# Projecte de recerca bàsica o aplicada
# PAC3 — Tercera Prova d'avaluació continuada

Cognoms: *Rodríguez Guerra*
Nom: *Juan José*

- Per a dubtes i aclariments sobre l'enunciat, adreceu-vos al consultor responsable de la vostra aula.
- Cal lliurar la solució en un fitxer OpenOffice.org, PDF o RTF fent servir una de les plantilles lliurades conjuntament amb aquest enunciat. Adjunteu el fitxer a un missatge adreçat a **l'espai d'avaluació de l'aula virtual**.
- El fitxer ha de tenir l'extensió *.odt* (OpenOffice.org), *.pdf* (PDF) o *.rtf* (RTF), segons el format en què feu el lliurament.
- La data límit de lliurament és el **20 de gener** (a les 24 hores).

## Respostes

L'article està a partir de la pàgina 9.

Quant a la revista o congrés en el qual es podria enviar l'article, sempre i quan fos d'interès suficient per a això, es troben el International Workshop on OpenMP 2011, en el qual s'exposen els avenços sobre OpenMP I les seves aplicacions.

Altra lloc on es podria enviar l'article és a SpringerLink, a IEEE Transactions on Parallel and Distributed Systems, o a la publicació Lecture Notes in Computer Science, entre d'altres.

Entre les persones a les quals enviaria el treball per a una revisió estarien els següents:

**Josep Jorba Esteve**, Enginyer Informàtic Superior per la UAB. Màster en Arquitectura i processament paral·lel per la UAB. Doctor Enginyer per la UAB. Professor d'Estudis d'Informàtica, Multimèdia I Telecomunicacions de la UOC. Entre les seves publicacions trobem:

Artícles de revistes:

- GUTIÉRREZ, G.; DARADOUMIS, A.; JORBA, J. (2007). " A Conceptual Model for Grid Learning Services Automatic Composition ". Lecture Notes in Computer Science. Núm. 4805. Pág. 40- 41. ISSN: 03029743
- JORBA, J.; MARGALEF, T.; LUQUE, E. (2007). " Search of Performance Inefficiencies in Message Passing Applications with KappaPI 2 tool ". Lecture Notes in Computer Science. Pág. 409- 419. ISSN: 03029743.
- GUTIÉRREZ, G.; DARADOUMIS, A.; JORBA, J. (2006). " Semantic Description of Grid Based Learning Services ". Lecture Notes in Computer Science. Núm. 4331. Pág. 509- 518. ISSN: 03029743.

Llibres i monografies:

- SUPPI, R.; JORBA, J.; MAS, J.; MEGÍAS, D. (2005). Proyecto de administración de redes y sistemas operativos basados en GNU/Linux . Editorial UOC .
- JORBA, J.; SUPPI, R. (2004). Administración avanzada de GNU/Linux . Editorial UOC . ISBN: 84-9788-116-8.

Capítols de llibres:

- JUAN, A. A.; FAULIN, J.; RIERA, D.; MASIP, D.; JORBA, J. (2009). " A Simulation-based Methodology to assist Decision-makers in real Vehicle Routing Problems ". En: CORDEIRO, J. Proceedings of the 11th Int. Conf. on Enterprise Information Systems (indexed by ISI Proceedings). INSTICC. Pág. 1- 12. ISBN: N/A.
- JUAN, A. A.; FAULÍN, F. J.; RIERA, D.; MASIP, D.; JORBA, J. (2009). " A Simulation-based Methodology to assist Decision-makers in real Vehicle Routing Problems ". En: FILIPE, J.; CORDEIRO, J. Proceedings of the 11th International Conference on Enterprise Information Systems. INSTICC. Pág. 212- 217. ISBN: 978-989-8111-84-5.
- LÁZARO, D.; MARQUÈS, J. M.; JORBA, J. (2008). " An architecture for decentralized service deployment ". Proceedings of the Second International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2008).
- JUAN, A. A.; FAULIN, J.; JORBA, J.; GRASMAN, S.; BARRIOS, B. (2008). " SR-2: A Hybrid Intelligent Algorithm for the Vehicle Routing Problem ". Proceedings of the 8th International Conference on Hybrid Intelligent Systems. Pág. 78- 83. ISBN: 978-0-7695-3326-1.
- GUTIÉRREZ, G.; DARADOUMIS, A.; JORBA, J. (2007). " Semantic Description and Matchmaking of Learning Grid Services ". En: PIERRI, A.; ORCIUOLI, F.; GAETA, M.; CAPUANO, N.; RITROVATO, P.; MIRANDA, S.; SALERNO, S. The Learning Grid Handbook . Pág. 37- 54. ISBN: 978-1-58603-829-8.
- LÁZARO, D.; MARQUÈS, J. M.; JORBA, J. (2007). " Decentralized Service Deployment for Collaborative Environments ". En: BAROLLI, L.; TJOA, M. The first Internationa Coference on Complex, Intelligent and Software Intensive Systems. Pág. 229- 234. ISBN: 0-7695-2823-6.

Documents Científics:

- JORBA, J.; MARGALEF, T.; LUQUE, E. (2005). " Performance Analysis of Parallel Applications with KappaPI 2 ".
- JORBA, J.; MARGALEF, T.; LUQUE, E. (2005). " Automatic Performance Analysis of Message Passing Applications Using the KappaPI 2 tool ".

Contribucions a congresos:

- JUAN, A. A.; MASIP, D.; RIERA, D.; FAULIN, J.; JORBA, J. (2009). " A Simulation-based Methodology to assist Decision-makers in real Vehicle Routing Problems ". En: 11th Int. Conf. on Enterprise Information Systems. INSTICC. Milan, 6 mayo.
- GUTIÉRREZ, G.; DARADOUMIS, A.; JORBA, J. (2009). " Discovering Social Relationships and Intentions in Web Forums Using Natural Language Processing Techniques ". En: International Conference WebScience'09: Society On-Line. Athens, 18 marzo.
- LÁZARO, D.; MARQUÈS, J. M.; JORBA, J. (2009). " A Best-effort Mechanism for Service Deployment in Contributory Computer Systems ". En: International Conference on Complex, Intelligent and Software Intensive Systems. Fukuoka, 18 marzo.
- GUTIÉRREZ, G.; DARADOUMIS, A.; JORBA, J. (2008). " Learning grid: herramientas de aprendizaje basadas en tecnología grid y web semántica ". En: Congreso Nacional de Ingeniería y Arquitectura 08 (IA08). Morelia, Michoacán, México, 27 noviembre.
- GUTIÉRREZ, G.; DARADOUMIS, A.; JORBA, J. (2008). " Toward a semantic approach for automatic composition of Learning Grid Services ". En: International Workshop on Web/Grid Information and Services Discovery (WGISD-2008). IEEE Computational Intelligence Society. BARCELONA, 6 marzo.
- GUTIÉRREZ, G.; DARADOUMIS, A.; JORBA, J. (2007). " Exploring Semantic Description and Matching Technologies for Enhancing the Automatic Composition of Grid based Learning Services. ". En: 5th International Workshop on Ontologies and Semantic Web for E-Learning (SWEL'07 @ AIED'07). AIED Conference 2007. Los Angeles, 9 julio.
- LÁZARO, D.; MARQUÈS, J. M.; JORBA, J. (2007). " Decentralized Service Deployment for Collaborative Environments ". En: The first Internationa Coference on Complex, Intelligent and Software Intensive Systems. Secure Business Austria and DEXA Society. Viena, 10 abril.

Projectes:

- Encàrrecs Gestió Professorat
- Free Technology Academy (FTA)
- Red Nacional de e-Ciencia

- Computación de altas prestaciones y su aplicación a la ciencia e ingeniería computacional
- Red Temática para la Coordinación de Actividades Middleware en Grids (TIN2005-25849-E)
- Cyted Grid: Tecnología GRID como motor de desarrollo regional
- PCC-Grid: Portal de Ciencia Colaborativo en Tecnologías Grid
- Kaleidoscope Network of Excellence Computer Support for Collaborative Learning (CSCL)
- Kaleidoscope Network of Excellence Learning Grid SIG
- Procesamiento de altas prestaciones: arquitectura, entornos de desarrollo y aplicaciones. CICYT (TIN2004-03388), 2004-2007
- SPREAD (EGV1-CT-2001-00043)

**Eduard Ayguadé Parra**: Enginyer en Telecomunicacions per la UPC, 1986. PhD en Ciencies de la Computació per la UPC, 1989. Professor des del 1997 en la UPC, Departament d'Arquitectura de Computadors. Director Associat des del 2006 del Centre de Supercomputació de Barcelona (BSC – CNS), Departament de Ciències de la Computació.

Projectes Nacionals:

- Diseño de Arquitecturas Paralelas de Alta Velocidad a Bajo Coste (PA85-0314), Ministeri d'Educació d'Espanya, 1985-1989. Investigador.
- Diseño e Implementación de un Sistema Multiprocesador con Buses Multiplexados, Telefonica, 1985-1986. Investigador.
- Sistema Multiprocesador Fuertemente Acoplado, Institut Català de Tecnología, 1989-1991. Investigador.
- Arquitectura, Herramientas de Programación y Sistemas Operativos para Multiprocesadores (TIC 89-392), Ministeri d'Educació d'Espanya, 1990-1992. Investigador.
- Arquitectura y Compilación para Supercomputadores (TIC 92-880), Ministeri d'Educació d'Espanya, 1992-1995. Investigador i coordinador d'activitats de “Compiler for Parallel Architectures”.
- Acción Integrada Hispano-Britanica 1995-1996: Distribución Automática de Datos para Computadores de Altas Prestaciones (HB163-B), Ministeri d'Educació d'Espanya, 1995-1996. Coordinador de Projecte.
- Computación de Altas Prestaciones (TIC 95-429), Ministeri d'Educació d'Espanya, 1995-1998. Investigador i coordinador d'activitats de “Compiler for Parallel Architectures”.
- Computación Paralela para el Modelado de Problemas Industriales (TIC96-1630-CE), Ministeri d'Educació d'Espanya, 1996-1999. Coordinador de Projecte.
- Acción Integrada Hispano-Austríaca 1997: Técnicas Avanzadas para la Distribución de Datos (HU1996-10), Ministeri d'Educació d'Espanya, 1997. Coordinador de projecte.

