الجمهـوريــــــــــــــــــــة العربيـــــــــــــــــــــة الســــــــــــــــــــــورية

المعهد العالي للعلوم التطبيقية والتكنولوجيا

قســــــــــــــــــــــم المعلوميــــــــــــــــــــــات

العام الدراسي 2025/2024

**مقرر الحوسبة المتوازية**

الوظيفة الأولى

البرمجة المتوازية – النياسب المتعددة

Parallel programming - multithreading

تقديم الطالب

المهنّد ياسر حافظ

03/09/2024

**المحتويات**

[إيجاد الأعداد الأوليّة 1](#_Toc178459121)

[-1.1 خوارزمية التقسيم إلى مجالات Chunks method 1](#_Toc178459122)

[1.1.1- فكرة الخوارزمية 1](#_Toc178459123)

[2.1.1- التحويل إلى خوارزمية متوازية 1](#_Toc178459124)

العتاد المستخدم

تم تطبيق اختبارات الأداء على عتاد له المواصفات التالية:

الجدول 1 مواصفات العتاد المستخدم

|  |  |
| --- | --- |
| Intel® Core™ i5-6200U @ 2.30GHz | CPU |
| 2 | Cores |
| 4 | Logical processors |
| 12 GB | RAM |

السؤال الأول

إيجاد الأعداد الأوليّة

تم تطبيق ثلاث خوارزميات لإيجاد الأعداد الأولية ضمن مجال ما واختبار أدائها في حالات مختلفة.

1.1- خوارزمية التقسيم إلى مجالات Chunks method

تعرض هذه الفقرة الخوارزمية الأولى التي تم تطبيقها لإيجاد الأعداد الأولية.

1.1.1- فكرة الخوارزمية

يتم في هذه الخوارزمية العبور على جميع الأعداد الصحيحة في المجال واختبار كون كل عدد أولياً أم لا من خلال اختبار قابلية قسمته على جميع الأعداد الصحيحة في المجال .

إن تعقيد هذه الخوارزمية في الحالة الأسوأ Worst case هو .

2.1.1- التحويل إلى خوارزمية متوازية

تقوم الفكرة على تقسيم المجال إلى مجموعة من المجالات الجزئية تبعاً لعدد النياسب، ويقوم كل نيسب بإيجاد الأعداد الأولية ضمن مجاله.

من الجدير بالذكر أن الأعداد الأوليّة يتم تخزينها في لائحة List موجودة ضمن الصف ChunksMethod، وبالتالي يقوم كل نيسب يقوم بتنفيذ ChunksMethodRunnable بالتعديل على هذه اللائحة؛ ولضمان تحقيق التعديل بشكل متزامن ودون حدوث أخطاء أو تضاربات Conflicts، تم استخدام التعليمة synchronized.

3.1.1- اختبار الخرج

تم إجراء اختبار وحدة Unit test للتحقق من صحة تطبيق هذه الخوارزمية على قيم مختلفة للعدد وتم تجاوز الاختبار بشكل صحيح.

4.1.1- اختبارات الأداء

تم تطبيق عدد من الاختبارات لتقييم أداء الخوارزمية على قيم مختلفة للعدد وعدد النياسب.

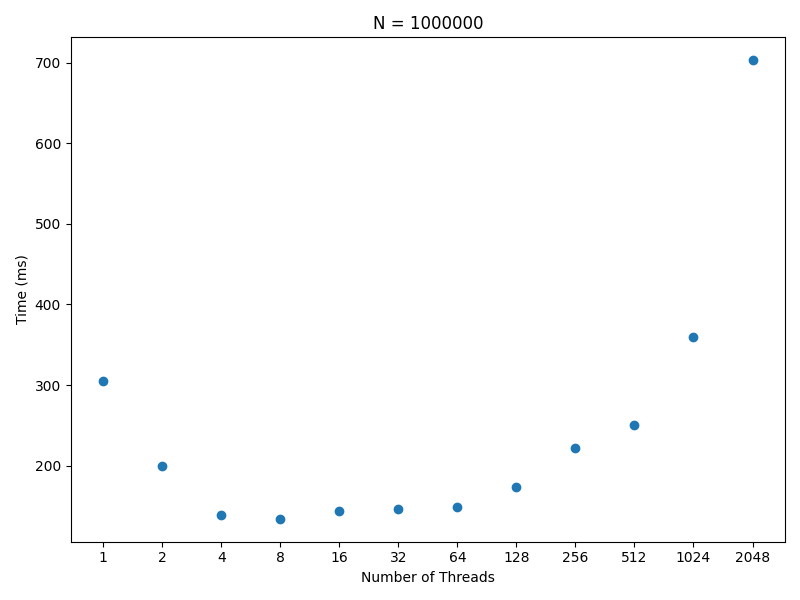
يتم في كل اختبار تطبيق الخوارزمية عدد من المرات وقياس الزمن اللازم للتنفيذ وحساب الزمن الوسطي المستغرق.

