实验一 矩阵相乘 实验报告

07111403 朴泉宇 1120141831

一、实验目的

通过用 C++、Java、Python、Haskell 四种语言，运用简单的程序运算（在本实验中选择矩阵相乘，时间复杂度 $O(n^3)$），结合较大数据规模，通过统计程序运行时间来得出相应语言的程序运行性能结论；再结合在编码时的编码难度来较粗略地分析这四种编程语言。

二、实验环境

CPU：i7 – 4710MQ，八核，2.50GHz

RAM：8G

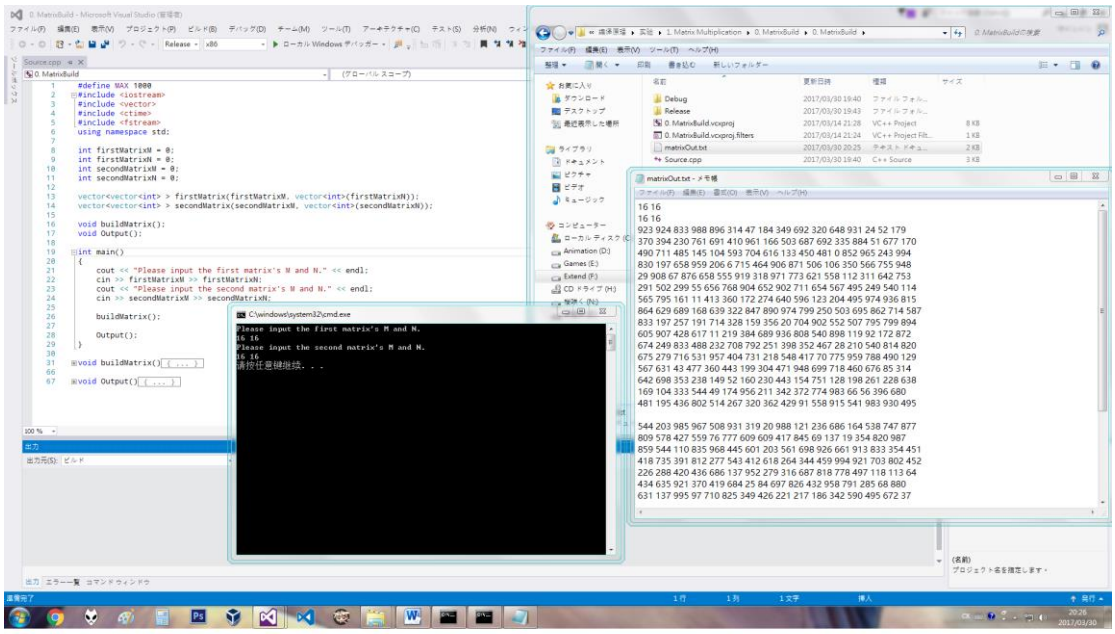
Cache：L1 Code：32KB，L1 Data：32KB，L2：256KB，L3：6MB

三、实验过程

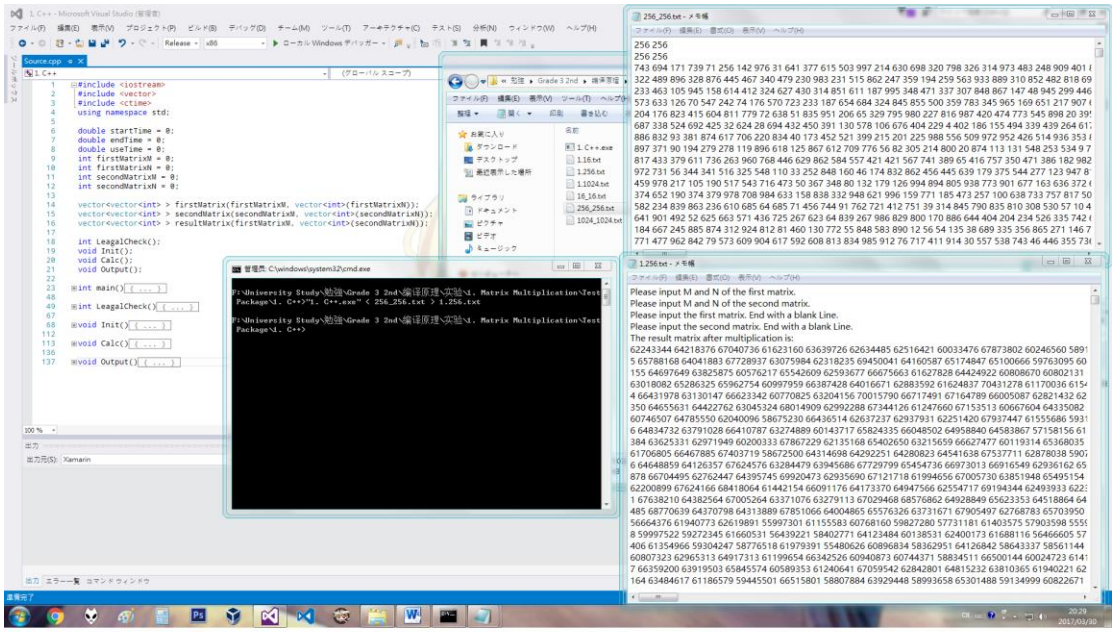
1. 用 C++编写矩阵生成程序（已确定在接下来的矩阵相乘程序中的输入格式）。
2. 用 C++、Java、Python、Haskell 分别编写矩阵相乘程序，C++在 Release 模式下生成二进制（.exe）文件，Java 生成 Main.class 文件，Python 生成*.py 文件，Haskell 生成二进制文件。
3. 利用重定向，将已经生成的矩阵输入，并将结果矩阵输出至相应文件。
4. 等待程序运行完成，并输出在 $O(n^3)$（即两个矩阵相乘的算法部分）所消耗的时间，单位 ms（Haskell 除外，Haskell 计算了输出文本的时间）。
5. 每种语言，在不同规模矩阵下，各运行多组，并计算出平均时间，统计于表格。
6. 得出相应结论。