- Explotación de Paralelismo y Multiprogramación (TIC97-1445-CE ), Ministeri d'Educació d'Espanya, 1997-2000. Coordinador de projecte.
- Acció Integrada ACI amb University of Illinois 1997: High Performance Compilers, Govern Català, 1997. Coordinador de projecte.
- Acció Integrada ACI amb University of California at Santa Cruz 1998: Multithreaded Architectures, Govern Català, 1998.Coordinador de projecte.
- Acció Integrada ACI amb University of Illinois 1998: High Performance Compilers for Multithreaded Architectures, Govenr Català, 1998. Coordinador de projecte.
- Computación de Altas Prestaciones II (TIC 98-511), Ministeri d'Educació d'Espanya 1998- 2001. Investigador i coordinador d'activitats de “Compiler for Parallel Architectures”.
- Computación de Altas Prestaciones III (TIC2001-0995), Ministeri d'Educació d'Espanya 2002-2004. Investigador I coordinador d'activitats de “Compiler for Parallel Architectures”.
- Computación de Altas Prestaciones IV (TIN2004-07739), Ministeri de Ciència I Tecnologia d'Espanya 2004-2007. Investigador i coordinador d'activitats de “Programming Models and Compilers” .
- Supercomputación y eCiencia Consolider Program (CSD2007-00050), Ministeri de Ciència i Innovació. 2007- 2012.
- Computación de Altas Prestaciones V (TIN2007-60625), Ministeri de Ciència i Tecnologia d'Espanya 2007-2012. Consolidador de projecte. Investigador i coordinador d'activitats de “Programming Models and Compilers”.

<u>Projectes Europeus:</u>

- Performance-Critical Applications of Parallel Architectures APPARC, Basic Research Action BRA 6634, European Comission ESPRIT III, 1992-1995. Researcher.
- Advanced Computer Training and Technology Transfer, ACT-UETP University Enterprises Training Partnership, European Comission COMETT 92/1/7414/Ca, 1992-1994. Researcher.
- Training in Advanced Computing Applications, Tools and Architectures, ACT-UETP University Enterprises Training Partnership, European Comission COMETT 94/1/8353/Ca, 1994-1995. Researcher.
- PARANDES - Parallel Software for Applications in Chilean Industry, European Comission ESPRIT ITDC’94, 1995-1997. Researcher.
- PARALIN - Parallel Computing Modelling for Industrial Problems, European Comission INCO-DC, 1996-1998. Researcher.
- NANOS - Effective Integration of Fine-grain Parallelism Exploitation and Multiprogramming, European Comission ESPRIT Long Term Research LTR 21907, 1996-1999. Researcher.

- INTONE – Innovative OpenMP Tools for Non-experts (IST-1999-20252), European Comisión (Information Society Technologies). 2000-2003. Project coordinator.
- POP – Performance Portability of OpenMP (IST-2001-33071) , European Comisión (Information Society Technologies). 2001-2004. Researcher.
- SARC – Scalable Computer Architecture. EU Commission. Information Society Technologies, IST- 2006-27648. January 2006 - December 2009. Researcher.
- ACOTES – Advanced Compiler Technologies for Embedded Streaming. EU Commission, Information Society Technologies, IST-2006-034869. June 2006 - May 2009. Researcher.
- VELOX – An Integrated Approach to Transactional Memory on Multi- and Many-core Computers.
- EU Commission, Information Society Technologies, IST-2007-216852. January 2008-December 2010.

Entre les seves moltes publicacions mencionaré les següents:

- Milos Milovanovic, Roger Ferrer, Vladimir Gajinov, Osman S. Unsal, Adrián Cristal, Eduard Ayguadé, Mateo Valero. Nebelung: Execution Environment for Transactional OpenMP.
- International Journal of Parallel Programming vol. 36 no. 3. pp. 326-346. ISSN: 0885-7458. Springer, 2008.
- Nikola Vujic, Marc Gonzalez, Xavier Martorell and Eduard Ayguadé. Automatic Pre-Fetch and Modulo Scheduling Transformations for the Cell BE Architecture. 21st Annual Workshop on Languages and Compilers for Parallel Computing (LCPC). University of Alberta. July 31- August 2, 2008. In Lecture Notes in Computer Sciences, vol. 5335/2008, Springer-Verlag, 2008.
- Ferad Zyulkyarov, Sanja Cvijic, Osman Unsal, Adrian Cristal, Eduard Ayguadé, Tim Harris and Mateo Valero. WormBench - A Configurable Workload for Evaluating Transactional Memory Systems. MEDEA -- MEmory performance: DEaling with Applications, systems and architecture (workshop held in conjunction with PACT-2008). Toronto, Canada. October 25-29, 2008.
- Marc Gonzalez, Nikola Vujic, Xavier Martorell, Eduard Ayguadé, Alexandre E. Eichenberger, Tong Chen, Zehra Sura, Tao Zhang, Kevin O'Brien, and Kathryn O'Brien. Hybrid Access-Specific Software Cache Techniques for the Cell BE Architecture. The 17th International Conference on Parallel Architectures and Compilation Techniques (PACT). Toronto, Canada. October 25-29, 2008.
- Roger Ferrer, Marc Gonzalez, Federico Silla, Xavier Martorell, Eduard Ayguadé. Evaluation of Memory Performance on the Cell BE with the SARC Programming Model. MEDEA – Memory performance: DEaling with Applications, systems and architecture (workshop held in conjunction with PACT-2008). Toronto, Canada. October 25-29, 2008.

- Eduard Ayguade, Nawal Copty, Alejandro Duran, Jay Hoeflinger, Yuan Lin, Federico Massaioli, Ernesto Su, Priya Unnikrishnan, Guansong Zhang. A Proposal for Task Parallelism in OpenMP.
- International Workshop on OpenMP (IWOMP 2007). Beijing, China. June 2007. In Lecture Notes in Computer Science, vol. 4935, pp. 1-12. Springer-Verlag, 2008.
- Eduard Ayguadé, Alejandro Duran, Jay Hoeflinger, Federico Massaioli and Xavier Teruel. An Experimental Evaluation of the New OpenMP Tasking Model. LCPC 2007: 20th International Workshop on Languages and Compilers for Parallel Computing. Urbana (IL), USA. October 2007. In Lecture Notes in Computer Science. Vol. 5234/2008, pp. 63-77. ISBN 0302-9743, Springer- Verlag, 2008.
- Eduard Ayguadé, Nawal Copty, Alejandro Duran, Jay Hoeflinger, Yuan Lin, Federico Massaioli, Xavier Teruel, Priya Unnikrishnan, Guansong Zhang. The Design of OpenMP Tasks. IEEE Transactions on Parallel and Distributed Systems, Jun 2008. IEEE Computer Society.
- Alejandro Duran, Julita Corbalán and Eduard Ayguadé. Evaluation of OpenMP Task scheduling strategies. International Workshop on OpenMP (IWOMP-2008). Purdue University, West Lafayette (IN, USA). May 12-14, 2008. In Lecture Notes in Computer Sciences, vol. 5004/2008, pp. 100-110. ISBN 978-3-540-79560-5, Springer-Verlag, 2008.
- Alejandro Duran, Josep M. Perez, Eduard Ayguadé, Rosa M. Badia and Jesus Labarta. Extending the OpenMP Tasking Model to Allow Dependent Tasks. International Workshop on OpenMP (IWOMP-2008). Purdue University, West Lafayette (IN, USA). May 12-14, 2008. In Lecture Notes in Computer Sciences, vol. 5004/2008, pp. 111-122. ISBN 978-3-540-79560-5, Springer-Verlag, 2008.
- Xavier Teruel, Priya Unnikrishnan, Xavier Martorell, Eduard Ayguadé, Raul Silvera, Guansong Zhang and Ettore Tiotto. OpenMP Tasks in IBM XL compilers. CASCON-2008, Toronto (Canada). October 27-30, 2008.

<u>Algunes de les seves participacions més recents en Conferències i revistes són:</u>

- MULTIPROG-2008 – First Workshop on Programmability Issues for Multi-Core Computers. Held in conjunction with the 3rd International Conference on High-Performance Embedded Architectures and Compilers (HiPEAC 2008). Goteborg, Sweden (1/2008). Organizing committee.
- CC 2008 – International Conference on Compiler Construction. Budapest. Hungary (4/2008). Program Committee.
- IWOMP 2008 – 4th International Workshop on OpenMP. Purdue University, West Lafayette, IN (USA) (12-14/5/2008). Publications Co-Chair and Program Committee.
- ICS’08 –22nd ACM International Conference on Supercomputing. Island of Kos, Greece (7- 12/6/2008). Workshops and Tutorials Chair.

- LCPC 2008 – 21st International Workshop on Languages and Compilers for Parallel Computing. Edmonton, Alberta (Canada) (31/7-2/8/2008). Program Committee.
- ICPP 2008 – International Conference on Parallel Processing. Portland (OR), USA (8-12/9/2008). Program Committee (Compilers and Languages Track)
- MULTIPROG-2009 –Second Workshop on Programmability Issues for Multi-Core Computers. Held in conjunction with the 4th International Conference on High-Performance Embedded Architectures and Compilers (HiPEAC 2009). Paphos, Cyprus, (25-28/1/2009). Organizing committee.
- HiPEAC 2009 – The 4th International Conference on High Performance and Embedded Architectures. Paphos, Cyprus, (25-28/1/2009). Program Committee.
- IWOMP-2009 – 5th International Workshop on OpenMP. Technische Universität Dresden (Germany) (3-5/6/2009). Program Committee.
- LCPC 2009 – 22nd International Workshop on Languages and Compilers for Parallel Computing. University of Delaware, Newark, Delaware (USA) (8-10/10/2009). Program Committee.
- OpenMP Implementation and Performance Issues, Eduard Ayguade (CEPBA), Mats Brorsson (KTH), Sven Karlsson (KTH), Xavier Martorell (CEPBA) and Marc Gonzalez (CEPBA). Tutorial at the Fourth European Workshop on OpenMP (EWOMP 2002). Rome (Italy). September 2002.
- Performance Analysis Tools and Techniques. Panel session organizer and chair. Fourth European Workshop on OpenMP (EWOMP 2002). Rome (Italy). September 2002.
- Research Exhibit “CEPBA - European Center for Parallelism of Barcelona”. IEEE/ACM Supercomputing’02, Baltimore (USA). November 2002.
- What are the necessary ingredients of scalable OpenMP programming?. Panel session member. European Workshop on OpenMP (EWOMP’2003). Aachen (Germany). September 2003.
- Experiences in Exploiting Nested Parallelism in OpenMP. Invited talk. Workshop on OpenMP: Experiences and Implementations (WOMPEI’2003). Tokyo (Japan). October 2003.
- Programación de Arquitecturas Paralelas. Invited talk. Summer School at the Universidad de Castilla-La Mancha. July 7-9, 2004. Albacete (Spain)
- Arquitecturas y Modelos de Programación para Multicore. Invited talk with Alex Ramirez. Jornadas de Paralelismo. September 17, 2008. Castellon de la Plana (Spain).

