نحو مزيدٍ من تخفيض زمن الخوارزمية Stable Matching Algorithm للجدولة المتعددة النوى المتباينة

طالب الدراسات العليا: م. أبيّ سلوم كلية الهندسة المعلوماتية – جامعة دمشق

الدكتور المشرف: د.م. نزار الحافظ

1- ملخص

في البنى المتعددة النوى المتباينة والمتماثلة توجد مجموعة من خوارزميات الجدولة. من بين تلك الخوارزميات Stable Matching Algorithm (SMA) خوارزمية المطابقة المستقرة التي تُستخدم في البنى المتعددة النوى المتباينة وتعتمد على تصنيف المهام والنوى وفق قائمة أولويات [1].

في بحث سابق [11] قمنا بتحليل الأداء الزمني لخوارزمية SMA من حيث زمن تنفيذ المهام، وبيّنا أن استخدام SMA يحقق توفيراً زمنياً (بنسبة 74%) في بعض الحالات وزيادة في زمن التنفيذ (61%) في حالات أخرى.

نقدم في هذا البحث مساهمةً جديدة تتمثل بتحقيق مزيد من التوفير في زمن التنفيذ بتعديل خوارزمية SMA. فقد أجرينا أربعة تعديلات على الطريقة التي تعتمد عليها الخوارزمية في إسناد المهام على النوى. أظهرت نتائج المحاكاة انخفاضَ زمن التنفيذ انخفاضًا هامًّا في بعض الحالات، بنسبة 206%. نفّذنا التجارب باستخدام المحاكي sniper [10] لبنيان متعدد النوى (متماثلة أو متباينة) مع تغيير ترددات عمل هذه النوى. وفي كل تجربة أجريناها، قارنّا بين النتائج الزمنية لكل من التعديلات الأربعة وأزمان التنفيذ في حالة الجدولة من دون اعتماد SMA وحالة الجدولة باعتماد SMA. وأتاحت هذه النتائج استنتاج خوارزميتين معدلتين من SMA تعطيان أفضل أداء زمني.

كلمات مفتاحية: خوارزمية SMA، المحاكي Sniper، نوى متماثلة (homogeneous cores)، نوى متباينة (heterogeneous cores)، خوارزمية PSO، خوارزمية SFLA، خوارزمية LTF، خوارزمية DSR.

A further reduction in the time of the Stable Matching Algorithm in a heterogeneous multi-core architecture

1- Abstract

In heterogeneous and homogeneous multi-core architectures there is a set of scheduling algorithms. One such algorithm is the Stable Matching Algorithm (SMA), which is used in heterogeneous multi-core architectures and is based on the classification of tasks and cores according to a list of priorities[1].

In a previous paper [11], we analyzed the time performance of the SMA algorithm in terms of task execution time, and showed that using SMA achieves time savings (74%) in some cases and an increase in execution time (61%) in other cases.

In this paper, we present a new contribution, which is to achieve more savings in execution time by modifying the SMA algorithm. We've made four updates to the way that the algorithm depends on to assign tasks to cores. The simulation results showed a significant decrease in the execution time in some cases, by 206%. We carried out the experiments using the sniper simulator [10] of multi-core architecture (heterogeneous and homogeneous) with changing the working frequencies of these cores. In each experiment we conducted, we compared between the execution time of each of the four updates and the execution times of the case that we do not use SMA and of the case that we use SMA to schedule task. These results made it possible to deduce two modified algorithms of SMA algorithm those give the best timing performance.

Keywords: SMA algorithm, sniper simulator, homogeneous cores, heterogeneous cores, PSO algorithm, SFLA algorithm, LTF algorithm, DSR algorithm.

2- مقدمة

إن معظم المعالجات المتعددة النوى الموجودة هي أنظمة ذات نوى متماثلة (homogeneous cores)، وهذا يؤدي إلى وجود نقطة سلبية تتمثل في ضعف أداء هذه المعالجات عند استخدام التطبيقات المعقدة ذات مسلك التنفيذ الوحيد [1]. في المقابل، يتحسّن الأداء باستعمال النوى ذات الحجم الكبير، لأن زيادة حجم النواة يمكّن من تنفيذ مسالك أكثر في الوقت نفسه حتى لو كانت تلك المسالك ذات أولويات مختلفة. وهي أيضًا تقلل عدد مرات مقاطعة مسالك التنفيذ ذات الأولويات المنخفضة حتى عند وجود مسالك تنفيذ ذات أولويات أعلى منها [11]. وتبين مقارنة أجريت بين المعالجات المتعددة النوى المتماثلة والمعالجات المتعددة النوى المتباينة أن الأنظمة المتباينة النوى تتفوق في الأداء على الأنظمة المتماثلة النوى بمقدار 63% ([7]). لذلك تميل الأبحاث للاتجاه نحو استخدام المعالجات المتعددة النوى غير المتماثلة التي تدمج أنواع مختلفة من النوى في شريحة واحدة [11]، وهذا النوع من البنيان يؤدي إلى تحسين كبير في الأداء بتحقيق توازن حمل عمل بين النوى، وتخفيض تردد (frequency) المعالج لتخفيض استهلاك الطاقة (power).

