مجلة تكريت للعلوم الهندسية 23 (4) (2016) (23-30)

TJES

ISSN: 1813-162X
مجلة تكريت للعلوم الهندسية
متاحة على الموقع الالكتروني: http://www.tj-es.com

دراسة العوامل المؤثرة بنسبة الإصابة في الذاكرة المخبئية ضمن نظام متعدد المعالجات

عمار علي زقزوق
جامعة البعث – حمص – سوريا
azakzok@albaath-univ.edu.sy
ammar.zakzouk@gmail.com

الخلاصة

يقدم هذا البحث دراسة تحليلية استنتاجية على المستوى المادي والبرمجي لنظام بمعالجات متعددة متجانسة "متماثلة، متناظرة". وذلك من خلال تحديد العوامل "البارامترات" الأساسية المؤثرة في الأداء من حجم الذاكرة المخبئية وعدد المعالجات وبروتوكولات ترابط "تماسك" الذاكرات المخبئية ومقارنتها، وإجراء محاكاة وتقويم لقيم تجريبية متعددة لهذه البارامترات. من أجل تسريع المعالجة التفرعية "المتوازية" وتحسين الأداء لتحقيق التنفيذ في الزمن الحقيقي، فقد تم الحصول على قيم توافقية لهذه العوامل كنتائج تجريبية.

الكلمات الدالة: الذاكرة المخبئية، نسبة عدم الإصابة، المعالجة التفرعية، المعالجات المتعددة، بروتوكولات التماسك، الزمن الحقيقي، بنية الحاسوب.

Study the Influential Factors in the Hit Rate of Cache Memory for a Multiprocessors System

Abstract

This paper presents an analytical and deductive study at software and hardware level for a symmetric multiprocessors system. Basic influential factors in the performance as cache memory size, processors number, and coherence protocols of cache memories are defined and compared, and a different simulations and evaluations of multiple experimental values of these parameters have been carried out also. In order to speed up the parallel processing and to improve the performance for realizing the execution in real time, harmonious values of these factors are obtained as experimental results.

Key word: Cache Memory, Miss Rate, Parallel Processing, Multiprocessors, Coherence Protocols, Real Time, Computer Architecture.

المقدمة

لقد بلغ تطور المعالجات وحيدة النواة (Unicore Processors) مرحلة متقدمة جداً، ما أدى إلى تحسين أداء (Performance Improvement) البنى المادية بشكل كبير[1]. إلا أن التحسين على المستوى المادي اصطدم بقيود طبيعية عدة: فالحرارة والتشويش الكهرومغناطيسي الناتجين عن زيادة سرعة تردد الساعة (Clock Frequency Speed) يؤثران في كثافة الترانزستورات (Transistors Intensity) على الشريحة "الدارة المتكاملة" (Integrated Circuit: IC)، كما أن الانتقال من تقنية 16 بتاً إلى 32 بتاً ثم إلى 64 بتاً، وكذلك زيادة سرعة الناقل "الممر" (Bus Speed)، لم يلبيان متطلبات عصر السرعة وتعدد المهام (Multitasking)، وخاصة فيما يتعلق بأنظمة المعالجة المعقدة كما في المخدمات (Servers) ومحطات العمل (Workstations)[2،3]. لكننا لسنا بصدد ذكر طرق لحل هذه المشكلات، لأنها لا تؤدي إلى التخلص التام منها. ما أدى إلى توجه الباحثين نحو محاولة تحسين الأداء على المستوى البرمجي مع المادي، من خلال تكامل البنية المادية والبنية البرمجية (Software and Hardware Architecture Integration)، والتي تمثلت بادئ الأمر بالمعالجة المتوازية (Parallel Processing) بديلاً عن المعالجة التسلسلية (Serial "Scalar" Processing)[4]. كما أن تحقيق المعالجة المتوازية باستخدام معالج وحيد النواة هو توازٍ وهمي "افتراضي" (Pseudo "Virtual" Parallelism)، إذ

يقوم المعالج هنا بتنفيذ مهام عدة من خلال تقسيم "مشاركة" الزمن (Time Slicing "Sharing") بين هذه المهام. وبالتالي، إن تحقيق الزمن الحقيقي (Real Time) بشكل فعلي في المعالجة المتوازية لا يمكن أن يتم إلا باستخدام وحدات معالجة متكاملة ومستقلة عن بعضها[6،5].

يوجد بشكل عام طريقتان أساسيتان لزيادة معدل تنفيذ التعليمات في المعالج: الأولى هي زيادة سرعة المعالج (Processor Speed)، وبالتالي إنقاص زمن تنفيذ التعليمة (Instruction Cycle)، والثانية هي زيادة عدد التعليمات التي يمكن تنفيذها بآن واحد[2].

من هنا جاء التفكير باستخدام تقنية جديدة تحول عملية معالجة البيانات والتعليمات من تسلسلية (SISD: Von Neumann Machine) إلى تفرعية (SIMD, MISD, and MIMD)[1] بغية تقليل الزمن المنقضي، لأن التفرع كما نعلم أسرع من التسلسل[7]. إذ يمكن تحقيق التوازي على مستويات عدة، بدءاً من التعليمة نفسها (المعالجات الأنبوبية "التدفقية، الضخية": Pipeline Processors) مروراً بمجموعة من التعليمات (المعالجات السلمية الفائقة: Superscalar Processors) وصولاً إلى الإجرائيات (المعالجات المصفوفية والمعالجات الشعاعية: Array Processors and Vector Processors) وحتى البرامج (Multicores Processors and Multiprocessors)[10]. وذلك سعياً في تحسين عامل التسريع (Speed up Factor) مقارنة بالمعالجات وحيدة النواة والتسلسلية، وصولاً إلى تطوير أنظمة تفرعية متكاملة على المستوى المادي والبرمجي، من أجل تحقيق متطلبات الزمن الحقيقي[4].