# Estudi de comparació de rendiment de Benchmark multicore i multithreading amb OpenMP 2.5 i OpenMP 3.0

Juan José Rodríguez Guerra

Universitat Oberta de Catalunya
Av. Tibidabo, núm. 39-43. 08035 Barcelona.
jrodriguezgue@uoc.edu

**Resum.** El rendiment en la computació és cada vegada més important en el desenvolupament científic, concretament en disciplines de la ciència on és necessària l'aplicació de càlcul intensiu. El processament paral·lel i els sistemes de memòria compartida juguen cada vegada més un paper molt important a l'hora d'aplicar aquests càlculs, estalviant temps i recursos a l'hora d'obtenir els resultats. En aquest article es presenta un estudi comparatiu del rendiment de dues computadores multicore i multithreading utilitzant el model de memòria compartida d'OpenMP 2.5 i el nou model de memòria compartida OpenMP 3.0. Per a això hem desenvolupat un benchmark basat en multiplicació de matrius parametritzable per tal d'avaluar el comportament dels dos models de memòria compartida, mitjançant la multiplicació clàssica, multiplicació per blocs i, finalment, la multiplicació per tasques (OpenMP 3.0).

## 1. Introducció.

L'evolució de les arquitectures de computadors s'ha anat orientant cap als processadors multicore degut al guany de rendiment, l'eficiència en l'ús dels recursos de computació, i l'estalvi d'energia que aquest paral·lelisme de maquinari suposa.
Concretament, la combinació d'arquitectures multicore amb la capacitat multithreading d'aquests cores ha suposat un avenç molt important en el camp de la computació, permetent un ús més eficient del temps de processament de les CPUs que ha multiplicat el rendiment d'aquestes, així com un estalvi energètic, ja que no calen freqüències de rellotge de CPU tan elevades [1][2].
És en aquest context on aquest treball es desenvolupa. És important intentar conèixer la idoneïtat de l'aplicació de les arquitectures multicore i multithreading per tal d'intentar aplicar cada configuració a aquelles tasques que millor s'adeqüen a cadascuna d'elles.
Aquest estudi compararà el rendiment de dues arquitectures mitjançant el model de memòria compartida per a programació paral·lela OpenMP [3]. Per una banda, s'analitzarà el comportament de l'arquitectura d'Intel Pentium IV Core 2 Quad Q9300, 2 x Dual core, memòria cau L1 de 32 KB per core, i L2 de 3 MB x Dual core, amb 4 GB RAM [*4*], i per altra banda l'arquitectura de Sun (ara Oracle)

SPARC Enterprise T5120 UltraSPARC T2, 4 cores reals, 8 threads x core, memòria cau L1 de 8KB, i L2 = 4 MB, amb 8 GB de RAM [5].
Per a l'estudi comparatiu de rendiment de les dues plataformes s'ha utilitzat una primera versió de benchmark desenvolupat amb aquest objectiu i basat en una clàssica multiplicació de matrius parametritzable. Aquest microbenchmark permet fer multiplicacions de matrius de nombres enters i multiplicacions de matrius de nombres de coma flotant, així com modificar la grandària de les matrius, adaptar les matrius a l'arquitectura de la memòria cau de la plataforma on s'executa, i realitzar la multiplicació per blocs de diferents mides.
A més, el microbenchmark dóna l'opció d'escollir la versió d'OpenMP que es farà servir per al càlcul, OpenMP 2.5 [6], basat en el model clàssic fork , o bé OpenMP 3.0 [7], que contempla el nou paradigma basat en tasques.
La resta d'aquest estudi s'organitza com segueix. Las Secció 2 presenta el model de memòria compartida i programació paral·lela amb OpenMP 2.5 i 3.0. La Secció 3 descriu la metodologia del microbenchmark utilitzat per a l'obtenció de resultats. La Secció 4 dóna l'anàlisi comparatiu dels resultats de rendiment de les dues arquitectures amb les dues versions d'OpenMP, v. 2.5 i v. 3.0. Per últim, la secció 5 presenta les conclusions.

## 2. Descripció general del model OpenMP v. 2.5 i v.3.0.

El model OpenMP i la seva API (Application Program Interface) s'han creat per al suport multiplataforma de programació de memòria compartida paral·lela en els llenguatges C, C++ i Fortran, per a qualsevol arquitectura, entre les quals s'inclouen plataformes Unix i plataformes Windows. Darrere del projecte hi ha un fort grup de corporacions cormecialitzadores de maquinari i programari que fan de l'OpenMP un model portable i escalable, el qual proporciona als programadors d'aplicacions de memòria compartida i paral·leles una interfície flexible que els permet desenvolupar, tant aplicacions per a computadors d'escriptori, com per a supercomputadors [3].
OpenMP consisteix en unes directives de compilador, unes rutines d'entorn d'execució i unes variables d'entorn. L'especificació està mantinguda per l'“OpenMP Architecture Review Board” (Taula de revisió de l'arquitectura d'OpenMP). Proporciona avantatges com ara: un bon rendiment i escalabilitat, un estàndard madur de facto, gran portabilitat, poc esforç de programació, i permet que el programa sigui paral·lelitzat incrementalment. L'OpenMP està totalment dissenyat per a arquitectures multicore. El seu model de memòria permet a tots els threads accedir a la mateixa memòria compartida global i les dades poden ser compartides o privades. Si són compartides tots els threads tenen accés. Per contra, les dades privades només són accessibles pel thread propietari, la transferència de dades és transparent al programador i la sincronització és en la

seva majoria implícita. La versió d'OpenMP 2.5 es basa en el model “fork and join” en el qual existeix un thread principal, anomenat màster, del qual sorgeixen els threads esclaus o “workers” quan s'entra en una zona paral·lela del programa. Quan els threads “workers” finalitzen una feina hi ha una fase de sincronització amb el thread màster, i després hi poden haver més zones paral·leles que segueixen el mateix comportament [*8*].
En el cas de la versió 3.0 d'OpenMP, apareix un nou element anomenat tasca. Una tasca és una instància de codi executable amb el seu entorn de dades generada quan un thread troba un constructor o pragma de tasca, o bé un constructor que inicia una zona de codi paral·lel. Quan un thread executa una tasca produeix una regió de tasca. Una regió de tasca està composta de tot el codi que es troba durant l'execució d'una tasca. Una regió paral·lela està composta d'una o més regions de tasca implícita, o tasca generada quan en el codi es troba, durant l'execució, un constructor de tasca [*8*]. Altra peculiaritat de les tasques és que poden estar o no lligades a un thread. En el cas d'estar lligades a un thread, només aquest podrà executar la tasca. En cas contrari, l'execució de la mateixa la pot començar un thread de l'equip de threads, però aquest pot deixar l'execució i després un altre thread la pot continuar, la pot tornar a abandonar, o pot finalitzar-la [*9*].
El nostre estudi ha estat realitzat amb les dues versions d'OpenMP, amb el llenguatge de programació C, i el suport que el compilador de GNU GCC i la seva llibreria libGOMP donen a OpenMP v. 2.5, a partir de la seva versió 4.2, i a OpenMP v. 3.0, a partir de la seva versió 4.4 [*10*][*11*].

## 3. Metodologia del microbenchmark aplicat.

Per a la realització de l'estudi hem dissenyat un microbenchmark parametritzable basat en multiplicació de matrius per tal d'avaluar el rendiment de les dues arquitectures. Tot el microbenchmark es compon de sis petits programes. Un primer conjunt d'aquests petits programes s'ha utilitzat per a l'obtenció dels temps d'execució de les multiplicacions sense paral·lelitzar, és a dir, són programes que s'executen de forma seqüencial, sense aprofitar les arquitectures multicore i multithreading. Tenim un segon grup de programes format pels dos microbenchmarks principals, un per a operacions amb nombres enters, i un altre per a operacions amb nombres de coma flotant, el quals apliquen el model d'OpenMP en les seves dues versions.
En el primer grup de programes seqüencials tenim programes que fan la multiplicació de matrius directament, la multiplicació de matrius tenint en compte l'arquitectura de la memòria cau per tal d'avaluar l'impacte que aquesta té sobre les operacions, i la multiplicació de matrius per blocs, tant per a nombres enters com per a nombres de coma flotant. Aquests programes tenen paràmetres de compilació com ara la mida de la memòria que s'utilitzarà per a les tres matrius

involucrades en l'operació, el nombre de blocs que es farà servir (hem definit 4, 8 o 16 com a possibles), i el nombre de bytes per a una línia de memòria cau.
En el segon grup de programes tenim els dos microbenchmarks que fan les diferents operacions per a nombres enters, o per a nombres en coma flotant. Aquest dos microbenchmarks reben paràmetres en línia de compilació per tal de definir la seva execució, entre ells estan: la mida de la memòria que ocuparan les tres matrius involucrades en la multiplicació, si s'aplicarà OpenMP 2.5 o OpenMP 3.0, el nombre de threads que es farà servir (poden ser 2, 4, 6, 8, o 10), el paràmetre que indica el nombre de bytes per línia de memòria cau (aquest paràmetre pot, o no, aparèixer), i el nombre de blocs que s'utilitzarà (paràmetre que pot o no aparèixer i que pot tenir els valors 4, 8 o 16).
En el cas d'una execució en la qual s'aplica OpenMP v. 2.5 s'executa la multiplicació amb una construcció tal com es mostra en el següent pseudocodi [6]:

```
#pragma omp parallel default (none)
                    shared (matriu4, matriu5, matriu6)
                    firstprivate (lines, colums)
                    priv ate (ii, jj, kk, sumaI)
          {
          #pragma omp for schedule (dynamic) nowait
          per a ii = 0 fins que ii < lines fer
               per a jj = 0 fins que jj < colums fer
                    per a kk = 0 fins que kk < colums fer
                         sumaI = sumaI + (matriu4 [ii][kk] *
                                         matriu5 [kk][jj])
                         incrementar kk

                    matriu6 [ii][jj] = sumaI
                    incrementar jj
               incrementar ii
          }
```

Per a una multiplicació per blocs amb OpenMP v. 2.5, el pseudocodi que s'executarà es fa tal com es mostra seguidament [6]:

```
void mulblock () // Multiplica el bloc i acumula el resultat
     {
     ...

     #pragma omp parallel for
                 shared (ii, jj, kk, matriu4, matriu5, matriuAux)
                 private (i, j, k, sumaI) sched ule (dynamic)
          per a i = ii fins que i < (ii + BLOCKSIZE) fer
               per a j = jj fins que j < (jj + BLOCKSIZE) fer
                    per a k = kk fins que k < (kk + BLOCKSIZE) fer
                         sumaI = sumaI + (matriu4 [i][k] *
                                         matriu5 [k][j])
                         incrementar k

                    matriuAux [i - ii][j - jj] += sumaI;
                    incrementar j
               incrementar i
     }
```

```
...
 per a ii = 0 fins que ii < INTLCMB fer
      per a jj = 0 fins que jj < INTLCMB fer
           clearmatriu (); // Neteja la matriu auxiliar
           per a kk = 0 fins que kk < INTLCMB fer
                mulblock ();
                kk = kk + BLOCKSIZE

           copymatriu (); // Posa el resultat en la matriu destí
           jj = jj + BLOCKSIZE
    ii = ii + BLOCKSIZE
 ...
```

