	7.26	1.81	6.31
lidipar9	500	7.24	1.81	5.24

Tabla 4.12: Resumen de SeqMsg(PVM) con Cuatro Máquinas y n = 2000 en el LIDI.

La Figura 4.33 y la Tabla 4.13 muestran el perfil y el resumen de la ejecución del programa paralelo con cómputo y comunicaciones secuenciales utilizando todas las computadoras para resolver una multiplicación de matrices de orden n = 2000. Es decir que se mantienen el algoritmo y el tamaño de las matrices (respecto de la Figura 4.32 y la Tabla 4.12) pero se aumenta al doble la cantidad de computadoras que se utilizan en paralelo para resolver el problema. En teoría, el tiempo total de cómputo debería reducirse a la mitad del que se tiene en la Figura 4.32, pero no solamente no es la mitad sino que es mayor.

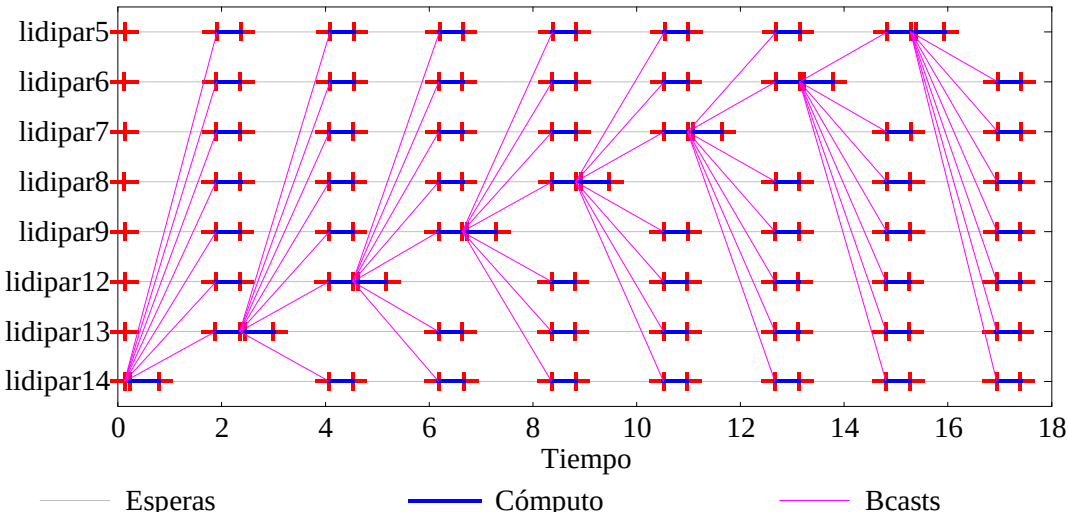


Figura 4.33: Perfil de SeqMsg(PVM) con Ocho Computadoras y n = 2000 en el LIDI.

Comparando los valores de la Tabla 4.12 y la Tabla 4.13:

- En la Tabla 4.12 (donde se utilizan cuatro computadoras) se nota que cada computadora utiliza un poco más de tiempo en cómputo local que en comunicaciones. La proporción de tiempos es aproximadamente 45% - 55%.
- La situación cambia cuando se utilizan ocho computadoras, tal como se muestra en los valores de la Tabla 4.13, ya que cada computadora utiliza para las comunicaciones un promedio de más de tres veces mayor que el tiempo que utiliza para cómputo local. Nuevamente se debe recordar que la cantidad de datos que se deben comunicar son exactamente los *mismos* y además vía mensajes broadcast en una red de interconexión que es capaz de manejar mensajes broadcast a nivel físico.

Nombre	*Filas*	*Tot. Cómp.*	*Por It.*	*Tot. Msg.*
lidipar14	250	3.78	0.47	13.49
lidipar13	250	3.75	0.47	13.52
lidipar12	250	3.73	0.47	13.54
lidipar9	250	3.74	0.47	13.54
lidipar8	250	3.73	0.47	13.55
lidipar7	250	3.74	0.47	13.54
lidipar6	250	3.72	0.46	13.56
lidipar5	250	3.73	0.47	12.08

Tabla 4.13: Resumen de SeqMsg(PVM) con Ocho Máquinas y n = 2000 en el LIDI.

Es interesante notar lo que sucede cuando se resuelve la multiplicación de matrices de orden n = 3200. La Figura 4.34 y la Figura 4.35 muestran los perfiles de ejecución del programa paralelo con cómputo y comunicaciones secuenciales que resuelven una multiplicación de matrices de orden n = 3200 utilizando cuatro y ocho computadoras respectivamente. Evidentemente, el tiempo de ejecución con cuatro computadoras (poco más de 90 segundos, Figura 4.34) es bastante mayor que el tiempo de ejecución con ocho computadoras (aproximadamente 50 segundos, Figura 4.35), pero esto no asegura por sí solo que el rendimiento es aceptable.

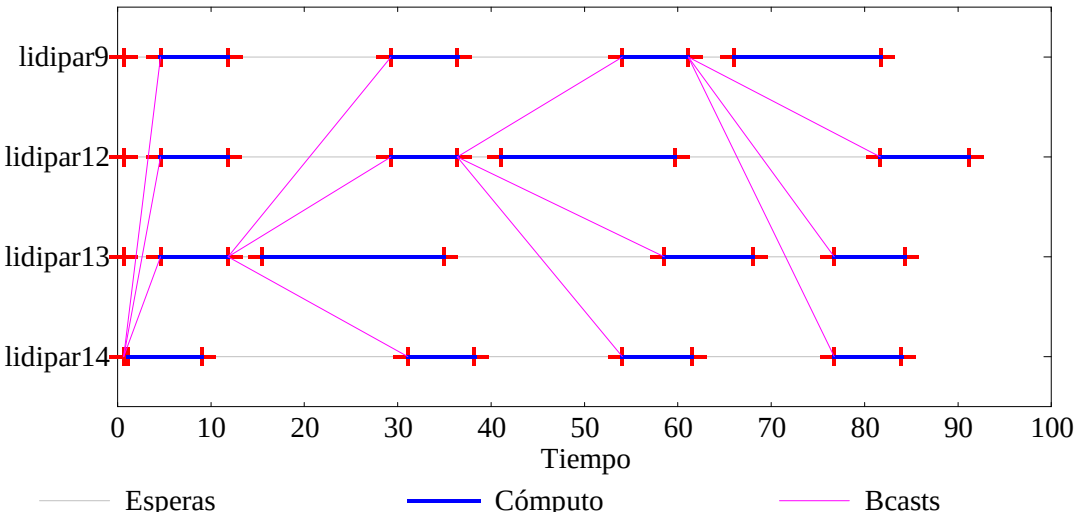


Figura 4.34: Perfil de SeqMsg(PVM) con Cuatro Computadoras y n = 3200 en el LIDI.

La información gráfica de la Figura 4.34 se completa con los datos de la Tabla 4.14 que muestra el resumen de tiempos de ejecución y, de manera análoga, la información gráfica de la Figura 4.35 se completa con los datos de la Tabla 4.15 que también muestra el resumen de tiempos de ejecución.

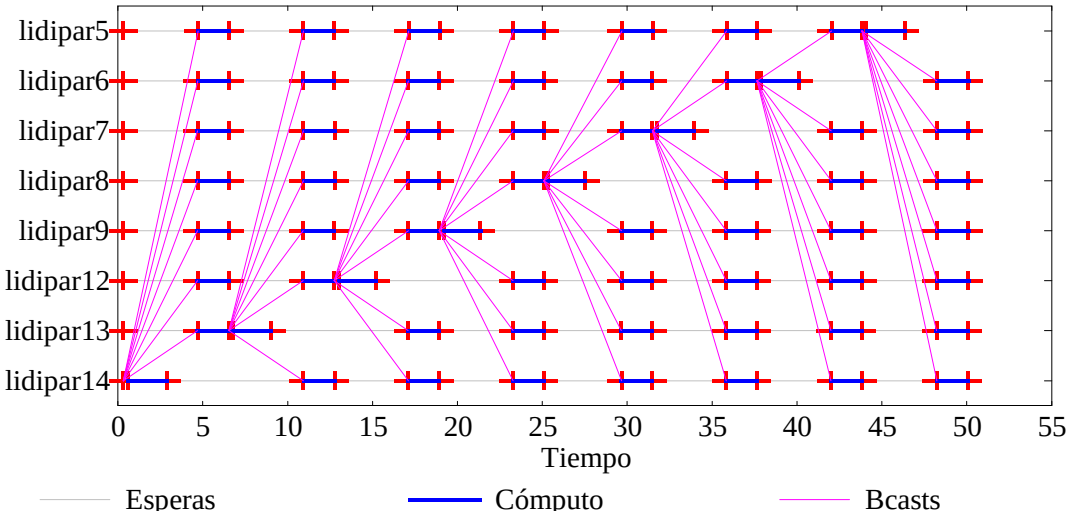


Figura 4.35: Perfil de SeqMsg(PVM) con Ocho Computadoras y *n* = 3200 en el LIDI.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
lidipar14	800	30.20	7.55	55.40
lidipar13	800	43.00	10.75	42.79
lidipar12	800	43.24	10.81	49.37
lidipar9	800	38.20	9.55	44.69

Tabla 4.14: Resumen de SeqMsg(PVM) con Cuatro Máquinas y *n* = 3200 en el LIDI.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
lidipar14	400	14.97	1.87	35.13
lidipar13	400	14.98	1.87	35.12
lidipar12	400	14.97	1.87	35.13
lidipar9	400	14.94	1.87	35.17
lidipar8	400	14.95	1.87	35.16
lidipar7	400	14.98	1.87	35.14
lidipar6	400	14.89	1.86	35.22
lidipar5	400	15.01	1.88	31.47

Tabla 4.15: Resumen de SeqMsg(PVM) con Ocho Máquinas y *n* = 3200 en el LIDI.

Se puede apreciar en la Figura 4.34 que hay períodos de cómputo que son sustancialmente mayores que otros. Más específicamente, en las computadoras **lidipar13**, **lidipar12**, y **lidipar9**, el período de cómputo posterior al envío del mensaje broadcast es mayor que todos los demás períodos de cómputo locales. Por la definición misma del algoritmo, en todos los períodos de cómputo se lleva a cabo la misma tarea y por lo tanto el tiempo de

cómputo debería ser igual.