ملاحظة: يوجد المزيد من اختبارات الأداء ضمن الرماز لم يتم عرضها جميعها.

1.4.1.1- حالة

الجدول 2 نتائج الأداء بحالة

|  |  |  |  |
| --- | --- | --- | --- |
| **Average time(ms)** | **Threads** | **Average time(ms)** | **Threads** |
| 149 | 64 | 305 | 1 |
| 173 | 128 | 199 | 2 |
| 222 | 256 | 139 | 4 |
| 250 | 512 | 134 | 8 |
| 359 | 1024 | 144 | 16 |
| 703 | 2048 | 146 | 32 |

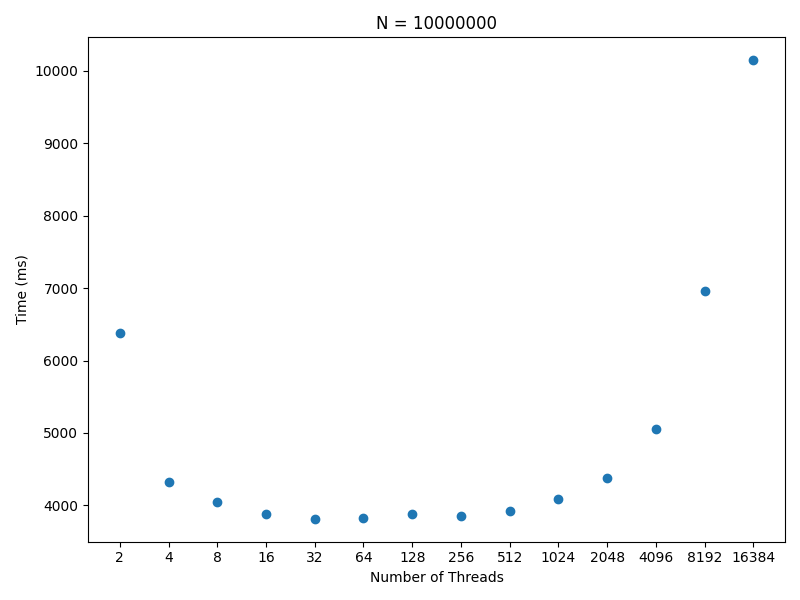


الشكل 1 مخطط الأداء بحالة

2.4.1.1- حالة

الجدول 3 نتائج الأداء بحالة

|  |  |  |  |
| --- | --- | --- | --- |
| **Average time(ms)** | **Threads** | **Average time(ms)** | **Threads** |
| 3855 | 256 | 6381 | 2 |
| 3928 | 512 | 4329 | 4 |
| 4094 | 1024 | 4045 | 8 |
| 4380 | 2048 | 3884 | 16 |
| 5061 | 4096 | 3815 | 32 |
| 6956 | 8192 | 3824 | 64 |
| 10148 | 16384 | 3885 | 128 |

