TP:随机流 – 机会模型 – 准随机数 – 并行性
生成并行的伪随机数流 与专业科学图书馆的对抗 – CLHEP

PJ:CLHEP 库,2D 简单积分计算代码（参见 Simu ZZ2）

实验室的主要目标:面对专业规模的模拟库（高能物理:CLHEP)并进行分布式随机计算。请注意:在回答问题之前请仔细阅读主题和提示。从为 CLHEP 提议的版本开始,这个旧版本可以使用常见的 C++ 编译器（特别是 G++)进行移植。如果您使用个人 Windows 计算机,如有必要,请在 Windows（或其他版本的 Linux)下安装 Ubuntu -

https://korben.info/tuto-pour-installer-linux-sous-windows-10-avec-wsl.html和开发包。

1)安装库（观察目录的层次结构,源代码、包含文件、测试代码在哪里,编译后库的位置在哪里（检查日期 - 旧版本是 2005 年的,如果过程顺利,您将拥有今天的 .a 和 .so ） ,编译、链接,然后查看创建的新目录、其路径的库的创建……使用代码测试该库。实验结束时给出的 C++ 以及提供的测试代码(testRand.cc)

2) 使用 saveStatus 方法归档 Mersenne Twister (MT) 生成器的一些状态文件。将这些状态与少量抽奖（例如 10 个号码）分开。使用restoreStatus 方法测试恢复到给定状态。我们找到相同的伪随机数来重现一个序列,对其进行调试等。位对位的可重复性对于计算中的调试是必要的（如果我们无法再调试,计算就死了.....）。根据记录,伪随机数的生成器是确定性算法,它们只是或多或少良好质量的机会的“模拟” ,但可以（必须)根据其初始状态再现相同的序列。

3) 编写代码来保存 20 亿次抽奖中的 10 个单独状态（使用 saveStatus）。

4) 创建一个代码,使用蒙特卡洛方法链接 10 次球体体积（半径 1)计算的重复。每次模拟绘制 10 亿个点来计算近似值,每次复制需要 20 亿个数字,这是在问题 3 中预先计算的。每次复制通过将生成器恢复为预先计算的状态,使用独立的伪随机数流（使用每次复制开始时的restoreStatus方法）。测量时间

顺序计算（10 x 20 亿次打印）。

5) 使用“序列分割”技术并行化执行,同时计算 10 个复制。最简单的方法是保持程序顺序并使用多任务系统执行“SPMD”（简单程序多数据）。在我们的例子中,改变的数据只是随机流。考虑如何做到这一点,然后提出一种在具有独立随机流的多核服务器（0 到 X 个逻辑核心)上并行运行 10 个模拟复制的方法。我们可以有一个程序,通过将状态文件作为参数来计算球体（3D)体积的估计值,并在标准输出上给出与该随机序列相对应的估计值（标准输出的结果可能突然变成重定向到一个文件）。每个命令行都可以在后台运行（&请参阅下面的示例命令行。10 个模拟的命令可以分组在一个脚本中,该脚本将并行运行所有复制。

```
$ simuSV3D MTStatus-7>EstimationSV3D-7&
```

可选问题：

6.a) 自己发现不同的并行化技术或库　使用现有代码的示例返回问题 5（例如，使用蒙特卡罗技术在 1,2 和 3 维中模拟穿过一滴水的“慢”中子在第六部分中，您可以使用主要并行库之一，而不是使用每个使用流并在 Unix 下并行启动的独立顺序程序： pthread、OpenMP（或 OpenMPI）通过启动问题5的模拟（根据其他课程的进度进行工作）。

6.b) 生物信息学应用：使用机器翻译生成单词和句子的一部分，其中的字符是随机绘制的。例如，编写一个尝试随机生成寡肽的简单程序：“gattaca” - https://fr.wikipedia.org/wiki/Bienvenue_%C3%A0_Gattaca

