实验报告

实验名称（利用 GEM5 评测系统性能）

班级 智能 1602 学号 201608010708 姓名田宗杭

实验目标

利用 GEM5 仿真 x86-64 系统，测量 FFT 程序运行时间，与实际 x86-64 系统的测试结果进行比较。

实验要求

* 采用 C/C++编写程序，选择合适的运行时间测量方法
* 根据自己的机器配置选择合适的输入数据大小 n，保证足够长度的运行时间
* 回答思考题，答案加入到实验报告叙述中合适位置

思考题

1. GEM5 是什么？怎么使用？
2. 利用仿真器评测系统性能的优点是什么？缺点是什么？
3. GEM5 仿真系统测得的程序性能与实际系统测得的程序性能差别大不大？可能的原因是什么？

实验内容

GEM5 的安装和使用

GEM5 的官方文档在[这里](http://gem5.org/Documentation)，包括了安装和使用说明。

GEM5 网站也提供了 x86-64 系统 Linux 镜像下载，可以在 GEM5 的[下载](http://gem5.org/Download)页面找到。

大家也可以参考[这篇博客文章](https://blog.csdn.net/u012822903/article/details/62444286)及其相关文章，了解如何利用 GEM5 进行 Linux 全系统模拟和运行自己的测试程序。

利用 GEM5 测试 FFT 程序在 x86-64 系统上的性能

在安装好 GEM5，并掌握如何使用 GEM5 运行自己的测试程序后，可以在 GEM5 上运行 FFT 程序，测试其性能。建议首先使用实验一中的单线程 FFT 程序，然后再尝试实验二中的多线程 FFT 程序，记录测试数据。

测试

测试平台

在如下机器上进行了测试：

部件	配置	备注
CPU	core i7-6700U	
内存	DDR3 8GB	
操作系统	Ubuntu 18.04 LTS	中文版

测试记录

单(多)线程 FFT 程序的测试参数如下：

参数	取值	备注
数据规模	64	
线程数目	1	

FFT 程序运行过程的截图如下：

```
tian@ubuntu:~$ cd Downloads
tian@ubuntu:~/Downloads$ cd gem5-40c18bb90501
tian@ubuntu:~/Downloads/gem5-40c18bb90501$ sudo build/X86/gem5.opt configs/examp
le/fs.py
[sudo] password for tian:
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 compiled Dec  9 2018 21:58:14
gem5 started Dec 19 2018 16:26:42
gem5 executing on ubuntu, pid 2577
command line: build/X86/gem5.opt configs/example/fs.py

Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assign
ed (512 Mbytes)
info: kernel located at: /home/tian/Downloads/gem5-40c18bb90501/fs-image/binarie
s/x86_64-vmlinux-2.6.22.9
Listening for com_1 connection on port 3456
      0: rtc: Real-time clock set to Sun Jan  1 00:00:00 2012
0: system.remote_gdb: listening for remote gdb on port 7000
warn: Reading current count from inactive timer.
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
```

```
tian@ubuntu:~$ cd Downloads
tian@ubuntu:~/Downloads$ cd gem5-40c18bb90501
tian@ubuntu:~/Downloads/gem5-40c18bb90501$ cd util/term/tian@ubuntu:~/Downloads/gem5-40c18bb90501/util/term$ sudo ./m5term 127.0.0.1 3456
[sudo] password for tian:
==== m5 slave terminal: Terminal 0 ====
Linux version 2.6.22.9 (blackga@nacho) (gcc version 4.1.2 (Gentoo 4.1.2)) #2 Mon Oct 8 13:13:00 PDT 2007
Command line: earlyprintk=ttyS0 console=ttyS0 lpj=7999923 root=/dev/hda1
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
 BIOS-e820: 000000000009fc00 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 0000000020000000 (usable)
 BIOS-e820: 0000000020000000 - 00000000c0000000 (reserved)
 BIOS-e820: 00000000fffff0000 - 0000000100000000 (reserved)
end_pfn_map = 1048576
kernel direct mapping tables up to 100000000 @ 8000-d000
DMI 2.5 present.
Zone PFN ranges:
  DMA             0 ->     4096
  DMA32        4096 ->  1048576
  Normal    1048576 ->  1048576
early_node_map[2] active PFN ranges
    0:        0 ->      159
    0:      256 ->   131072
Intel MultiProcessor Specification v1.4
MPTABLE: OEM ID:  MPTABLE: Product ID:  MPTABLE: APIC at: 0xFEE00000
Processor #0 (Bootup-CPU)
I/O APIC #1 at 0xFEC00000.
Setting APIC routing to flat
Processors: 1
swsusp: Registered nosave memory region: 000000000009f000 - 00000000000a0000
swsusp: Registered nosave memory region: 00000000000a0000 - 0000000000100000
Allocating PCI resources starting at c4000000 (gap: c0000000:3fff0000)
Built 1 zonelists.  Total pages: 127570
Kernel command line: earlyprintk=ttyS0 console=ttyS0 lpj=7999923 root=/dev/hda1
Initializing CPU#0
PID hash table entries: 2048 (order: 11, 16384 bytes)
```

```
(none) / # ls
bin    dev  fft      home  lib32  lost+found  opt   root  sys  usr
boot   etc  fft.cpp  lib   lib64  mnt         proc  sbin  tmp  var
(none) / # ./fft
```