توجد مجموعة من خوارزميات الجدولة للأنظمة المتباينة والأنظمة المتماثلة، وتبين نتائج الدراسات [8],[7],[6],[5],[2] أن لهذه الخوارزميات نقاط سلبية ونقاط إيجابية. فمثلا معظم خوارزميات الجدولة على المعالجات المتعددة النوى المتباينة لا تهتم بمعدل استهلاك الطاقة، وبعضها يهتم بجدولة مهام زمن حقيقي فقط، وبعضها الآخر يهتم بجدولة مهام عادية لا تعمل في الزمن الحقيقي.

من ناحية أخرى، يتيح استعمال بعض خوارزميات الجدولة في المعالجات المتعددة النوى المتماثلة، مثل خوارزمية Dynamic Slack Reclamation (DSR) "استرداد الركود ديناميكياً" وخوارزمية Largest Task First(LTF) "المهمة الأكبر أولا" [9] تحقيقَ توفيرٍ كبيرٍ في الطاقة مقارنة بباقي خوارزميات جدولة المهام في المعالجات المتعددة النوى المتماثلة.

تُستعمل خوارزمية SMA لجدولة المسالك في الأنظمة ذات النوى المتباينة. وسنقدم في هذا البحث تحليلاً لسلبية هذه الخوارزمية، ثم نعطي حلاً للتخلص من هذه السلبية. نفّذنا تجاربنا باستعمال المحاكي Sniper على بنيان X86 متعدد النوى. يتيح Sniper إجراء محاكاة على النوى المتماثلة أو المتباينة على حد سواء، بخلاف محاكيات أخرى تعمل على أنظمة نوى متماثلة فقط. أيضا يسمح Sniper بالمحاكاة على عشرات أو مئات النوى للتطبيقات ذات الذاكرة المشتركة، سواء للتطبيقات المتعددة المسالك أو التطبيقات ذات حمل العمل المختلف. وهو أسرع من باقي المحاكيات الموجودة [10]. ويولِّد Sniper مخططاً بيانياً يظهر عدد الدورات الحسابية المستهلكة في مكونات النظام المختلفة وفق فواصل زمنية محددة، وهذا يؤدي إلى فهم أفضل لتأثير هذه المكونات في أداء النظام ككل.

3- أعمال ذات صلة

من بين مجموعة خوارزميات الجدولة في المعالجات المتعددة النوى التي درسها الباحثون باهتمام كبير خوارزمية Stable Matching Algorithm (SMA) [1]، وخوارزمية Particle Swarm Optimization Algorithm (PSO) [5]، والخوارزمية الجينية genetic algorithm(GA) ، وغيرها.

تقوم خوارزمية SMA على تحسين الانتقاء الديناميكي لنوى المعالج المتباينة (heterogeneous) باستخدام مجدول، وذلك بغرض تحسين الأداء وتقليل استهلاك طاقة النوى.

وتقوم خوارزمية PSO بتخفيض كل من الطاقة المستهلكة وزمنِ أفضل جدولة لمهام الزمن الحقيقي في حالة وحدات المعالجة المتباينة. وتبين نتائج التجارب أن خوارزمية PSO توفر طاقة بمقدار 40% إلى 50% مقارنة بـ (SFLA) Shuffled Frog Leaping Algorithm (خوارزمية قفزات الضفدع المخلوطة).

تُستعمل الخوارزمية الجينية [6] لجدولة التنفيذ المتوازي على المعالجات المتعددة النوى المتباينة (heterogenous)، وتحقق كفاءة في استخدام النوى وفي تقليل زمن التنفيذ.
أيضا تُستعمل خوارزمية probability based scheduling (PbS) (الجدولة الاحتمالية) [7] لجدولة مهام الزمن الحقيقي غير الدورية في الأنظمة المتعددة المعالجات المتماثلة، وتمتاز بانها تقلل عدد المعالجات المستخدمة، وترفع كفاءة استخدام المعالجات وتخفض الطاقة المستهلكة.
لخوارزمية Parallel Hierarchical Hungarian Algorithm (الجدولة الهرمية المتوازية الهنغارية) [8] ميْزة قابلية التدرج، إذ تعمل على المعالجات التي تدعم تقنية التدرج الديناميكي للفولطية والتردد (dynamic voltage and frequency scaling) (DVFS). تُستعمل هذه الخوارزمية لجدولة مسالك التنفيذ على المعالجات المتعددة النوى المتباينة. وتبين التجارب أن هذه الخوارزمية أسرع في جدولة مسالك التنفيذ بمقدار 150 مرة من خوارزمية الجدولة الهنغارية غير المتوازية.
أخيرا تخفّض خوارزمية Dynamic Slack Reclamation(DSR) "استرداد الركود ديناميكيا" استهلاك الطاقة في المعالجات المتعددة النوى المتماثلة (homogeneous). وعند استعمالها مع خوارزمية Largest Task First(LTF) "المهمة الأكبر أولا" [9] يمكنها تحقيق أفضل توزيع لمهام الزمن الحقيقي على النوى وتخفيض استهلاك الطاقة بمقدار 60% إلى 90% مقارنة باستخدام أي من الخوارزميتين LTF, DSR على حدى.

