Java 8最快的垃圾蒐集器是什麼？

本文由 ImportNew - paddx 翻譯自 javacodegeeks。歡迎加入翻譯小組。轉載請見文末要求。

OpenJDK 8 有多種 GC（Garbage Collector）算法，如 Parallel GC、CMS 和 G1。哪一個才是最快的呢？如果在 Java 9 中將 Java 8 默認的 GC 從 Parallel GC 改為 G1 （目前只是建議）將會怎麼樣呢？讓我們對此進行基準測試。

基準測試方法

運行相同的代碼六次，每次使用不同的VM參數（-XX:+UseSerialGC, -XX:+UseParallelGC, -XX:+UseConcMarkSweepGC, -XX:ParallelCMSThreads=2, -XX:ParallelCMSThreads=4, -XX:+UseG1GC）。

每次運行大概花費55分鐘。

其它VM參數：-Xmx2048M -server

OpenJDK版本：1.8.0_51（當前最新的版本）

軟件：Linux version 4.0.4-301.fc22.x86_64

硬件：Intel? Core? i7-4790 CPU @ 3.60GHz

每次運行13個?OptaPlanner?規劃問題方案。每次運行時間為5分鐘。前30秒用於JVM預熱，不計算在內。

解決規劃問題不涉及 IO （除了啟動時需要幾毫秒來加載輸入信息）。單個 CPU 使用完全飽和。通常會創建許多存活時間很短的對象，GC 之後就會回收這些對象。

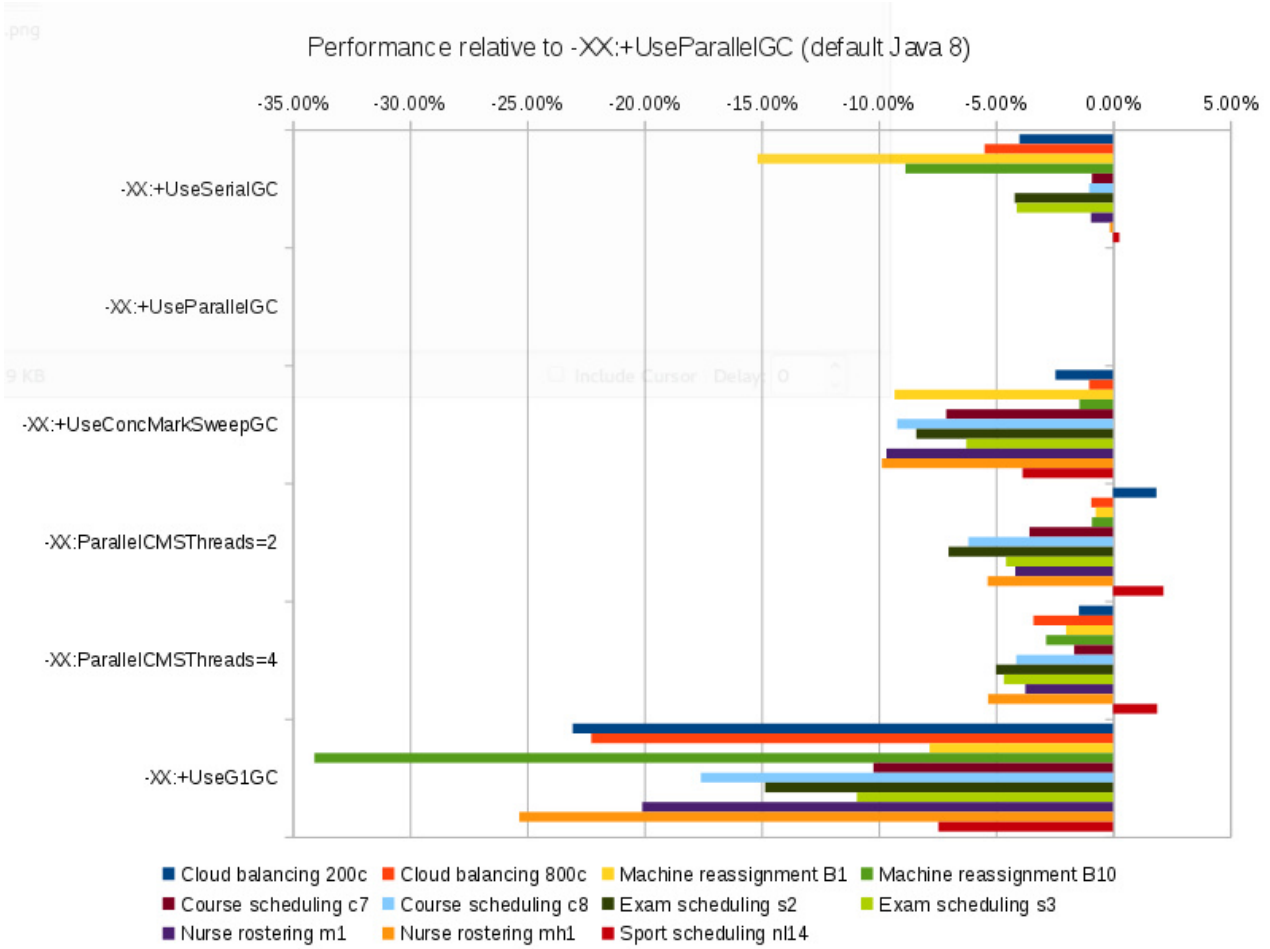
衡量標準可以是計算每毫秒的得分，越高越好。計算一個擬議規劃解決方案是一個不可小覷的問題：涉及到大量的計算，包括每個實體與其他所有實體的衝突檢測。