FFT 程序的输出

![图 1 程序输出](../lab1/perf_ls.png)

n	x[]	X[]	f
0	+0.000	+0.000	0
1	+2.834	+0.000	1
2	+1.295	+32.000	2
3	+1.269	+0.000	3
4	+0.166	+0.000	4
5	+2.570	+32.000	5
6	-0.222	+0.000	6
7	+1.756	+0.000	7
8	+1.000	+0.000	8
9	+0.839	+0.000	9
10	-2.064	+0.000	10
11	-1.145	+32.000	11
12	+2.014	+0.000	12
13	+1.305	+0.000	13
14	+1.345	+0.000	14
15	-0.449	+0.000	15
16	+2.000	+0.000	16
17	-0.840	+32.000	17
18	+0.579	+0.000	18
19	+0.194	+0.000	19
20	+0.599	+0.000	20
21	-2.808	+0.000	21
22	-3.912	+0.000	22
23	-1.122	+0.000	23
24	-1.000	+0.000	24
25	-0.205	+0.000	25
26	-2.070	+0.000	26
27	+0.907	+0.000	27
28	-1.248	+0.000	28
29	+0.158	+32.000	29
30	+0.530	+0.000	30
31	+2.444	+0.000	31
32	+0.000	+0.000	32
33	-2.444	+0.000	33
34	-0.530	+0.000	34
35	-0.158	+32.000	35
36	+1.248	+0.000	36
37	-0.907	+0.000	37
38	+2.070	+0.000	38
39	+0.205	+0.000	39
40	+1.000	+0.000	40
41	+1.122	+0.000	41
42	+3.912	+0.000	42
43	+2.808	+0.000	43
44	-0.599	+0.000	44
45	-0.194	+0.000	45
46	-0.579	+0.000	46

```
 47      +0.840   +32.000 47
 48      -2.000   +0.000  48
 49      +0.449   +0.000  49
 50      -1.345   +0.000  50
 51      -1.305   +0.000  51
 52      -2.014   +0.000  52
 53      +1.145   +32.000 53
 54      +2.064   +0.000  54
 55      -0.839   +0.000  55
 56      -1.000   +0.000  56
 57      -1.756   +0.000  57
 58      +0.222   +0.000  58
 59      -2.570   +32.000 59
 60      -0.166   +0.000  60
 61      -1.269   +0.000  61
 62      -1.295   +32.000 62
 63      -2.834   +0.000  63
it took 0.000576 seconds
(none) / #
```

在原虚拟机系统上的运行时间为：

```
 35      -0.158   +32.000 35
 36      +1.248   +0.000  36
 37      -0.907   +0.000  37
 38      +2.070   +0.000  38
 39      +0.205   +0.000  39
 40      +1.000   +0.000  40
 41      +1.122   +0.000  41
 42      +3.912   +0.000  42
 43      +2.808   +0.000  43
 44      -0.599   +0.000  44
 45      -0.194   +0.000  45
 46      -0.579   +0.000  46
 47      +0.840   +32.000 47
 48      -2.000   +0.000  48
 49      +0.449   +0.000  49
 50      -1.345   +0.000  50
 51      -1.305   +0.000  51
 52      -2.014   +0.000  52
 53      +1.145   +32.000 53
 54      +2.064   +0.000  54
 55      -0.839   +0.000  55
 56      -1.000   +0.000  56
 57      -1.756   +0.000  57
 58      +0.222   +0.000  58
 59      -2.570   +32.000 59
 60      -0.166   +0.000  60
 61      -1.269   +0.000  61
 62      -1.295   +32.000 62
 63      -2.834   +0.000  63
it took 0.000157 seconds
```