4- خوارزمية المطابقة المستقرة Stable Matching Algorithm (SMA)

تُستعمل خوارزمية SMA لتوزيع المهام المتباينة الأحمال على النوى المتباينة (heterogeneous). في هذه الطريقة تمتلك كل من مجموعة المهام ومجموعة النوى قائمة أولويات خاصة بها. ويقوم المجدول وفق SMA باستقصاء أولويات وتوفر النوى، وذلك لاختيار أفضل زوج (مهمة – نواة) قبل إسناد مهمة على نواة معيّنة. يبيّن الشكل 1 خطوات الخوارزمية.

Stable Matching Algorithm(SMA) [1]:

```
1: Input: priority lists of tasks and cores
2: Initialize each core to be free.
3: while (some core is free and hasn't assigned to every task)
   {
4:    Choose such a core c
5:    t = 1st core on c's list to whom c has not yet assigned
6:    if (t is not assigned)
7:          choose c and t to be assigned
8:    else if (t prefers c to its assigned task c' and c' is free)
9:          choose c and t to be assigned، and c' to be free
10:   else
11:         t rejects c
    }
12: Output: stable tasks to cores mapping
```

الشكل 1. خوارزمية SMA.

تأخذ الخوارزمية قائمة الأولويات كدخل في الخطوة 1. وفي الخطوة 2 تقوم الخوارزمية بإخلاء كل النوى من أجل إسناد الأعمال وفقا لأولوياتها. تستمر حلقة while في الخطوة 3 حتى يكتمل الربط بين كل المهام والنوى. وفي الخطوة 4 يجري اختيار نواة، وفي الخطوة 5 تُجلب مهمة لتنفذ على تلك النواة. في الخطوات 6، 8، 10 لدينا اختبار شرطي. إذا كانت نتيجة اختبار الشرط هي true (الخطوة 6)، والشرط هو أن المهمة لم تُسنَد إلى نواة أخرى، فإنه يجري إسناد المهمة t إلى النواة المختارة في الخطوة 7، وإذا وجدت المهمة نواة أخرى c أفضل من النواة c'، فيجري إسناد المهمة t إلى النواة c

وإخلاء النواة c' (الخطوة 9). وإذا لم تتوافق أولويات المهمة مع النواة فإن المهمة سوف ترفض الإسناد إلى النواة (الخطوة 11).

4-1- التعديلات على خوارزمية SMA

للحصول على نتائج زمنية أفضل مما حصلنا عليه مع استخدام خوارزمية SMA لجدولة المهام في البنى المتعددة النوى المتباينة، قمنا بإجراء أربعة تعديلات على الطريقة التي تعتمد عليها خوارزمية SMA في إسناد المهام على النوى. في جميع هذه التعديلات الأربعة يقوم المسلك الذي ينهي تنفيذه بإرسال إشارة إلى باقي المسالك تدل على انتهاء تنفيذه، وبناء على تلك الإشارة تُنقل المسالك بطريقة معينة تختلف من تعديل لآخر. وفيما يلي هذه الطرق:

في التعديل الأول: عند استلام إشارة انتهاء التنفيذ، نقوم بالمرور على كل النوى التي تعمل بتردد أقل من تردد عمل نواة مرسل الإشارة فننقل المسلك الموجود عليها إلى النواة التي لها تردد عمل أعلى مباشرة من ترددها.

في التعديل الثاني: تجري عملية النقل كما في التعديل الأول، لكن في حالة الجدولة التي لا نستخدم فيها خوارزمية SMA. فعند استلام إشارة انتهاء التنفيذ، نقوم بالمرور على كل النوى التي تعمل بتردد أقل من تردد عمل نواة مرسل الإشارة فننقل المسلك الموجود عليها إلى النواة التي لها تردد عمل أعلى مباشرة من تردد النواة التي كان يعمل عليها المسلك في حالة الجدولة من دون استعمال خوارزمية SMA.

في التعديل الثالث: تجري عملية نقل المسالك كما في التعديل الأول، لكن مع تحديد عتبة لحجم العمل الذي أنجزه المسلك، بحيث يُنقل المسلك الذي لم يتجاوز تلك العتبة. يُعَدّ المسلك أنه وصل إلى العتبة المحددة إذا أنجز 4/3 حجم العمل المسند له.