هذه التحسينات أدت إلى التوازي المتعدد. مثلاً، يؤدي متحكم DMA (الوصول "الولوج، النفاذ" المباشر إلى الذاكرة: Direct Memory Access) عمل I/O (دخل/خرج: Input/Output) بالتنسيق مع CPU (وحدة المعالجة المركزية: Central Processing Unit)[6]. وقد ابتُكرت في هذا المجال تقنيات بدأت بإضافة معالجات أخرى على اللوحة الأم (Mother Board) نفسها، وعرف هذا باسم تعدد المعالجات (Multiprocessors)[5]. وبالوقت نفسه، تم التوجه إلى تصميم وتطوير معالجات متعددة النوى (Multicores Processors) لرفع الأداء وتقليل الحرارة الناتجة على مستوى المعالج كعنصر أساسي في أي نظام تفرعي، وذلك بمكاملة نواتين أو أكثر ضمن معالج واحد على المقبس (Socket) ذاته[8]. وقد قامت إنتل (Intel) باستخدام تقنية الخيوط "المسالك، المسارد، النياسب" الفائقة جداً (Hyper Threading) ضمن هذه المعالجات، وذلك بسبب الحاجة إلى تحقيق أداء أعلى دون رفع استهلاك الطاقة (Power Consumption) الكهربائية وتكاليف إضافية، وكذلك القلق من موضوع الحرارة المبددة (Temperature Dissipation) في النظام[1].

وقد اقترنت هذه التحسينات باهتمام الباحثين بشكل أكبر بالذاكرة المخبئية "كاش"[2] (Cache Memory)، وما تلعبه من دور هام وأساسي في تحسين الأداء، إن كان على مستوى معالج واحد أو معالجات عدة. حيث تستخدم في عمليات الجلب المسبق للتعليمات والمعطيات (Data and Instructions Prefetch)، والتنبؤ "التخمين" (Prediction)، وتتبع الأثر كما في عمليات القفز والتفرع (Branch and Jump Trace)، وكذلك كذاكرة مشتركة (Shared Memory) تستثمر في عمليات الاتصال والتزامن (Communication and Synchronization) بين نوى متعددة داخل معالج واحد، وأيضاً ضمن نظام بمعالجات متعددة[3]. يكمن أحد الأهداف الأساسية للكاش في الحفاظ على معدل إصابة (Hit Rate) عالٍ أو معدل عدم إصابة (Miss Rate) منخفض، وفي هذه الحالة لا تستخدم المعالجات الممر بشكل متكرر كثيراً.

هدف البحث (Research Aim)

يهدف البحث إلى تقديم دراسة تحليلية استنتاجية لنظام بمعالجات متعددة متجانسة (Symmetric Multi-Processor: SMP) عن طريق إجراء عمليات محاكاة (Simulation) باستخدام SMPCache. إذ يتم تطبيق برامج اختبار معيارية (Benchmarks) تعتمد مبدأ تقفي الأثر لمحاكاة سلوك نظام بذاكرة مشتركة عبر ناقل أساسي، ما يسمح بدراسة تأثير العديد من العوامل "البارامترات" ("Parameters" Factors) على نسبة عدم الإصابة في الكاش، وهي: حجم الكاش (Cache Memory Size) وعدد المعالجات (Processors Number) وبروتوكولات التماسك (Coherence Protocols). وذلك سعياً في الوصول إلى تحديد البارامترات الأفضل لعناصر النظام، بما يحقق متطلبات السرعة في الأداء والمعالجة على التوازي في الزمن الحقيقي.

لقد تم التركيز في بحثنا هذا على دراسة المعالجات المتعددة كونها تمثل حلاً جيداً للأنظمة الضخمة والمعقدة، والتي تتطلب العمل ضمن الزمن الحقيقي بسبب استقلالية الموارد، كما في المخدمات مثلاً.

توصيف النظام (System Description)

لقد أخذت البنية متعددة المعالجات دوراً هاماً في الدراسات الأخيرة، إذ تم تطوير هذه التقنية بوضع معالجات عدة على اللوحة الأم نفسها عن طريق وضع كل معالج في مقبس خاص به، ولكل معالج موارده الخاصة والمستقلة عن المعالجات الأخرى[1].

تتجلى أهمية تعدد المعالجات من خلال الأمور التالية:

- تنفيذ المهام المستقلة بمعالجات متزامنة، وبذلك تزداد نسبة العمل وعدد المستخدمين.

[1] SISD: Single Instruction Single Data.
SIMD: Single Instruction Multiple Data.
MISD: Multiple Instruction Single Data.
MIMD: Multiple Instruction Multiple Data.
وذلك وفق تصنيف فلاين (Flynn's Taxonomy).

[2] سنستخدم هذه الكلمة ضمن ثنايا البحث للإشارة إلى الذاكرة المخبئية.

