

Advanced Data Structure

Homework 0

实验结果复现

运行环境: VMWare 17.04 _Ubuntu 22.04

```
cdm@cdm-virtual-machine:~/Desktop/DAS/hw0$ time ./cppTest
93326215443944152681699238856266700490715968264381621468592963895217
59999322991560894146397615651828625369792082722375825118521091686400
00000000000000000000000000000000

real    0m0.002s
user    0m0.002s
sys     0m0.000s
cdm@cdm-virtual-machine:~/Desktop/DAS/hw0$ time python3 hw0.py
93326215443944152681699238856266700490715968264381621468592963895217
59999322991560894146397615651828625369792082722375825118521091686400
00000000000000000000000000000000

real    0m0.007s
user    0m0.000s
sys     0m0.007s
cdm@cdm-virtual-machine:~/Desktop/DAS/hw0$
```

测试结果显示，C++代码的执行时间显著短于Python代码。此外，在CPU资源占用方面，C++代码主要消耗用户态（user mode）的CPU时间，而Python代码则在较大程度上依赖内核态（kernel mode）的CPU时间。这一发现指向了两种语言在运行时性能及资源调度方面的根本差异。

Q1:为什么使用c++写的程序会比使用python写的程序快?

A1: 在评估C++和Python程序性能时，由上述实验结果结合相关资料分析我们可以归纳出以下关键因素导致C++程序通常比Python程序执行更快：

1. C++代码通常是编译型语言：

- C++代码在执行前会被编译成机器码，这通常意味着它与操作系统的交互较少，因为更多的工作是直接为用户空间完成的，往往在执行计算密集型任务时有更好的性能。

2. Python代码通常是解释型语言：

- Python代码在执行时通常需要一个解释器（如CPython），解释器可能会进行更多的系统调用，比如内存分配和I/O操作，这些都会增加 sys 时间。

3. 库函数和系统调用：

- C++代码在使用标准库或自定义代码执行任务时可能会更倾向于直接在用户空间操作，而不是依赖操作系统的功能。
- 相比之下，Python 的标准库和第三方模块很多时候是对操作系统功能的封装，因此在运行时会有更多的系统调用。

4. 优化级别:

- C++编译器在编译时进行了大量优化以减少不必要的系统调用和其他开销。
- Python解释器可能没有那么多优化，或者优化的重点不同，导致更多系统资源的使用。

5. 静态类型与动态类型:

- C++是静态类型的语言，类型检查在编译时完成，这意味着在运行时不需要进行类型检查，减少了运行时的开销。
- Python是动态类型的语言，它在运行时需要进行类型检查，这增加了额外的运行时开销。

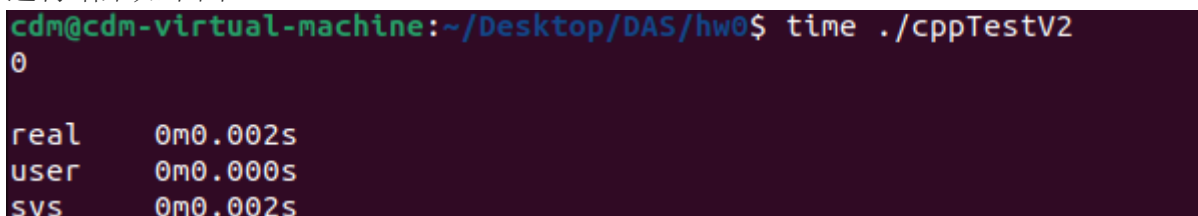
6. 内存管理:

- C++提供了更细致的内存管理机制，开发者可以控制内存的分配和释放，这有助于提高效率。
- Python通过自动垃圾回收来管理内存，这是在性能上可能会带来一些损失。

Q2: 去掉"test.cpp"中的#include<gmpxx.h>，并将mpz_class改成int后，计算100的阶层的运行结果如何？为什么会出现这样的结果？

A2:

1. 运行结果如下图:



```
cdm@cdm-virtual-machine:~/Desktop/DAS/hw0$ time ./cppTestV2
0

real    0m0.002s
user    0m0.000s
sys     0m0.002s
```

结果错误但运行总时长未明显改变，时间更加集中于sys内核态。

2. 原因分析:

- 运行结果错误主要是由于使用int类型进行计算时，由于int类型在多数系统上是32位的，其最大能够表示的数值有限，无法存储像100的阶乘这样巨大的结果。100的阶乘结果远超32位整数能表示的数值范围，导致整数溢出，从而得到错误的运行结果。而在原始使用mpz_class的情况下，mpz_class是GMP库中提供的一个用于表示任意大小的整数的类，可以存储100的阶乘这样大的数。

- 而运行时长分配的改变主要是由于GMP库专门针对大数计算进行了高度优化，GMP库执行大数操作时，主要消耗的是用户空间的CPU时间（即user时间），因为它执行的是计算密集型任务，而不涉及过多的系统调用。而尝试用int来计算100的阶乘，结果会因为整数溢出而不正确。在这种情况下，程序可能会不断地尝试进行无效的计算，或者遇到其他问题（如内存访问错误），这可能会导致频繁的系统调用，比如操作系统处理这些错误的系统调用。这些系统调用会增加sys时间。

Q3:除了编程语言外，影响程序运行快慢还可能有哪些因素？

A3:

1. 硬件资源:

- CPU速度和架构
- 内存大小和速度
- 存储设备的速度（HDD vs. SSD）
- 显卡（对于图形密集型程序）

2. 操作系统:

- 操作系统的效率
- 操作系统的调度策略
- 系统负载和资源管理

3. 编译器/解释器优化:

- 编译器的优化能力
- 解释器执行代码的效率

4. 代码效率:

- 算法的选择和实现
- 数据结构的选择
- 编码实践，比如循环展开和函数内联

5. 并发和并行计算:

- 多线程和多进程的使用
- 分布式计算
- GPU加速

6. I/O操作:

- 磁盘I/O的频率和效率
- 网络延迟和带宽

7. 外部库和依赖:

- 第三方库的性能
- 系统调用的开销

8. 内存管理:

- 内存分配和释放策略
- 内存泄漏和碎片化

9. 编译器指令和标志:

- 优化级别
- 特定硬件指令集的使用

10. 程序的配置和环境:

- 系统环境变量的设置
- 程序的运行时配置

11. 运行时环境:

- 其他正在运行的应用程序
- 系统的安全措施，如防火墙和杀毒软件