Per últim, la multiplicació per blocs i amb OpenMP v. 3.0 fa que cada multiplicació de bloc sigui una tasca. La construcció de la multiplicació és tal com segueix [*7*]:

```
per a ii = 0 fins que ii < INTLCMB fer
     per a jj = 0 fins que jj < INTLCMB fer
          clearmatriu (); // Neteja matriu auxiliar
          #pragma omp parallel
               shared (matriu4, matriu5, matriu6, matriuAux, kk)
               firstprivate (ii, jj)
               {
               #pragma omp single nowait
                    {
                    per a kk = 0 fins que kk < INTLCMB fer
                       #pragma omp task firstprivate (ii, jj, kk)
                       mulblock30 (kk);
                       kk = kk + BLOCKSIZE

                    copymatriu (); // Posa resultat en destí
                    }
               }
          jj = jj + BLOCKSIZE
     ii = ii + BLOCKSIZE
```

Tant les variables que controlen els bucles, les que controlen el temps d'execució, com les que contenen les pròpies matrius, són globals. En el cas de les matrius el fet que siguin globals fa que puguem superar el límit de pila que tindríem si fossin variables locals. A més, la definició de les variables que contenen les matrius es fa en funció de la quantitat de memòria que s'indica en el paràmetre anteriorment citat de línia de compilació. Són matrius quadrades i si en línia de compilació apareix el paràmetre que indica multiplicació per blocs, només s'executarà aquesta modalitat si la mida de memòria d'aquest paràmetre és superior a 6 MB.

## 4. Resultats experimentals.

Les proves de rendiment s'han fet per a dues arquitectures: Intel Pentium IV  Core 2 Quad Q9300, 2 x Dual core, memòria cau L1 de 32 KB per core, L2 de 3 MB x Dual core 2.50 GHz, amb 4 GB RAM [*4*], i Sun (ara Oracle) SPARC Enterprise T5120 UltraSPARC T2, 4 cores

reals 1.2 GHz, 8 threads x core, memòria cau L1 de 8KB, L2 = 4 MB, amb 8 GB de RAM [5].
Totes les compilacions s'han realitzat amb GNU C Compiler, GCC, versions 4.3.3 i 4.4.5, tant en l'arquitectura Intel com en l'arquitectura SPARC.
Seguidament es mostren gràfiques dels SpeedUps obtinguts per a 2, 4, 6, 8, 10, 16 i 32 threads per a les operacions amb multiplicació de matrius amb nombres enters i per a les multiplicacions de matrius de nombres en coma flotant per a diferents mides d'ocupació de memòries en les dues arquitectures i amb OpenMP v. 2.5 i 3.0. Tots els SpeedUps s'han calculat amb la fórmula: Speedup = $T_1$ / $T_{p,k}$, on $T_1$ és el temps mesurat per a una execució amb un únic core, i $T_{p,k}$ és el temps mesurat amb p cores i k threads [12].
Per a les proves realitzades només per a les memòries cau s'ha pogut comprovar que, si es té en compte la mida de línia de la memòria cau a l'hora de definir les matrius, la rapidesa de les operacions es multiplica. També hem pogut veure la diferència de rapidesa entre la memòria cau de nivell 1, L1, i la cau de nivell 2, L2.
La diferència de l'SpeedUp entre les dues arquitectures és molt apreciable, ja que l'arquitectura PIV funciona a 2.50 Ghz, mentre que l'arquitectura UltraSPARC T2 és de 1.2 Ghz, essent més ràpida fins als 8 threads l'arquitectura P IV.
Hem pogut calcular les sobrecàrregues i l'eficiència de les memòries cau i hem comprovat novament que l'arquitectura UltraSPARC T2 té més sobrecàrrega amb operacions amb la memòria cau. La seva sobrecàrrega comença a baixar fins a menys de 0 a partir de 8 threads, tot i que amb operacions amb nombres de coma flotant arriba un moment, amb 6 threads, que aquesta s'estabilitza i no baixa més, cosa que té sentit ja que només hi ha una Float Point Unit per core i a partir de 4 threads aquesta s'ha de compartir entre totes les peticions dels diversos threads [5].
La sobrecàrrega la calculem amb la fórmula: Sobrecàrrega = $T_{p,k} - T_1/p$, on $T_1$ és el temps mesurat per a una execució amb un únic core, p és el nombre de cores, en aquest cas 4, i $T_{p,k}$ és el temps mesurat amb p cores i k threads [12]; i l'eficiència s'ha calculat només per a 4 cores, ja que les dues màquines tenen 4 cores reals, amb la fórmula següent: Eficiència = $T_1$ / ($p*T_p$), on $T_1$ és el temps mesurat per a una execució amb un únic core, p és el nombre de cores, en aquest cas 4, i $T_p$ el temps mesurat amb p cores [12].
Quant veiem les operacions amb quantitats de memòria superiors veurem aquestes gràfiques.
En les operacions amb la memòria cau, al ser quantitats de memòria petites i ràpides, amb baixa latència, té gran influència la rapidesa dels cores, essent el PIV el doble de ràpid que la UltraSPARC, tot i que aquesta última compensa el seu desavantatge de procés amb la capacitat de procés multithreading.
Les gràfiques següents mostren els SpeedUps de les multiplicacions de matrius, de nombres enters i de coma flotant, amb OpenMP v. 2.5, amb OpenMP v. 2.5 i per blocs, i amb OpenMP v. 3.0 amb tasques

(cada bloc de la multiplicació per blocs s'ha convertit en una tasca) amb un total d'ocupació de memòria de 250 MB.

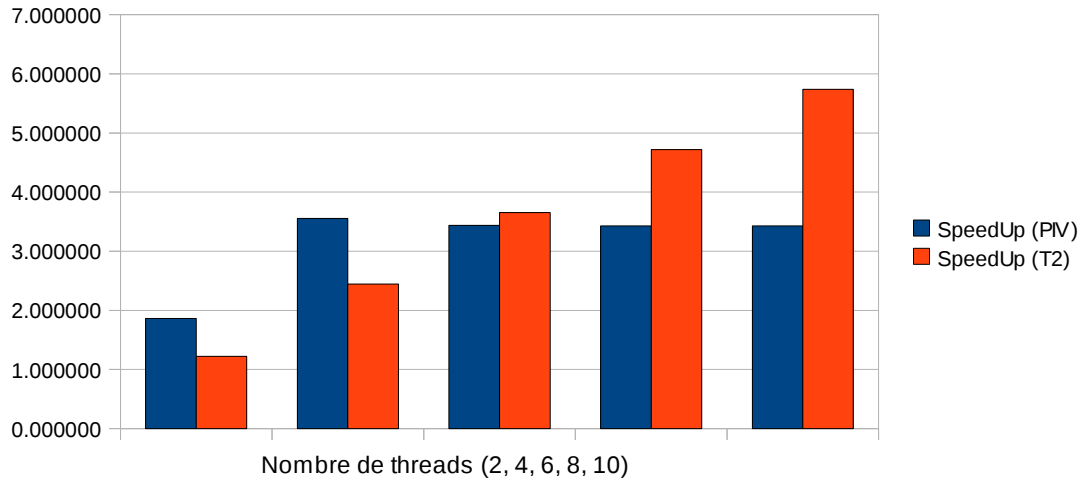


**Fig.1** SpeedUp multiplicació de matrius amb OpenMP v. 2.5, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

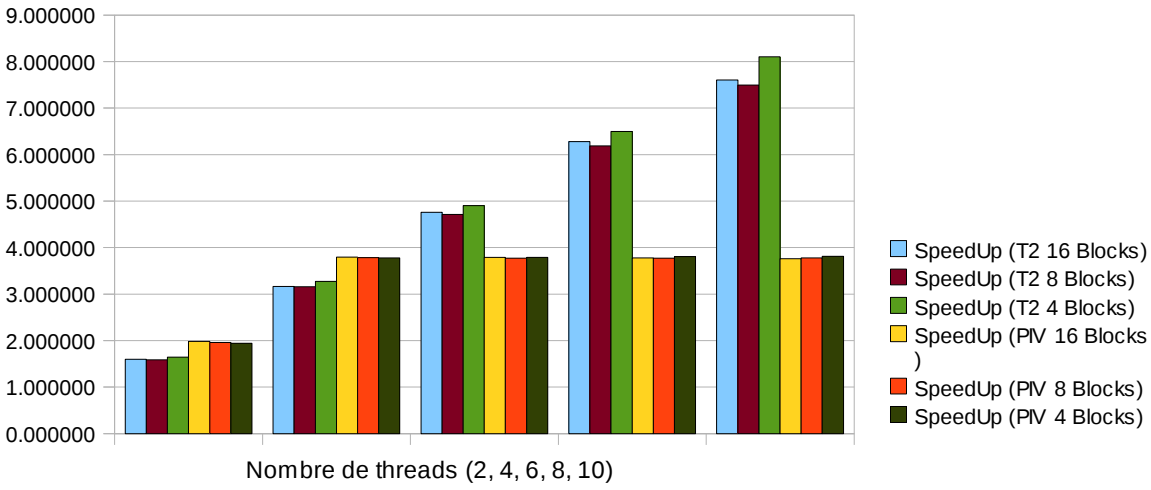


**Fig.2** SpeedUp multiplicació per blocs de matrius amb OpenMP v. 2.5, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

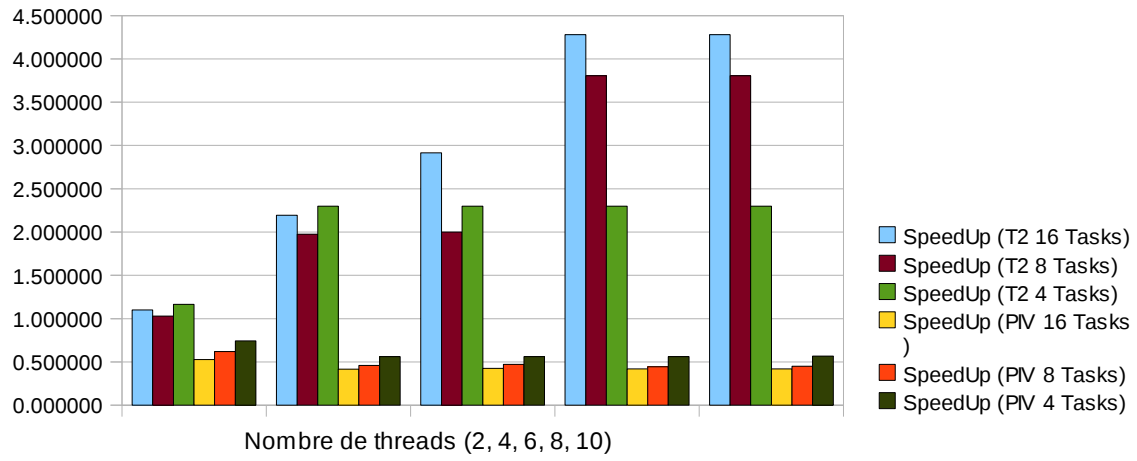


**Fig.3** SpeedUp multiplicació de matrius amb OpenMP v. 3.0 i tasques, amb nombres enters i un ús de memòria de 250MB entre les tres matrius de l'operació.

