قياس الأداء في البنى المتوازية للمعالجات التفرعية

الدكتور رزق غانم[1]

الملخص

إن من أكبر المشجعات على استخدام النظم التفرعية هو تقليص زمن التنفيذ الخاص بالحسابات للبرامج ويعتمد زمن التنفيذ للبرنامج التفرعي على عوامل عدّة متضمنة بنية منصة التنفيذ، والمفسر، ونظام التشغيل وبيئة البرمجة التفرعية ونموذج البرمجة التفرعية المستخدم وحتى خصائص البرنامج التطبيقي كمكان الرجوع إلى الذاكرة الخاص بالبرنامج والتبعيات بين الحسابات التي ستنفد في الواقع. هذه العوامل كلّها يجب أن تؤخذ بالحسبان عندما يتم تطوير البرمجيات التفرعية وفي الحالات كلّها فإنَّه من الصعب الجمع بين تلك العوامل معاً.

ولتنفيذ تطوير البرمجيات التفرعية وتحليلها يستخدم غالباً قياس الأداء أو الفعالية، لذلك اِسْتُخْدِمَ في هذا البحث هذا القياس لتحليل النسخ المختلفة ومقارنتها من البرمجيات التفرعية .

الكلمات المفتاحية: المعالجات المتوازية ذات الأنابيب والمعالجات الفائقة، المعالجات الشعاعية، البرمجة التفرعية، مستوى المعطيات التفرعية.

[1] دكتور في قسم الحواسيب والأتمتة، كلية الهندسة الميكانيكية والكهربائية، جامعة دمشق سورية.

Measurement in parallel structures of multiprocessors

[1]Dr. Rzek Ghanem

Abstract

One of the most encouraging uses of parallel systems is to reduce the execution time of the calculation of the programs. The implementation time for the sub-program depends on several factors including the structure of the implementation platform ,interpreter, the operating system, the parallel programming environment, and model used. Even the characteristics of the applied software as a reference to the program's memory and the dependencies between the calculations that will actually run. All these factors must be taken into account when the parallel software is developed, though it is very hard to combine all these factors together.

In order to implement the development and analysis of the parallel software, it is often used to measure performance or effectiveness which will used to analyze and compare the different versions of the parallel software.

Key words: Parallelism, Pipelining, Superscalar processor, Vector processor, Data Level Parallelism, parallel environment.

[1]Assistant Professor –Department of Computer& Automation Engineering- Electrical faculty Damascus university.

1-مقدمة

لطالما دعت الحاجة إلى ابتكار حلول ذات وثوقيه عالية وسريعة لمشاكل هندسية عدّة ذات مقياس كبير، مكنت التطورات السريعة التي حدثت في السنوات الأخيرة في تكنولوجيا الحاسب الآلي والاتصالات من تأمين الدعم التقني الأساسي لمشكلات عدّة في مجال المعالجة المتوازية والمعالجة الموزعة التي كانت سابقاً تستهلك كثيراً من الوقت عند معالجتها بالطرق التقليدية المتسلسلة. تُنفَّذ الحسابات المتوازية عادةً على آلة متوازية، أو على SAN (شبكة لكامل النظام system area network)، في حين تعتمد الحسابات الموزعة والتحكم بشكل عام على الشبكات المحلية LAN والشبكات الواسعة WAN

(wide area network)، وعلى الإنترنت

2-هدف البحث

هَدَفَ هذا البحث إلى دراسة تطور الأداء الخاص بنظم الحواسيب التي تعمل على التوازي و العلاقات المبينة لتطور وحدة المعالجة المركزية فضلاً عن معياري MIPS & MFLOPS، وفعالية المعالجات التي تستخدم هرمية الذواكر، ثم مقاييس الأداء للبرامج التفرعية.

3-أنظمة المعالجة المتوازية والمعالجة الموزعة

كانت المعالجة المتوازية والمعالجة الموزعة ومازالت أحد موضوعات البحث المهمة منذ عدة عقود. على الرغم من سرعة الحواسب الالكترونية الخارقة

(super computer) ذات المعالج الوحيد، إلّا أنَّ كلفتها المادية عالية جداً ويتعلق أداؤها بسعة ذواكرها. مع التطور المطرد في تكنولوجيات الحاسوب (الكمبيوتر) والاتصالات، يُسْتَبْدَلُ بالحواسب الالكترونية الخارقة ذات المعالج الوحيد عناصر معالجة متوازية ومعالجة موزعة أقل كلفة و أكثر فعالية.

