

实验报告

实验名称（利用 **GEM5** 评测系统性能）

智能 1501 201508010513 王鑫淼

实验目标

利用 GEM5 仿真 x86-64 系统，测量（多线程）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

1、安装 **gem5**

1) 安装库文件：

```
sudo apt-get install mercurial scons swig gcc m4 python python-dev  
libgoogle-perftools-dev g++ libprotobuf-dev
```

安装好编译环境：sudo apt-get install build-essential

2) 下载安装 **gem5** 源码：

下载: hg clone <http://repo.gem5.org/gem5>

安装编译 gem5 的各个架构: scons build/X86/gem5.opt

```
[ SHCXX] nomali/lib/MMU.cc -> .os  
[ SHCXX] nomali/lib/nomali_api.cc -> .os  
[ AR] -> nomali/libnomali.a  
[ RANLIB] -> nomali/libnomali.a  
[ LINK] -> x86/gem5.opt  
scons: done building targets!
```

2、Gem5 的 FS（全系统）模拟

1) 下载文件

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>

下载 alpha 对应的全系统文件

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

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

m5_system_2.0b3 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。

2) 修改文件

修改.bashrc 文件，添加环境变量：

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

修改 SysPath.py 文件：

```
except KeyError:  
    paths = [ '/dist/m5/system', '/home/wxm/gem5/fs-image' ]
```

修 Benchmarks.py 文件：

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

3) 启动 FS

启动 FS: sudo build/X86/gem5.opt configs/example/fs.py

进入 gem5/util/term，安装 m5term

make

sudo make install

sudo ./m5term 127.0.0.1 345

```
wxm@ubuntu:~/gem5/util/term$ sudo ./m5term 127.0.0.1 3456
[sudo] wxm 的密码:
==== m5 slave terminal: Terminal 0 ====
Linux version 2.6.22.9 (blackga@nacho) (gcc version 4.1.2 (Gent
Command line: earlyprintk=ttyS0 console=ttyS0 lpj=7999923 root=
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)
```

3、gem5 运行 X86 编译的测试程序

1) 挂载:

挂载操作系统: sudo mount -o,loop,offset=32256 fs-image/disks/linux-x86.img /mnt

挂载可执行的程序文件: sudo cp mountfile/fft1 /mnt

执行 umount 操作: sudo umount /mnt

2) 启动 FS 编译文件:

~/gem5\$ sudo build/X86/gem5.opt configs/example/fs.py

~/gem5/util/term\$ sudo ./m5term 127.0.0.1 3456

运行: ./fft1

```
mounting filesystems...
loading script...
Script from M5 readfile is empty, starting bash shell...
(none) / # ./fft1
```

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。建议使用实验一中的单线程 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
----	----	----

部件	配置	备注
CPU	core i5-5200U	
内存	DDR3 2GB	
操作系统	Ubuntu 16.04 LTS	中文版

测试记录

本地实验结果

```

1018 -2.112 +0.000 1018
1019 -1.820 +512.000 1019
1020 -1.497 +0.000 1020
1021 -1.147 +0.000 1021
1022 -0.776 +512.000 1022
1023 -0.392 +0.000 1023
time: 8.391000 ms

```

FS 下运行结果

```

1020 -1.497 +0.000 1020
1021 -1.147 +0.000 1021
1022 -0.776 +512.000 1022
1023 -0.392 +0.000 1023
time: 30.000000 ms
(none) / # █

```

思考解答

思考题 1:

gem5 是一种 CPU 模拟器

gem5 模拟器提供了四个不同的 CPU 模型，两个不同的系统模型以及两个不同的内存系统模型，并且支持多种指令集（ARM、ALPHA、MIPS、Power、SPARC 和 x86），其中可以再 ARM、ALPHA 和 x86 三种架构上运行 Linux。gem5 的许可证是基于 BSD 的，这就为工业界和学术界的合作搭建了一个好的桥梁。虽然开发一个 Full-system 的模拟器是很复杂的，但 gem5 正在借助开源的强大合作力不断完善自身功能。

思考题 2: 优缺点:

优点: 省钱省时间，节省资源

缺点: 制作难度大、仿真时间过长

思考题 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很大

可能的原因是在模拟器中编译源代码时，一条指令需要多条指令去解释

分析和结论

从测试记录来看，实际系统上测得的执行时间是 8.391ms，FFT 程序在 GEM5 上的执行时间是 30ms，与实际系统上测得的执行时间相比，FS 下的执行时间是实际系统的执行时间的 3.58 倍，在实际选择中，也许用模拟器来测试功能是否完善是比较可行的，测试性能的话，与实际情况相比，可能会存在比较大的误差，这是因为在模拟器中编译源代码时，一条指令需要多条指令去解释。