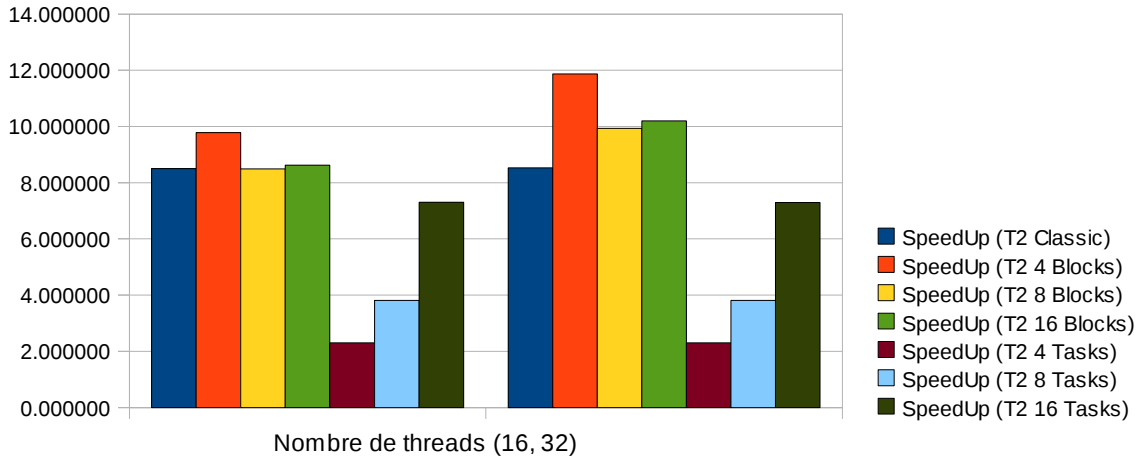


**Fig.4** SpeedUp multiplicació de matrius amb OpenMP v. 2.5 i 3.0 amb tasques, amb nombres enters i un ús de memòria de 250MB entre les tres matrius.

Per a l'arquitectura Intel PIV podem veure a les Fig. 1 i 2 que no hi ha molta diferència en la variació de l'SpeedUp entre la multiplicació clàssica i la multiplicació per blocs amb OpenMP v. 2.5 i nombres enters. Tornem a veure que l'augment de rendiment es dóna de 2 a 4 threads, llavors es produeix un estancament per la limitació d'un thread per core. En el cas de la Fig. 3, veiem com amb OpenMP v. 3.0, la multiplicació de matrius mitjançant tasques pateix una baixada molt important de l'SpeedUp a causa de la sobrecàrrega introduïda per OpenMP v. 3.0. Això ens pot indicar dues coses, per una banda el requeriment de fer més proves per tal de trobar la grandària i el nombre de tasques òptim per a aquesta arquitectura, així com l'òptima construcció d'aquestes tasques, i per l'altra, que les tasques no són adequades per a tot tipus de treballs [9].
Quant a l'arquitectura UltraSPARC T2 i les multiplicacions amb OpenMP v. 2.5 clàssica i per blocs, veiem com, tot i que els seus cores

són la meitat de ràpids que l'arquitectura PIV, la seva capacitat multithreading permet que obtinguem millors SpeedUps per a operacions amb enters.
Quan es fa la multiplicació per blocs, com es pot veure a la Fig. 2, conforme augmenta el nombre de threads, i el nombre de blocs que fem servir, augmentem l'SpeedUp gràcies a què tenim més threads dedicats a les tasques, i a què aquests blocs són més petits i això millora la localitat de les dades a la memòria i es treu més profit de les memòries cau, reduint els accessos a la memòria RAM de la màquina [5]. Però tornem a veure que amb OpenMP v. 3.0, l'ús de les tasques introdueix una sobrecàrrega que fa variar molt l'SpeedUp de 4 a 8 o 16 blocs. Tornem a veure la necessitat de definir més la construcció de les tasques, el seu scheduling, la grandària de cada tasca en el nostre cas, si s'utilitzen tasques lligades o no als threads, etc [9].
La Fig. 4 descriu el comportament de l'SpeedUp per a la UltraSPARC T2 amb 16 i 32 threads, màxims threads d'aquesta T2, per multiplicació clàssica i per blocs amb OpenMP v. 2.5, i amb tasques d'OpenMP v. 3.0. Tornem a veure la sobrecàrrega introduïda per l'OpenMP 3.0, repercutint l'SpeedUp amb 4 i 8 tasques, cosa que reafirma el nostre argument anterior respecte a les tasques.

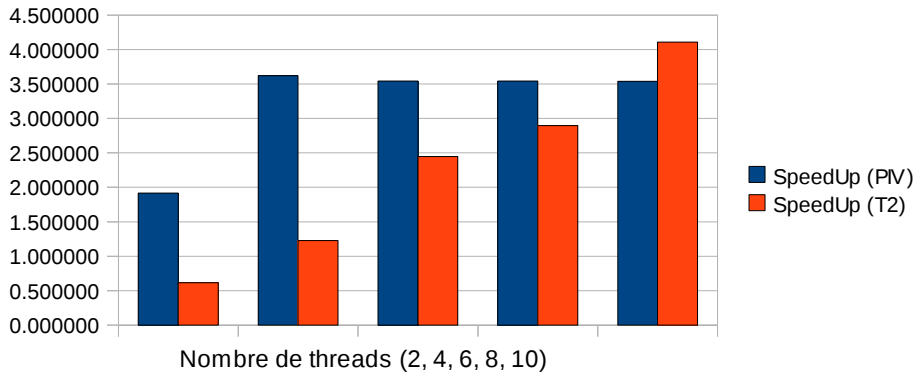


**Fig.5** SpeedUp multiplicació de matrius amb OpenMP v. 2.5 amb nombres de coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

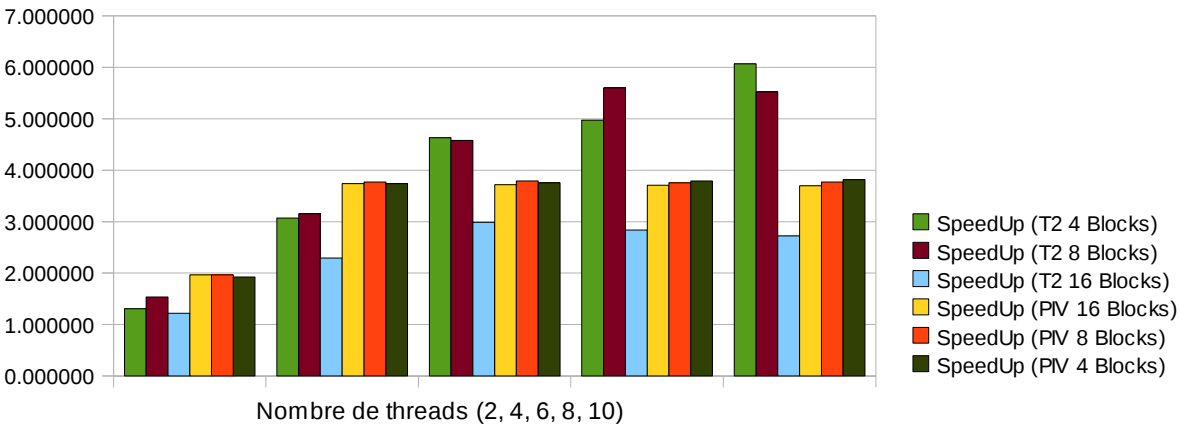


**Fig.6** SpeedUp multiplicació per blocs de matrius amb OpenMP v. 2.5, nombres de coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

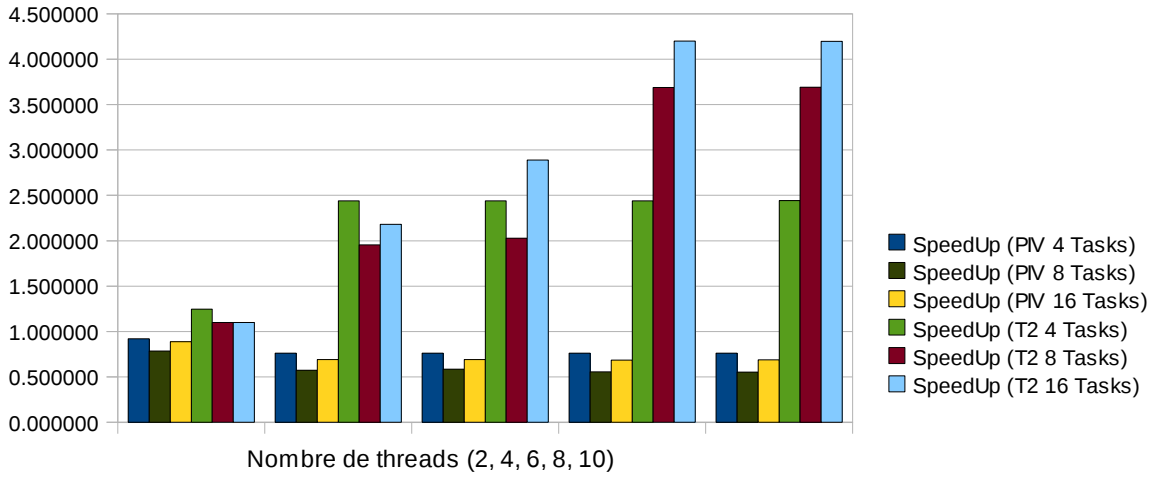


**Fig.7** SpeedUp multiplicació de matrius amb OpenMP v. 3.0 (tasques), nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