➢ الإقلال من تكامل المعالجات في نظام وحيد، لاشتراكه في موارد النظام، مثل الذاكرة والأقراص ووحدات الربط بالشبكات.
➢ تجزئة العمل الوحيد الكبير إلى مهام عدة متشابهة تنفذ في وقت واحد، من أجل السرعة في التطبيق، إذ أصبحت القدرة الحسابية للحاسوب تقاس بعدد العمليات التي يمكن أن ينفذها بالثانية على أعداد الفاصلة العائمة (Floating-point Operations Per Second: FLOPS) ومضاعفاتها.

تبعاً لنمط الاتصال في شبكة التوصيل "الربط" الداخلية (Interconnection Network) التي تربط معالجات عدة، تقسم الأنظمة متعددة المعالجات إلى: أنظمة الذاكرة المشتركة "التشاركية" (Shared Memory) بمجال عنونة وحيد، وأنظمة تمرير الرسائل (Messages Passing) بمجالات عنونة متعددة. يتحقق الاتصال في أنظمة الذاكرة المشتركة بالكتابة والقراءة في ومن ذاكرة مشتركة بين جميع المعالجات الموجودة في النظام، بينما يتم الاتصال في أنظمة تمرير الرسائل بإرسال واستقبال رزم (Packets) بين المعالجات[1،7،9].

تشكل الذاكرة التشاركية أحد التصنيفات الأساسية للمعالجات المتعددة، حيث تتشارك المعالجات بذاكرة عامة (Global Memory)، وتقوم هذه الذاكرة بتحقيق التزامن والتنسيق بين المعالجات من خلال عمليات القراءة والكتابة، وتتمتع أنظمة الذاكرة المشتركة بالمزايا الآتية:
➢ سهولة في البرمجة تؤدي إلى مرونة في تصميم المترجم (Compiler)، وفي تطوير التطبيقات التي تستخدم النظام.
➢ كلفة قليلة لتحقيق اتصال بعرض حزمة (Bandwidth) جيد.

لكن عند تصميم بنية نظام ذاكرة تشاركية، تبرز مشكلتان أساسيتان هما:

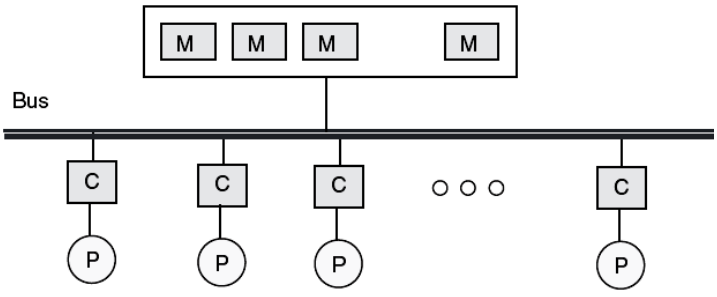
✦ انخفاض الأداء (Performance Degradation):
بسبب محاولة معالجات عدة الوصول إلى الذاكرة المشتركة باللحظة نفسها. وجارٍ تقدم العمل والبحث حالياً على حل مشكلة التنازع "التضارب" (Conflict "Contention, Collision") هذه من خلال تطوير تقنيات استخدام الذاكرات المخبئية.

✦ الترابط (Coherence):
تتم هذه العملية من خلال امتلاك نسخ متعددة من البيانات ونشرها على كافة الذاكرات المخبئية، حيث تكون نسخ المعطيات مترابطة إذا كانت قيمها متساوية في جميع هذه الذاكرات. فمثلاً، إذا كتب أحد المعالجات فوق إحدى هذه النسخ، تصبح هذه النسخة متضاربة مع بقية النسخ الموجودة في الذاكرات الأخرى. وجارٍ تقدم العمل والبحث حالياً على حل مشكلة الترابط هذه من خلال تطوير بروتوكولات تماسك مختلفة، من أهمها بروتوكولات التنصت (Snooping Protocols).

المعالجات المتعددة المتجانسة (SMP)

تكون الذاكرة في نظام الوصول المتماثل إلى الذاكرة (Uniform Memory Access: UMA) قابلة للنفاذ من قبل جميع المعالجات عن طريق شبكة الربط، وذلك بالطريقة نفسها التي ينفذ فيها معالج واحد إلى ذاكرته، أي أن جميع المعالجات يكون لها زمن الوصول ذاته إلى موقع الذاكرة[7]. في هذه الحالة، يمكن أن تكون شبكة الربط المستخدمة ممراً وحيداً أو ممرات عدة أو مفاتيحاً متصالبة (Crossbar Switches)[2،10]. بسبب الوصول المتكافئ إلى الذاكرة المشتركة، فإننا ندعو هذه الأنظمة أيضاً بأنظمة SMP، إذ كل معالج له الفرصة نفسها في الكتابة إلى الذاكرة أو القراءة منها، كما أن سرعة الوصول إلى الذاكرة متساوية لكل المعالجات[1].