Una de las principales razones para que una computadora tenga menor rendimiento (asumiendo que se realiza exactamente la misma tarea, como en este caso), es la utilización de la memoria swap que se ha explicado antes. Por lo tanto, la explicación más *aceptable* está relacionada justamente con la memoria. Dado que para llevar a cabo un mensaje broadcast se utiliza PVM, cada una de las rutinas de comunicaciones tiene su propio requerimiento de memoria agregado (overhead), básicamente para almacenamiento de los mensajes en *buffers* (memoria intermedia) que PVM se encarga luego de transferir entre las máquinas. En general es aceptado que la cantidad de memoria extra en este sentido es como mínimo igual a la cantidad de datos que se transfieren. Esta sobrecarga (overhead) de memoria hace que la memoria principal disponible no sea suficiente y por lo tanto se utiliza el espacio de memoria swap. En realidad, esto genera una reducción del rendimiento de

- Cómputo, dado que parte de los datos a procesar quizás están asignados en memoria swap.
- Comunicaciones, dado que parte de los datos a transferir quizás están asignados en memoria swap.

Y es por esta razón los períodos de cómputo posteriores a un broadcast son “más lentos” que los demás.
A diferencia de lo que ha sucedido hasta ahora (en las redes locales del CeTAD y del LQT), el tiempo total de comunicaciones en promedio es bastante mayor para cuatro máquinas que para ocho, lo cual se puede comprobar cuantitativamente con los datos de la Tabla 4.14 y la Tabla 4.15 respectivamente.

El perfil y el resumen de ejecución para el mismo problema pero con seis máquinas que se muestran en la Figura 4.36 y en la Tabla 4.16 confirman que el problema de rendimiento anterior (que se puede visualizar en la Figura 4.34) está relacionado con la memoria ya que

- Todos los períodos de cómputo en todas las computadoras utilizan aproximadamente el mismo tiempo de ejecución.
- El tiempo total de comunicaciones es menor que para cuatro y ocho computadoras.

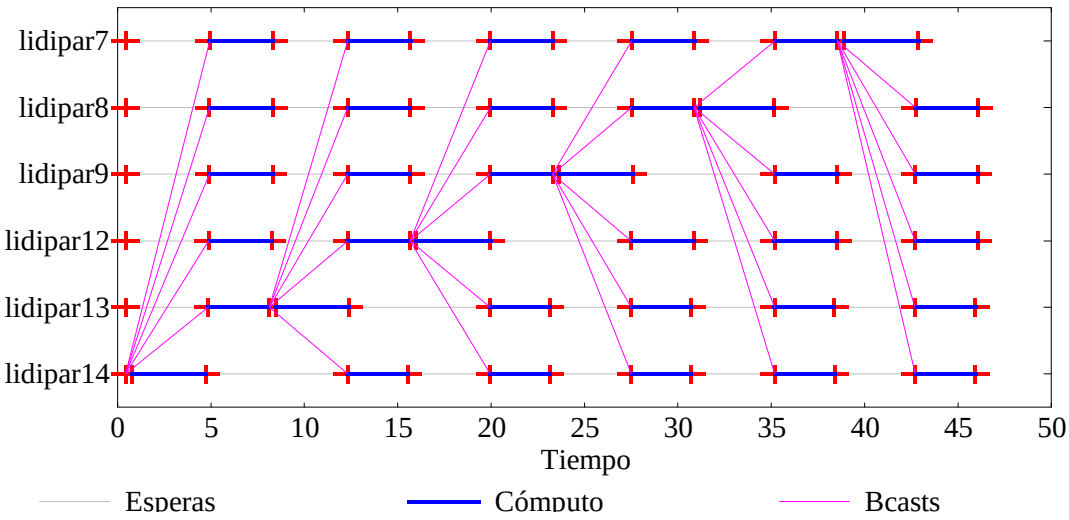


Figura 4.36: Perfil de SeqMsg(PVM) con Seis Computadoras y n = 3200 en el LIDI.

Nombre	*Filas*	*Tot. Cómp.*	*Por It.*	*Tot. Msg.*
lidipar14	534	19.99	3.33	25.53
lidipar13	534	19.98	3.33	25.53
lidipar12	533	20.75	3.46	24.89
lidipar9	533	20.73	3.46	24.91
lidipar8	533	20.75	3.46	24.90
lidipar7	533	20.77	3.46	21.68

Tabla 4.16: Resumen de SeqMsg(PVM) con Seis Máquinas y n = 3200 en el LIDI.

4.8 Rendimiento Real de las Redes Locales Utilizando "UDP"

Dado que el problema de rendimiento es generado casi exclusivamente por las comunicaciones, se llevaron a cabo experimentos específicos para evaluar el rendimiento de los mensajes broadcast y también punto a punto utilizando la biblioteca PVM. Si bien los resultados se muestran detalladamente en el Apéndice C, las principales conclusiones son [120]:

- Las formas de enviar un mismo mensaje a más de un proceso destino cuando cada proceso está asignado a una máquina distinta se implementan con múltiples mensajes punto a punto. Tanto
  - la operación de multicast, **pvm_mcast()**,
  - como la operación de broadcast en un grupo, **pvm_bcast()**,

  implican que, como mínimo, el mismo mensaje es enviado m veces desde la computadora origen (donde está ejecutándose el proceso que envía el mensaje) hacia las m máquinas donde hay al menos un proceso destino del mensaje. Si, por ejemplo, un mensaje broadcast o multicast tiene cinco receptores y cada uno de estos procesos receptores está ejecutándose en una máquina distinta (y distinta de la máquina en donde se ejecuta el proceso que envía el mensaje), el tiempo total del mensaje será aproximadamente igual a cinco veces el tiempo del mismo mensaje si se envía a otro proceso que se ejecuta en otra máquina.
- El tiempo de latencia de los mensajes depende de las computadoras origen y destino del mismo. Sin embargo, para mensajes suficientemente grandes el tiempo de latencia no tiene relevancia con respecto al tiempo de transferencia de los datos y por lo tanto el tiempo total del mensaje es independiente de las máquinas que se comunican.

La rutina de comunicaciones broadcast propuesta está orientada al aprovechamiento de las características físicas (específicamente broadcast) de las redes Ethernet que la biblioteca de comunicaciones (PVM) no utiliza. Esto tiene como consecuencia directa que el tiempo de comunicaciones broadcast con PVM es mucho mayor que el esperado y por lo tanto no se tienen buenos resultados de rendimiento total. En este punto se tienen varias alternativas,

siendo las dos más importantes:
1. Utilizar otra biblioteca de pasaje de mensajes, tal como alguna implementación de MPI, que es lo que se suele recomendar para este tipo de arquitecturas paralelas.
2. Implementar una rutina de mensajes broadcast (y eventualmente toda una biblioteca de comunicaciones colectivas) para utilizar explícitamente la capacidad de broadcast de las redes Ethernet.

La utilización de otra biblioteca de pasaje de mensajes tiene, en principio, un inconveniente fundamental desde el punto de vista de rendimiento, o de "predicción" de buen rendimiento de los mensajes broadcast [123]. En el caso específico de MPI, es claro que el rendimiento es dependiente de la implementación. Más específicamente, la implementación será la que determina el grado de aprovechamiento de las características de las redes Ethernet para los mensajes broadcast. En este sentido, MPI y en particular todas sus implementaciones comparten *cierto grado de incertidumbre* respecto del rendimiento de los mensajes broadcast con todas las demás bibliotecas de pasaje de mensajes, incluida PVM. La diferencia en este caso son los experimentos específicos que se llevaron a cabo, y que determinaron las características de rendimiento de los mensajes broadcast (y multicast) para PVM y no para las demás bibliotecas. De hecho, es bastante difícil que las bibliotecas de pasaje de mensajes sean optimizadas para las características de las redes Ethernet dado que:
- En general, las bibliotecas son propuestas de una forma u otra como estándares para las máquinas paralelas de pasaje de mensajes y por lo tanto no tiene sentido orientarlas hacia un tipo específico de redes de interconexión. De hecho, tanto PVM como MPI se han implementado para distintos tipos de máquinas paralelas y por lo tanto no tiene sentido orientarlas *a priori* a las redes de interconexión Ethernet.
- En general, las bibliotecas proveen una gran cantidad de rutinas de comunicación. Aunque en teoría se puede afirmar que con las primitivas **send** - **receive** para la comunicación de procesos punto a punto es suficiente, también se ha concluido que existe una gran variedad de rutinas de comunicación que se consideran útiles y hasta necesarias en algunos casos. Quizás el ejemplo más claro al respecto sea la propia definición del estándar MPI. En este contexto, es muy difícil orientar u optimizar una o una clase de rutinas de comunicaciones para una o una clase de redes de interconexión sin producir una biblioteca excesivamente costosa (en términos de desarrollo, mantenimiento, etc.) y/o *demasiado* específica.

Por estas razones, se eligió implementar una rutina de mensajes broadcast entre procesos de usuario con un conjunto de premisas de diseño e implementación, de forma tal que [123] [132]:
- Aproveche el *propio* broadcast de las redes Ethernet, y de esta manera se optimiza en cuanto a rendimiento. Dado que el algoritmo depende exclusivamente de los mensajes broadcast, al aprovechar el broadcast de las redes Ethernet automáticamente se tiene una buena expectativa en cuanto a escalabilidad, ya que se espera que el tiempo de comunicaciones se mantenga constante y no aumente de manera proporcional a la cantidad de computadoras que se utilizan.
- Sea suficientemente simple para no imponer una carga demasiado pesada en cuanto a implementación y mantenimiento. Además, es claro que la simplicidad *per se* hace un gran aporte para el rendimiento óptimo. Por otro lado, la propuesta es suficientemente específica como para que la implementación sea simple.