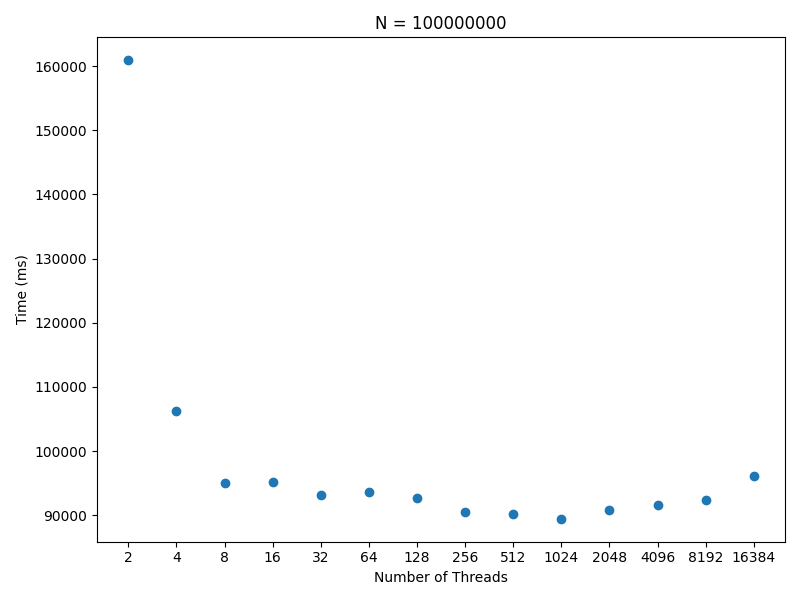


الشكل 2 مخطط الأداء بحالة

3.4.1.1- حالة

الجدول 4 نتائج الأداء بحالة

|  |  |  |  |
| --- | --- | --- | --- |
| **Average time(ms)** | **Threads** | **Average time(ms)** | **Threads** |
| 90520 | 256 | 160905 | 2 |
| 90171 | 512 | 106176 | 4 |
| 89448 | 1024 | 94980 | 8 |
| 90859 | 2048 | 95150 | 16 |
| 91669 | 4096 | 93232 | 32 |
| 92434 | 8192 | 93612 | 64 |
| 96146 | 16384 | 92616 | 128 |