3-1 الأنظمة المتوازية

يمثل النظام المتوازي التركيب الفيزيائي من عملية المعالجة المتوازية. ونميز بين نوعين من الأنظمة المتوازية. الأول هو الآلة المتوازية، والثاني هو عبارة عن شبكة حاسوب (كمبيوتر) مثل SAN مخصصة لأجل المعالجة المتوازية. كلا النوعين يتألف من عدد من المعالجات المرتبطة مع بعضها بعضاً في حيز فيزيائي صغير. يوجد كثير من الآلات المتوازية المتوافرة في الأسواق، منها الحواسب الالكترونية الضخمة mainframe مثل Cray، أو الحواسب المتوازية الشاملة مثل SGI. فضلاً عن الحواسب المصنعة خصّيصى والمزودة بمعالجات متعددة. عند استخدام شبكة كمبيوتر في عمليات المعالجة المتوازية، فإنها تمثل آلة متوازية وهمية بحيث تكونْ الحواسب جميعها الموجودة على الشبكة معالجات لهذه الآلة الوهمية أو الافتراضية.

إن روابط الاتصال بين معالجات نظام متوازٍ تكون عادةً قصيرة جداً؛ على سبيل المثال، المعالجات لآلة متوازية تتوضع على اللوحة الأم نفسها. إيصال المعلومات بين معالجات النظام المتوازي موثوق للغاية، وإذا أردنا أخذ التأخير الزمني لنقل المعلومات بعين الاعتبار، فمن الممكن التنبؤ به. إن النية الأساسية من استخدام النظام المتوازي هي زيادة سرعة العمليات الحاسوبية؛ وذلك عبر تشغيل أكثر من معالج في آنٍ واحد. بكلمات أخرى، الهدف الوحيد من تطبيق النظام المتوازي هو الحصول على حل سريع عبر السماح لعدة معالجات بالعمل بوقت متزامن على المهمة المطلوبة.

3-2 الأنظمة الموزعة

بشكل مشابه للنظام المتوازي، يشكّل النظام الموزع التدابير الفيزيائية المتخذة لأجل عملية المعالجة الموزعة. إلّا أنَّه يختلف عنه بأنَّ النظام الموزَّع يتألف عادةً من

شبكة حاسوب موزعة جغرافياً على مساحة كبيرة. الحواسب الالكترونية للنظام الموزع ليس بالضرورة أن تكون متشابهة من حيث الأداء. يمكن أن يستخدم النظام الموزَّع لأجل عملية اكتساب المعلومات؛ على سبيل المثال، يمكن أن يكون النظام الموزع عبارة عن شبكة من الحساسات لأجل القيام بعمليات قياس بيئية، إذْ تقوم مجموعة من الحساسات الموزعة جغرافياً بالحصول على معلومات عن حالة البيئة في تلك المنطقة، فضلاً عن إمكانية معالجة هذه المعلومات. يمكن أن يستخدم النظام الموزع لأجل عمليات التحكم و الأتمتة لأنظمة واسعة الانتشار (large scale) مثل حجوزات الطيران وأنظمة التنبؤ بالأحوال الجوية.

إن الرسائل المنتقلة عبر شبكة اتصالات البيانات للنظام الموزع يُتحكم بها بنمط موزع عبر مجموعة من حواسب النظام الموزع. عادةً تكون خطوط الاتصال بين الحواسب الالكترونية في النظام الموزع طويلةً جداً كما تُبَدَّلُ بيانات الاتصالات مرات عدّة والتي يمكن أن تتعرض لتشويش بسبب عدة إشارات ضجيج. بشكل عام، يُصمم النظام الموزع بحيث يكون قادراً على العمل بشكل صحيح مع وجود روابط اتصال محدودة، غير الموثوق بها أحياناً، وغالباً مع عدم وجود آلية تحكم مركزي. كما أنَّ التأخير الزمني الحاصل عند نقل البيانات بين الحواسب الموزعة يمكن أن يكون من المستحيل التنبؤ به، خاصة مع عملية معالجة موزعة، التي لها متطلبات زمنية صارمة مثل التحكم بالتوتر أو الاستطاعة في أنظمة القدرة.

3-3 مقارنة بين المعالجة المتوازية والمعالجة الموزعة

في الماضي كان الهدف الوحيد من المعالجة المتوازية هو الحصول على حل أسرع. العمليات الحاسوبية المتوازية يمكن أن تُنفّذ عن طريق الحواسب الالكترونية العملاقة التفرعية(المتوازية) مثل Cray، أو عن طريق الحواسب الالكترونية المتوازية مثل Silicon Graphic Systems، أو عن طريق الحواسب الشخصية المزودة بمعالجات عدّة.