分子生物学的一些提醒可以在下面找到： https://www.supagro.fr/ress-tice/ue1-ue2_auto/Bases_Biologie_Moleculaire_v2/co/0_module_bases_biologie_moleculaire_V2_1.html

每次抽奖都会给出一个核酸碱基　用具有 4 种可能性的字母表示（“A”、“C”、“G”、“T”）。很容易计算可能的组合数量。模拟这个过程并计算获得 1 的尝试次数

时代 成功找到序列：“AAATTTGCGTTCGATTAG”（在单个复制中）。然后进行 40 次独立重复，得出平均试验次数和置信半径。首先，不要重置发生器并按顺序工作。然后使用“序列分割”技术，恢复适当间隔的 TM 状态以进行并行工作。当您完成多次模拟后，您将能够计算概率平均值和标准差，理想情况下给出置信区间。

与精确概率进行比较。尝试随机生成一个可理解的句子：“机会不会写信息。机会编写程序（或基因）更不容易。根据旧的ASCII码，提前估计计算时间以及偶然获得该字符串的概率。最后，估计“偶然”拥有人类及其所有功能基因的精确链条的概率　我们可以认为它包含 30 亿个核酸碱基。而不是考虑时间和生成非常非常长的句子的概率

易于理解，我们可以寻找“随机”找到一本书中包含章节、段落、句子等的连贯文本的概率。

CLHEP 安装提示：

安装 CLHEP 库（以 .tgz 格式提供）。检查文件夹的层次结构、源文件的位置、包括...安装前后– 列出文件和文件夹并注明日期。安装前查看以下所有提示。下面的 Unix 命令 tar（磁带存档）将以详细格式解压缩 .tgz。

```
$ tar zxvf CLHEP-Random.tgz
```

在安装过程中，您将首先进行顺序安装。首先，在已创建的Random目录中找到 configure 脚本，在进行编译之前启动它，检查并记下总编译时间。

```
./configure --prefix=$PWD //其中 PWD 是安装目录时间 make
                                                    // 查看顺序编译需要多少时间
```

然后，删除创建的目录 (rm –fr ./Random)。现在您将并行编译以充分利用您用于实验的 SMP 计算机的全部潜力。etud 服务器建议使用许多逻辑核心。检查并注意并行编译可以节省多少时间（尽管你们都共享同一台机器）。

用户时间大致相同,系统时间也大致相同,但是由于这种并行性,实际时间（挂钟） -您的实际等待时间- 将会减少。

```
./configure --prefix=$PWD //其中 PWD 是安装目录时间 make –jXXX make install
                                                    // 指定使用 XXX 个逻辑核心进行并行编译
                                                    // Makefile 的这条规则完成安装
```

测试此文件末尾提出的代码以及包中提出的主要测试（完整测试 - 代码位于已安装库的测试目录中）。为了编译代码,您需要查看“include”和“lib”库的安装位置。如果安装正确,您将找到 CLHEP 库的两个版本。两个文件的日期/时间将与您的编译相对应。一个是带有“.a”扩展名的静态版本（例如 libm.a 表示标准数学库的静态版本）。您可以使用常规ar Unix 命令查看静态库内的目标文件。例如： ar –t libm.a列出了标准 C 数学库的所有目标文件。

```
libCLHEP-Random-2.1.0.0.a
```

另一个将是动态版本“.so” 用于共享对象 这对应于 Windows 下的 DLL（动态链接库）。在此版本中,每个编译后的目标文件只有一个版本将加载到所有可执行文件的内存中。对象将由执行进程在运行时共享和重新输入。

```
libCLHEP-随机-2.1.0.0.so
```

要编译测试文件（如下所示）,您可以先从单独的编译（“-c”选项)开始,以避免链接阶段。 “-I”选项给出了我们找到包含目录的路径 在本例中,我们假设 testRand.cpp 文件位于包含目录的正上方。您可以使用绝对路径（这会比较长）。

```
g++ -c testRand.cc –I./include
```