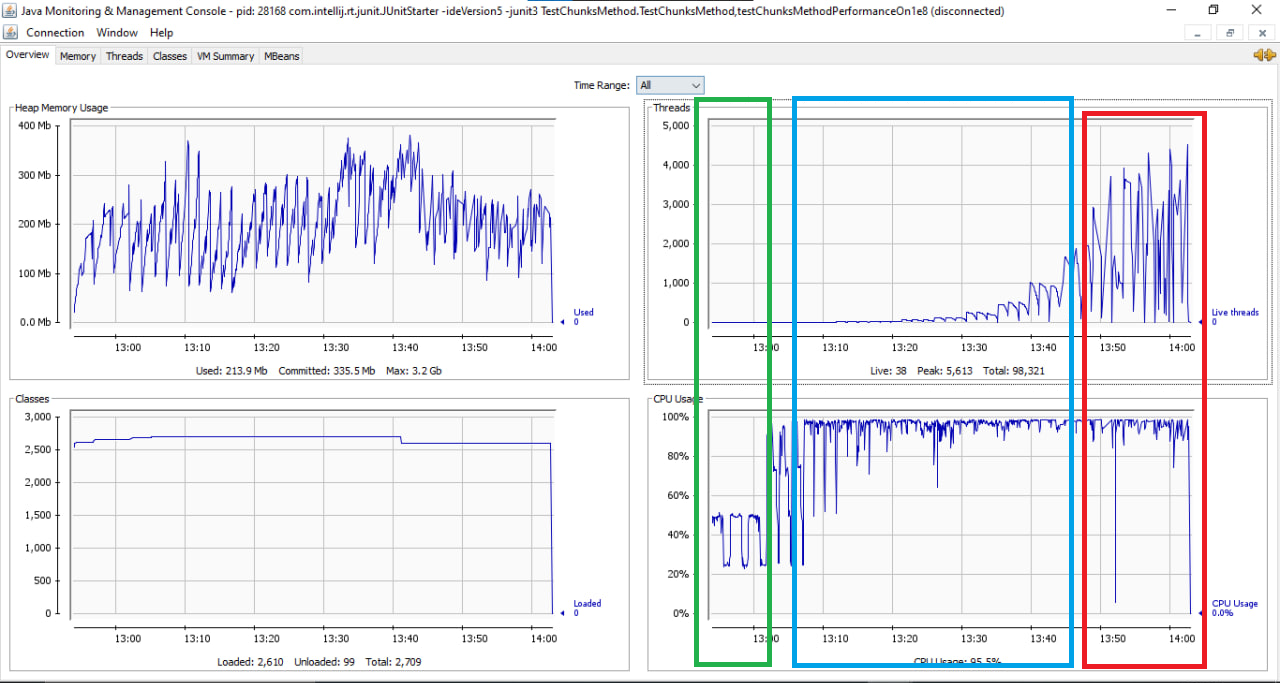


الشكل 3 مخطط الأداء بحالة

4.4.1.1- مراقبة الأداء بحالة

تم مراقبة أداء إجرائية الاختبار في هذه الحالة باستخدام الأداة JConsole، وتم تدقيق استخدام وحدة المعالجة والذاكرة بالإضافة إلى عدد النياسب المفعّلة Live threads.

تم ضمن إجرائية الاختبار إيجاد الأعداد الأولية في المجال باستخدام عدد مختلف من النياسب في كل مرة، وفي كل مرة كان يتم إيجاد الأعداد الأولية 3 مرات وقياس الزمن اللازم ومن ثم يتم حساب المتوسط.



الشكل 4 مراقبة الأداء في حالة

من الجدير بالملاحظة في الشكل السابق هو أنه في بداية تنفيذ الاختبار (باللون الأخضر) كان عدد النياسب قليلاً وبالتالي لم يتم استخدام كامل موارد وحدة المعالجة من قبل الإجرائية، أما مع نهاية الاختبار (باللون الأحمر) نلاحظ تخبطاً في عدد النياسب التي تعمل وذلك لأنه في هذه المرحلة أصبح عدد النياسب كبيراً (8192 و16384) وبالتالي لم يتمكن نظام التشغيل من جدولة هذه النياسب للعمل معاً وذلك بسبب محدودية العتاد (أقصى عدد من النياسب المفعلّة هو 5614 رغم أنه يوجد اختبارات تطلب استخدام عدد أكبر من النياسب) وبالتالي أصبحت جدولة عدد كبير من النياسب تشكل عبء إضافي على عملية التنفيذ ولم يتم تفعيلها معاً (أي أنه تم تأجيل جدولة بعض النياسب) وهذا ما يفسر التخبط في عدد النياسب في مرحلة نهاية الاختبار.

أما في المرحلة الوسطى (باللون الأزرق) فيمكن بوضوح تتبع عدد النياسب التي تعمل في كل مرحلة اختبار (أي في كل مرة نحدد فيها عدد النياسب)، كما يوضح الشكل التالي:



الشكل 5 النياسب المفعلة في بعض مراحل الاختبار

نلاحظ وجود النياسب 3 مرات وهو عدد المرات التي نقوم فيها بتطبيق الخوارزمية.

نلاحظ كذلك انخفاض عدد النياسب تدريجياً نحو الصفر وذلك لانتهاء عملها.

5.4.1.1- مراقبة الأداء بحالة مع 1024 نيسب

تم مراقبة أداء إجرائية الاختبار في الحالة التي حصلنا فيها على أفضل زمن وسطي، وهي حالة استخدام 1024 نيسب.