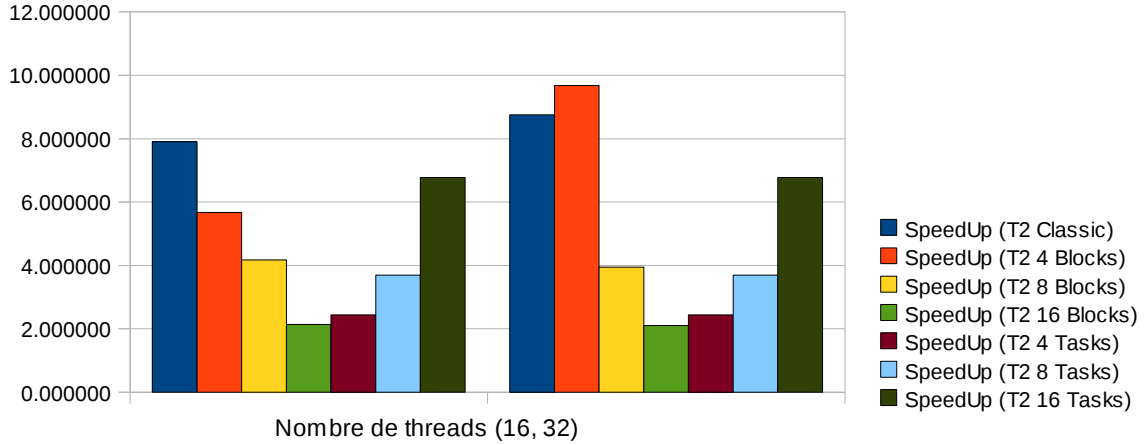


**Fig.8** SpeedUp multiplicació de matrius amb OpenMP v. 2.5 i 3.0, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

Quant a l'SpeedUp de les operacions de nombres en coma flotant podem veure un patró similar en l'arquitectura PIV, on a partir de 4 threads aquest no augmenta, i no hi ha diferència significativa quant a la variació de blocs. No passa el mateix amb l'arquitectura UltraSPARC T2, on veiem un creixement de l'SpeedUP per cada vegada que augmentem en 2 el nombre de threads, així, partint d'un SpeedUp d'una mica més de 0.5 amb 2 threads, acabem amb 10 threads amb un SpeedUp de més de 4.0. Com es pot veure en la Fig. 5 de la multiplicació clàssica i, com es pot veure en la Fig. 8, amb més threads veiem un SpeedUp que arriba a casi 9 amb 32 threads. Amb operacions amb blocs, Fig. 6 i 8, veiem que amb OpenMP v. 2.5. i amb 4 blocs obtenim un continu creixement de l'SpeedUP al augmentar el nombre de threads, cosa que passa, però menys si fem servir 8 blocs, i no passa amb 16 blocs. Aquest comportament es pot relacionar amb: la característica del maquinari de la UltraSPARC T2 del nombre de Float Point Unit que té, una per cada core, amb una millor localitat de les dades a la memòria amb 4 blocs, i amb el fet de

tenir un core dedicat només a un thread. Conforme augmentem el nombre de blocs i de threads es genera més sobrecàrrega de gestió dels threads, més peticions de sincronització entre ells i més competència entre els threads per accedir a les FPU de la T2, així com més accessos de memòria per part del major nombre de threads [5].
Per a les operacions amb tasques, veiem a la Fig. 7 i 8, que guanyem SpeedUp amb l'increment de tasques, però tornem a veure que OpenMP v. 3.0 introdueix una sobrecàrrega extra de gestió de les tasques que perjudica l'SpeedUp en comparació amb la multiplicació per blocs de les Fig. 6 i 8, amb el mateix nombre de blocs que tasques. Donat aquest resultat és convenient per a un estudi futur mirar d'optimitzar la construcció de les tasques, la seva grandària i el seu scheduling per tal de trobar la definició més adequada i eficient al benchmark que s'executi.
Seguidament podem veure a les figures següents les eficiències, $Eficiència = T_1 / (p*T_p)$, $T_1$ és el temps mesurat per a una execució amb un únic core, p és el nombre de cores, i $T_p$ el temps mesurat amb p cores [12], calculades de les diferents multiplicacions, nombres enters i coma flotant, amb OpenMP v. 2.5 i OpenMP v. 3.0.

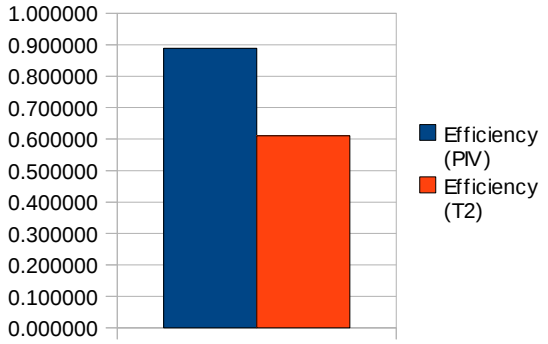


**Fig.9** Eficiència multiplicació de matrius clàssica amb OpenMP v. 2.5, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

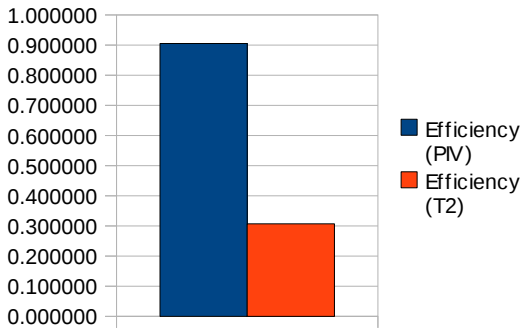


**Fig.10** Eficiència multiplicació de matrius clàssica amb OpenMP v. 2.5, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

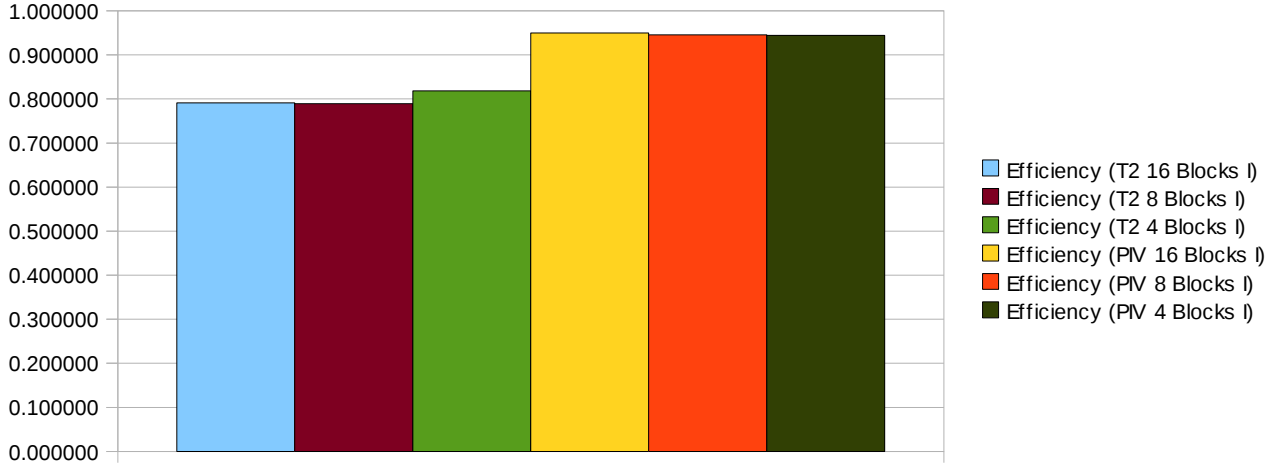


**Fig.11** Eficiència multiplicació de matrius per blocs amb OpenMP v. 2.5, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

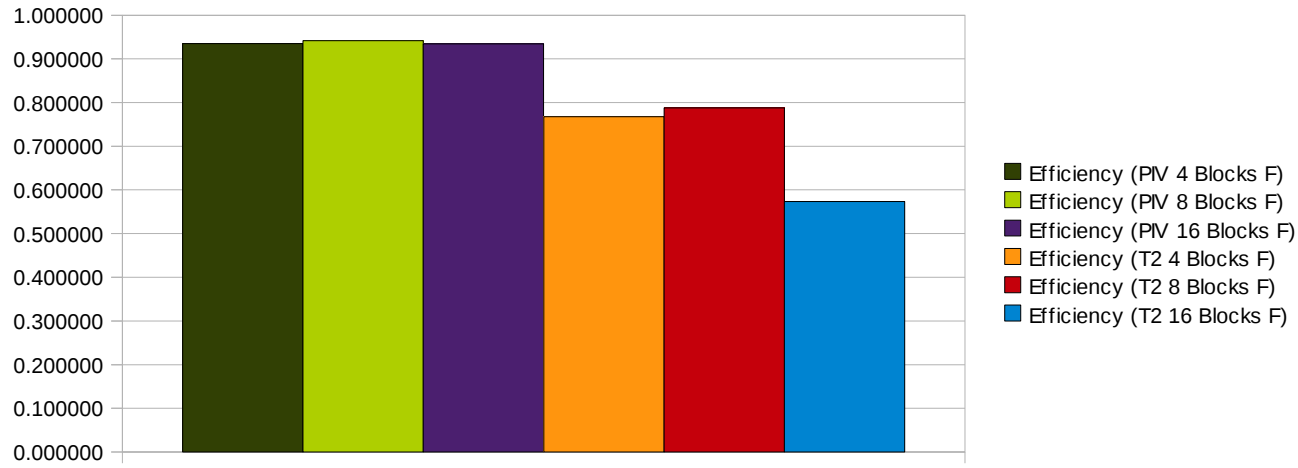


**Fig.12** Eficiència multiplicació de matrius per blocs amb OpenMP v. 2.5, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

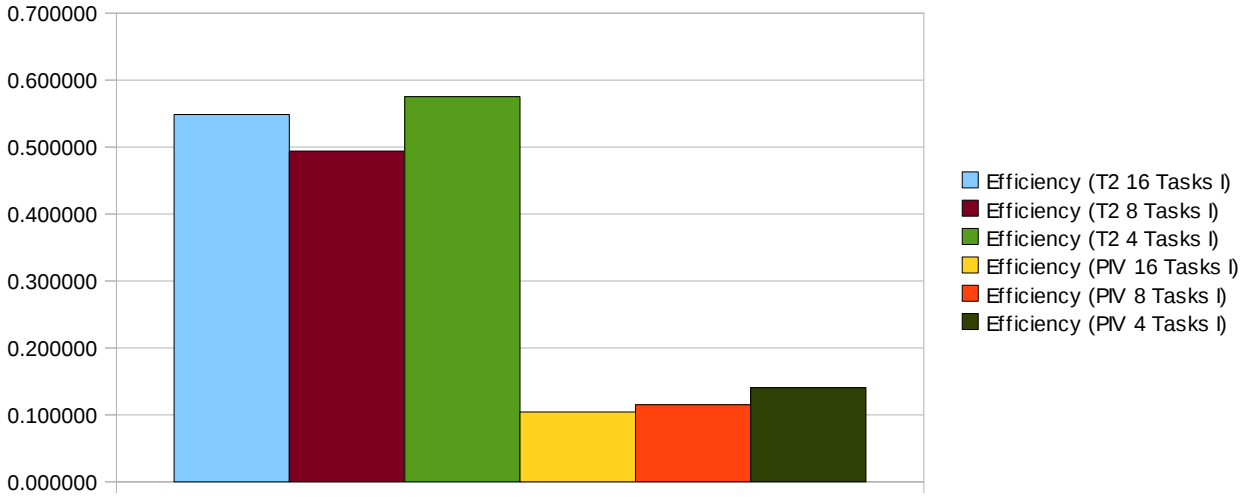


**Fig.13** Eficiència multiplicació de matrius amb tasques, OpenMP v. 3.0, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