وعادةً ما نجد أنَّ اتصالات البيانات في أنظمة المعالجة المتوازية موثوق بها جدّاً، وذلك لأنَّ هذه الخطوط تكون قصيرة جداً في مثل هذه الأنظمة. لذلك فإنَّه غالباً ما تصمم خوارزميات التوازي بحيث تكون متزامنة تماماً، أمَّا في حالة تطبيق اللامتزامن فكي نضمن التزامن في كل عملية تكرار، يجب على كل معالج ألا ينتقل إلى التكرار التالي حتى يستقبل جميع البيانات المفروض إرسالها من قبل المعالجات الأخرى المعنية.

فالحواسيب المتوازية هي تقنية استخدام أكثر من حاسب أو استخدام حاسب بأكثر من معالج لحل المشكلة. والدافع لاستخدامها هو تنفيذ الحسابات بطريق أسرع فيستطيع n حاسب يعمل بوقت واحد حساب النتيجة أسرع n مرة، وكمية كبيرة من الذاكرة المتوافرة.

4-عامل التسريع speedup factor

$$S(n) = \frac{\text{Execution time using one processor (single processor system)}}{\text{Execution time using a multiprocessor with n processors}} = \frac{t_s}{t_p}$$

إذْ Ts زمن التنفيذ لمعالج واحد و TP زمن التنفيذ لمعالجات عدّة ويظهر من النسبة التالية ان S(N) تزداد عند استخدام معالجات عدّة.

وتحسب S(n) كالآتي:

$$S(n) = \frac{\text{Number of computational steps using one processor}}{\text{Number of parallel computational steps with n processors}}$$

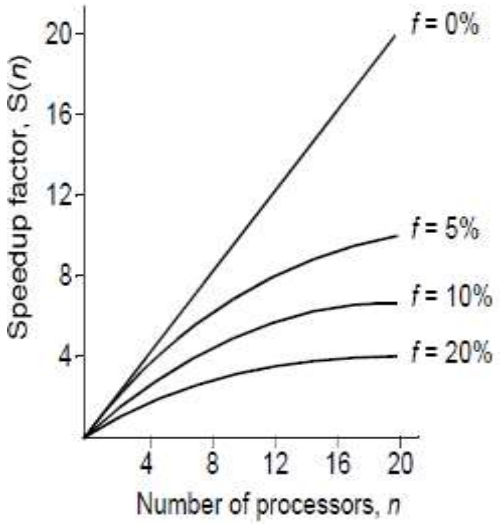
التسريع الأعظمي عند استخدام n معالج(تسريع خطي)

4-1-التسريع الأعظمي حسب قانون امدال:

يحسب عامل التسريع كالآتي:

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1+(n-1)f}$$

التسريع حسب عدد المعالجات: يبيّن الشكل (1) التسريع كتابع لعدد المعالجات



الشكل (1) التسريع حسب عدد المعالجات

يلاحظ أنَّه حتى عند استخدام عدد لانهائي من المعالجات فإنَّ عامل التسريع الأعظمي يكون محدوداً بـ 1/f

5-تحليل قياس الأداء للبرامج التفرعية [6] Performance Analysis of Parallel Programs

إن أداء نظام الحاسب هو من أهم العوامل المسهمة في تطور الحواسيب وبحسب وجهات النظر فإنَّ هناك العديد من القياسات التي يمكن أن نحدد من خلالها الفعالية.

فمثلاً فإن مستخدمي الحاسب يهتمون بزمن الاستجابة الأصغري إذْ يعرف زمن الاستجابة بأنَّه الوقت بين بدء تنفيذ البرنامج وانتهاء تنفيذ البرنامج.

من وجهة نظر أخرى فإنَّ مراكز الحسابات الضخمة تهتم بمردود خرج عالٍ إذْ يعرف مردود الخرج بأنَّه الرقم الوسطي لوحدات العمل التي نستطيع تنفيذها في وحدة الزمن.

1.5 اختيار معايير دراسة أداء نظام وتقييمه [2]

لدراسة أداء أنظمة أو لمقارنة الأنظمة المختلفة لغرض معين، يجب علينا أولاً تحديد بعض المعايير ،التي غالبا ما تسمى هذه المعايير المقاييس في تقييم الأداء.

تحتاج الحالات المختلفة مجموعات مختلفة من المقاييس. وهكذا، مقاييس اختيار المشكلة هي مهمة جداً. وما هو أكثر من ذلك، قد يكون للقياس نفسه أوزان مختلفة في حالات مختلفة. فعلى سبيل المثال، يستخدم عادة زمن الاستجابة للقياس، والذي يعكس مدى سرعة النظم في إكمال الخدمات المقدمة من قبله في نظم التحكم في الوقت الحقيقي مما كانت عليه في نظم الشبكة الذي قد يكون أكثر اهتماماً في مقاييس مثل الإنتاجية، وعند تصميم أي نظام لتقديم الخدمات يجب أن تدرس مختلف الخدمات واحدة تلو الأخرى.