四、运行效果

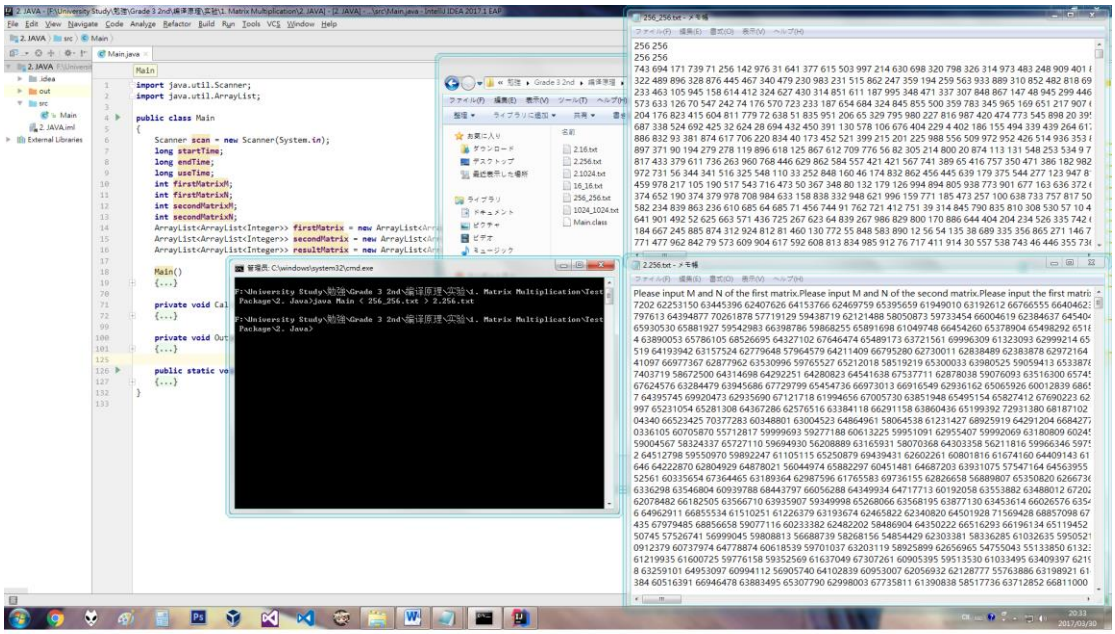
1. MatrixBuild



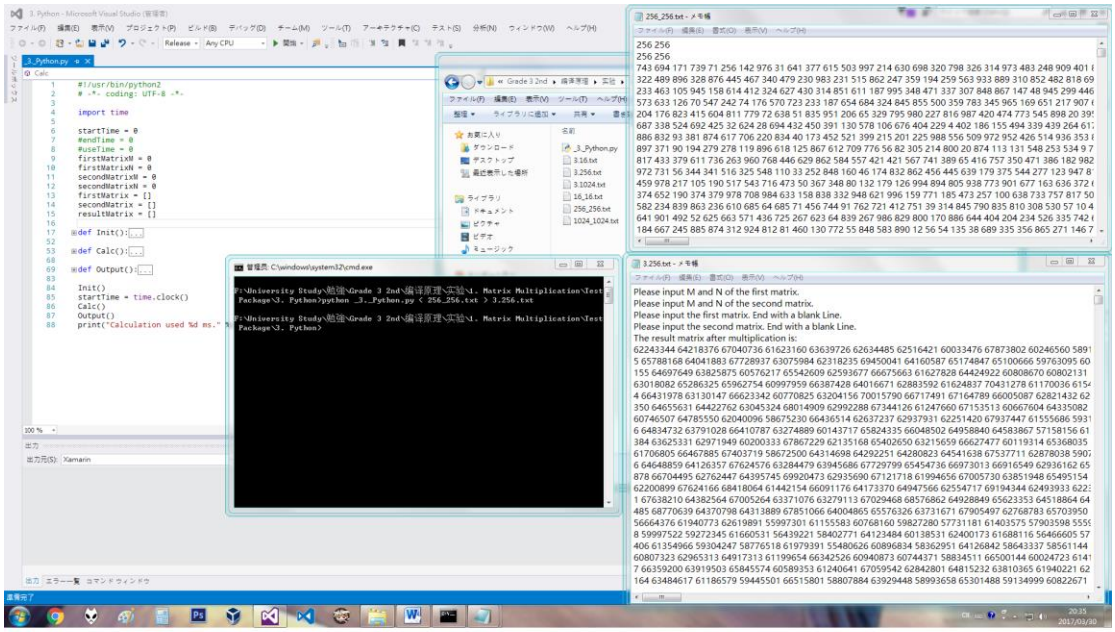
2. C++



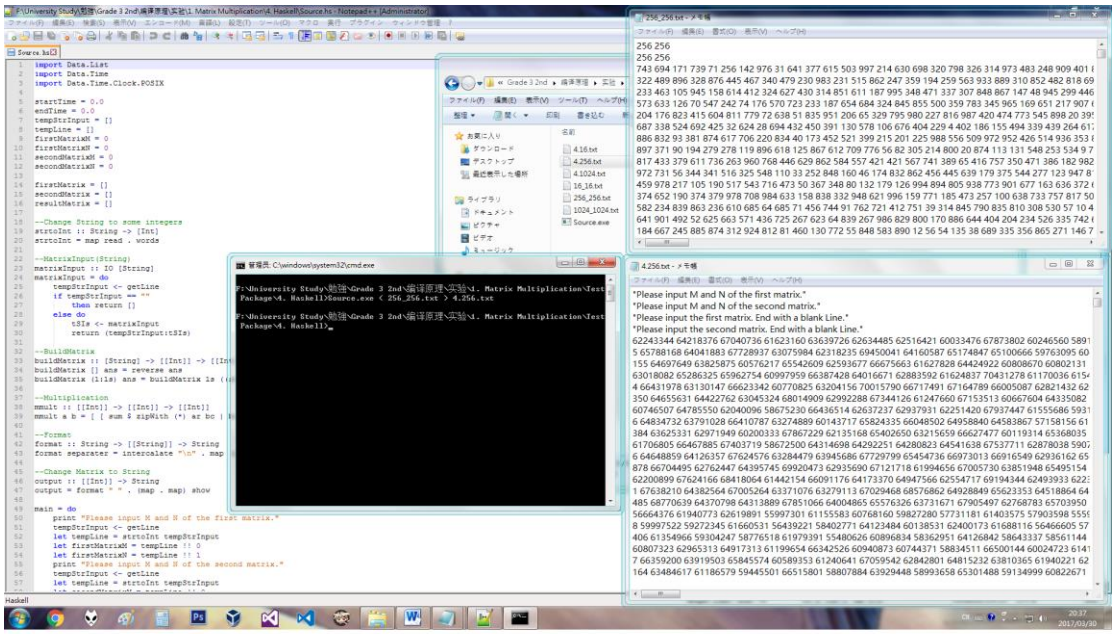
3. Java



4. Python



5. Haskell



五、程序运行性能对比分析（时间单位：ms）

1. C++

	16 × 16	256 × 256	1024 × 1024
1	0	18	4292
2	0	18	4880
3	0	17	4971
4	0	17	4885
5	0	18	4496
6	0	18	5145
7	0	17	5061
8	0	18	4406
Avg	0	17.625	4767

2. Java

	16 × 16	256 × 256	1024 × 1024
1	0	188	28514
2	0	187	26607
3	0	187	26227
4	0	203	29401
5	0	187	29492
6	0	187	26404
7	0	203	29630
8	0	202	25822
Avg	0	193	27762.125

3. Python

	16 × 16	256 × 256	1024 × 1024
1	2	5669	373673
2	2	5492	371021
3	2	5482	365833
4	2	5505	372684
5	2	5463	367594
6	2	5524	376105
7	2	5452	369736
8	2	5635	369314
Avg	2	5527.75	370745

4. Haskell

	16 × 16	256 × 256	1024 × 1024
1	2	2091	167317
2	0	2073	168149
3	0	2162	165560
4	0	2053	166683
5	0	2073	167538
6	0	2110	165814
7	0	2095	165797
8	0	2066	165501
Avg	0.25	2090.375	166544.875