يعد استخدام الممر البنية الأبسط لنظام الذاكرة المشتركة المتماثلة، كما هو مبين في الشكل (1). ويتم تخفيض النزاع على الممر من خلال جلب التعليمات والبيانات من الذاكرة العامة المشتركة (Memory: M) ووضعها في كاش (Cache: C) خاصة بكل معالج (Processor: P) قدر المستطاع، وهذا هو النوع الأكثر شيوعاً في أنظمة الذاكرة المشتركة. وهنا تكون الذاكرات المخبئية عالية السرعة، حيث توصل كل منها بالمعالج من جهة وبالممر من جهة أخرى، إذ يمكن تأمين نسخ محلية للتعليمات والمعطيات بمعدل سريع. في حال وجد المعالج التعليمات والمعطيات التي يريدها في الكاش الخاصة به، نقول عندها أن معدل الإصابة يساوي 100%، وبالمقابل نحصل على نسبة عدم إصابة عندما لا نستطيع الحصول على المعلومات من الكاش، وإنما يجب نسخها من الذاكرة العامة عبر الممر، ومن ثم إلى المعالج الذي طلبها[1،7].



الشكل (1) نظام ذاكرة مشتركة من النوع UMA-SMP

بروتوكولات التنصت (Snooping Protocols)

تقوم هذه البروتوكولات بمراقبة حركة البيانات في الناقل والذاكرات المخبئية (مراقبة فعاليات الممر: Bus Activities Monitoring) بشكل مستمر لإدارة ومزامنة التواصل بين معالجات عدة، ثم القيام بتنفيذ الأوامر المتوافقة مع هذه الأحداث، وذلك لمنع حدوث حالات تنازع، وبالتالي المحافظة على سلامة البيانات المشتركة بين عناصر النظام[7].

ينتقل البلوك في الذاكرة من حالة إلى أخرى نتيجة حدوث العمليات التالية: Read-Miss, Read-Hit, Write-Miss, Write-Hit. تعني Miss إما عدم إصابة أو أن البلوك موجود في الكاش ولكن يتم إبطاله (Invalidated) من قبل أحد المعالجات الأخرى أثناء قيامه بعملية كتابة إلى هذا البلوك المشترك. تختلف بروتوكولات التنصت على الممر من حيث عملها على تعطيل "إلغاء" (Invalidation)، أي أنه إذا قام أحد المعالجات بكتابة بيانات إلى موقع ذاكرة محدد، فإن هذه البيانات سوف تلغى من باقي الذاكرات المخبئية الأخرى، أو تعديل "تحديث" (update) النسخ المشتركة في الذاكرات المخبئية البعيدة عند إجراء عملية كتابة إلى الذاكرة المحلية، أي عند تعديل البيانات في الذاكرة الرئيسة (Main Memory) من قبل أحد المعالجات عندها يتم تحديث هذه البيانات في كل المعالجات. كما تختلف أيضاً بالمكان الذي سيتم الحصول على المعطيات منه في حال حدوث عدم إصابة، إذ تؤخذ هذه المعطيات من كاش بعيدة أخرى في بعض الحالات، أو من الذاكرة الرئيسة نفسها في حالات أخرى. فيما يلي بعضاً من هذه البروتوكولات التي تحافظ على ترابط الذاكرات المخبئية، والمستخدمة ضمن البحث:

✦ MSI (Modified Shared Invalidated)

يعتمد هذا البروتوكول على الإبطال من أجل ذاكرات مخبئية ذات كتابة عكسية (write-back)، وهو يستخدم ثلاث حالات للبلوك من أجل أية عملية كتابة عكسية للكاش، وهي:

I: البلوك موجود في الكاش، ولكن يتم إبطاله.

S: البلوك بحالة عدم تعديل في إحدى الذاكرات المخبئية أو أكثر، والنسخة الموجودة في الذاكرة الرئيسة مماثلة للموجودة في هذه الذاكرات.

M (أو Dirty): معالج وحيد فقط يملك البلوك بحالته المتاحة في الكاش، والبلوك في الذاكرة الرئيسة يكون بقيمة قديمة غير نافعة، وليس هناك أي من المعالجات يملك هذا البلوك بحالته المتاحة (S أو M).

✦ MESI (Modified Exclusive Shared Invalidated)

هذا البروتوكول هو تحسين للبروتوكول MSI، إلا أنه يضيف الحالة (E) التي تعني بأن كاشاً وحيدة تملك نسخة من البلوك ولم يتم تعديلها (النسخة نفسها في الذاكرة الرئيسة)، وذلك باتجاه تقليل عدد إجراءات الناقل التي تحدث نتيجة بروتوكول تماسك الكاش.

✦ Dragon

هذا البروتوكول معتمد على التحديث من أجل ذاكرات مخبئية ذات كتابة عكسية، وهو يستخدم أربع حالات للبلوك هي: E و S لهما المعنى نفسه في MESI، بالإضافة إلى SC (Shared Clean) و SM (Shared Modified). SC تعني أنه من المحتمل أن معالجين أو أكثر (متضمنة هذه الكاش) يملكون هذا البلوك ضمن ذاكراتهم المخبئية، والذاكرة الرئيسة قد تكون أو لا تكون محدثة (up-to-date). SM تعني أنه من المحتمل أن معالجين أو أكثر (متضمنة هذه الكاش) يملكون هذا البلوك ضمن ذاكراتهم المخبئية، والذاكرة الرئيسة غير محدثة، ومن مسؤولية هذا المعالج تحديث الذاكرة الرئيسة عندما يتم استبدال البلوك من الكاش الخاصة به. كما في MESI، تكون الإشارة S ضرورية.