عند تقديم خدمة ما طلبت من النظام، سيكون لدينا ثلاث نتائج محتملة:

(1) يكمل النظام الخدمة بشكل صحيح؛

(2) يكمل النظام الخدمة مع الخطأ؛

(3) يخفق النظام في أداء الخدمة.

نحتاج في كل حالة من الحالات الثلاث السابقة مجموعة من المقاييس الفردية. ويمكن عدّ كل منهما على حدة . بالنسبة إلى الحالة الأولى يكمل النظام الخدمة بشكل صحيح وهنا يمكن عادة للأداء أن يقاس بواسطة مقياس الوقت نسبة للموارد الذي يمثل الوقت الذي يستغرقه النظام لإكمال الخدمة والمعدل الذي ينفذ به الخدمة والموارد اللازمة للنظام لإكمال الخدمة، وفي هذه الحالة

فإن زمن الاستجابة، والإنتاجية، و مقدار الإفادة أو الخدمية هي ما تستخدم على نطاق واسع في دراسات الأداء، أمَّا في الحالة الثانية يكمل النظام الخدمة مع أخطاء هنا يجب قياس احتمالية حصول الأخطاء، وفي أحيان عدّة يجب تصنيف هذه الأخطاء ودراسة كل صنف على حدة، أمَّا في الحالة الأخيرة يخفق النظام في تحقيق الخدمة فإنَّ احتمالية الخطأ هي من القياسات المهمة إذْ أنَّ أن النظام يتكون من عدد من المكونات فإنه من المهم لنا تحديد مصدر حصول الأخطاء.

6.معياري MIPS & MFLOPS

إن قياس الأداء المستخدم في أنظمة الحواسب يدعى ب MIPS rate(Million Instruction Per Second) (مليون تعليمة بالثانية) سنرمز لعدد التعليمات في البرنامج A بالرمز $n_{instr}(A)$ وبزمن وحدة المعالجة المركزية الخاص بالمستخدم بـ $T_{U-CPU}(A)$عندها نعرف ال MIPS rate للبرنامج A بالعلاقة

$$MIPS(A) = \frac{n_{instr}(A)}{T_{U-CPU}(A).10^6}$$

التي يمكن تحويلها إلى العلاقة :

$$MIPS(A) = \frac{r_{cycle}}{CPI(A).10^6}$$

إذْ $r_{cycle} = 1/t_{cycle}$ هو معدل نبضات الساعة للمعالج ولهذا فإنه كلما كان المعالج أسرع هذا يعني معدلات MIPS أكبر ولأنَّ CPI عدد الدورات الخاص بالتعليمة يختلف من برنامج إلى آخر فإنه من الممكن أن يعتمد معدل ال MIPS أيضا على البرنامج A.

يعتبر تقييم MIPS للأداء من التقييمات الضعيفة نظراً إلى أنّه يأخذ بالحسبان فقط عدد التعليمات إذْ إنَّ التعليمات القوية والفعالة غالبا تحتاج لزمن تنفيذ أطول ولهذا يفضل هذا المعيار للمعالجات ذات التعليمات البسيطة .

ثانيا إن معدل MIPS الخاص بالبرامج لا يتوافق بالضرورة مع زمن التنفيذ الخاص بها

من أجل المبرمجين الذين يهتمون بالحسابات العلمية فإننا نهتم بمعيار MFLOPS (Milloion Floating-point Operation Per Second) إذْ يعرف بالعلاقة

$$MFLOPS(A) = \frac{n_{flp-op}(A)}{T_{U-CPU}(A).10^6} \ [1/s]$$

إذْ $n_{flp-op}(A)$ هو عدد الفواصل العشرية المنفذة من قبل البرنامج A هذا المعيار لا يعتمد على عدد التعليمات المنفذة كما معيار MIPS ولكن على عدد العمليات الحسابية ذات الفاصلة العشرية المنفذة خلال تنفيذ التعليمة بينما لا تملك التعليمات التي لا تمتلك عمليات على الفواصل العشرية أي تأثير في هذا المعيار ،إذْ إنَّ عدد العمليات الفعال والمنفذ هو المستخدم في معيار MFLOPS فهو يوفر مقارنة عادلة نوعا ما بين نسخ البرامج المختلفة التي تقوم بتنفيذ نفس العمليات ويشير معدل ال MFLOPSالأعلى إلى زمن تنفيذ أسرع، وفي الحالات كلّها

فإنَّ هذا المعدل لا يعطي نتائج دقيقة عند مقارنة البرامج التي تحتوي على عمليات كالقسمة والجذر التربيعي، نظرا إلى أنَّ حسابات قيم هذه العمليات تأخذ زمناً طويلاً، وتحسب بالطريقة نفسها دوماً. وبشكل عام فإنَّ معيار الـ MFLOPS مناسب لمقارنة البرامج التي تنفذ العمليات نفسها ذات الفاصلة العائمة.