5. 总表

	16 × 16	256 × 256	1024 × 1024
C++	0	17.625	4767
Java	0	193	27762.125
Python	2	5527.75	370745
Haskell	0.25	2090.375	166544.875

六、语言易用性和程序规模对比分析

1. C++

C++在学习难度、编码难度以及相应的程序性能上都出类拔萃。但相应的，C++是比较底层的语言，故其程序规模就相比其他三种语言都麻烦了很多。但是 C++也在与时俱进，在最近的 C++11 以及 C++14 里，C++有了大量的新特性，其在兼容 C 以及旧的 C++的同时依旧能发挥面向对象的特性，C++的编码难度，以及代码规模都有效地降低了。

我们知道，C 以及 C++，由于其底层的特性，运行速度快，故常作为驱动程序、引擎程序的编写。其特性之优，由此次矩阵相乘的简单实验便可看出。C++是一门可以信赖的语言，但也需要程序员的高度细致。

2. Java

Java 是一门纯面向对象的语言。面向对象的特性使得掌握了面向对象程序设计的程序员可以快速地入门和开发，易用性强，程序规模也适中，而且支持跨平台。Java 还是当今移动平台巨头 Android 的语言。这些特性也使得 Java 的市场占有率居高不下。

但是 Java 也有其缺点。在进行矩阵相乘的时候，我关注了资源管理器，发现每当 Java 运行一次，就会占用约 400MB 的内存。我们也可以从 Android 的移动端不难发现，Android 程序的问题在于内存的大量使用。

Java 由于其广泛的应用，且有效的控制、检错等性质，是一门非常好用的语言。

3. Python

   Python 比起前面两种语言，就像是我们在学习算法的时候的伪代码一样，个人认为语法更加的高级，写脚本或者写应用程序等，一目了然。易用性高，程序规模小，成就了当今 Python 的火热。但相应的，其在此次实验中的时间得分是最差的，反映了其为了达到“高级”而所付出的代价。

   Python 的特性使得其能嵌入到非常多的设计软件中，较快速地写脚本，来自动化一些设计过程，当然还有在网络中的广泛应用。Python 是一门接近人类认知的语言，非常好用，但在性能方面却乏善可陈。

4. Haskell

   Haskell 和上边三种语言都不同，Haskell 作为函数式语言，整个的程序建立思路都和前三种有着本质区别，最典型的例子就是 Haskell 没有循环结构。学习函数式思想需要有一定的数学基础，这抬高了这门语言的学习难度，进而降低了 Haskell 的易用性。

   但我们知道，数学一直是简洁而优美的，这也是 Haskell 的特性——矩阵相乘的代码，包含 import 部分以及空行，只需要 4 行代码，干净简练，令人称道。而且 Haskell 语言的惰性特性，使得其运行时间要比 Python 好很多。

   Haskell 的函数特性，数学特性，使得其易用性较低，适合学习而不适合放到市场；但其简练而优美的特性却令人赞叹，而且性能也尚可。Haskell 并不好用，但我相信一旦掌握了函数式的思想，Haskell 的优美必将更令我们震惊。

5. 总结

   C++：易用性较高，程序规模较大，快速高效，编码细致度非常高。

   Java：易用性较高，程序规模中等，比较高效但空间占用较大，编码细致度中等。

   Python：易用性很高，程序规模较小，不大高效，编码压力较小。

   Haskell：易用性较低，程序规模非常小，比较高效，学习成本较高，编码压力较大。

七、心得体会

C++、Java、Python、Haskell，四门语言各有各的特点，各自在程序世界中发挥着不同的作用，人们也不断改进现有语言，设计不同的语言，使得编程能够变得更加简单而高效。虽然目前看来并没有万能的高级语言，但这是侧面证明了守恒的定律；而我们发现一些语言在特定的领域能够有着更加得心应手且高效的应用。这种模块化思想也是程序设计的优秀思想。

我们发现，目的不在于学习一门语言，而在于学习这门语言所运用的程序设计思想。比如能够较熟练运用 C++面向对象特性的程序员也可以快速地上手 Java 的程序开发中，这便是掌握面向对象程序设计思想的效果。

Python 是迈向更“高级”的程序设计语言的一门语言，但在程序性能上却不尽人意，这引发我们思考——语言的高级性和性能上的平衡点在哪里？我们该如何去寻找并实现这个平衡点？

Haskell，Dijkstra 曾高度评价这门语言。在这次的简单实验中，我也确实感受到了函数式程序设计的魅力，简练优雅而不失性能。但是正如上边所说，学会编写 Haskell 以及其他的函数式程序设计语言 Scala 等等，需要的不是在语法上的简单学习，而是函数式思想的建立以及熟练运用，这份思想才是函数式程序设计语言的根本。

程序之海浩瀚无穷，还需要我们进一步去探索、实践。

2017.03.30