分析和结论

1. GEM5 是什么？怎么使用？

GEM5 模拟器是用于计算机系统架构研究的模块化平台，包括系统级架构以及处理器微体系结构。GEM5 是一款模块化的离散事件驱动全系统模拟器，它结合了 M5 和 GEMS 中最优秀的部分，是一款高度可配置、集成多种 ISA 和多种 CPU 模型的体系结构模拟器。M5 是由 Michigan 大学开发的一款开源的多处理机模拟器，受到了业内的广泛关注，很多高水平论文都采用 M5 作为研究工

具。另一方面，Wisconsin 推出的 GEMS 能够对储存层次进行详细而灵活的模拟，包括对多种不同的 cache 一致性协议和互联模型的支持。目前的 GEM5 是 M5 和 GEMS 的一个紧耦合版本。GEM5 已经能够支持多种商用 ISA，包括 X86、ARM、ALPHA、MIPS、Power、SPARC 等，并且能够在 X86、ARM、ALPHA 上加载操作系统。

GEM5 的全系统仿真测试使用过程：

进行全系统仿真

1.进入 gem5 目录下，编译 X86 架构

```
root@ubuntu14:/home/zzh/gem5# scons build/X86/gem5.opt
```

2.下载 X86 架构对应的全系统文件，也就是 disk

```
wget http://www.m5sim.org/dist/current/x86/x86-system.tar.bz2
```

3.下载 alpha 对应的全系统文件，这是因为后面会用到里面解压出来的一个文件。

```
wget http://www.m5sim.org/dist/current/m5_system_2.0b3.tar.bz2
```

4.gem5 目录下新建个文件夹 fs-image，进入该目录，解压 x86-system.tar.gz

```
root@ubuntu14:/home/zzh/gem5# mkdir fs-image
root@ubuntu14:/home/zzh/gem5# cd fs-image
root@ubuntu14:/home/zzh/gem5/fs-image# tar -xjf x86-system.tar.bz2
```

5.我第三步下载的 alpha 对应的全系统文件——m5_system_2.0b3 放在 Download 目录下，将其解压，并将 disks 目录下的 linux-bigswap2.img 放到 x86-system 解压后的 disks 目录下。

```
root@ubuntu14:/home/zzh/gem5/fs-image#                                    cp
/home/zzh/Downloads/m5_system_2.0b3/disks/linux-bigswap2.img disks/
```

6.修改.bashrc 文件，添加下面的环境变量

```
root@ubuntu14:~# ls -a
.  ..  .bash_history  .bashrc  .profile  .viminfo
root@ubuntu14:~# vim .bashrc

export M5_PATH=$M5_PATH:/home/zzh/gem5/fs-image/
```

7.使环境变量生效

```
root@ubuntu14:~# source .bashrc
```

8.进入 gem5 目录下，启动 X86 架构的 FS 模式

9.错误：Can’t find a path to system files.
解决办法：修改 gem5 目录下的 configs/common/SysPath.py，将目录改成你现在的目录

```
except KeyError:
            path = [ '/dist/m5/system', '/n/poolfs/z/dist/m5/system' ]
except KeyError:
            path = [ '/dist/m5/system', '/home/zzh/gem5/fs-image' ]
```

10.错误：Can’t find file ‘x86root.img’ on path.
解决办法：修改 Benchmarks.py 文件，将 x86root.img 改为 linux-x86.img

```
ubuntu14:~/gem5/configs/common$ vim Benchmarks.py

elif buildEnv['TARGET_ISA'] == 'x86':
            return env.get('LINUX_IMAGE', disk('x86root.img'))


elif buildEnv['TARGET_ISA'] == 'x86':
            return env.get('LINUX_IMAGE', disk('linux-x86.img'))
```

11. ubuntu14:~/gem5$sudo build/X86/gem5.opt configs/example/fs.py --kernel=x86_64-vmlinux-2.6.22.9 --disk-image=linux-x86.img

12.打开另一个终端，用于连接 FS 模拟的全系统。这里使用的是 m5term，先编译安装这个工具，在 gem5 的目录下有。

```
ubuntu14:~/gem5$ cd util/term/
ubuntu14:~/gem5/util/term$ make cc  -o m5term term.c
ubuntu14:~/gem5/util/term$ sudo make install
[sudo] password：
install -o root -m 555 m5term /usr/local/bin
```