7.مقاييس الأداء للبرامج التفرعية[5]

إن زمن التشغيل program runtime الخاص ببرنامج على منصة تشغيل معينة هو عامل مهم لقياس أداء وفعالية البرنامج التفرعي ،ويعرف زمن التنفيذ التفرعي parallel runtime Tp(n) لبرنامج بأنَّه الزمن الفاصل بين بدء البرنامج وانتهاء تنفيذ البرنامج في المعالجات المشتركة

بتنفيذه جميعها، ويعرف هذا الزمن عادة بالنسبة إلى عدد منتهٍ من المعالجات المشتركة في تنفيذ البرنامج وقدره مثلاً p، وحسب بنية ومنصة التشغيل فإنَّ زمن التشغيل يمكن أن يجمع الأزمنة الآتية:

1-زمن التشغيل الخاص بتنفيذ الحسابات المحلية الخاصة بكل معالج مشترك بالتنفيذ أي الحسابات التي يقوم بها كل معالج بشكل منفصل باستخدام البيانات الموجودة في ذاكرته المحلية

2-زمن التشغيل اللازم لتبادل البيانات بين المعالجات، مثلاً تُنَفَّذُ عمليات الاتصال بين المعالجات في حال وجد لدينا فضاء عنونة موزع.

3- زمن التشغيل اللازم للمزامنة بين المعالجات المشتركة عند محاولة الوصول إلى بنى بيانات مشتركة مثلاً فضاء عنونة مشترك.

4-أزمان الانتظار التي تحصل بسبب توزيع الأحمال غير المتكافئ بين المعالجات أو بسبب انتظار المعالجات دورها للوصول إلى معلومات مشتركة لضمان عدم حصول تضارب.

8.التسريع وقياس الفعالية[5],[6]

تعدُّ كلفة برنامج تفرعي $C_p(n)$ لحجم دخل n يُنفَّذُ بواسطة p معالج بالشكل الآتي:.$C_p(n) = p \cdot T_p(n)$

إذْ إنَّ $C_p(n)$ هو عبارة عن قياس المقدار الكلي للعمل المنجز من قبل المعالجات كلِّها؛ ولذلك فإنَّ كلفة البرنامج تدعى أيضاً بالعمل، أو بنتاج زمن التشغيل للمعالج

نقول عن البرنامج التفرعي بأنَّه مثالي الكلفة – cost optimal إذا كان:

$C_p(n) = T^*(n)$هذا يعني أنَّه إذا تم تنفيذ العدد الكلي نفسه للعمليات برنامج تسلسلي له زمن تشغيل $T^*(n)$

1.8 التسريع[2]

من أجل عملية تحليل البرامج التفرعية يمكن إجراء عملية مقارنة بين أزمان التنفيذ التفرعية مع زمن التنفيذ التسلسلي، وهذه المقارنة تتم عادة عبر حساب نسبة التسريع speed up $S_p(n)$

ويمكن تعريف نسبة التسريع الخاصة ببرنامج تفرعي بزمن تنفيذ تفرعي $T_p(n)$ بالعلاقة:

$S_p(n) = \frac{T^*(n)}{T_p(n)}$ ،

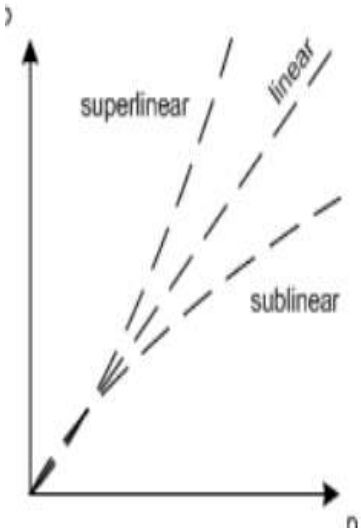
إذْ إنَّ p هو عدد المعالجات المشتركة بالتفريع التي تحاول حل مشكلة ذات حجم n و $T^*(n)$ هو زمن التنفيذ لأفضل تنفيذ تسلسلي ممكن لحل المشكلة المطروحة نفسها.