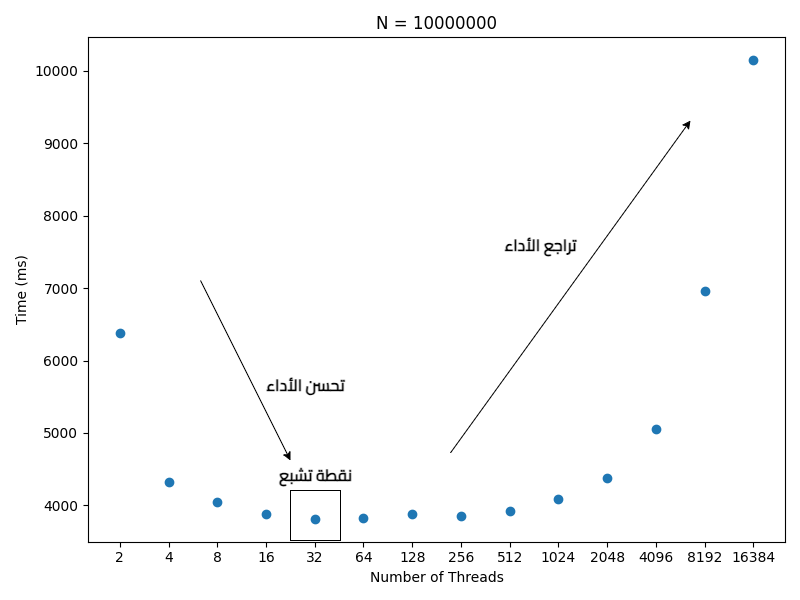
الشكل 6 مراقبة الأداء بحالة مع استخدام 1024 نيسب

نلاحظ من الشكل السابق أنه قد تم استخدام 1024 نيسب وكانت مفعّلة ومجدولة جميعها، وأن استهلاك وحدة المعالجة في جميع مراحل التنفيذ كان تقريباً 95% أي أنه يتم استخدام كامل موارد وحدة المعالجة.

بالنسبة لاستخدام الذاكرة نلاحظ تزايده مع تقدم التنفيذ، وذلك بسبب عملية تخزين الأعداد الأولية ضمن لائحة List يتم تخزينها ضمن الذاكرة؛ حيث أنه مع تقدم التنفيذ يتم تخزين المزيد من الأعداد الأولية في اللائحة.

6.4.1.1- تحليل نتائج اختبارات الأداء

نلاحظ من النتائج السابقة أن زيادة عدد النياسب له تأثير إيجابي على الأداء؛ حيث نلاحظ انخفاض زمن التنفيذ مع زيادة عدد النياسب إلى حد معين (نقطة تشبّع)، ولكن عند تجاوز هذا الحد يصبح زيادة عدد النياسب ذو أثر سلبي على الأداء وذلك لأنه يشكل حملاً إضافياً على العتاد بسبب ما يستلزم من عمليات جدولة بالإضافة إلى الاستهلاك الزائد للذاكرة بسبب الحاجة إلى تخصيص بعض الموارد المستقلة لكل نيسب (كالمكدس Stack).



الشكل 7 تغير الأداء تبعاً لعدد النياسب

ضمن هذه الخوارزمية؛ تم إسناد أطوال مجالات متساوية لكل نيسب، وفي رأيي هذه الطريقة هي الأنسب وذلك لأنه لا فرق بين نيسب وآخر ولا يمكننا التحكم بآلية جدولة النياسب ضمن وحدة المعالجة، ويمكننا أن نلاحظ أن استخدام نياسب مع أطوال مجالات كبيرة نسبياً (الجزء الأول من المخطط السابق) يعيدنا إلى حالة مشابهة لحالة البرمجة التسلسلية، وكذلك استخدام نياسب مع أطوال مجالات صغيرة نسبياً (الجزء الأخير من المخطط السابق) يؤدي إلى أداء غير مرضٍ بسبب الزيادة الكبيرة في عدد النياسب، لذا من الأفضل استخدام مجالات متوسطة الطول نسبياً.

كذلك، عند مراقبة الأداء تمت ملاحظة تأثير مواصفات العتاد على أداء البرنامج، وخصوصاً عند ملاحظة عجز نظام التشغيل عن جدولة عدد كبير من النياسب معاً، وهذا الأمر متعلق ببنية وحدة المعالجة وعدد النوى Cores والمعالجات الافتراضية Logical processors ضمنها.

لذا؛ يجب دراسة مواصفات العتاد جيداً قبل التفكير باستخدام البرمجة المتوازية لأن تحقيق أداء جيد بحاجة إلى دراسة جيدة لعدد النياسب الواجب استخدامها بالإضافة إلى حجم العمل المسند إلى كل نيسب بما يتوافق مع مواصفات العتاد من حيث وحدة المعالجة والذواكر.

2.1- خوارزمية غربال إيراتوستين Sieve of Eratosthenes