- Con el máximo de portabilidad posible, para ser utilizada si es posible incluso en el contexto de las redes de interconexión que no sean Ethernet.
- Implementada e instalada desde modo usuario, sin cambiar el sistema operativo (el *kernel*) y sin que sea necesario obtener permisos especiales (*superuser*). Es normalmente aceptado que los mejores resultados en cuanto a rendimiento se obtienen adaptando el kernel y/o con la posibilidad de manejar las prioridades de los procesos, tal como en [31] [25] [GAMMA]. Estas posibilidades se descartan dado que:
    - En general, las bibliotecas de uso libre no utilizan estas características y por lo tanto sería como cambiar el contexto de desarrollo de software paralelo. Básicamente, un usuario que haya utilizado siempre PVM nunca tuvo ni tiene por qué obtener prioridades especiales y aún cambiar el mismo sistema operativo.
    - La propuesta original se dirige a redes de computadoras instaladas y por lo tanto cada computadora no necesariamente tiene como objetivo único y/o principal el cómputo paralelo. De hecho, se puede tener el caso de distintos administradores para cada una de las computadoras a utilizar en paralelo y esto produce, por lo menos, un múltiple trabajo de administración que en general no es fácil de resolver.
- Eventualmente pueda ser extendida a toda una biblioteca de comunicaciones colectivas, tal como otras propuestas [15] [14] [16] pero orientada específicamente a las redes de interconexión Ethernet.

La mayoría (sino *todas*) las premisas anteriores se cumplen al basar todo el diseño y la implementación de la rutina de broadcast en el protocolo estándar UDP (User Datagram Protocol) [95] sobre IP (Internet Protocol) [96] ya que:

- UDP permite enviar un mismo dato o conjunto (paquete) de datos a múltiples destinos a nivel de aplicaciones de usuario.
- Tal como se verificó en todas las máquinas utilizadas, la implementación del protocolo UDP aprovecha directamente la capacidad de broadcast de las redes Ethernet.
- En principio, parece razonable que el broadcast implementado directamente como parte del protocolo UDP tenga mejor rendimiento que el implementado por un usuario. Si en una red ATM, por ejemplo, se tiene la posibilidad de utilizar UDP es muy probable que el broadcast de UDP sea *mejor* (en términos de rendimiento) que el que se pueda implementar desde los procesos de usuario. Aunque el rendimiento no se tenga en cuenta, siempre que exista una implementación del protocolo UDP se podrá utilizar el broadcast propuesto, independientemente de que la red de interconexión sea Ethernet o no [93].
- La interfase de usuario provista por los sockets es suficientemente simple y ampliamente extendida a todas las versiones de UNIX, como para simplificar la implementación de la rutina de broadcast, aún cuando se deben resolver problemas relacionados con *sincronización* de procesos (en una misma y en diferentes computadoras) y *confiabilidad* de las comunicaciones.
- Los protocolos UDP, TCP e IP son fácilmente utilizables desde los procesos de usuario, al menos en las computadoras estándares de las redes locales instaladas.

En resumen, se dispone de *nueva* rutina de mensajes broadcast basada en UDP y portable como mínimo a todas las versiones de UNIX utilizadas en todas las redes locales en las cuales se lleva a cabo la experimentación. Con esta nueva rutina de mensajes broadcast se

vuelven a realizar los mismos experimentos y los resultados se muestran en las subsecciones que siguen.

Los resultados de la experimentación muestran varias características que no han sido encontradas hasta ahora. La aparición de la mayoría de estas características está dada por:

- El rendimiento marcadamente superior del broadcast basado en UDP sobre el de la biblioteca PVM. Esto genera que el tiempo de comunicaciones sea comparable (al menos en el mismo orden de magnitud) con el del cómputo y por lo tanto el rendimiento de cómputo de cada computadora es importante dado su peso en el tiempo de ejecución total. Hasta ahora, el tiempo de comunicaciones ha sido tan superior que básicamente todo o la gran mayoría del tiempo de ejecución paralela se debe a las comunicaciones.
- La heterogeneidad de las computadoras, lo cual en sí mismo aporta un grado importante de innovación a lo que generalmente se muestra sobre rendimiento de máquinas paralelas.

Dado que ahora la heterogeneidad y más específicamente las diferencias de rendimiento de cómputo de las computadoras se vuelve relevante, la mayoría de las nuevas características aparecen en la red local del CeTAD, que es la más heterogénea de las tres en las que se lleva a cabo la experimentación.

En general, en las redes locales del LQT y del LIDI se repiten algunas características que aparecen en la red local del CeTAD o directamente no aparecen. A lo sumo se acentúa en el caso de la red local del LQT la influencia de los requerimientos de cómputo cuando los problemas a resolver son (o pueden ser) mayores por la mayor cantidad de memoria disponible. Por esta razón los resultados de la experimentación en la red local del CeTAD serán explicados con el mayor nivel de detalle posible y en el caso de las otras dos redes locales se verán solamente las diferencias en cuanto a “comportamiento” siempre desde el punto de vista del rendimiento.

4.8.1 Red Local del CeTAD

Con el objetivo de presentar y separar mejor los resultados de la experimentación en la red local del CeTAD, todo el análisis de los datos obtenidos se divide en las dos subsecciones que siguen de acuerdo con el tamaño del problema. La primera se dedica a los resultados de speedup tomando como referencia el tiempo de ejecución de la multiplicación de matrices de orden $n = 2000$. Posteriormente se analizan los resultados de speedup tomando como referencia el tiempo de ejecución de la multiplicación de matrices de orden $n = 3200$.

4.8.1.1 Matrices de 2000×2000 Elementos

La Figura 4.37 muestra los valores de speedup obtenidos en la red local del CeTAD por los algoritmos implementados utilizando UDP, SeqMsg(UDP) y OverMsg(UDP) para matrices de orden $n = 2000$ junto con los que se mostraron anteriormente en la Figura 4.17.

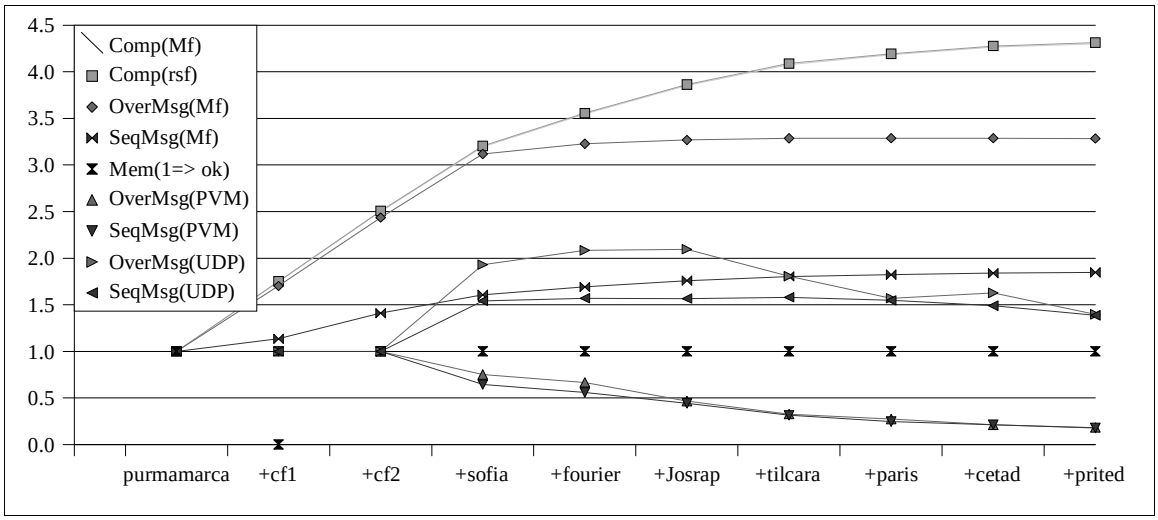


Figura 4.37: Speedup de los Algoritmos con UDP en la Red del CeTAD para n = 2000.

Tal como se puede apreciar en la Figura 4.37, el rendimiento de los algoritmos SeqMsg(UDP) y OverMsg(UDP) (que son los únicos que se agregan con respecto a los que se, muestran en la Figura 4.17), varía dependiendo de la cantidad de computadoras que se utilizan. Esta variación de los algoritmos no parece, *a priori*, estar relacionada. Inicialmente se analizarán detalladamente los resultados del algoritmo con los períodos de cómputo y comunicaciones secuenciales, SeqMsg(UDP), desde tres puntos de vista:

- La relación de los resultados de rendimiento obtenidos comparándolos con los que se obtienen al utilizar la biblioteca de comunicaciones PVM.
- La capacidad del algoritmo de utilizar al máximo la capacidad de cómputo de las máquinas, en los períodos de cómputo.
- La capacidad del algoritmo de utilizar al máximo la capacidad de la red de comunicaciones, en los períodos de comunicaciones.

Posteriormente, el mismo tipo de análisis de resultado (siempre desde el punto de vista del rendimiento en general) se hará con el algoritmo orientado al aprovechamiento de la capacidad de solapar cómputo con comunicaciones, OverMsg(UDP).

SeqMsg(UDP). El algoritmo que lleva a cabo los períodos de cómputo y comunicaciones de manera secuencial, SeqMsg(UDP), no parece alejarse demasiado de la estimación del speedup máximo que le corresponde, SeqMsg(Mf), al menos hasta que se utilizan las diez máquinas disponibles.

La Tabla 4.17 muestra el resumen de ejecución cuando se utilizan seis máquinas para multiplicar matrices de 2000×2000 elementos con este algoritmo. Comparando los valores que se muestran en la Tabla 4.17 con los que se muestran en la Tabla 4.4, donde lo único diferente es que se utiliza la rutina de broadcast provista por PVM se puede ver que:

- En términos de tiempo de cómputo local son muy similares, cuando se utiliza PVM el promedio total es de 15.61 segundos y con la rutina basada directamente en UDP el promedio es de 14.31 segundos.
- Los tiempos de comunicaciones son absolutamente diferentes, cuando se utiliza PVM el promedio total es de 65.12 segundos y con la rutina basada directamente en UDP el promedio es de 15.69 segundos, un poco más de cuatro veces menor.

Nombre	*Filas*	*Tot. Cómp.*	*Por It.*	*Tot. Msg.*
purmamarca	555	14.15	2.83	15.89
cf1	426	14.14	2.83	15.86
cf2	426	14.09	2.82	15.91
sofia	394	14.40	2.88	15.71
fourier	199	14.78	2.96	15.08

Tabla 4.17: Resumen de SeqMsg(UDP) con Cinco Máquinas y n = 2000 en el CeTAD.

La diferencia en tiempo de comunicaciones evidentemente está dada por la forma en que PVM resuelve los mensajes broadcast: múltiples mensajes punto a punto entre las computadoras involucradas. Pero más allá de la comparación con la implementación basada en PVM, es necesario verificar si cada computadora se utiliza al máximo de su capacidad y si la red de interconexión se utiliza con rendimiento óptimo. Este análisis se hará para el caso en el cual se utilizan las diez computadoras, dado que el rendimiento disminuye y se pueden identificar un poco mejor las causas de esta degradación del rendimiento.