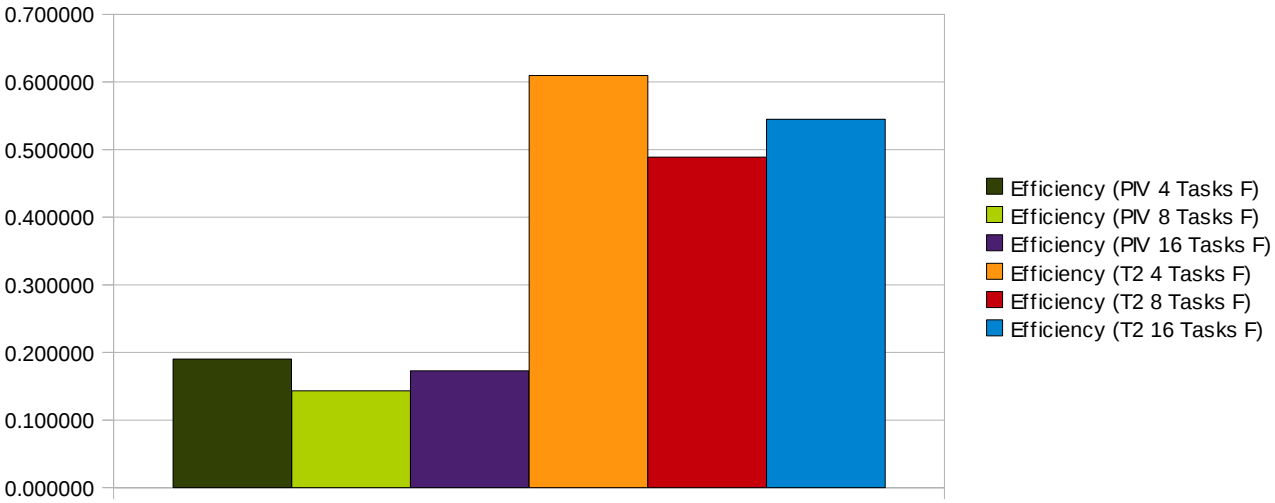


**Fig.14** Eficiència multiplicació de matrius amb tasques, OpenMP v. 3.0, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

Per últim, les gràfiques de les sobrecàrregues (Sobrecàrrega = $T_{p,k} - T_1/p$, on on $T_1$ és el temps mesurat per a una execució amb un únic core, p és el nombre de cores, en aquest cas 4, i $T_{p,k}$ és el temps mesurat amb p cores i k threads [*12*]), obtingudes amb les diferents multiplicacions de matrius clàssica i per blocs amb OpenMP v. 2.5, i amb tasques amb OpenMP v. 3.0.

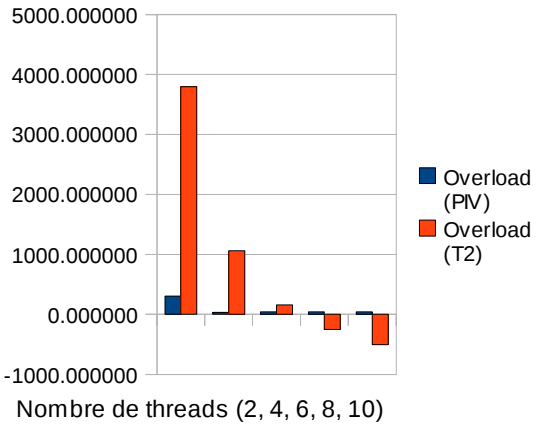


**Fig.15** Sobrecàrrega multiplicació de matrius clàssica amb OpenMP v. 2.5, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

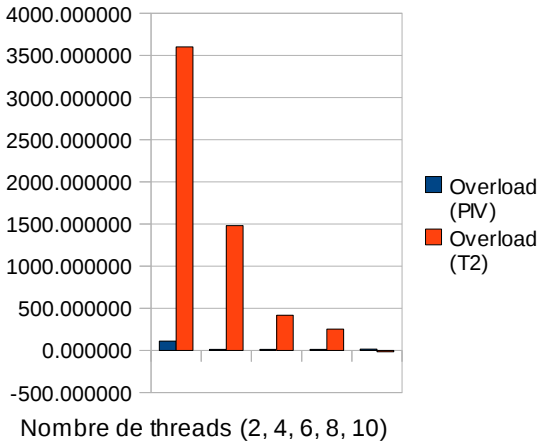


**Fig.16** Sobrecàrrega multiplicació de matrius clàssica amb OpenMP v. 2.5, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

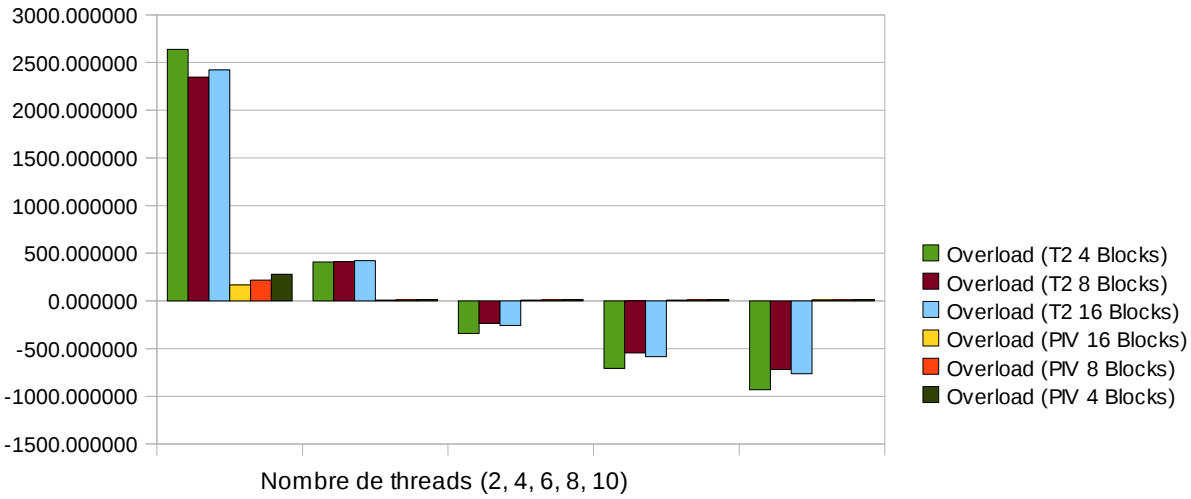


**Fig.17** Sobrecàrrega multiplicació de matrius per blocs amb OpenMP v. 2.5, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

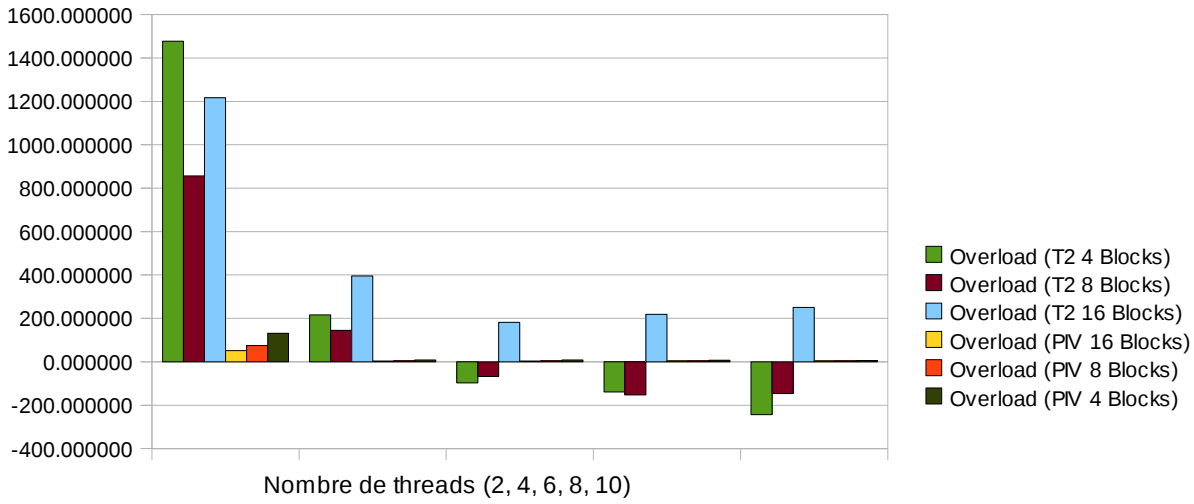


**Fig.18** Sobrecàrrega multiplicació de matrius per blocs amb OpenMP v. 2.5, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

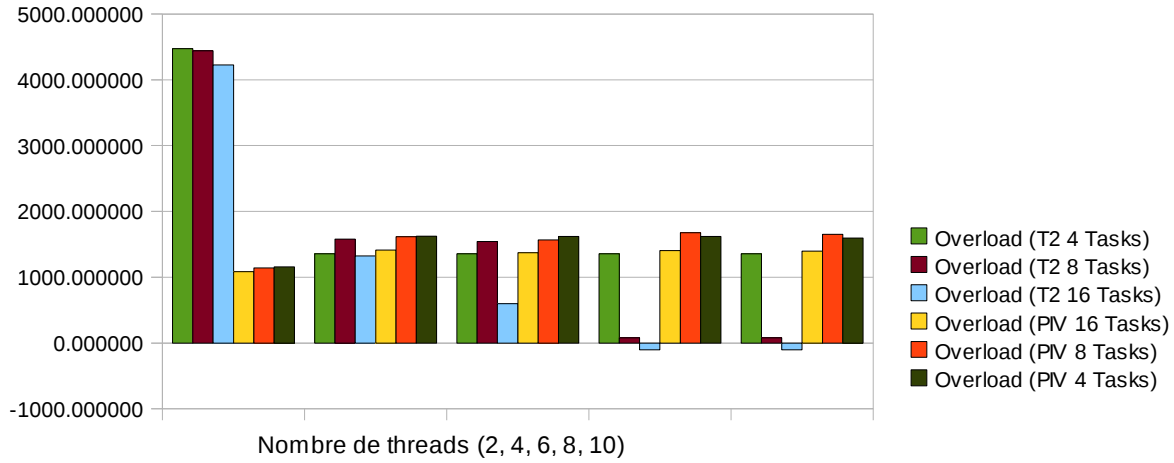


**Fig.19** Sobrecàrrega multiplicació de matrius amb tasques, OpenMP v. 3.0, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