يظهر التسريع للتنفيذ التفرعي نسبة حفظ زمن التنفيذ الذي يمكن الحصول عليه باستخدام التنفيذ التفرعي ل p معالج معال على التفرع بالمقارنة بأفضل تنفيذ تسلسلي ممكن نظرياً تتحقق العلاقة $S_p(n) \leq p$

إنَّ البحث عن الخوارزمية التسلسلية الفُضْلَى قد لا يكون بالأمر السهل مطلقا إذْ إنَّ الخوارزمية قد لا تكون معروفة بعد، أو غير محددة بالنسبة إلى مشكلة ما. ولسبب آخر قد توجد خوارزمية، ولكن يختلف زمن التنفيذ الخاص بها حسب الدخل المعطى لها، أو لها شروط معينة على الدخل كي تعمل بكفاءة، وقد يحتاج تنفيذ الخوارزمية إلى كثير من الجهد المعقد.

لأجل الأسباب السابقة يحسب التسريع غالباً باستخدام نسخة تسلسلية من التنفيذ التفرعي عوضاً عن أفضل خوارزمية تسلسلية ممكنة. وعملياً عندها يمكن أن يتحقق لدينا حالة superliner speedup، أي تتحقق المعادلة $S_p(n) > p$ والسبب هو سلوك ذاكرة الكاش، لنفرض أنَّنا سنستخدم التنفيذ التفرعي هذا يعني أنَّ البرنامج سيقوم بإعطاء كل معالج قسماً من البيانات ويتم اختيار هذا التقسيم بحيث يقوم المعالج بالحسابات المطلوبة على جزء

البيانات الخاص به، ومن الممكن أن يتم ملاءمة البيانات مع الكاش الوحيدة الموافقة لمعالج وحيد عند التنفيذ تسلسلياً هذا يؤدي إلى حصول إخفاقات في الكاش خلال الحسابات ولكن عندما يكون لدينا عدد من المعالجات تقوم بتنفيذ البرنامج نفسه، بالكمية نفسها من البيانات على التفرع عندها من الممكن ملاءمة قطع البيانات مع ذاكرة الكاش المحلية لكل معالج؛ وهذا يقود إلى تقليل عمليات إخفاق الكاش، ولكن هذه الحالة لا تحصل بالغالب ولا حتى حالة ال linear speedup التي يتحقق فيها $S_p(n) = p$ إذْ نحتاج إلى الاهتمام أكثر بإدارة التفرع من حيث تبادل البيانات بين المعالجات، وتزامن نقل البيانات بينها، ومراقبة أزمان الانتظار. وبشكل أو بآخر قد يحتاج تنفيذ البرنامج بشكله التفرعي عدداً من العمليات يفوق تنفيذ البرنامج نفسه بشكله التسلسلي، والشكل (2)يوضّح العلاقة بين عدد المعالجات P ونسبة التسريعSp



الشكل (2)العلاقة بين عدد المعالجات P ونسبة التسريعSp

2.8 الفعالية

هناك قياس مرادف للأداء للبرامج التفرعية هو قياس الفعالية، إذْ تقيس الفعالية التقسيم الزمني الأفضل بين المعالجات وتعرف الفعالية بشكلها المبسط بأنّها نسبة التسريع على عدد المعالجات التفرعية إذْ يتحقق الاستخدام الفعال لهذه المعالجات. ويمكننا أيضاً التعبير عن الفعالية بواسطة تابع الكلفة إذْ يكون

$$E_p(n) = \frac{T^*(n)}{C_p(n)} = \frac{S_p(n)}{p} = \frac{T^*(n)}{p.\,T_p(n)}$$

إذْ$T^*(n)$ هو زمن التنفيذ التسلسلي لأفضل خوارزمية تسلسلية و$T_p(n)$ هو زمن التنفيذ التفرعي الخاص ب p معالج

1.2.8 قانون أمبدال Amdahl's Law

من غير الممكن إنقاص الزمن اللازم لتنفيذ برنامج تفرعي عبر تخصيص مصادر تفرعية فقط، وكما رأينا فإنَّ عدد المعالجات يشكل حدّاً للتسريع الذي يمكننا الحصول عليه حد أخر هو تبعية البيانات التي تنفذها الخوارزمية والذي يمكن أن يحد مندرجة التفريع.