如果此阶段没有问题,您可以进一步在编译后添加链接阶段。如果我们删除“-c”选项并使用“-L”选项精确指定lib目录的路径,则可以完成此操作。然后我们使用输出“-o”选项请求可执行结果

```
g++ testRand.cc –I./include –L./lib –o myExe
```

这还不够,因为我们没有提到我们在此链接阶段使用 CLHEP 库。（就像我们忘记为使用数学库函数的 C 程序精确指定“-lm”一样）。根据服务器安装环境（每年都会随着更新而变化）,您必须找到可以正常工作的正确编译行。下面是一些行;第一行是在动态链接环境中。然后我们越来越趋向于静态。

```
（1)g++ -o myExe testRand.cc –I./include –L./lib -lCLHEP-Random-2.1.0.0
（2)g++ -o myExe testRand.cc –I./include –L./lib -lCLHEP-Random-2.1.0.0 -static
（3） g++ -o myExe testRand.cc –I./include –L./lib ./lib/libCLHEP-Random-2.1.0.a
（4)g++ -o myExe testRand.cc –I./include ./lib/libCLHEP-Random-2.1.0.a
```

当一切正常时,您可以进入测试目录并编译testRandom.cc文件,这样可以更广泛地使用库中提出的所有对象和方法。在现代 C++ 编译器上,您可能必须为退出函数包含stdlib.h 。这是通过#include <cstdlib>实现的（请注意,在标准名称中添加了“c” ,并且在包含旧 C 库时,现代 C++ 中未提及“.h”扩展名）。如果您熟悉 Unix,则可以使用ldd Unix 命令查看可执行文件的依赖项 - 您可能会看到必须更新 LD_LIBRARY_PATH 全局变量以指定正确的 PATH 并启用与共享对象库的动态链接。

```
导出 LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PWD/CLHEP/lib
```

编译后,可执行文件将让您了解该库对于模拟主要概率的蒙特卡罗模拟的实用性。

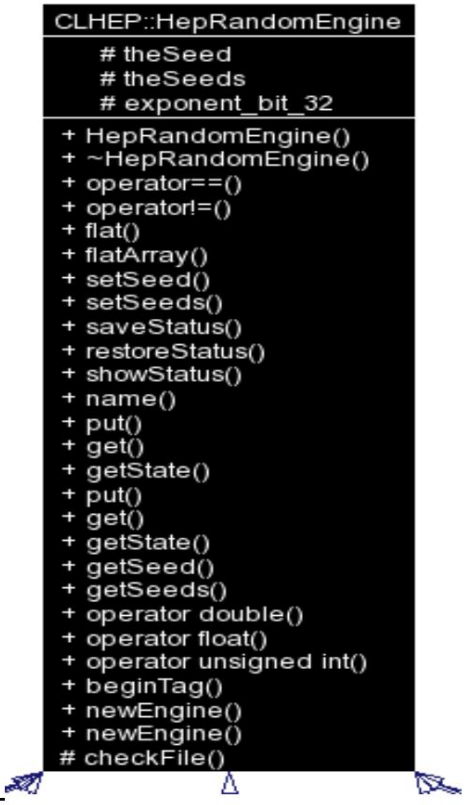
CLHEP程序的理想编译:注意g++版本。查看已安装文件的结

构和位置,并调整包含路径-I和库路径-L。下面是一个示例:

```
CLHEP_DIR=安装目录路径
g++ -o testRandom testRandom.cc -I$CLHEP_DIR/Random/
        include -L$CLHEP_DIR/Random/lib

        -lCLHEP-随机-2.1.0.0
```

如果需要,请添加-static以将库包含在可执行文件中,或者在启动编译的程序之前将库的路径添加到 LD_LIBRARY_PATH 环境变量中



超一流

要绘制数字,请使用以下方法： flat();