La Tabla 4.18 muestra el resumen de ejecución cuando se utilizan todas las máquinas (diez) para multiplicar matrices de 2000×2000 elementos con el algoritmo que lleva a cabo cómputo y comunicaciones secuenciales, SeqMsg(UDP). Comparando los valores de la Tabla 4.18 con los de la tabla anterior se puede notar que:

- En términos de cómputo, no hay muchos cambios dado que las cinco computadoras que se agregan en realidad tienen muy poca capacidad de cálculo con respecto al total de las demás. La cantidad total de procesamiento de las computadoras es directamente proporcional a la cantidad de filas asignadas, que a su vez está dada por la velocidad relativa de cada computadora, y las primeras cinco máquinas tienen a su cargo el 82% del total.
- El tiempo de comunicaciones ha aumentado de 15.69 segundos con cinco máquinas a 21.2 segundos con diez máquinas, lo cual representa una penalización en el rendimiento de las comunicaciones de un poco más de 35%. Evidentemente un 35% de degradación en el rendimiento de las comunicaciones es mucho mejor que el 500% que implica utilizar PVM pero de todas maneras parece muy elevado. Sin llegar a detalles muy específicos de lo que sucede con los mensajes broadcast en términos de rendimiento se puede decir que:
    - A medida que más procesos estén involucrados en comunicaciones colectivas y en particular en mensajes broadcast, es de esperar una mayor penalización en términos de rendimiento. Cuando todos los procesos están ejecutándose en computadoras diferentes y además las computadoras están interconectadas con una red Ethernet de 10 Mb/s la degradación será mayor.
    - Las computadoras que se incorporan son relativamente mucho más lentas que las demás. Aún cuando la capacidad de transferencia de las placas de red sea de 10 Mb/s independientemente de las computadoras en las que estén instaladas, está demostrado al menos experimentalmente que como mínimo el tiempo de latencia de los mensajes depende de la capacidad de cómputo de las máquinas. A medida

que aumenta la cantidad de computadoras, el tamaño de los mensajes disminuye y por lo tanto aumenta la importancia relativa del tiempo de latencia en el tiempo total de cada mensaje.

- Y estos factores (más máquinas y con mayor tiempo de latencia de comunicaciones), se combinan para llegar a poco más de 35% de penalización en el rendimiento de las comunicaciones.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	454	11.81	1.18	21.63
cf1	349	11.84	1.18	21.58
cf2	349	11.82	1.18	21.60
sofia	324	11.97	1.20	21.48
fourier	164	12.67	1.27	20.72
Josrap	142	12.10	1.21	21.43
tilcara	104	12.08	1.21	21.27
paris	48	11.59	1.16	21.53
cetad	38	11.98	1.20	21.03
prited	28	12.65	1.27	19.75

Tabla 4.18: Resumen de SeqMsg(UDP) con Diez Máquinas y n = 2000 en el CeTAD.

En este punto se podría analizar cuál es el rendimiento de *esta* red de interconexión para los mensajes broadcast cuando se utilizan cinco y diez máquinas. Se debe recordar que en todos los casos la cantidad total de datos a transferir está dada por la cantidad de elementos de la matriz B (C = A×B). En el caso de matrices de orden n = 2000 y con números en punto flotante de precisión simple, la cantidad dada en bytes es exactamente 4×2000^2. Desde el punto de vista del rendimiento, entonces:

- Cuando se utilizan las cinco computadoras con mayor capacidad de cálculo del CeTAD el tiempo total de las comunicaciones en cada máquina es en promedio 15.69 segundos. Esto representa un ancho de banda real entre procesos de un poco más de 8 Mb/s. Por lo tanto se tiene menos de 20% de penalización respecto del máximo absoluto del hardware en la comunicación de los mensajes broadcast entre procesos.
- Cuando se utilizan todas las computadoras del CeTAD, el tiempo total de las comunicaciones en cada máquina es en promedio 21.2 segundos. Esto representa un ancho de banda real entre procesos de un poco más de 6 Mb/s. Por lo tanto se tiene menos de 40% de penalización respecto del máximo absoluto del hardware en la comunicación de los mensajes broadcast entre procesos.

Hasta aquí se ha simplificado la métrica de tiempo de los mensajes al asumir que el tiempo dedicado a la transferencia de los datos entre las computadoras es igual al tiempo de espera por la comunicación de los datos en cada computadora. Sin embargo, es necesario recordar que el tiempo de comunicaciones del algoritmo en cada computadora en realidad es el tiempo durante el cual se debe esperar a que se complete un mensaje broadcast. Esto

significa que todo desbalance de carga de procesamiento genera que en algunas computadoras se espera más o menos por los datos. En la Tabla 4.18, por ejemplo, se puede notar que en general se cumple que las computadoras con mayor tiempo dedicado a cómputo, **fourier** y **prited**, tienen menor tiempo de comunicaciones, porque en realidad una parte del tiempo de comunicaciones que se contabiliza en las demás computadoras es este tiempo "extra" de cómputo que las computadoras utilizan para procesamiento de los datos de las matrices. Dado que el desbalance de carga en términos de tiempo de cómputo no llega a ser del 10% este tiempo al menos por ahora no se tiene en cuenta para discriminarlo como diferente del tiempo de las comunicaciones.

Además de considerar el rendimiento de las comunicaciones, evidentemente también se puede considerar la evaluación de dos aspectos que tienen una gran influencia en el rendimiento paralelo: el rendimiento de cómputo de cada una de las máquinas en particular y el balance de carga *real* de la máquina paralela.

Rendimiento de cómputo local-secuencial de SeqMsg(UDP). Desde el punto de vista del rendimiento de cómputo de cada máquina se tendría que analizar cuál es la penalización con respecto a la máxima capacidad de procesamiento por utilizar las computadoras en paralelo. En este sentido, quizás el factor más importante a identificar o cuantificar es la "interferencia" de las comunicaciones sobre el rendimiento de cómputo secuencial de cada computadora. Expresado de otra manera, desde el punto de vista del rendimiento no es lo mismo *solamente* computar-procesar datos que llevar a cabo períodos de cómputo *y* períodos de comunicaciones. Es claro que la memoria cache, por ejemplo, debe ser compartida o utilizada en todos los períodos, y por lo tanto no es lo mismo hacer toda una multiplicación de matrices que procesar un tercio, comunicar datos, procesar el segundo tercio, comunicar datos y procesar el último tercio.

Cuando se tienen máquinas heterogéneas, el impacto de las comunicaciones sobre el rendimiento de cómputo no necesariamente será el mismo. Siguiendo con el ejemplo de la utilización de la memoria cache, en principio el impacto en el rendimiento de cómputo puede ser distinto dependiendo del tamaño de memoria cache. En el Apéndice A se puede comprobar que las máquinas del CeTAD tienen una gran *variedad* de tamaños y tipos de memoria cache.

Se puede evaluar el rendimiento de cómputo local de cada computadora utilizando los valores que resumen la ejecución paralela para cinco y diez máquinas dados en la Tabla 4.17 y en la Tabla 4.18 respectivamente. La Tabla 4.19 muestra los Mflop/s de cada máquina cuando se considera la paralelización de una multiplicación de matrices de 2000×2000 elementos en cinco computadoras con el algoritmo SeqMsg(UDP). La Tabla 4.20 muestra los Mflop/s cuando se utilizan todas las computadoras. Es importante notar que:

- El rendimiento en cada computadora permanece relativamente invariante a medida que se agregan máquinas y por lo tanto a medida que se disminuye la granularidad. Se debe recordar que aumentar la granularidad implica aumentar la cantidad de mensajes para este algoritmo y por lo tanto se tiene mayor cantidad de períodos de cómputo en los cuales se llevan a cabo menor cantidad de operaciones.
- El rendimiento de cada computadora es bastante cercano al óptimo, tomando como el óptimo el que se tiene cuando *solamente* se ejecutan operaciones de cálculo

secuenciales. En este sentido, se tiene como referencia la experimentación realizada para evaluar la capacidad de cómputo relativa y que para el caso de las máquinas del CeTAD se muestra en la Figura 4.4 y en la Figura 4.5 para **purmamarca** en particular.

Nombre	***Filas***	***Tot. Cómp.***	***Mflop/s***
purmamarca	555	14.15	314
cf1	426	14.14	241
cf2	426	14.09	242
sofia	394	14.40	219
fourier	199	14.78	108

Tabla 4.19: Mflop/s de SeqMsg(UDP) con Cinco Máquinas y *n* = 2000 en el CeTAD.

Nombre	***Filas***	***Tot. Cómp.***	***Mflop/s***
purmamarca	454	11.81	307
cf1	349	11.84	236
cf2	349	11.82	236
sofia	324	11.97	216
fourier	164	12.67	104
Josrap	142	12.10	94
tilcara	104	12.08	69
paris	48	11.59	33
cetad	38	11.98	25
prited	28	12.65	18

Tabla 4.20: Mflop/s de SeqMsg(UDP) con Diez Máquinas y *n* = 2000 en el CeTAD.

Balance de carga de SeqMsg(UDP). Como se puede calcular a partir de los valores que resumen los tiempos de ejecución paralelos en las diferentes tablas, aparecen algunas diferencias en cuanto a tiempo dedicado a cómputo local que debería explicarse de acuerdo con el balance de carga del definido para el algoritmo. Las diferencias de tiempo de cómputo local que se pueden identificar son:

- En la Tabla 4.19 el menor tiempo de cómputo es el de **cf1**, 14.09 segundos, y el mayor tiempo de cómputo es el de **fourier**, 14.78 segundos. El porcentaje de tiempo que implica este desbalance es menor al 5% del menor tiempo de cálculo.
- En la Tabla 4.20 el menor tiempo de cómputo es el de **paris**, 11.59 segundos, y el mayor tiempo de cómputo es el de **fourier**, 12.67 segundos. El porcentaje de tiempo que implica este desbalance es menor al 10% del menor tiempo de cálculo.