التعديل الرابع: هو مزيجٌ من التعديلين الثاني والثالث. فيجري تحديد عتبة لحجم العمل الذي أنجزه المسلك، وعند استلام إشارة انتهاء التنفيذ، نقوم بالمرور على كل النوى التي تعمل بتردد أقل من تردد عمل نواة مرسل الإشارة فإن كان المسلك الموجود على النواة لم يتجاوز إنجازه العتبة المحددة، فإنه ننقله إلى النواة التي لها تردد عمل أعلى مباشرة من تردد النواة التي كان يعمل عليها المسلك في حالة الجدولة من دون استعمال خوارزمية SMA.

4-2- تقييم أداء التعديلات على خوارزمية SMA

من أجل تقييم التعديلات على خوارزمية SMA، استندنا إلى برنامج حسابي يقوم بضرب مصفوفتين مربعتين A(n,n) ، B(n,n) في حال عدة قيم لـ n هي: 20, 40, 100, 200, 400, 600, 800. من أجل كل قيمة لـ n وزعنا مهمة الضرب على ستة مسالك (threads)، منها أربعة كبيرة ومسلكين صغيرين. عدد الأسطر والأعمدة للمسالك الكبيرة يساوي ضعف عدد الأسطر والأعمدة للمسالك الصغيرة على الترتيب. وعدد الأسطر والأعمدة في أي مسلك (كبير أو صغير) هو نفسه. فمثلاً في حالة n تساوي 20 يُسنَد أربعة أسطر من المصفوفة A وأربعة أعمدة من المصفوفة B إلى كل مسلك كبير، وسطرين من المصفوفة A وعمودين من المصفوفة B إلى كل مسلك صغير.

من أجل التقييم، استخدمنا المحاكي sniper، وهو محاكي للبنيان المتعدد النوى المتباينة [10] يتيح بإجراء مجموعة من المهام، نذكر منها:

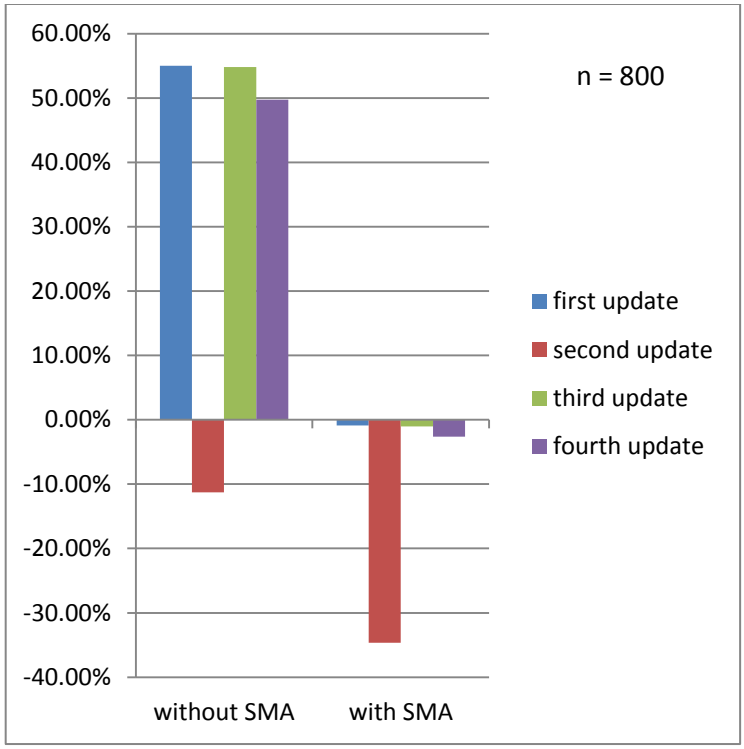
- إنشاء بنيان متعدد النوى (حتى 64 نواة)
- تغيير مستويات تردد العمل للنوى