منهجية ومتطلبات المحاكاة والتطبيق

Methodology and Requirements of (Simulation and Implementation)

يتم تطبيق برامج اختبار معيارية باستخدام SMPCache الذي يعتمد مبدأ تقفي الأثر لمحاكاة سلوك نظام SMP بذاكرة مشتركة عبر ناقل أساسي، ما يسمح بدراسة تأثير البارامترات المذكورة أعلاه على نسبة عدم الإصابة في الكاش الموجودة ضمن كل معالج.

✦ Benchmarks

تمثل العديد من التطبيقات التفرعية الحقيقية لتقفي الأثر بعشرات ملايين الوصولات "الرجوعات" (References) إلى الذاكرة، أي عدد المرات التي يتعامل (يتصل، يرجع) فيها التطبيق إلى الذاكرة من أجل القراءة و/أو الكتابة، كما هو مبين في الجدول (1).

الجدول (1) ملفات تتبع الأثر (Benchmarks)

ملف تتبع الأثر	عدد الرجوعات للذاكرة	عدد عمليات الكتابة	عدد عمليات القراءة	عدد التعليمات	الوصف
FFT	7,451,717	40705	83653	115642	تطبيق تفرعي يحاكي الحركة الديناميكية للسوائل.
Simple	27,030,092	36112	104408	99480	إصدار تفرعي من تطبيق بسيط.
Speech	11,771,664	57057	182943	0	---
Weather	31,764,036	22784	122163	95053	تطبيق تفرعي لحالة الطقس يُستخدم للتنبؤ بالأرصاد الجوية، وهو الإصدار التسلسلي من NASA.

✦ Configurations Files:

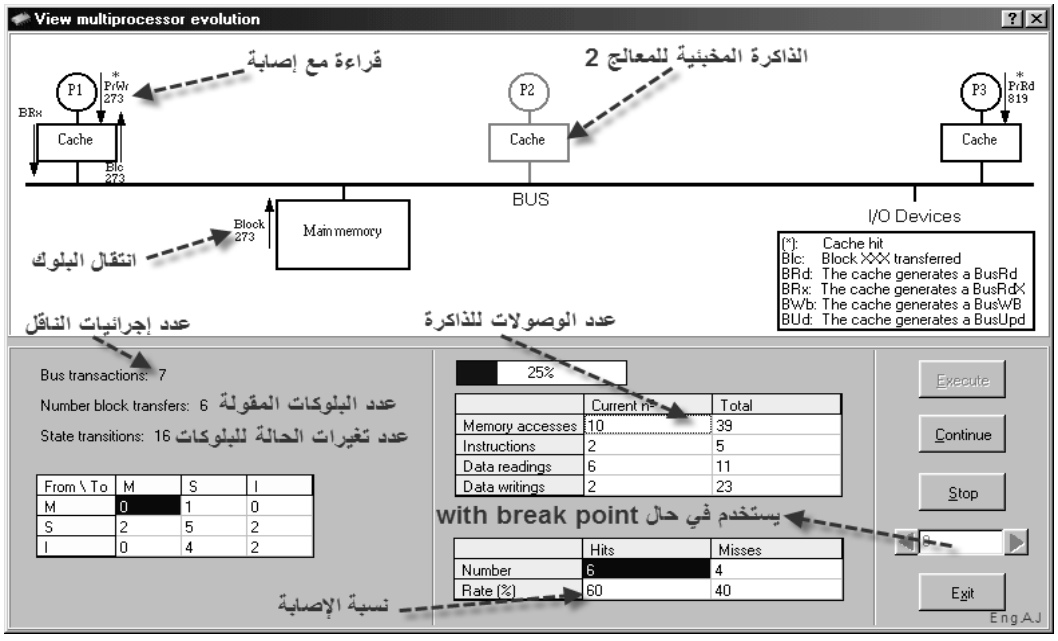
تمثل ملفات الإعدادات "التصميم" (Configurations "Design")، وهي تستخدم في اختيار إعدادات مختلفة لتصميم بنية معينة، إذ تخزن هذه الإعدادات ضمن ملف بلغة ASCII من أجل الاحتفاظ به للتحميل اللاحق، كما هو موضح في الجدول (2). يتم في هذه الحالة كتابة الإعدادات العددية ضمن هذا الملف، مثل عدد المعالجات وطول الكلمة وعدد الكلمات بالبلوك وعدد البلوكات بالذاكرة الرئيسة وعدد البلوكات بالكاش والتقابل (Mapping) وعدد مجموعات الكاش وسياسة الاستبدال (Replacement Policy) وبروتوكول التماسك وتحكيم الناقل (Bus Arbitration).

النتائج التجريبية وتحليلها (Experimental (Results and Analysis

لدى إعداد وتطبيق Benchmarks و Configurations Files (مع عدد رجوعات يساوي 240000) على النظام المصمم وفق الشكل (1)، تظهر لدينا حالة النظام كما هو موضح في الشكل (2)، إذ يسمح هذا الأمر بمراقبة تصور عام لنمو "تطور، تقدم" المعالجات المتعددة (Multiprocessors Evolution)، وذلك حسب إعدادات التصميم.