Es claro que aunque desde el punto de vista teórico se puede llegar a una ecuación o expresión que permita un balance de carga exacto, pero que se encuentren diferencias en

tiempo de ejecución. Solamente a modo de ejemplo, las computadoras **cf1** y **cf2** son *iguales* (Tabla 4.1 y Apéndice A), y tanto en la Tabla 4.19 como en la Tabla 4.20 se puede comprobar que las computadoras tienen asignada la misma carga de procesamiento, pero sin embargo tienen distintos tiempos de cómputo local. En general, independientemente del tipo de computadoras, algoritmos y formas de balancear la carga siempre se tendrá un mínimo de diferencias en cuanto a tiempos de ejecución.

Otra fuente de un mínimo desbalance de carga está dada por la definición misma del algoritmo. Tal como se explica en el capítulo anterior, el balance de carga se implementa asignando la cantidad de filas de la matriz resultado que corresponden a la capacidad de cálculo relativa de cada computadora. Por lo tanto, la cantidad de datos a calcular en cada computadora está dada en *cantidad de filas* de la matriz resultado. Sin embargo, no siempre la capacidad de cálculo relativa de las computadoras se pueden expresar como múltiplos unas de otras (o múltiplos de una de referencia) y además no siempre la cantidad total de filas podrá ser distribuida completamente siguiendo esta estrategia. Por lo tanto, el factor de corrección “de una fila” en la distribución de los datos que se explica en el capítulo anterior se puede mencionar como causante de un posible desbalance.

Por último, aunque no necesariamente menos importante otra posible fuente para el desbalance de carga es el que implica la propia heterogeneidad de las computadoras. Aunque cada computadora se mantiene en cuanto a capacidad de cálculo muy cercana al óptimo, no necesariamente esta “cercanía” será igual en todos los casos. Expresado de otra manera, todas las computadoras tienen un porcentaje mínimo de penalización de rendimiento en cuanto a cálculo secuencial cuando se hace cómputo paralelo (por código y *otros* procesos en la computadora necesarios para comunicaciones) y este porcentaje puede variar dependiendo de las características físicas de las computadoras. La variación en este porcentaje de penalización también genera cierto desbalance.

A pesar de que todas las fuentes de desbalance de carga son acumulativas, y a pesar de la gran heterogeneidad en cuanto a capacidad de cálculo de las computadoras, lo peor que se tiene en la experimentación en cuanto a desbalance de carga es menor al 10% del mejor tiempo de cómputo local.

OverMsg(UDP). Todo el detalle que se ha hecho en cuanto al análisis de resultados corresponde al algoritmo con los períodos de cómputo y comunicación secuenciales en cada una de las máquinas. Lo que sucede con el algoritmo que está orientado al solapamiento de las comunicaciones con el cómputo local es diferente y en la Figura 4.37 es evidente que algunos valores son bastante diferentes de los que se esperan. Para poder hacer una comparación más completa de los resultados obtenidos por los dos algoritmos y además identificar mejor las diferencias con la implementación basada en PVM se muestra primero el resumen de la ejecución paralela con cinco máquinas en la Tabla 4.21. Comparando estos valores con los de la Tabla 4.6 correspondientes a la implementación utilizando el broadcast de PVM, las conclusiones son similares a las que resultaron de la comparación de las implementaciones del algoritmo SeqMsg, es decir SeqMsg(PVM) con SeqMsg(UDP):

- Los tiempos de cómputo son casi los mismos, es decir que en términos de rendimiento de cómputo local cada computadora tiene casi la misma capacidad de cálculo (en Mflop/s, por ejemplo).

- Los tiempos de comunicaciones son muy diferentes, en este caso el rendimiento del broadcast utilizando UDP es aproximadamente diez veces mejor que el de PVM.

Nombre	*Filas*	*Tot. Cómp.*	*Por It.*	*Tot. Msg.*
purmamarca	555	16.63	3.33	5.96
cf1	426	17.30	3.46	5.45
cf2	426	17.31	3.46	5.42
sofia	394	18.29	3.66	4.33
fourier	199	16.53	3.31	5.98

Tabla 4.21: Resumen de OverMsg(UDP) con Seis Máquinas y n = 2000 en el CeTAD.

Cuando se comparan los valores de la Tabla 4.21 con los de la Tabla 4.17 se puede notar que:

- Los tiempos de cómputo local de OverMsg(UDP) son mayores que los de SeqMsg(UDP), lo cual es esperable ya que no solamente se llevan a cabo transferencias de datos entre computadoras sino que además en un mismo intervalo de tiempo los procesos de cómputo en cada computadora se ejecutan con los procesos de comunicaciones y por lo tanto la competencia por la CPU y la memoria cache, por ejemplo, es mayor y eso se traduce en mayor tiempo de cómputo.
- Los tiempos de comunicaciones de OverMsg(UDP) son aproximadamente un tercio de los obtenidos con SeqMsg(UDP). Dado que en realidad la misma cantidad de datos se transfieren entre las computadoras, gran parte de cada transferencia de datos transcurre *mientras* se está resolviendo un cálculo parcial de la matriz resultado en cada computadora. En este sentido, al menos para cinco computadoras, el solapamiento de las comunicaciones puede considerarse satisfactorio.

Sin embargo, como se puede notar claramente en la Figura 4.37, a partir de la inclusión de **tilcara** el rendimiento no solamente es menor que el esperado sino que además disminuye notablemente. Aunque se puede comprobar con la inclusión de cada una de las máquinas a partir de **tilcara** (**paris**, **cetad** y **prited**), lo que sucede se puede identificar de forma bastante clara directamente con el resumen de la ejecución de todas las máquinas que se muestra en la Tabla 4.22.

Tal como lo muestra la Tabla 4.22, el tiempo de cómputo local de las seis computadoras con mayor capacidad de cálculo del CeTAD (**purmamarca**, **cetadfomec1**, **cetadfomec2**, **sofia**, **fourier** y **Josrap**) es bastante similar entre sí, aproximadamente 13±0.45 segundos. El tiempo de cómputo de las cuatro computadoras restantes es bastante mayor y con mayor disparidad entre sí, aproximadamente 20±5 segundos. La explicación de lo que sucede es bastante sencilla y tiene relación con la competencia por los recursos mencionada antes. En este algoritmo que lleva a cabo las comunicaciones de manera solapada con el cómputo local, es claro que deben existir uno o más procesos en cada computadora que se ejecutan (transfieren datos) mientras se resuelve un período de cálculos parciales. Esto significa que durante la ejecución de un período de cálculo los recursos más importantes (CPU y memoria cache) serán compartidos con el o los procesos de comunicaciones. Esto como

mínimo genera una degradación del rendimiento de cálculo, pero además, según los valores de la Tabla 4.22 algunas computadoras pueden solapar cómputo y comunicaciones mejor que otras.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	454	12.39	1.24	21.08
cf1	349	12.99	1.30	20.48
cf2	349	13.01	1.30	20.46
sofia	324	13.45	1.34	20.07
fourier	164	13.28	1.33	20.18
Josrap	142	12.82	1.28	20.80
tilcara	104	18.34	1.83	15.08
paris	48	16.91	1.69	16.39
cetad	38	23.26	2.33	9.90
prited	28	24.81	2.48	7.91

Tabla 4.22: Resumen de OverMsg(UDP) con Diez Máquinas y n = 2000 en el CeTAD.

Por lo tanto se llega a que las diferencias en el rendimiento de cómputo solapado con comunicaciones dependen de varios factores, como la arquitectura del bus del sistema o del método de entrada/salida con el cual los datos se comunican a través de las placas de interfase de red instaladas. Si bien es muy difícil explicar más detalladamente lo que sucede en cada computadora, esto significa que algunas computadoras en realidad no son capaces de solapar *realmente* cómputo local con comunicaciones sino que secuencializan el cómputo con las comunicaciones y por lo tanto su rendimiento es el mismo que el que tienen con el algoritmo de cómputo y comunicaciones secuenciales. Incluso puede ser peor porque con OverMsg se tiene mayor sobrecarga (overhead) a nivel de procesos que se ejecutan y que el sistema operativo tiene que manejar (generando mayor cantidad de cambios de contexto, por ejemplo).

Si bien desde el punto de vista del rendimiento total esta explicación no es satisfactoria (de hecho, impone un límite físico dado que depende del hardware de la propia computadora) el propio algoritmo con cómputo solapado con las comunicaciones se *transforma* en un *benchmark*. Expresado de otra manera, OverMsg(UDP) permite de manera automática e independiente de las computadoras conocer con bastante precisión si son capaces de solapar cómputo con comunicaciones o no. De hecho, en este caso hasta se podría dar una cuantificación aproximada que puede estar basada en las diferencias de tiempo empleado para cómputo en cada computadora. Este benchmark toma mayor relevancia cuando se tiene en cuenta que se están utilizando computadoras de cómputo secuencial estándares, donde la posibilidad de solapar cómputo con comunicaciones vía una o más interfases de red no parece estar entre los objetivos prioritarios del diseño.

Aunque se tiene una explicación aproximada a la pérdida de rendimiento de cómputo paralelo a partir de la inclusión de **tilcara**, parece razonable analizar con un poco más de

detalle lo que sucede con el rendimiento de cómputo de cada una de las máquinas en particular y el balance de carga *real* de la máquina paralela.

Rendimiento de cómputo local-secuencial de OverMsg(UDP). Como era de esperar, y por las razones que ya se expusieron, el rendimiento de cálculo de las computadoras disminuye respecto de la situación óptima, es decir cuando solamente se realiza cómputo local. Dado que las máquinas son heterogéneas, la disminución también es *heterogénea*, dependiente de la arquitectura de cada computadora. Dejando de lado las máquinas que secuencializan el cómputo y las comunicaciones (a partir de **tilcara**), puede decirse que el costo de solapar las comunicaciones con el cómputo local es menor que el beneficio, dado que de hecho, el rendimiento total aumenta.

Balance de carga de OverMsg(UDP). Dado que el balance de carga se hace utilizando el rendimiento de cómputo secuencial y por la interferencia *heterogénea* de los procesos de transferencia de datos sobre los de cómputo, el balance de carga de OverMsg(UDP) puede llegar a ser peor que el de SeqMsg(UDP). Quitando del análisis del balance de carga real a las computadoras que no pueden solapar las comunicaciones con cómputo local de manera "aceptable" desde el punto de vista del rendimiento, el desbalance no llega a ser del 10% en el caso de las computadoras del CeTAD. Sin embargo, cuando este desbalance llegue a ser significativo se puede utilizar la información del resumen de ejecución para redefinir la distribución de los datos y distribuir los datos en función del rendimiento de cómputo asumiendo solapamiento de comunicaciones.