要保存状态：

saveStatus(nomDechier);

要将发电机恢复到某种状态：

恢复状态（nomDeFichier）；

潜水艇

CLHEP 库提供了许多生成器,根据目前的知识,只有 2 个生成器是正确的:Ranlux64Engine（非常慢，“豪华”级别为 64,以及 Matsumoto 的 Mersenne Twister - 不加密）。这是该库的生成器列表。

詹姆斯·兰登

三兰特

DRand48Engine

双随机

Hurd160发动机

MTwistEngine

非随机引擎

拉内库引擎

冉石发动机

Ranlux64引擎

瑞朗克斯引擎

Hurd288引擎

随机引擎

CLHEP 代码示例：

在二进制文件中生成数字（用于使用 G. Marsaglia 的 DIE HARD 软件进行测试）

```
// 在二进制文件中生成数字（用于 G. Marsaglia 的 DIE HARD 测试）

#包括<sys/types.h>
#包括<sys/stat.h>

#包括<fcntl.h>
#包括<限制.h>
#包括<unistd.h>

#include CLHEP/Random/MTwistEngine.h // CLHEP 库中 MT 的标头

int主要()
{
    CLHEP::MTwistEngine * mtRng = 新 CLHEP::MTwistEngine();

    整数 飞德；
    双倍的 飞瑞；
    无符号整数iRn；

    // 用于测试 Die Hard 的二进制文件 fId = open( ./qrngb ,O_CREAT|
    O_TRUNC|O_WRONLY,S_IRUSR|S_IWUSR);

    // Die Hard 测试针对 300 万个数字
    对于（int i = 1; i < 3000000; i++）
    {
        fRn = mtRng->平(); // 使用 MT 统一生成
        iRn = (无符号整数) (fRn * UINT_MAX);

        写入（fId,&iRn， sizeof（无符号整数））；
    }

    关闭（fId）；

    删除mtRng；

    返回0；
}
```

如果您想添加生成器,请参考:当中间乘法结果超过 32 位时,您将需要使用 long long。

这可能会在编译过程中造成问题 (ANSI ISO C99),如有必要,您必须修改configure.in – Makefile.am文件,并使用引导程序 (./bootstrap)考虑这些修改。

要考虑到修改,请记住使用rm –fr lib*删除库,或者更明智的是,在正确的目录中调整并执行 make 命令。然后你可以再次运行make –jXX并进行 make install。

仅供参考 SFMT:Saito 研究员和 Matsumoto 的前博士生与他一起提出了一种性能更好的发电机,包括 2 周期 216091 -1,生成速度
在大多数架构上速度是原来的两倍 (2006)。 2009 年提出了 GP-GPU 生成器,最近提出了周期更短（内存占用更低)的 Tiny MT。

网址： http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html

附录 – 球体的体积 – 简单的蒙特卡罗代码

```
13  /** ------------------------------------------------- *
14    * computeSphereVolume                               *
15    *                                                   *
16    * @brief  This function computes the volume of a sphere *
17    *         using Monte Carlo simulation (Radius = 1)  *
18    *         Analytically V = 4/3πR³                    *
19    *                                                   *
20    * @param  The number of Points                      *
21    *                                                   *
22    * @return An approximation of the sphere surface    *
23    * ------------------------------------------------- **/
24
25  double computeSphereVolume(int numOfPoints)
26  {
27      int insideSphere = 0;
28
29      // Check if the points are inside the French
30      for(int i = 0; i < numOfPoints; i++)
31      {
32          double x = random_double();
33          double y = random_double();
34          double z = random_double();
35
36          double distance = sqrt(x * x + y * y + z * z);
37
38          if (distance <= 1.0) insideSphere++;
39      }
40
41      // Volume of the cube = 8 (with side length 2)
42      return (8. * (double) insideSphere / (double) numOfIPoints);
43  }
```