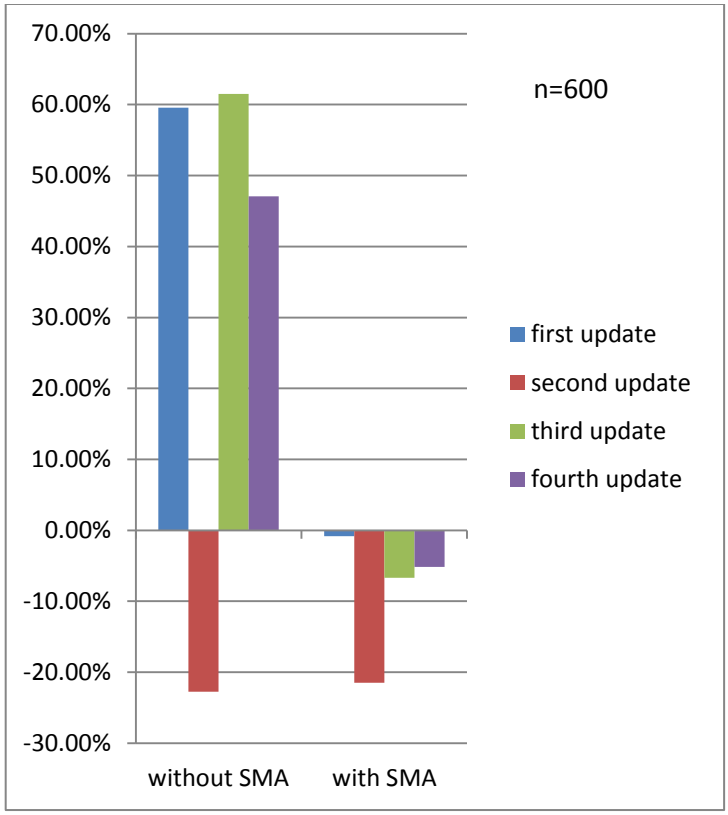
- نقل المسالك بين النوى

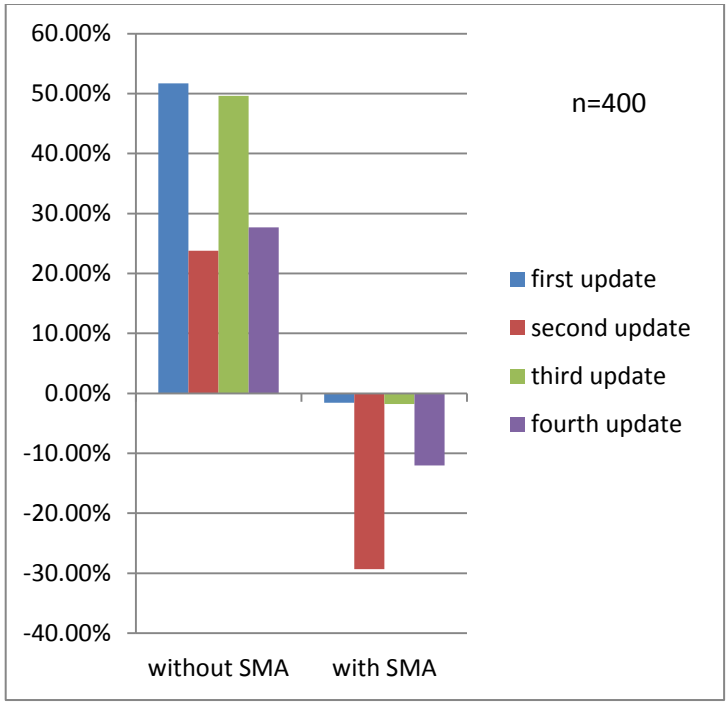
قمنا بضبط إعدادات المحاكي كما يلي:

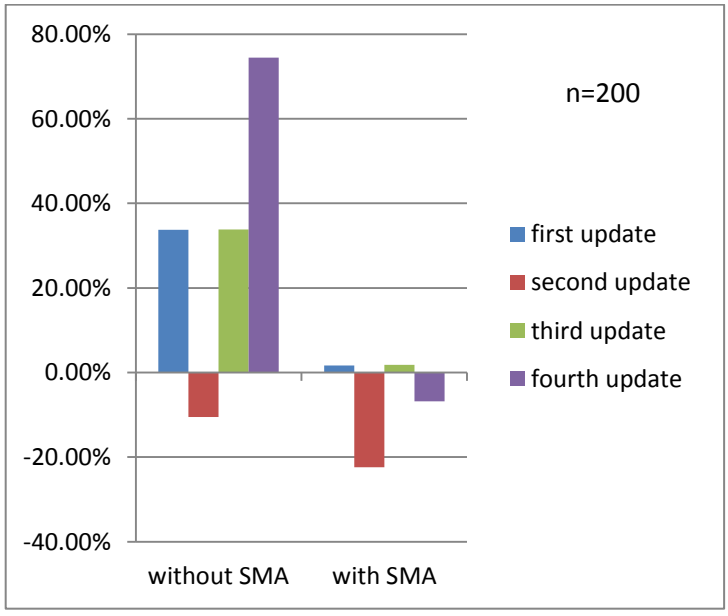
- عدد النوى سبعة
- ترددات النوى 1GHZ, 2.66GHZ, 3GHZ, 5GHZ, 10GHZ, 20GHZ ,1GHZعلى الترتيب (تردد النواة الأولى هو 1GHZ)