4.8.1.2 Matrices de 3200×3200 Elementos

Cuando se toma como referencia el tiempo de ejecución del mayor tamaño de problema que se puede resolver en la computadora de mayor capacidad de cómputo, es decir matrices de orden n = 3200 en **purmamarca**, los valores de speedup obtenidos se muestran en la Figura 4.38 junto con los que se mostraron Figura 4.18.

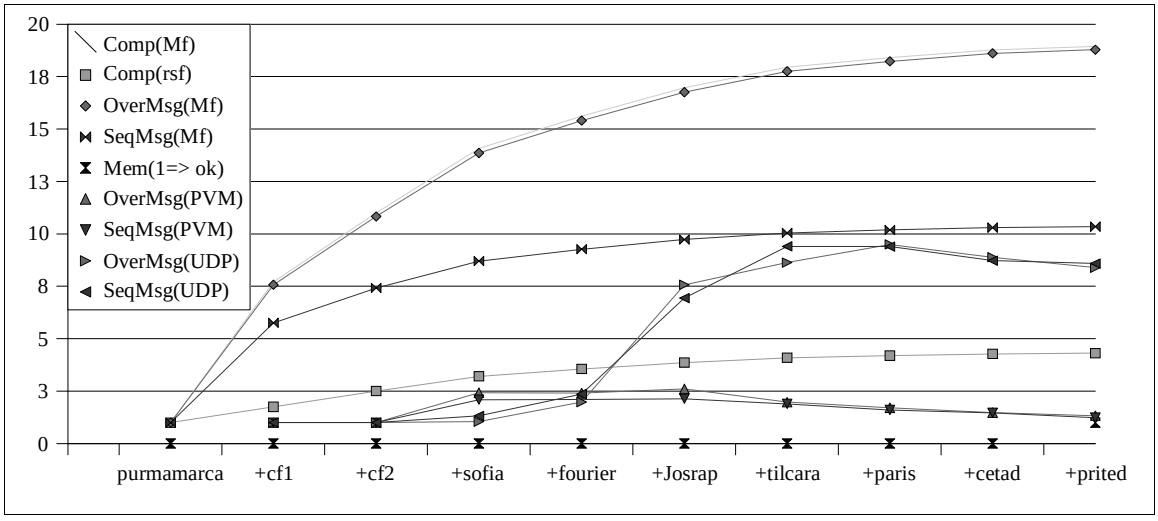


Figura 4.38: Speedup de los Algoritmos con UDP en la Red del CeTAD para n = 3200.

Como se puede notar en la Figura 4.38, en ningún caso se supera la estimación de SeqMsg(Mf) que es el rendimiento óptimo teniendo en cuenta solamente el aporte de cómputo de cada una de las computadoras sin tener en cuenta ningún tiempo relacionado con las comunicaciones.

El muy bajo rendimiento obtenido hasta la inclusión de **fourier** está directamente relacionado con los requerimientos de memoria del algoritmo en cada computadora. Específicamente, **cetadfomec1**, **cetadfomec2** y **fourier** tienen solamente 32 MB de memoria instalada y por lo tanto son las que tienen mayor probabilidad de que se utilice espacio de memoria swap durante los cálculos parciales de la matriz resultado.

Los valores que resumen la ejecución paralela que se muestran en la Tabla 4.23 aclaran que **fourier** es la computadora que requiere mucho tiempo más que las demás para resolver los cálculos locales.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	887	56.94	11.39	334.81
cf1	682	78.11	15.62	316.04
cf2	682	79.75	15.95	314.29
sofia	631	57.31	11.46	334.65
fourier	318	193.88	38.78	224.60

Tabla 4.23: Resumen de OverMsg(UDP) con Cinco Máquinas y n = 3200 en el CeTAD.

Los valores de la Tabla 4.23 también muestran que una gran cantidad de tiempo de ejecución en todas las máquinas está dedicado a la espera de la transferencia de los mensajes broadcast y en este caso también este tiempo puede estar afectado por la utilización del espacio de swap durante la ejecución.

De alguna manera, la estimación de memoria, Mem, en la Figura 4.38, está indicando que recién con todas las computadoras es probable que no haya problemas de memoria. La necesidad de recurrir a la memoria swap genera que

- Tanto los procesos de cómputo como los de comunicaciones pueden tener parte de su código ejecutable en memoria secundaria y por lo tanto pueden ser suspendidos en algún momento de su ejecución para recuperar el código.
- Tanto los procesos de cómputo como los de comunicaciones pueden tener parte de sus datos en memoria secundaria y por lo tanto pueden ser suspendidos en algún momento de su ejecución para recuperar esa información.
- Para todos los procesos el tiempo de swap genera un retraso en cuanto a tiempo de ejecución, pero en el caso de los procesos de comunicaciones la penalización puede ser mayor dado que pueden llegar a perder datos que se les están enviando durante el tiempo de swap.

Según la Figura 4.38, a partir de la inclusión de **Josrap**, el rendimiento es mucho mayor, aunque no tiene mayores variaciones con la inclusión de las demás computadoras. A partir de la inclusión de **Josrap** no se utiliza la memoria swap o la utilización no penaliza

significativamente la ejecución de los procesos de cómputo y de comunicaciones. La Tabla 4.24 muestra el resumen de la ejecución paralela con siete computadoras. Se tiene un gran avance en rendimiento de cálculo así como en el rendimiento de las comunicaciones.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
purmamarca	771	56.16	8.02	45.35
cf1	593	77.28	11.04	25.67
cf2	593	78.11	11.16	25.37
sofia	549	59.70	8.53	42.75
fourier	276	58.26	8.32	44.41
Josrap	242	52.93	7.56	47.07
tilcara	176	73.54	10.51	29.25

Tabla 4.24: Resumen de OverMsg(UDP) con Siete Máquinas y n = 3200 en el CeTAD.

En este caso particular, el tiempo total de cómputo está "dominado" por el tiempo de cómputo local de la computadora **cetadfomec2**, que sigue utilizando el espacio de memoria swap. Dado que los requerimientos de memoria sobre **fourier** se han disminuido respecto de la utilización de cinco computadoras, su tiempo de ejecución dedicado a cómputo ha disminuido notablemente respecto del que se muestra en la Tabla 4.23.

Por lo tanto, cuando deja de tener influencia la utilización de memoria swap en una o más máquinas en el tiempo total de cómputo paralelo se comienza a tener el rendimiento cercano al esperado del algoritmo que intercala períodos de cómputo con períodos de comunicación. En el caso específico de SeqMsg(UDP) esto es así hasta que el tiempo de comunicaciones se vuelve muy relevante o domina el tiempo total, lo cual está dando una idea de que la granularidad del problema no es adecuada para más allá de ocho máquinas. Aún así no se pierde demasiado en rendimiento. En el caso específico de OverMsg(UDP), una vez que se pueden aprovechar las máquinas a su capacidad máxima (o a un porcentaje relativamente alto de la capacidad máxima), se comienzan a utilizar las computadoras que no pueden solapar cómputo con comunicaciones y por lo tanto se tiene el mismo rendimiento que con SeqMsg(UDP).

Si se toma como referencia la estimación de memoria para establecer la cantidad de computadoras (en una elección *a priori*), Mem en los gráficos de speedup, se deberían utilizar todas las máquinas. El rendimiento obtenido para la multiplicación de matrices de 3200×3200 elementos utilizando las diez máquinas del CeTAD es aproximadamente nueve veces el rendimiento de **purmamarca** para el mismo tamaño de memoria. Este rendimiento se podría considerar "superlineal" dado que la suma de las potencias de cálculo relativas de todas las máquinas del CeTAD es menos de cinco veces la capacidad de cálculo de **purmamarca**. Una vez más, se debe recordar que el tiempo de referencia para este cálculo de speedup es el tiempo de ejecución "real" de multiplicación para matrices de orden n = 3200 que en **purmamarca** implica la utilización de memoria swap y la penalización es muy grande con respecto a la máxima capacidad de procesamiento de esta computadora (sin utilización de espacio de memoria swap).

4.8.1.3 Conclusiones Generales de la Experimentación en CeTAD

En un primer análisis, desde el punto de vista del rendimiento y de la escalabilidad *global* de los algoritmos, los resultados que se obtienen en la red local del CeTAD no parecen ser muy alentadores, ya que tanto SeqMsg(UDP) como OverMsg(UDP) a partir de una determinada cantidad de computadoras disminuyen en vez de aumentar (Figura 4.37 y Figura 4.38). Sin embargo, con la experimentación realizada se puede obtener información muy importante, además de la que se refiere estrictamente a rendimiento y escalabilidad global :

- El algoritmo OverMsg(UDP) se puede utilizar como *benchmark* para identificar cuáles computadoras son capaces de solapar efectivamente cómputo con transferencias de datos.
- En general, la estimación de speedup óptimo no es la que finalmente se obtiene por la sobrecarga (overhead) que imponen las placas de red, el sistema operativo, etc., pero conociendo las computadoras que son capaces de solapar cómputo con comunicaciones será posible definir la cantidad de computadoras a utilizar para un problema dado. Más específicamente: *cuáles*.
- La estimación de requerimientos de memoria, aunque no exacta, puede ser útil cuando es probable que algunas o todas las computadoras tengan problemas en cuanto a la utilización del espacio de memoria swap. Cuando se tiene esta información será posible elegir qué algoritmo utilizar (SeqMsg o OverMsg) dependiendo de si las computadoras a incluir en el cómputo paralelo pueden o no solapar efectivamente cómputo con comunicaciones.

4.8.2 Red Local del LQT

La mayoría de las cosas que se explicaron en el contexto de la experimentación en la red local del CeTAD son aplicables (quizás en otra escala) a la experimentación en la red local del LQT.

La Figura 4.39 muestra los valores de speedup obtenidos en la red local del LQT por los algoritmos implementados utilizando UDP, SeqMsg(UDP) y OverMsg(UDP), para matrices de orden $n = 5000$ junto con los que se mostraron anteriormente en la Figura 4.19. La estimación de speedup para el algoritmo de cómputo y comunicaciones secuenciales es casi exacta respecto de lo que se obtiene en la experimentación, lo cual es muy satisfactorio. Además, confirma que el problema del bajo rendimiento de SeqMsg(PVM), es decir del algoritmo implementado utilizando la biblioteca PVM, es justamente el broadcast de PVM que no aprovecha las características de las redes Ethernet.

