الجمهورية العربيـة السـورية
المعهد العالي للعلوم التطبيقية والتكنولوجيا
اختصاص شبكات ونظم تشغيل
العام الدراسي 2022/2023

عملي البرمجة المتوازية

اللوائح المترابطة المفرّزة والمتزامنة

تقديم

بشار حسين

تاريخ: 23/05/2023

السؤال الأول

لتحقيق متطلبات تسجيل عدد الطلبات الناجحة والفاشلة لكل عملية قمنا بما يلي:

أصفنا خاصيتين للصنف TestThread وهما success وfailure بحيث يصبح كل Object يخزن عدد محاولاته الناجحة والفاشلة من كل عملية.

ضمن التابع testHelp قمنا بإضافة 3 لوائح جديدة كما يلي:

```
List<AddThread> xAdd = new ArrayList<>();
List<ContainThread> xContain = new ArrayList<>();
List<RemoveThread> xRemove = new ArrayList<>();
```

وبعد الانتهاء من تنفيذ كل عملية قمنا بالمشي على اللائحة وعد نتيجة كل object من النجاح والفشل.

ولمعرفة طول اللائحة الجديد بعد عمليات الإضافة هو نفسه عدد العمليات الناجحة من Add.

ولمعرفة طول اللائحة الجديد بعد عمليات الإزالة هو نفسه طول اللائحة بعد عمليات الإضافة مطروحاً منه عدد العمليات الناجحة من Reomve.

النتائج

- طول اللائحة 20000 وعدد النياسب 4.

	زمن تنفيذ Contain	زمن تنفيذ Remove	زمن تنفيذ Add
طريقة المزامنة	3130	2490	2237
طريقة RWLock	225	557	1378
طريقة Lock	2935	2888	1342

- طول اللائحة 20000 وعدد النياسب 8.

	زمن تنفيذ Contain	زمن تنفيذ Remove	زمن تنفيذ Add
طريقة المزامنة	3620	4034	1698
طريقة RWLock	800	2665	1445
طريقة Lock	3000	2429	1323

- طول اللائحة 40000 وعدد النياسب 4.

	زمن تنفيذ Contain	زمن تنفيذ Remove	زمن تنفيذ Add
طريقة المزامنة	21659	9210	9973
طريقة RWLock	3567	29938	6098
طريقة Lock	30580	23536	10109

- طول اللائحة 40000 وعدد النياسب 8.

	زمن تنفيذ Contain	زمن تنفيذ Remove	زمن تنفيذ Add
طريقة المزامنة	22071	3342	11440
طريقة RWLock	4484	18706	9676
طريقة Lock	23819	14751	9670

ملاحظات حول النتائج

في البداية نلاحظ أن طول اللائحة بعد تنفيذ عمليات الإضافة ليس نفسه الطول المفترض لأن اللائحة تحتوي على أعداد صحيحة مختلفة فقط بينما نياسب الإضافة قد تحتوي على أعداد متشابهة وبالتالي ينتج طول اللائحة اقل من عدد الأعداد التي تم توليدها.

كل طريقة من الطرق لها ميزاتها فمثلا طريقة Synchronization هي بسيطة جداً في التطبيق والكتابة ولكنها قد تخلق مشاكل في الأداء بسبب lock contention. أما طريقة Lock فتوفر طريقة قابلة للإدارة أكثر من Synchronization مما يحسن الأداء بسبب تقليل lock contention ولكن يجب الانتباه إلى طريقة الإدارة لتجنب deadlocks أو مشاكل مزامنة أخرى.

طريقة ReadWriteLock تقوم على أنه لا مشكلة إن تمت عملية القراءة من قبل نيسبين أو أكثر في نفس الوقت مما يوفر كثيرا من وقت انتظار النيسب ويظهر أثر هذا التحسين في البيئات التي تكون فيها عمليات القراءة أكثر من عمليات الكتابة بكثير. وفي حالتنا يظهر هذا الأثر جلياً في أوقات تنفيذ عملية contain مقارنة بـadd وremove حيث أنها تكون أقل بكثير دائماً.

سيناريوهات فاشلة:

لتنفيذ الطلب الأول وحساب حالات الفشل والنجاح لكل عملية فكرنا بطرق أخرى للتنفيذ:

1- أنشأت متغيرين من نوع static باسم success و failure في كل صنف من الأصناف AddThread و ContainThread و RemoveThread وقمنا بالتعديل على المتغيرات طبعاً عند تنفيذ العمليات. ولكن هذه الطريقة لم تعمل لأنه نحتاج إلى إضافة lock للمتغيرات لأنها أصبحت كمورد مشترك يتم التعديل عليه من قبل مختلف النياسب.

2- يمكن إضافة المتغيرات ضمن الأصناف الخاصة باللوائح ولكن عندها سيتوجب علينا إضافة writeLock في عملية Contain في حالة ReadWriteLock ولم نفضل استخدام هذه الطريقة لأنها تلغي التأثير المفيد لحالة ReadWriteLock.