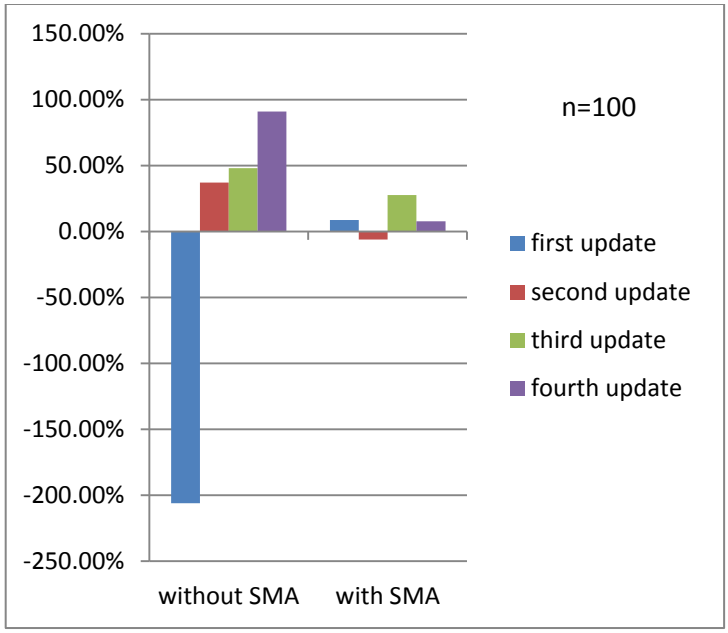
قمنا بمجموعة من التجارب، وفي كل تجربة قارنّا النتائج الزمنية لكل من التعديلات الأربعة وحالة الجدولة من دون SMA وحالة الجدولة باعتماد SMA. يبيّن الشكل 2 نتائج التجارب.

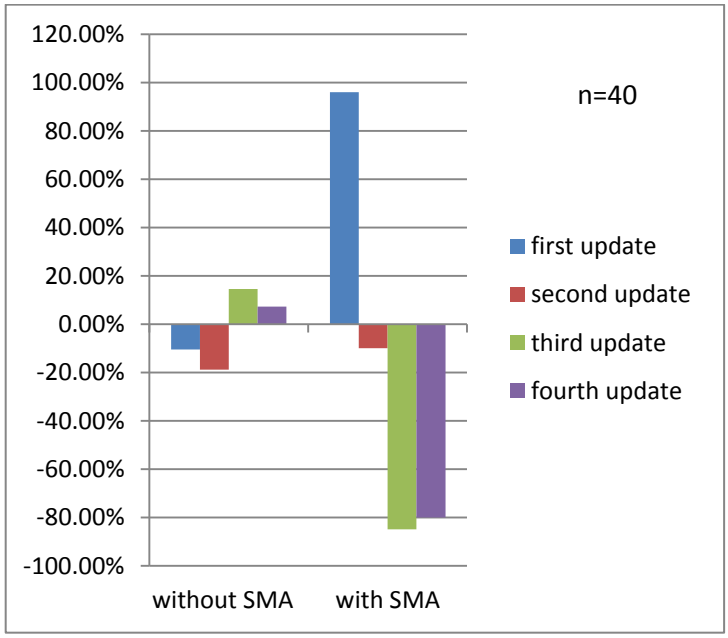

60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
-10.00%
-20.00%
-30.00%
-40.00%
without SMA
with SMA
n = 800
first update
second update
third update
fourth update

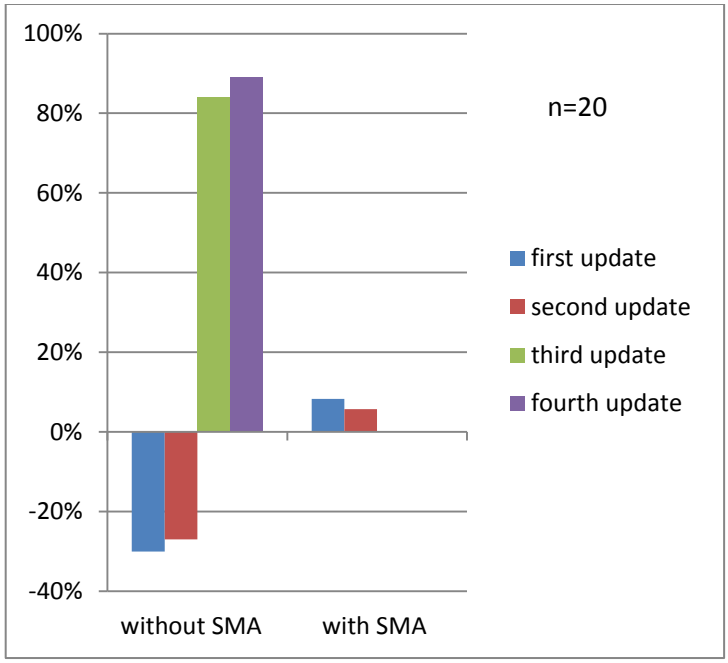

70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
-10.00%
-20.00%
-30.00%
without SMA
with SMA
n=600
first update
second update
third update
fourth update


60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
-10.00%
-20.00%
-30.00%
-40.00%
without SMA
with SMA
n=400
first update
second update
third update
fourth update


80.00%
60.00%
40.00%
20.00%
0.00%
-20.00%
-40.00%
without SMA
with SMA
n=200
first update
second update
third update
fourth update


150.00%
100.00%
50.00%
0.00%
-50.00%
-100.00%
-150.00%
-200.00%
-250.00%
without SMA
with SMA
n=100
first update
second update
third update
fourth update