الجدول (2) خيارات إعدادات النظام

عدد المعالجات	1, 2, 3, 4, 5, 6, 7, or 8
بروتوكول تماسك الكاش	MSI, MESI, or DRAGON
مخطط التحكيم	Random, LRU, or LFU
طول الكلمة	8, 16, 32, or 64 bits
عدد الكلمات بالبلوك	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024
عدد البلوكات بالذاكرة الرئيسة	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096,8192, 16384, 32768, 65536, 131072, 26214, 524288,1048576, 2097152, or 4194304
عدد البلوكات بالكاش	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, or 2048
التقابل	Direct, Set-Associative, or Fully-Associative
مجموعات الكاش	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, or 2048
سياسة الاستبدال	Random, LRU, FIFO, or LFU
استراتيجية الكتابة	Write-Back (for cache coherence protocols used)
مستويات الكاش بهرمية الذاكرة	1
المرجعية	To memory words
حجم البلوك الأعظمي	8KB
حجم الذاكرة الرئيسة الأعظمي	32GB
حجم الكاش الأعظمي	16MB



الشكل (2) عرض نمو المعالجات المتعددة

نلاحظ من الشكل (2) أنه فقط المعالج المحمل ببرنامج تتبع للذاكرة يؤثر على المحاكاة، وقد افترضنا وجود ثلاثة معالجات بنظام SMP، والمعالجين 1 و 3 تم تحميلهما بملفات تتبع. يكون اتجاه إجراء الناقل بشكل دائم من كاش معينة إلى الناقل، ومن المفترض أن كل كاش تراقب كل إجراء للناقل، علماً أننا نستخدم دائماً بروتوكولات التنصت لتماسك الكاش مع MSI، تولد الكاش 1 إجراء الناقل BusRdX من أجل القراءة. طلبات المعالج تذهب أيضاً باتجاه واحد، أي من المعالج إلى الكاش الخاصة به. نفترض أن المعالج يولد نوعين من الطلبات هما القراءة (PrRd) والكتابة (PrWr). كل من جلب التعليمة وقراءة المعطيات يعتبران على أنهما طلب قراءة PrRd. كل من القراءة والكتابة يمكن أن تكونا لبلوك الذاكرة الموجود أو غير الموجود في الكاش الخاصة بالمعالج. تمثل عملية الإصابة بالقراءة مع وجود النجمة (*). تتم عنونة البلوكات المنقولة بالشكل (Block XXX)، حيث XXX هو عنوان البلوك بالذاكرة الرئيسة. على سبيل المثال: بعد تنفيذ الإجراء BusRdX، تعطي الذاكرة الرئيسة بلوك الذاكرة 273.

يبين الجدول (3) الإعدادات اللازمة لإجراء المحاكاة، وسنعمل على تحليل النتائج الحاصلة وفق الآتي: المعالج الذي طلبها[1،7].

الجدول (3) خيارات إعدادات المحاكاة والتطبيق

	حجم الكاش	عدد المعالجات	بروتوكولات التماسك	نظام تحكيم الناقل	طول الكلمة	عدد الكلمات بالبلوك	عدد البلوكات ضمن الذاكرة الرئيسة	التخطيط	مجموعات الكاش	نظام الاستبدال
حجم الكاش	Variable	8	MESI	LRU= Scheme for bus arbitration	16 bits	32	524288	Set-associative	Four-way set associative caches	LRU= Replacement policy
عدد المعالجات	16KB	Variable	MESI							
بروتوكولات التماسك	16KB	8	Variable							

تأثير حجم الكاش (Cache Memory Influence)

من خلال النتائج التي حصلنا عليها والموضحة في الجدول (4)، نستنتج بعد دراستها وتحليلها أن نسبة عدم الإصابة تتناقص مع زيادة حجم الكاش. من أجل حجم كاش ضخم، سيتم استقرار نسبة عدم الإصابة، وهذا يبين عدم إصابة نتيجة التماسك (Coherence Miss)، والتي تكون مستقلة عن حجم الكاش. تبرهن هذه النتائج على أن المعطيات المشتركة تملك حيزاً أقل وموضعية مؤقتة أكثر من باقي أنواع المعطيات. وهذا يعني بشكل عام أن البرنامج التفرعي يبدي موضعية مؤقتة وحيزاً أقل من البرنامج التسلسلي، ونتيجة لذلك من الطبيعي أن تكون نسبة عدم الإصابة في نظام متعدد المعالجات أكبر منها لمعالج مفرد.

الجدول (4) تأثير حجم الكاش على نسبة عدم الإصابة

		حجم الكاش						
		1KB	2KB	4KB	8KB	16KB	32KB	
ملف تقفي الأثر	FFT	27.652	27.652	26.31	25.548	22.787	21.846	نسبة عدم الإصابة
	Simple	76.649	76.649	76.697	64.247	55.264	55.235	
	Speech	89.515	89.515	79.075	73.996	71.527	69.215	
	Weather	60.07	60.07	50.978	43.9	41.996	41.935	

تأثير عدد المعالجات (Processors Number Influence)

من خلال النتائج التي حصلنا عليها والموضحة في الجدول (5)، نستنتج بعد دراستها وتحليلها أن زيادة عدد المعالجات للتطبيقات التفرعية يزيد من نسبة عدم الإصابة. وهذا ممكناً بسبب البروتوكول القائم على الإبطال مثل MESI. فعدد معالجات أكثر يعني إمكانية أكبر ليتشارك عدد أكبر من الذاكرات المخبئية بالبلوك نفسه، وبما أن عملية كتابة واحدة تجبر باقي الذاكرات المخبئية على الإبطال لهذا البلوك، فهذا يسبب عدم إصابة جديدة، أي عدم إصابة نتيجة التماسك.