Es evidente también que el $O(n^3)$ en cuanto a requerimiento de operaciones de punto flotante de este problema es igual de influyente en el tiempo total de ejecución paralela que el $O(n^2)$ de la cantidad de datos a transferir por medio de los mensajes broadcast. Es por esta razón que tanto la estimación de speedup óptimo SeqMsg(Mf) como el speedup obtenido en la experimentación SeqMsg(UDP), es aproximadamente el 50% del speedup

óptimo independientemente del algoritmo utilizado y de la red de comunicaciones Comp(Mf).

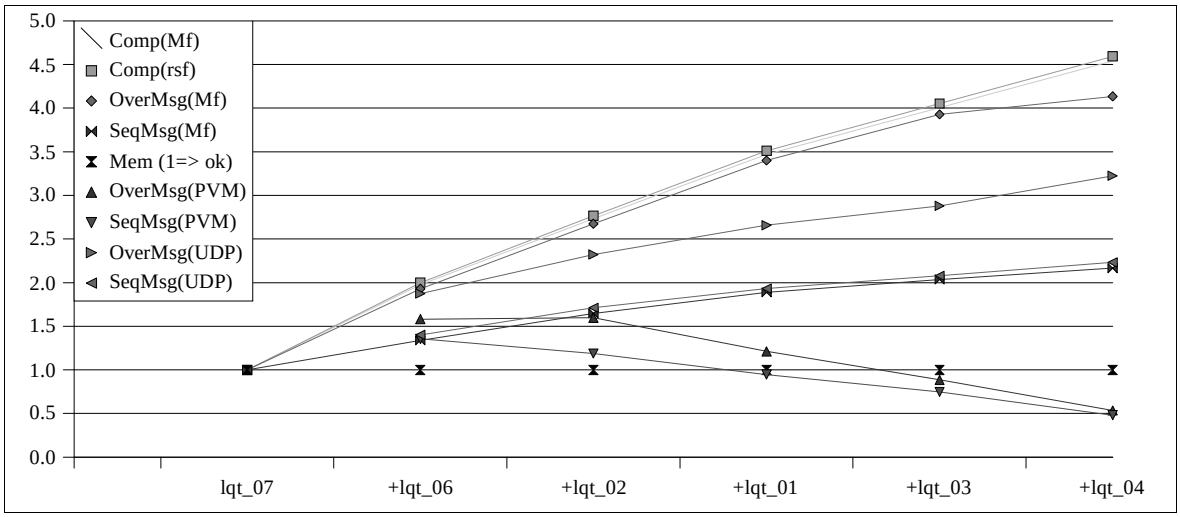


Figura 4.39: Speedup de los Algoritmos con UDP en la Red del LQT para n = 5000.

Aunque el algoritmo con cómputo y comunicaciones solapadas no llega al speedup óptimo calculado para este algoritmo, OverMsg(Mf), en la experimentación no solamente tiene mejor rendimiento que el algoritmo de cómputo y comunicaciones secuenciales sino que muestra al menos tres aspectos:

- La capacidad de cada una de las computadoras de solapar al menos en parte el cómputo local con las comunicaciones vía la red de interconexión. De hecho, tal como se puede visualizar en la Figura 4.39, el resultado final en términos de rendimiento tiene el efecto de solapar aproximadamente el 50% de las comunicaciones.
- La influencia de la sobrecarga que impone el sistema operativo entre otras capas de software no es despreciable, y de hecho es la que "consume" la diferencia entre OverMsg(Mf) y OverMsg(UDP).
- La granularidad de la aplicación hace que el speedup obtenido, OverMsg(UDP), se incremente a medida que se agregan mayor cantidad de computadoras. Expresado de otra manera, el tiempo de cómputo local es comparable con el tiempo de las comunicaciones y un buen porcentaje de las comunicaciones se puede realizar de manera solapada con el procesamiento.

Con respecto al solapamiento del cómputo con las comunicaciones, lo que sucede en realidad es lo mismo (o similar) que se explicó en la red del CeTAD. La Tabla 4.25 muestra el resumen de ejecución de SeqMsg(UDP) con todas las máquinas del LQT para resolver una multiplicación de matrices de orden n = 5000 en paralelo y la Tabla 4.26 muestra el resumen de ejecución de OverMsg(UDP).

De la comparación de los tiempos de ejecución que se muestran en cada tabla se llega a que:

- Los períodos de cálculo en la ejecución de OverMsg(UDP) son un poco mayores que los de SeqMsg(UDP) por la competencia por los recursos con los procesos que se encargan de las comunicaciones "en background".

- En la ejecución de OverMsg(UDP) un gran porcentaje de las comunicaciones se lleva a cabo *durante* el tiempo de cómputo local. Mientras que cada computadora debe esperar en promedio aproximadamente 87±3 segundos para la transferencia de datos durante la ejecución de SeqMsg(UDP), la espera es en promedio aproximadamente 25±4 segundos durante la ejecución de OverMsg(UDP).

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
lqt_07	1089	85.58	14.26	89.66
lqt_06	1089	86.36	14.39	88.94
lqt_02	835	87.94	14.66	87.41
lqt_01	811	90.54	15.09	85.17
lqt_03	589	90.15	15.02	85.20
lqt_04	587	89.94	14.99	85.36

Tabla 4.25: Resumen de SeqMsg(UDP) con Seis Máquinas y n = 5000 en el LQT.

Nombre	***Filas***	***Tot. Cómp.***	***Por It.***	***Tot. Msg.***
lqt_07	1089	91.40	15.23	28.69
lqt_06	1089	91.11	15.19	28.95
lqt_02	835	99.53	16.59	20.59
lqt_01	811	95.30	15.88	24.38
lqt_03	589	96.68	16.11	22.80
lqt_04	587	96.43	16.07	22.89

Tabla 4.26: Resumen de OverMsg(UDP) con Seis Máquinas y n = 5000 en el LQT.

La Figura 4.40 muestra los valores de speedup obtenidos en la red local del LQT por los algoritmos implementados utilizando UDP, SeqMsg(UDP) y OverMsg(UDP), para matrices de orden n = 9000 junto con los que se mostraron anteriormente en la Figura 4.20 (experimento que le *corresponde*).

Una vez más, la estimación del rendimiento óptimo del algoritmo de cómputo y comunicaciones secuenciales, SeqMsg(Mf), es casi exactamente la que se obtiene en la experimentación, que es SeqMsg(UDP). En el caso de la estimación del speedup óptimo para el algoritmo de cómputo y comunicaciones solapadas, OverMsg(Mf) es muy cercano al que se obtiene, OverMsg(UDP). La diferencia relativa entre los valores es bastante menor a la que se muestra en la Figura 4.39 y esto está dado por la influencia que tiene el $O(n^3)$ de requerimientos de cómputo sobre el $O(n^2)$ del requerimiento de comunicaciones. Y evidentemente esa diferencia hace que el porcentaje de tiempo de cómputo sea bastante superior al de las comunicaciones para matrices de 9000×9000 elementos que para matrices de 5000×50000 elementos.

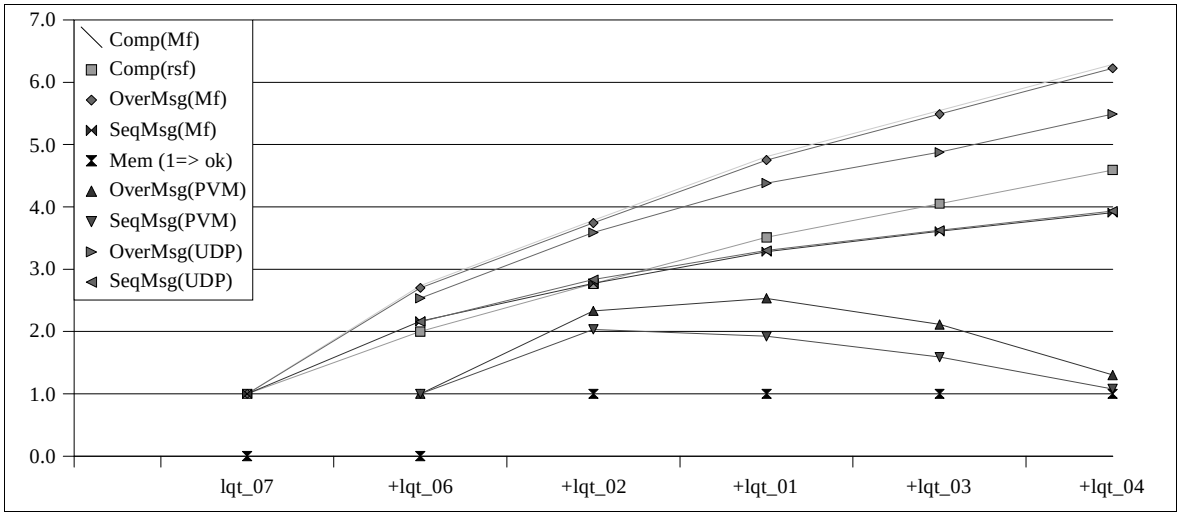


Figura 4.40: Speedup de los Algoritmos con UDP en la Red del LQT para n = 9000.

El rendimiento obtenido para la multiplicación de matrices de 9000×9000 elementos utilizando las seis máquinas del LQT es aproximadamente 5.5 veces el rendimiento de **lqt_07** para el mismo tamaño de memoria. Este rendimiento se podría considerar "superlineal" dado que la suma de las potencias de cálculo relativas de todas las máquinas del LQT es aproximadamente 4.5 veces la capacidad de cálculo de **lqt_07**. Una vez más, se debe recordar que el tiempo de referencia para este cálculo de speedup es el tiempo de ejecución "real" para matrices de orden n = 9000 que en **lqt_07** implica la utilización de memoria swap.

4.8.3 Red Local del LIDI

La Figura 4.41 muestra los valores de speedup obtenidos en la red local del LIDI por los algoritmos implementados utilizando UDP, SeqMsg(UDP) y OverMsg(UDP), para la multiplicación de matrices de orden n = 2000 junto con los que se mostraron anteriormente en la Figura 4.21.