120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
-20.00%
-40.00%
-60.00%
-80.00%
-100.00%
without SMA
with SMA
n=40
first update
second update
third update
fourth update



الشكل 2. مقارنة النتائج الزمنية للتعديلات الأربعة على خوارزمية SMA بالنتائج الزمنية لحالة الجدولة باستعمال SMA وحالة الجدولة من دون استعمال SMA، وذلك حسب الأبعاد المختلفة للمصفوفات (تمثّل النسب الموجبة زيادةً زمنية، والنسبُ السالبة تَحسُّنًا زمنيًّا)

يمكن جمع نتائج المخططات السابقة في مجموعة من الجداول التوضيحية المبينة أدناه:

N	التعديل الأول	التعديل الثاني	التعديل الثالث	التعديل الرابع
800	أسوأ بـ %55.1	أفضل بـ %11.29	أسوأ بـ %54.79	أسوأ بـ % 49.72
600	أسوأ ب %59.55 .	أفضل بـ %22.76	أسوأ بـ %61.49	أسوأ بـ %47.09
400	أسوأ بـ %51.7	أسوأ بـ %23.8	أسوأ بـ %49.63	أسوأ بـ %27.7
200	أسوأ بـ %33.77	أفضل بـ %10.55	أسوأ بـ %33.81	أسوأ بـ %74.44
100	أفضل بـ %206	أسوأ بـ %37	أسوأ بـ %48	أسوأ بـ %91
40	أفضل بـ %10.54	أفضل بـ %18.91	أسوأ بـ %14.57	أسوأ بـ %7.17
20	أفضل بـ %30	أفضل بـ %27	أسوأ بـ %84	أسوأ بـ %89

الجدول 1. مقارنة بين النتائج الزمنية للتعديلات الأربعة ونتائج الجدولة من دون استعمال SMA.

N	التعديل الأول	التعديل الثاني	التعديل الثالث	التعديل الرابع
800	أفضل بـ %0.87	أفضل بـ %34.66	أفضل بـ %1.04	أفضل بـ %2.64
600	أفضل بـ %0.81	أفضل بـ %21.47	أفضل بـ %6.67	أفضل بـ %5.17

400	مساواة	أفضل بـ %29.31	أفضل بـ %1.79	أفضل بـ %12
200	أسوأ بـ %1.68	أفضل بـ %22.4	أسوأ بـ %1.84	أفضل بـ %6.82
100	أسوأ بـ %8.72	أسوأ بـ %6.12	أسوأ بـ %27.54	أسوأ بـ %7.82
40	أسوأ بـ 96%	أسوأ بـ 10%	أفضل بـ 85%	أفضل بـ 80%
20	أسوأ بـ 8%	أسوأ بـ 6%	مساواة	مساواة

الجدول 2. مقارنة بين النتائج الزمنية للتعديلات الأربعة ونتائج الجدولة باستعمال SMA.

	التعديل الأول	**التعديل الثاني**	**التعديل الثالث**	**التعديل الرابع**
أعظم توفير زمني	%206	%27	–	–
أعظم زيادة زمنية	%59.55	%37	%84	%91

الجدول 3. مقارنة أعظم توفير زمني وأعظم زيادة زمنية تحققها التعديلات الأربعة مقارنة بحالة الجدولة من دون استعمال SMA.

	التعديل الأول	التعديل الثاني	التعديل الثالث	التعديل الرابع
أعظم توفير زمني	%0.87	%34.66	%85	%80
أعظم زيادة زمنية	%96	%10	%27.54	%7.82

الجدول 4. مقارنة أعظم توفير زمني وأعظم زيادة زمنية تحققها التعديلات الأربعة مقارنة بحالة الجدولة باستعمال SMA

من الجدول 2 نستنتج، في حالة الجدولة باستعمال SMA، أن التعديل الثاني يحقق أفضل توفير زمني (بين 21.47% و 34.66%)، مقارنة بالتعديلات الأول والثالث والرابع، في حالة الأبعاد الكبيرة للمصفوفة (بين 200 و 800). في المقابل، يحقق التعديلان الثالث والرابع أفضل توفير زمني (85% و 80% على الترتيب) في حالة أبعاد المصفوفة 40 × 40.

ومن الجدول ا نستنتج، في حالة الجدولة من دون استعمال SMA، أن التعديل الثاني يحقق أفضل أداء (بين 10.55% و 27%) في حالة جميع الأبعاد للمصفوفة (باستثناء 100 و 400)، في حين يحقق التعديل الأول توفيرًا زمنيًّا في حالة الأبعاد 100 وما دون، ذروته هي 206%. بالمقابل لا يحقق التعديلان الثالث والرابع أي توفير زمني.

من الجداول 1 إلى 4 أيضا ، نبيّن في الجدول 5 طريقة الجدولة التي تحقق أفضل توفير في الزمن ومقدار التوفير الموافق بحسب أبعاد المصفوفة، وذلك في حالة الجدولة باستعمال خوارزمية SMA.