الجدول (5) تأثير عدد المعالجات على نسبة عدم الإصابة

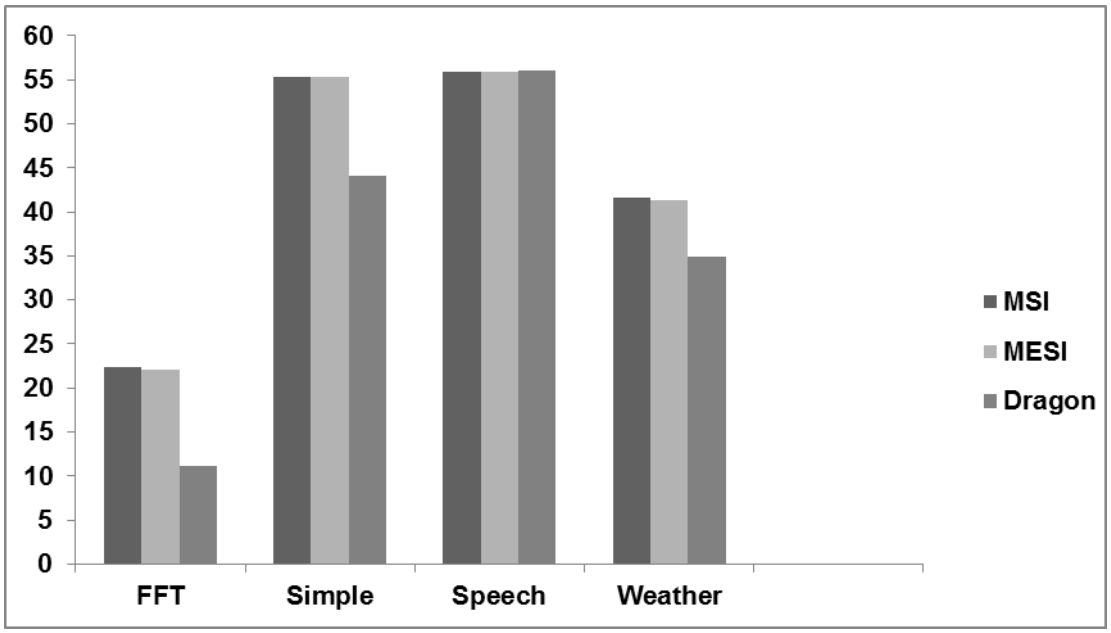
		عدد المعالجات				
		1	2	4	8	
ملف تقفي الأثر	FFT	0.28333	8.1833	12.038	23.21	نسبة عدم الإصابة
	Simple	21.297	36.177	49.508	55.062	
	Weather	7.03	35.968	41.363	50.464	

تأثير بروتوكولات التماسك (Coherence Protocols Influence)

من خلال النتائج التي حصلنا عليها والموضحة في الجدول (6) والشكل (3)، نستنتج بعد دراستها وتحليلها أن نسبة عدم الإصابة من أجل MSI و MESI هي نفسها. وهذا متلائم مع النظرية، لأن MESI هو تحسين لـ MSI (يضيف الحالة Exclusive) من أجل تخفيض عدد عمليات الناقل (Bus Transactions) بسبب بروتوكول التماسك. ولكن على الرغم من أن للبروتوكولين نسبة عدم الإصابة ذاتها، إلا أن MESI يولد تزاحماً أقل على الممر. أيضاً، من الملاحظ أن Dragon يملك أقل نسبة عدم إصابة. وهذا ممكناً، لأن Dragon يعتمد على التحديث، بينما MSI و MESI فهما يعتمدان على الإبطال. في بروتوكولات التحديث، في كل مرة تتم فيها الكتابة إلى موقع بحالة المشاركة من قبل معالج ما، يتم تحديث جميع الذاكرات المخبئية للمعالجات الأخرى التي تملك هذا البلوك. علاوة عن ذلك، يتم في Dragon تعديل التحديث للكتابة لكلمة واحدة فقط (للكلمة المعدلة فقط) بدلاً من كامل البلوك المنقول. بالمقارنة مع بروتوكولات الإبطال، عند عملية الكتابة يتم وضع حالة البلوك invalid بكافة الذاكرات المخبئية الأخرى، لذلك فإن هذه المعالجات ستحصل على البلوك عبر عدم الإصابة، وبذلك ستسبب تزاحماً أكبر على الممر.

الجدول (6) تأثير بروتوكولات التماسك على نسبة عدم الإصابة

		بروتوكول التماسك			
		MSI	MESI	Dragon	
ملف تقفي الأثر	FFT	22.405	22.149	11.170	نسبة عدم الإصابة
	Simple	55.256	55.264	44.073	
	Speech	55.964	55.972	56.068	
	Weather	41.568	41.379	34.887	



الشكل (3) تأثير بروتوكولات التماسك على نسبة عدم الإصابة

يمكن إيجاز ما توصلنا إليه من خلال توصيف العلاقة بين نسبة عدم الإصابة المرتبطة بحجم الكاش وعدد المعالجات المتعلقة بسرعة المعالجة وبروتوكولات التماسك المتأثرة بعرض حزمة الممر. وبالتالي، للحصول على الأداء المناسب من نظام متعدد المعالجات، يجب تحقيق التوازن بين طرفي العلاقة التالية[1]:

$$V(1-h).N \le BI \quad \text{.....(1)}$$

N: عدد المعالجات.
h: معدل الإصابة في الكاش.
B: عرض الحزمة.
(1-h): معدل عدم الإصابة.
I: دورة تشغيل المعالج (Fetches/Cycle).
V: سرعة المعالج (Fetches/Sec).
BI: عرض الحزمة الفعال (Fetches/Sec).