يقول قانون أمبدال عندما يجب تنفيذ جزء من برنامج بشكل تفرعي فإنَّ زمن التنفيذ التفرعي للبرنامج مكون من الكسر المكون لجزء تنفيذ البرنامج f إذْ 0<=f<=1 , مضروباً ب T*(n)، وزمن تنفيذ جزء الكسر المتبقي 1-f مفرعاً على p معالج هو $\frac{1-f}{p}$ T*(n) عندها تكون نسبة التسريع معطاة بالعلاقة الآتية:

$$S_p(n) = \frac{T^*(n)}{f.T^*(n)+\frac{1-f}{p}T^*(n)} = \frac{1}{f+\frac{1-f}{p}} \leq \frac{1}{f}$$

وحسب قانون أمبدال فإن عدد المعالجات لا يهم بقدر أجزاء البرنامج التي يجب أن تقسم وتنفذ تفرعياً

9.التوسعية في البرامج التفرعية [2][1]

مفهوم التوسعية: وهو قياس يوصف مدى تحسن الفعالية الذي يمكن أن نحصل عليه حسب الزيادة في عدد المعالجات، إن التوسعية تعتمد على العديد من العوامل كالخوارزميات والتنفيذ التفرعي وبالغالب فإنَّه من أجل

مشكلة ذات حجم محدود ندرس تأثير زيادة عدد المعالجات في سرعة حل هذه المشكلة، وتأتي أهمية التوسعية من المقدرة على معرفة هل كانت البرامج التي تحل مشكلات كبيرة ستنفذ بالسرعة نفسها من أجل برامج بمشكلات أصغر، إذْ نحاول أن تبقى الفعالية ثابتة عند ثبوت عدد المعالجات وحجم المشكلة.

إنَّ تحليل التوسع يؤدي دوراً مهماً في تقييم أداء النظم المتوازية الكبيرة الذي يدرس كيفية الاستخدام الفعال للمعالجات المقدمة من قبل النظام وعادة كلما كانت هذه المعالجات أكثر زادت الكفاءة .

10-التقنية المستخدمة

استخدمنا في هذا البحث برنامج فيجوال ستديو 6[٢٠١٠] [8] وذلك لعدة أسباب أهمها:

أنَّ استخدام أساس اتصالات ويندوز

(Windows Foundation Communication) التي تعدُّ واجهة برمجة تطبيقات

(Application Programming Interface) تستخدم لبناء تطبيقات موزعة. وتعدُّ هذه الخدمة جزءاً من إطار بيئة دوت نت، إذ توفر نموذجاً لواجهة برمجية موحدة لبناء تطبيقات موجهة تكون موزعة وتتصل عبر الويب[7] .كما أنَّ برنامج فيجوال ستديو 6 متطور يدعم برمجة الشبكات والويب، وهو سهل لبناء خدمات ويب (Web Services) تعمل على شبكة الإنترنت، كما له إمكانية بناء برامج تطبيقات موزعة تعمل على الشبكات المحلية.

11-مكونات النظام

يتكون النظام المقترح في هذا البحث من ثلاثة أقسام رئيسية هي:

• العميل (Client): البرنامج الرئيس الذي يتعامل معه المستخدم.

• المخدم (Server): البرنامج الذي يتعامل معه العميل، وليس للمستخدم دخل فيه.

• حامل المخدم (Server Loader): برنامج يقوم بتحميل المخدم وتشغيله في المحطات، إذْ تتمثل مهمة حامل المخدم في العمل في كل الأجهزة المشاركة في الحزمة (Cluster) وتحميل المخدم فيها. أمَّا المخدم فيختلف باختلاف البرنامج المتوازي (إذْ يكون لكل تطبيق متوازٍ مخدم خاص به)، ويستطيع المبرمج إضافة مخدم من واجهة المبرمج وتسجيله بالنظام ممَّا يجعله متاحاً للمستخدم العادي. بهذه الطريقة ستتكون لدينا مكتبة من البرامج المتوازية في شكل كائنات موزعة (Distributed Objects) يمكن للمستخدم العادي استدعائها دون أن يعرف كيف كتبت، وإنَّما فقط يدخل المعطيات المطلوبة. أمَّا مهمة توزيع المخدم في الشبكة فهي من اختصاص حامل الخادم الذي يقوم بتوزيع المهام بين الأجهزة المتوفرة بطريقة Round Robin. بالنسبة إلى العميل فهو يتصل بالمخدم من خلال حامل المخدمً، باختلاف المخدم وهذا ما يميز النظام بحيث يكون الاتصال مختلفاً في الأجهزة المكونة للحزمة الذي يتحدد بواسطة المستخدم للنظام.

الفكرة التي تمت تجربتها بالنظام مبنية على برنامج واحد وبيانات متعددة (SIMD)

(Single Instruction Multiple Data) لذلك يُكْتَب البرنامج الموزع بواسطة واجهة البرامج، ثم تُوَزَّعُ نسخ منه في الأجهزة الطرفية تلقائياً، ثم يقوم العميل بإرسال بيانات مختلفة إلى نسخ البرنامج الموزعة بالأجهزة الطرفية ويستلم النتائج ومن ثم يظهرها.