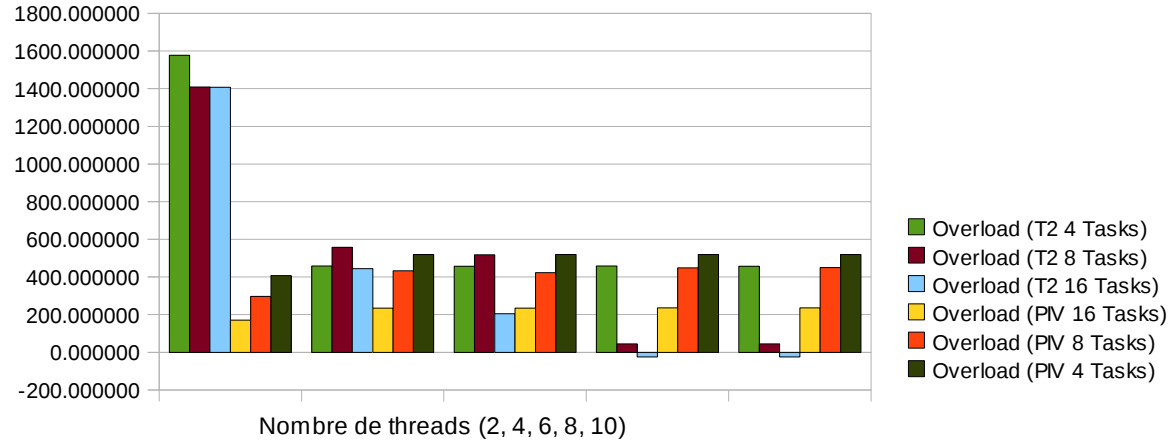


**Fig.20** Sobrecàrrega multiplicació de matrius amb tasques, OpenMP v. 3.0, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

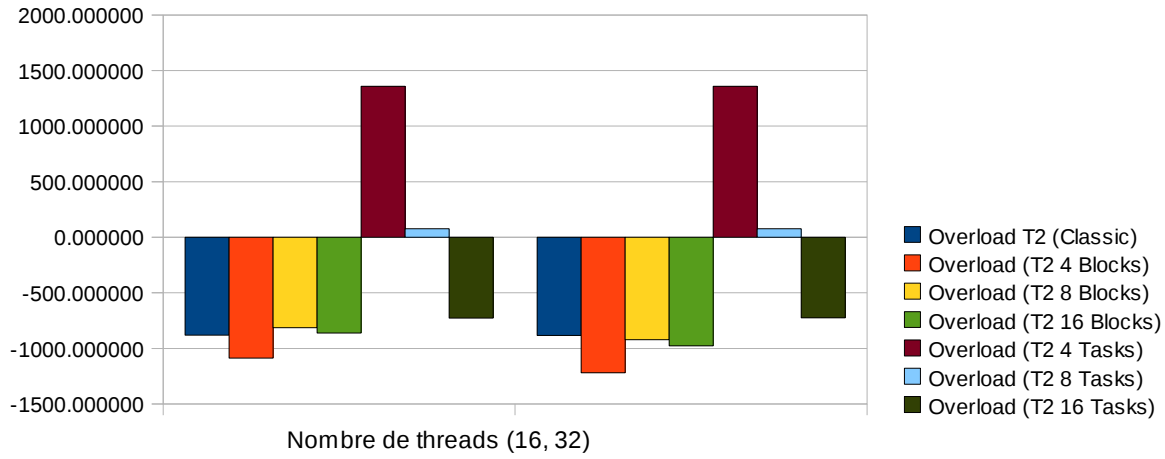


**Fig.21** Sobrecàrrega multiplicació de matrius clàssica, blocs i tasques, OpenMP v. 2.5 i 3.0, nombres enters amb un ús de memòria de 250MB entre les tres matrius de l'operació.

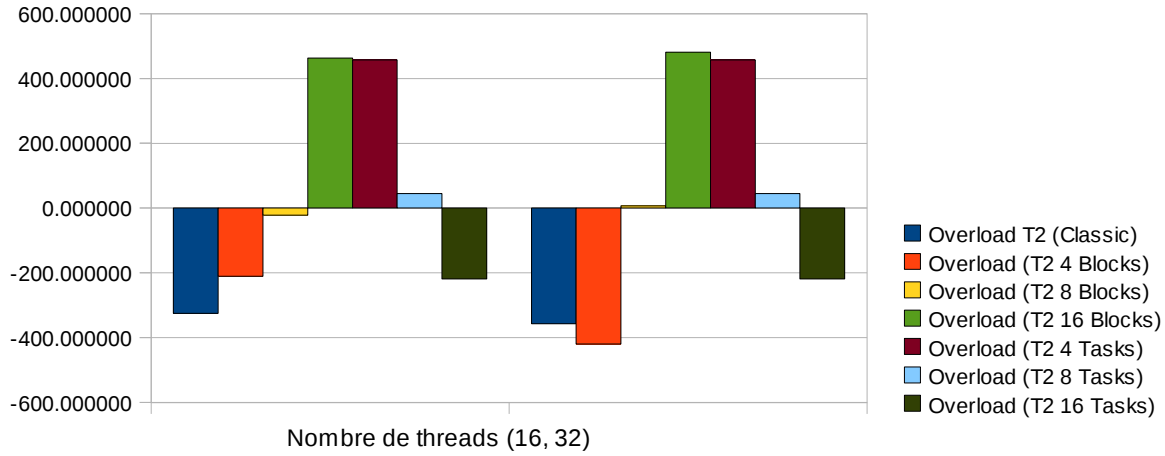


**Fig.22** Sobrecàrrega multiplicació de matrius classica, blocs i tasques, OpenMP v. 2.5 i 3.0, nombres coma flotant amb un ús de memòria de 250MB entre les tres matrius de l'operació.

## 5. Conclusions.

Les arquitectures multicore i multithreading tenen una aplicació clau ara i en un futur immediat i cal millorar les seves prestacions per a la computació intensiva científica. És important entendre el rendiment que aquestes computadores poden proporcionar i per a quins tipus d'aplicacions són més adequades. Amb aquest article hem intentat comparar les dues arquitectures Intel Pentium IV  Core 2 Quad Q9300 i SPARC Enterprise T5120 UltraSPARC T2 i s'ha intentat identificar les millores en l'arquitectura SPARC mitjançant la comparació de rendiment del model de memòria compartida OpenMP v. 2.5 amb OpenMP v. 3.0. En l'arquitectura  SPARC Enterprise T5120 UltraSPARC T2 hem pogut apreciar que la seva capacitat multithreading aporta molt bon rendiment amb operacions amb nombres enters en OpenMP

2.5 i hem apreciat que amb OpenMP 3.0 s'introdueix una sobrecàrrega important que fa baixar l'SpeedUp d'aquestes operacions. També hem comprovat, per a les operacions amb nombres de coma flotant, en la multiplicació clàssica i per blocs, amb OpenMP 2.5, un bon rendiment, tot i que amb blocs el millor rendiment s'experimenta amb 4 blocs, la qual cosa té sentit ja que la T2 només té 4 unitats de coma flotant i això permet un millor SpeedUp en multiplicació amb 4 blocs, millor localitat de memòria i un millor aprofitament dels recursos per part dels threads. Hem deduït que ampliar el nombre de FPU (Float Point Units) augmentaria la capacitat de càlcul, i un augment del nombre de cores incrementaria la capacitat multithreading de l'arquitectura SPARC, i tot plegat faria més potent aquesta arquitectura. També l'organització interna de la memòria cau té una gran influència en el rendiment, i la divisió o configuració de la mateixa per als cores influeix en el rendiment de la multiplicació per blocs. Per a estudis futurs, es proposa un estudi més profund de la influència de l'organització interna de la memòria cau en el rendiment de les operacions, i per al cas concret d'OpenMP v. 3.0, es proposa l'estudi més a fons, aplicant més benchmarking, de la definició de les tasques, el seu nombre més adequat, i el seu scheduling, per tal de trobar l'aplicació més adequada i eficient per al tipus d'operacions del nostre micro benchmark. Tot això requeriria de més proves per tal de definir la construcció, aquest scheduling i la grandària de les tasques, així com la combinació amb diferents mides de matrius, les quals utilitzarien diferents porcions de memòria, per tal de trobar les característiques més adequat i eficients per al model basat en tasques.

# Referències


1. José Antonio Moya: SOTT Open SPARC University Program, SUN Microsystems (d'abril 2009).
2. Shameem Akhter , Jason Roberts: Multi-Core Programming Increasing Performance through Software Multi-threading , Intel Corporation (Abril 2006).
3. The OpenMP API specification for parallel programming, http://openmp.org/wp/about-openmp/
4. Intel Core 2 Quad Processor Q9300 (6M Cache, 2.50 Ghz, 1333 Mhz FSB), http://ark.intel.com/Product.aspx?id=33922
5. OpenSPARC designers, developers and programmers: OpenSPARC Internals OpenSPARC T1/T2 Chip Multithreaded Throughput Computing, Sun Microsystems, Inc. , 4150 Network Circle, Santa Clara, CA 95054 , U.S.A. 650-960-1300 (October 2008)
6. OpenMP Architecture Review Board: OpenMP Application Program Interface, Versió 2.5 (May 2005)
7. OpenMP Architecture Review Board: OpenMP Application Program Interface, Versió 3.0 (May 2008)

8. Ruud van der Pas: An Overview of OpenMP, IWOMP 2010 CCS, University of Tsukuba , Tsukuba, Japan (June 14-16, 2010 )
9. Alejandro Duran: Tasking in OpenMP, Barcelona Supercomputing Center (June 2009)
10. Diego Novillo: OpenMP and automatic parallelization in GCC, Red Hut Canada, In: Proc. of the 2006 GCC Summit, Ottawa, Canada (2006)
11. GOMP – An OpenMP implementation for GCC, GNU Prject, Free Software Fundation (FSF), http://gcc.gnu.org/projects/gomp/
12. Ami Marowka: Performance of OpenMP Benchmarks on Multicore Processors, Algorithms and Architectures for Parallel Processing Lecture Notes in Computer Science, Volume 5022/2008, 208-219, DOI: 10.1007/978-3-540-69501-1_22 (2008)