بناءً على المعادلة (1)، يكون معدل عدم الإصابة في الكاش V(1-h) من أجل معالج واحد، و V(1-h).N

من أجل N معالجاً، ما يعرض الممر إلى الإشباع (عنق الزجاجة) لدى محاولة جميع المعالجات الوصول إلى الممر. لمنع ظهور مشكلة عنق الزجاجة، ولكي يصبح النظام فعالاً، يجب أن يتحقق شرط هذه المعادلة. وهذا هو الهدف من بحثنا الذي تمثل بتحديد وتحليل البارامترات المرتبطة بذلك، وذلك لاختيار الأنسب مادياً وبرمجياً حسب خصائص النظام المراد تحسينه أو تصميمه، وبالتالي إيجاد الحل الأمثل بما يحقق متطلبات النظام من حيث سرعة الأداء وقلة استهلاك الطاقة والحجم والكلفة (Cost)، وهذا هو المعيار الأساسي والمرجعي في تقاطع الأفكار والمقارنة بينها.

لتوضيح العلاقة بين نسبة عدم الإصابة المرتبطة بحجم الكاش وعدد المعالجات المتعلقة بسرعة المعالجة وبروتوكولات التماسك المتأثرة بعرض حزمة الممر وفق العلاقة (1)، سنقوم بإجراء التطبيق التالي:

لنفترض أنه لدينا نظام ذاكرة مشتركة يتمتع بالمواصفات الآتية:

معالجات تعمل بسرعة V=107 Instructions/Sec.

دورة تشغيل I=1.

ذاكرات مخبئية بمعدل إصابة 97%.

عرض حزمة BI=106 Cycle/Sec.

والمطلوب حساب:

1- معدل عدم الإصابة وعدد المعالجات؟

2- نسبة الإصابة التي تدعم ثلاثين معالجاً؟

سنعمل على إيجاد وتوضيح المطلوب وفق الآتي:

1- معدل عدم الإصابة: (1-h)=0.03.

عدد المعالجات: N≤106/(0.03*107)=3.33.

وبالتالي، النظام الذي لدينا يمكن أن يدعم ثلاثة معالجات فقط.

2- نسبة الإصابة:

h=1-BI/N.V=1-(106(1))/((30).(107))= 1-1/300=0.9967.

نلاحظ تزايد نسبة الإصابة بمعدل 2.8% نتيجة مضاعفة عدد المعالجات بمقدار عشرة.

الخاتمة (Conclusion)

النتائج الحاصلة تركز على الآليات المستخدمة للتوفيق بين زيادة أداء الحاسوب مقابل السعة المثالية للكاش، وكيفية إدارة واستثمار ذلك على مستويات عدة. وذلك سعياً في الحفاظ على معدل إصابة عالٍ أو معدل عدم إصابة منخفض في الذاكرات المخبئية الموجودة ضمن المعالجات، ما يساهم في زيادة الأداء. وبالنتيجة، تمثل الكاش حلاً مناسباً في الأنظمة التفرعية والمعقدة التي تتطلب العمل في الزمن الحقيقي.

المصادر (References)

1- Richard S. Morrison, “Cluster Computing: Architectures, Operating Systems, Parallel Processing, and Programming Languages”, GNU General Public License [39], 2003

2- Michael Kistler, Michael Perrone, Fabrizio Petrini, “Cell Multiprocessor Communication Network: Built for Speed”, IEEE Computer Society, 0272-1732/06/2006.

3- Dhruba Candra, Fei Guo, Seongbeom Kim, Yan Solihin, “Predicting Inter-Thread Cache Contention on a Chip Multi-Processor Architecture”, International Symposium on High Performance Computer Architecture (HCPA), Feb 2005.

4- Ahmed Amine Jerraya, Wayne Wolf, “Multiprocessor Systems-on-Chips (MPSoC) Technology”, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol. 27, No.10, October 2008.

5- Francis Cottet, Emanuel Grolleau, “Systèmes Temps Réel de Contrôle-Commande, Conception et Implémentation”, Dunod, 2005.

6- John L. Hennessy, David A. Patterson, “Computer Architecture: a Quantitative Approach”, Morgan Kaufmann, 2007.

7- Hesham El-Rewini, Mostafa Abd-El-Barr, “Advanced Computer Architecture and Parallel Processing”, Wiley, 2005.

8- Rakesh Kumar, Victor Zyuban, Dean M. Tullsen, “Interconnections in Multi-core Architectures: Understanding Mechanisms, Overheads and Scaling”, 0-7695-2270-X/05 IEEE 2005.

9- Daniel J. Sorin, Milo M. K. Martin, Mark D. Hill, David A. Wood, “Safety Net: Improving the Availability of Shared Memory Multiprocessors with Global Checkpoint/Recovery”, 29th Annual International Symposium on Computer Architecture (ISCA-29), Anchorage, AK, May 25-29, 2002.

10- Jadhav, S. S., “Advanced Computer Architecture and Computing”, Technical Publications Pune, 2009.