11-تنفيذ النظام

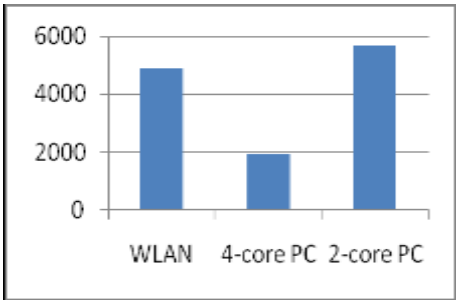
تم تشغيل النظام وتجربته على حواسيب مرتبطة مع بعضها بعضاً بشبكة محلية ومن ثم تنفيذه على شبكة منزلية تحتوي على أربعة حواسيب مرتبطة لاسلكياً عبر DSL.

تم تجربة النظام في البداية على حاسب واحد رباعي النواة (CPU 2.83GHz)، إذْ تم تنفيذ برنامج لحساب Pi على ١٠٠٠رقم (مرة باستخدام نواة واحدة، ومرة باستخدام نواتين، ثم بثلاث، ثم بأربعة) وكانت النتائج كما في الشكل(3)

الشكل (3) استخدام حاسب رباعي النواةّ

.نلاحظ أن أفضل أداء عند استخدام حاسب رباعي النواة.

وعند تجربة النظام على شبكة لاسلكية (WLAN) مكونة من حاسب ثنائي النواة (2.6 GHz) وحاسب أحادي النواة (3 GHz)، وحاسبين Netbook 1.66 Atom، كانت النتائج مقارنة مع تنفيذ البرنامج على حاسب ثنائي النواة (Core–2) وآخر رباعي النواة (core–4) كما في الشكل(4).



الشكل (4) تنفيذ البرنامج على شبكة لاسلكية

ونلاحظ أنَّ أفضل أداء كان عند استخدام حاسب رباعي النواة (core–4) .ممَّا سبق نلاحظ أنَّ أفضل أداء كان عند استخدام حاسب رباعي النواة، أي إنَّه كلما زاد عدد الأنوية في الحواسيب التي تعمل على التوازي كان الأداء أفضل.

الخلاصة والتوصيات

أُجْرِيَتْ تجارب مختلفة في شبكات محلية وشبكات لاسلكية، وكان الفارق واضحاً بين النتائج ممَّا يؤكد أنَّ أداء الحواسيب التي تعمل على التوازي (أكثر من نواة في الحاسب)أفضل من الحواسيب ذات النواة الواحدة، وكلما زاد عدد الأنوية زاد الاداء.

إنَّ استخدام برنامج متوازٍ يقوم بحساب أرقام Pi،(Computation of Digits of Pi)، بحيث يقوم كل معالج أو نواة بحساب مجموعة من هذه الأرقام بالتوازي يعطي أداء أفضل من استخدام اجهزة ذات نواة واحدة، لذلك لا بدَّ من وضع طريقة قياسية لإضافة برامج متوازية بحيث يمكن تحديثها ومن ثَمَّ يمكن للمبرمجين الإسهام بإضافة برامج تعمل على التوازي لتوافق المعالجات متعددة النواة وخاصة أنَّ الأجهزة المتوفرة في السوق كلّها الآن هي حاسبات متعددة النواة .

References

[1]Thomas Rauber ,(2010)- Parallel Programming For Multicore and Cluster System.Springer - Verlag Berlin Heidelberg , 642p.

[2] Hu Lei and Gorton Ian,2012- Performance Evaluation for Parallel Systems. University of NSW, Sydney Australia

[3] Hinton Geoffrey, (2014), Parallel models of associative memory,New York: Psychology Press. ISBN 978-1-315-80799-7

[4] Almasi, G.S. and A. Gottlieb ,(2015), Highly Parallel Computing. Benjamin.Cummings publishers, Redwood City, CA

[5] Flynn, Laurie J. (2016), "Intel Halts Development of 2 New Microprocessors". The New York Times,478p.

[6] Minh, Tran Ngoc, (2016), Workload modeling and performance evaluation in parallel systems,

[7] Brian J. S., Chee, Curtis Franklin Jr.(2016), CloudComputing Technology and Strategies of the Ubiquitous Data Center, Taylor and Francis Group،

[8] Isvan Novak and others, (2010), Visual Studio 2010 and .NET 4 Six-In-One, Wrox,

إيداع البحث	2019/2/11	Received
قبول البحث للنشر	2019/6/18	.Accepted for Publ