為了能在本地重複運行這些基準測試，可以從源碼進行構建，然後運行主類 GeneralOptaPlannerBenchmarkApp。

基準測試結果

執行結果

為了方便查看，我已經對每種 GC 與 Java 8 默認 GC（Parallel GC）進行了比較。



結果非常清楚：**默認（Parallel GC）是最快的**。

原始基準測試數據

Garbage Collector	Cloud balancing 200c	Cloud balancing 800c	Machine reassignment B1	Machine reassignment B10	Course scheduling c7	Course scheduling c8	Exam scheduling s2	Exam scheduling s3	Nurse rostering m1	Nurse rostering mh1	Sport scheduling nl14
`-XX:+UseSerialGC`	121211	102072	239278	54637	10821	14370	17095	10130	7389	6667	2234
`-XX:+UseParallelGC` (default Java 8)	126248	107991	282055	59944	10919	14517	17843	10564	7459	6676	2228
`-XX:+UseConcMarkSweepGC`	123150	106889	255775	59087	10142	13180	16346	9903	6738	6018	2142
`-XX:ParallelCMSThreads=2`	128591	106992	279968	59406	10530	13621	16591	10082	7148	6319	2276
`-XX:ParallelCMSThreads=4`	124415	104328	276401	58234	10738	13918	16952	10072	7180	6320	2270
`-XX:+UseG1GC` (default Java 9?)	97146	83952	259981	39522	9803	11965	15195	9410	5961	4985	2062
Dataset scale	120k	1920k	500k	250000k	217k	145k	1705k	1613k	18k	12k	4k

相對基準測試數據

Garbage Collector	Average	Cloud balancing 200c	Cloud balancing 800c	Machine reassignment B1	Machine reassignment B10	Course scheduling c7	Course scheduling c8	Exam scheduling s2	Exam scheduling s3	Nurse rostering m1	Nurse rostering mh1	Sport scheduling nl14
-XX:+UseSerialGC	-4.05%	-3.99%	-5.48%	-15.17%	-8.85%	-0.90%	-1.01%	-4.19%	-4.11%	-0.94%	-0.13%	+0.27%
-XX:+UseParallelGC (default Java 8)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
-XX:+UseConcMarkSweepGC	-6.23%	-2.45%	-1.02%	-9.32%	-1.43%	-7.12%	-9.21%	-8.39%	-6.26%	-9.67%	-9.86%	-3.86%
-XX:ParallelGCThreads=2	-2.67%	+1.86%	-0.93%	-0.74%	-0.90%	-3.56%	-6.17%	-7.02%	-4.56%	-4.17%	-5.35%	+2.15%
-XX:ParallelGCThreads=4	-2.94%	-1.45%	-3.39%	-2.00%	-2.85%	-1.66%	-4.13%	-4.99%	-4.66%	-3.74%	-5.33%	+1.89%
-XX:+UseG1GC (default Java 9?)	**-17.60%**	-23.05%	-22.26%	-7.83%	-34.07%	-10.22%	-17.58%	-14.84%	-10.92%	-20.08%	-25.33%	-7.45%
Dataset scale		120k	1920k	500k	250000k	217k	145k	1705k	1613k	18k	12k	4k

Java 9 默認應該為 G1 嗎？

有一種提議是在 OpenJDK9 的服務器端使用 G1 作為默認 GC。我第一反應就是拒絕該提議：

G1 的平均值要慢17.60%

G1 在每個數據集用例下都比較慢。

在最大數據集（Machine Reassignment B10）下，表現比其它數據集都要差，G1 慢了34.07%。

如果在開發機和服務器之間採用不同的默認 GC，則開發者基準測試的可信度就會下降。

另一方面，存在幾個需要注意的細節：

G1 關注是 GC 暫停的問題，而不是吞吐量。對於這些用例（計算量比較大），GC 暫停時長基本沒影響。

這是一個（基本是）單線程的基準測試。並行解決多個問題或採用多線程解決的基準測試，結果可能不同。

G1 推薦的堆內存至少是 6GB。而這次基準測試的堆內存是 2GB，即使在最大數據集（Machine Reassignment B10）也只需要這麼多內存。

海量計算只是 OpenJDK 的諸多功能中的一個：這是在社區廣泛爭論的一個問題。如果有其他方面（如網站服務）的證明，可能值得改變默認GC。但是，請首先向我展示你實際項目的基準測試。

結論

在 Java 8 中，對 OptaPlanner 用例來說，默認 GC（Parallel GC）通常情況是最好的選擇。

原文鏈接： javacodegeeks 翻譯： ImportNew.com - paddx
譯文鏈接： http://www.importnew.com/16533.html
[轉載請保留原文出處、譯者和譯文鏈接。]

關於作者： paddx

查看paddx的更多文章 >>