13.用如下命令连接，注意 3456 是个端口，是在上图里出现的端口 3456，默认情况下启动的第一个端口都是这个，再接着开启第二个就会是 3457。

```
zzh@ubuntu14:~/gem5/util/term$ sudo ./m5term 127.0.0.1 3456
```

编译并运行程序：
1.首先进入 gem5 目录下，可以新建一个文件夹 mountfile，专门用于存放后面的往系统里 mount 的文件。

```
ubuntu14:~/gem5$ mkdir mountfile
```

先把已经待编译文件放进该文件夹 mountfile，

```
ubuntu14:~/gem5$ sudo cp ~/code/TestGauss2/TestGauss2 mountfile/
```

2.挂载：

```
ubuntu14:~/gem5$ sudo mount -o,loop,offset=32256 fs-image/disks/linux-x86.img /mnt
```

3.显示一下/mnt，可以看到挂载好的操作系统

```
ubuntu14:~/gem5$ ls /mnt
bin   dev  home  lib32  lost+found  opt   root  sys  usr
boot  etc  lib   lib64  mnt         proc  sbin  tmp  var
```

4.将可执行的程序文件复制进挂载的系统

```
zzh@ubuntu14:~/gem5$ sudo cp mountfile/TestGauss2 /mnt
```

5.显示一下/mnt，可以看到已经在系统里了

```
zzh@ubuntu14:~/gem5$ ls /mnt
bin   dev  home  lib32  lost+found  opt   root  sys         tmp  var
boot  etc  lib   lib64  mnt         proc  sbin  TestGauss2  usr
```

6.在使用 linux 的 image 文件重新开启 gem5 之前，应该执行 umount 操作

```
zzh@ubuntu14:~/gem5$ sudo umount /mnt
```

7.然后再重新启动 gem5 的全系统

```
zzh@ubuntu14:~/gem5$ sudo build/X86/gem5.opt configs/example/fs.py
```

另一个终端：

```
zzh@ubuntu14:~/gem5/util/term$ sudo ./m5term 127.0.0.1 3456
```

8.这样就可以在系统里直接编译并执行程序

2. 利用仿真器评测系统性能的优点是什么？缺点是什么？

优点：
通过对计算机系统的建模、仿真及评估，科研人员可以验证新型体系结构设计;将该技术导入产品设计则可优化系统方案、降低开发风险并提升开发效率.作为一种系统评估手段，体系结构模拟器运行在宿主机上，通过加载测试程序来验证新的设计方案，发现其中潜在的缺陷，从而改进设计并有效控制风险. 体系结构模拟器通常使用软件方式对部分或全部计算机系统硬件建模，对体系结构的指令集架构、处理器、存储系统、网络传输拓扑结构等进行模拟，验证系统的功能和性能. 体系结构模拟器已成为系统研究和设计开发中不可或缺的工具.

缺点：
1、真机调试更好，因为真机调试就是在我们平常使用的手机上操作，测试更准确;模拟器在某些方面往往达不到真机的真实水平。
2、真机调试更能清晰真实的反映出开发过程中出现的问题;而模拟器性能比较差，在模拟器上不一定能发现。
3、真机测试更能支持横竖屏都方便，有一些情况模拟机不行。

4、搞 3D 图形图像时候，真机支持，虚拟机不一定支持 OpenGL ES。
5、真机调试速度快，模拟器速度慢
6、功耗模型，目前 GEMS 和 M5 可以使用外挂的功耗模型，比如 Orion 和 McPAT，但是 gem5 希望集成一个综合性的模块化的功耗模型。这凸显了功耗问题已经成为当前的体系机构研究和微处理器产业发展的瓶颈问题之一。
7、对各种 ISA/CPU/memory 模型的全面支持。目前的版本可能在某些配置下不能很好地运行。
8、并行化，显然目前 GEM5 的性能还不尽如人意。为此，需要做并行化的工作。
9、检查点导入，即使是用 GEM5 的简单 CPU 模型来进行模拟，其速度也明显慢于那些基于二进制翻译的模拟器，比如 QEMU 或者 SimNowTM 等等。模拟器中的状态检查点转移到 GEM5 中。用户可以将模拟快速推进到某个时刻点，然后再从这个时刻点开始继续模拟。如果采用类似 KVM 等虚拟机的话，可能获得比用二进制翻译器更高的性能。

3. GEM5 仿真系统测得的程序性能与实际系统测得的程序性能差别大不大？可能的原因是什么？

从测试记录来看，FFT 程序在 GEM5 上的执行时间是 0.000576s，与实际系统上测得的执行时间相比，大约为 3 倍。