N	الخوارزمية	التوفير الزمني
800	التعديل الثاني	%34.66
600	التعديل الثاني	%21.47
400	التعديل الثاني	%29.31
200	التعديل الثاني	%22.4
100	SMA	-
40	التعديل الثالث أو الرابع	%85-80
20	التعديل الثالث أو الرابع أو SMA	-

الجدول 5. أفضل خوارزمية والتوفير الزمني الموافق في حال الجدولة باستعمال SMA.

ونبيّن في الجدول 6 كذلك طريقة الجدولة التي تحقق أفضل توفير في الزمن ومقدار التوفير الموافق بحسب أبعاد المصفوفة، وذلك في حالة الجدولة من دون استعمال خوارزمية SMA.

N	الخوارزمية	التوفير الزمني
800	التعديل الثاني	%11.29
600	التعديل الثاني	%22.76
400	التعديل الثاني	-
200	التعديل الثاني	%10.55
100	التعديل الأول	%206
40	التعديل الثاني	%18.91
20	التعديل الأول	%30

الجدول 6. أفضل خوارزمية والتوفير الزمني الموافق في حالة الجدولة من دون SMA.

5- خلاصة

قدمنا في هذا البحث نتائج اختبار الأداء الزمني للتعديلات التي أجريناها على خوارزمية SMA. حيث قمنا بمقارنة نتائج هذه التعديلات مع نتائج حالة الجدولة التي نستخدم فيها SMA ومع نتائج حالة الجدولة التي لا نستخدم فيها SMA، وتبين من هذه المقارنات أن التعديل الثاني هو أفضل هذه التعديلات في كلا الحالتين وذلك عند استخدام أبعاد كبيرة للمصفوفات حيث أدى هذا التعديل إلى توفير زمني مقداره 27% مقارنة مع حالة الجدولة التي لا نستخدم فيها SMA كما أنه أدى إلى توفير زمني مقداره 34.66% مقارنة مع حالة الجدولة التي نستخدم فيها SMA، أما عند استخدام أبعاد صغيرة للمصفوفات فإن أفضل التعديلات مقارنة مع حالة الجدولة التي لا نستخدم فيها خوارزمية SMA هو التعديل الأول الذي أدى إلى توفير زمني بلغت أعظم قيمة له 206% بينما كان أفضل التعديلات مقارنة مع حالة الجدولة التي نستخدم فيها خوارزمية SMA هو التعديل الرابع الذي أدى إلى توفير زمني بلغت أعظم قيمة له 80%.

كما استنتجنا خوارزميتين معدلتين من SMA تعطي إحداهما أفضل أداء زمني في حال الجدولة باستعمال SMA، وتعطي الأخرى أفضل أداء زمني في حال الجدولة من دون استعمال SMA.

6- المراجع

[1] Zafar، M.R.(2016). Scheduling on Heterogeneous Multi-Core Processors Using Stable Matching Algorithm، in IJACSA.

[2] Zhang، W.، Bai، E.، He، H.، & Cheng، A.، M.K. (2015). Solving Energy-Aware Real-Time Tasks Scheduling Problem with Shuffled Frog Leaping Algorithm on Heterogeneous Platforms، in sensors.

[3] Karande، P.، Dhotre، S.S.، & Patil، S.(2014). Task Management for Heterogeneous Multi-core Scheduling، in IJCSIT.

[4] Kinsy، M.A.، Devadas، S.(2016). Algorithms for Scheduling Task-based Applications onto Heterogeneous Many-core Architectures، in Mit.

[5] Zhang، W.، Xie، H.، Cao، B.، & Cheng، A. M.K. (2014). Energy-Aware Real- Time Task Scheduling for Heterogeneous Multiprocessors with Particle Swarm Optimization Algorithm، in Hindawi.

[6] Radhamani، A.S.، Baburaj، E.(2013). performance efficient heterogeneous multi core scheduling strategy based on genetic algorithm، in ARPN.

[7] Anne، N.، Muthukumar، V.(2013). Energy Aware Scheduling of Aperiodic Real-Time Tasks on Multiprocessor Systems، in JCSE.

[8] Winter، J. A.، Albonesi، D. H.، & Shoemaker، C. A.(2010).Scalable Thread Scheduling and Global Power Management for Heterogeneous Many-Core Architectures، in ACM.

[9] Salom، O.، Alhafez، N.(2016). Enhancing Energy-Efficient Scheduling Algorithm on Multicore Processors، in Damascus University Journal.

[10] Akra, A., Sawalha, L.(2016). A Comparison of x86 Computer Architecture Simulators, in Computer Architecture and Systems Research Laboratory (CASRL).

[11] سلوم، أ.، الحافظ، ن.(2020). نحو تخفيض زمن الخوارزمية Stable Matching Algorithm للجدولة المتعددة النوى المتباينة. مجلة جامعة البعث – سورية – حمص.