Para comparar la estimación de los valores óptimos de speedup de los algoritmos, SeqMsg(Mf) y OverMsg(Mf) con los obtenidos, SeqMsg(UDP) y OverMsg(UDP) se debe recordar que los tiempos de comunicaciones se estiman en base a una red diez veces más rápida que las del CeTAD y del LQT. Esto a su vez genera que cualquier sobrecarga (overhead) sobre el tiempo de cómputo óptimo tendrá un impacto mayor en la red del LIDI que en las otras dos. Como conclusiones generales a partir de la Figura 4.41 se tienen:

- Tanto SeqMsg(UDP) como OverMsg(UDP) son menores a las estimaciones que les corresponden. El más alejado de la estimación es OverMsg(UDP).
- Los dos algoritmos tienen mejor rendimiento a medida que se aumenta la cantidad de computadoras. Es evidente que la mayor capacidad de la red de comunicaciones así como el rendimiento aceptable que se obtiene con los mensajes broadcast son factores importantes para que esto suceda.

- El algoritmo que está diseñado para solapar las comunicaciones con el cómputo local en cada una de las máquinas, OverMsg, es mejor que el de cómputo y comunicaciones secuenciales, SeqMsg. Esto a su vez indica que las computadoras son capaces de solapar efectivamente al menos en parte cómputo local con comunicaciones a través de la red de interconexión.

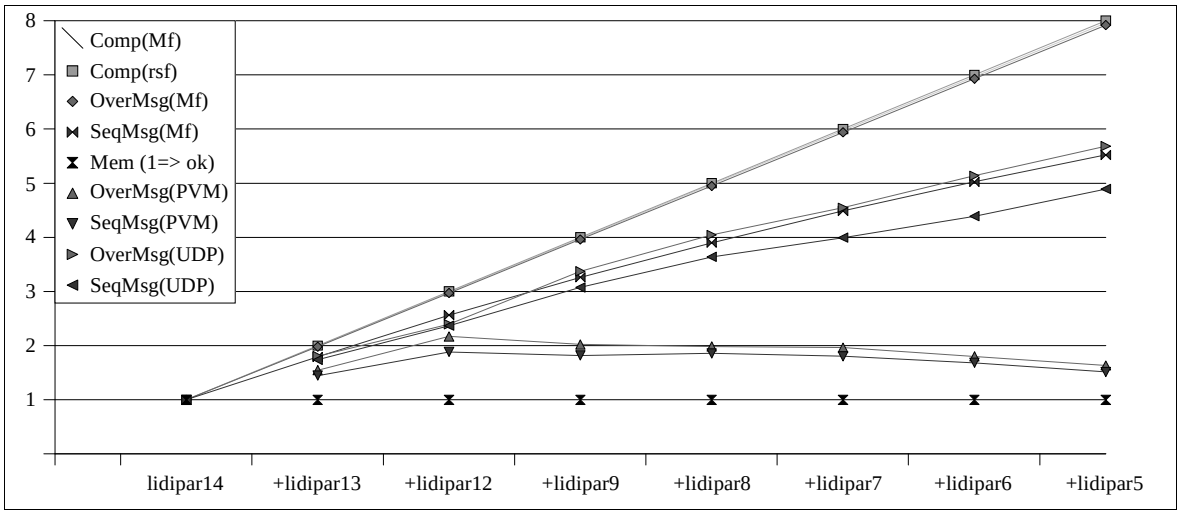


Figura 4.41: Speedup de los Algoritmos con UDP en la Red del LIDI para n = 2000.

La Figura 4.42 muestra los valores de speedup obtenidos en la red local del LIDI por los algoritmos implementados utilizando UDP, SeqMsg(UDP) y OverMsg(UDP), para la multiplicación de matrices de orden n = 3200 junto con los que se mostraron anteriormente en la Figura 4.22.

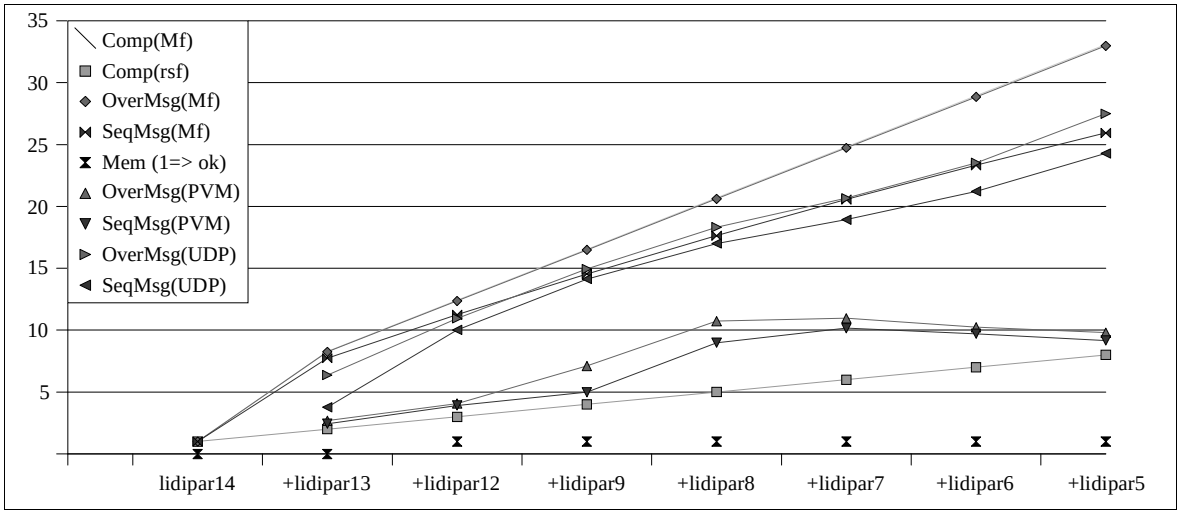


Figura 4.42: Speedup de los Algoritmos con UDP en la Red del LIDI para n = 3200.

Quizás éstos sean los mejores resultados en cuanto a rendimiento de todos los obtenidos. Tal como lo muestra la Figura 4.42, las ocho computadoras del LIDI pueden resolver una multiplicación de matrices de 3200×3200 elementos más de 25 veces más rápido que la

misma multiplicación en una de ellas (son todas iguales). De todas maneras, se debe recodar que el tiempo de referencia secuencial para la multiplicación de matrices de orden n = 3200 está penalizado con respecto al óptimo por la utilización de memoria swap durante los cálculos.

Una vez más, OverMsg(UDP) es mejor que SeqMsg(UDP) dado que las computadoras pueden efectivamente solapar comunicaciones y cómputo local y el algoritmo aprovecha también efectivamente esta capacidad. También se observa en la Figura 4.42 que siempre el rendimiento de los algoritmos mejora cuando se aumenta la cantidad de computadoras. Esto indica que, como era esperado, el algoritmo es escalable (al menos hasta ocho computadoras) y en particular la implementación de los mensajes broadcast entre procesos de usuario utilizando UDP también es escalable. A diferencia de lo que sucede con las matrices de orden n = 2000, para las matrices de orden n = 3200 los valores de speedup obtenidos son más cercanos a los estimados para los algoritmos, lo cual no hace nada más que confirmar la importancia del $O(n^3)$ de cómputo sobre el $O(n^2)$ de las comunicaciones que sobre la red local del LIDI se acentúa comparándola con las del CeTAD y del LQT.

4.9 Conclusiones-Resumen de la Experimentación

Una vez identificadas las características de las redes locales en cuanto a computadoras y redes de interconexión que se presentaron al principio del capítulo, se presentaron los resultados de la experimentación utilizando PVM como la biblioteca de pasaje de mensajes en general y de mensajes broadcast en particular. De esta experimentación surge que:

1. El rendimiento con PVM no es aceptable. En todas las redes locales, es decir independientemente de cantidad de máquinas, heterogeneidad u homogeneidad de las mismas, tamaños de matrices utilizados y rendimiento de la red de interconexión física los resultados fueron los mismos: el rendimiento empeora a medida que se utilizan mayor cantidad de computadoras.
2. Al hacer el perfil de ejecución y con un mínimo de instrumentación queda claro que el problema de rendimiento es siempre ocasionado por la rutina broadcast de la biblioteca PVM, que es implementada por múltiples mensajes punto a punto.

Se propone e implementa un mensaje broadcast basado directamente en el protocolo UDP, dado que en principio ninguna biblioteca de pasaje de mensajes de propósito general puede asegurar *a priori* rendimiento optimizado de los mensajes broadcast. Se repiten los mismos experimentos hechos anteriormente con PVM y se llega a que:

3. El algoritmo con períodos de cómputo y comunicaciones ejecutados de manera secuencial (SeqMsg) proporciona en casi todos los casos el rendimiento esperado. Las excepciones pueden darse en el caso de la utilización de memoria swap en algunas computadoras.
4. El algoritmo SeqMsg proporciona rendimiento que mejora a medida que se agregan computadoras en todas las redes, aún cuando el tamaño de las matrices no sea el mayor posible (o *escale* junto con la cantidad de computadoras).
5. El algoritmo organizado para solapar los períodos de cómputo y comunicaciones (OverMsg) tiene mejor rendimiento en todos los casos que SeqMsg, con lo cual el

rendimiento obtenido por este algoritmo puede considerarse aceptable en todas las redes locales utilizadas.

6. El Algoritmo OverMsg puede utilizarse de manera bastante sencilla como *benchamark* con dos propósitos:
   1. Identificar de manera sencilla la proporción de comunicaciones que puede ser llevada a cabo solapadamente (en *background*) mientras se realiza cómputo local en cada computadora.
   2. Las computadoras que no pueden ejecutar cómputo y comunicación de manera solapada y que por lo tanto penalizan el rendimiento paralelo de **toda** la red utilizada. En este sentido, OverMsg puede ser utilizado para descartar tales computadoras, o para dar un máximo de computadoras utilizables para una aplicación dada.
7. Tanto SeqMsg como OverMsg proporcionan speedup muy satisfactorio si la aplicación o, más específicamente, el tamaño de la aplicación hace que el rendimiento secuencial sea afectado por la memoria swap de la computadora en la cual se resuelve.
8. Tanto SeqMsg como OverMsg pueden utilizarse satisfactoriamente en clusters homogéneos como heterogéneos ya que en todos los casos obtienen rendimiento optimizado de
   1. Cómputo local de cada computadora y específicamente balanceado.
   2. Comunicaciones sobre la red de interconexión Ethernet con los mensajes broadcast.