实验一 学习使用 gem5 模拟器

1. 实验目的

1. 熟悉 gem5 模拟器实验环境
2. 初步理解 gem5 进行硬件仿真和模拟执行程序的过程
3. 学习分析 gem5 的输出统计文件
4. 对 gem5 的模拟执行程序流程有一个整体的理解

2. 实验内容

2.1 阅读 Introduction

- gem5 是一个模块化的离散事件驱动的**计算机系统模拟平台**。
  - gem5 的组件可以重新排列、参数化、扩展或替换。它将时间的流逝模拟为一系列离散事件。它的预期用途是以各种方式模拟一个或多个计算机系统。它是一个模拟器平台，可以根据需要使用尽可能多的预制组件来构建自己的模拟系统。
- gem5 主要是用 C++ 和 Python 编写的。
- 它可以在全系统模式（FS 模式）下模拟具有设备和操作系统的完整系统，或者在系统调用仿真模式（SE 模式）下直接由模拟器提供系统服务的用户空间程序。
- 对于在 CPU 模型上执行 Alpha、ARM、MIPS、Power、SPARC、RISC-V 和 64 位 x86 二进制文件的支持程度各不相同，包括两个简单的**单周期模型**、**乱序模型**和**顺序流水线模型**。
- 内存系统可以由缓存和交叉开关或 Ruby 模拟器灵活构建，Ruby 模拟器提供了更加灵活的内存系统建模。

2.2 Building gem5

- 需要对每个要模拟的 ISA 单独编译 gem5。如果使用 ruby-intro-chapter，则必须为每个缓存一致性协议单独编译。
- SCons 使用 SConstruct 文件（`gem5/SConstruct`）设置多个变量，然后使用每个子目录中的 SConscript 文件来查找和编译所有的 gem5 源代码。
- gem5 中的 SCons 脚本目前有 5 个不同的二进制文件可供构建：debug、opt 和 fast
  - debug: 构建时没有优化和调试符号。
  - opt: 这个二进制文件是用大多数优化构建的（例如，-O3），但包含调试符号。此二进制文件比调试快得多，但仍包含足够的调试信息以能够调试大多数问题。
  - fast: 使用所有优化构建（包括支持平台上的链接时优化）并且没有调试符号。
- 编译指令为：

  ```
  scons build/X86/gem5.opt -j9
  CPU_MODELS=AtomicSimpleCPU,TimingSimpleCPU,O3CPU,MinorCPU
  ```

  使用 scons 工具编译 gem5 模拟器，使用 9 个线程（-j9）并指定 CPU 模型为 AtomicSimpleCPU、TimingSimpleCPU、O3CPU 和 MinorCPU，编译完成后生成可执行文件 build/X86/gem5.opt.

  - AtomicSimpleCPU：最简单的 CPU 模型之一，每个周期只执行一个指令，不支持乱序执行或者多级流水线。

    - TimingSimpleCPU：支持基本的流水线和指令乱序执行，适用于性能测试和简单的系统仿真。
    - O3CPU：是 gem5 中性能最好的 CPU 模型之一，支持乱序执行、分支预测、多级流水线等高级特性，适用于性能测试和复杂系统仿真。
    - MinorCPU：支持乱序执行、分支预测、多级流水线等高级特性，但相对于 O3CPU，MinorCPU 是一种更简单、更可扩展的 CPU 模型。它将 CPU 拆分为多个独立的阶段，可以方便地插入新的功能，适用于复杂系统仿真和研究。
- 编译完成：

```
scons: done building targets.
*** Summary of Warnings ***
Warning: Header file <png.h> not found.
         This host has no libpng library.
         Disabling support for PNG framebuffers.
Warning: Deprecated namespaces are not supported by this compiler.
         Please make sure to check the mailing list for deprecation
         announcements.
Warning: Couldn't find HDF5 C++ libraries. Disabling HDF5 support.
```

2.3 创建一个简单的配置脚本

gem5 二进制文件以一个 Python 脚本作为参数，用于设置和执行仿真。在此脚本中，用户需要创建要模拟的系统、创建系统的所有组件并指定所有组件的参数。

gem5 的模块化设计是围绕 SimObject 类型构建的。仿真系统中的大多数组件都是 SimObjects：CPU、高速缓存、内存控制器、总线等。gem5 将所有这些对象从其 C++ 实现导出到 Python。因此，从 Python 配置脚本中，用户可以创建任何 SimObject，设置其参数，并指定 SimObject 之间的交互。

使用 gem5 模拟器来创建一个简单的系统并运行一个"Hello World"程序。具体步骤如下：

1. 导入 m5 对象和 System 对象
2. 创建系统：创建一个系统对象并设置它的时钟频率和电压域，然后指定内存大小和内存模式。
3. 创建 CPU：创建一个 X86TimingSimpleCPU 对象作为 CPU，并创建一个内存总线（ SystemXBar 对象）以将 CPU 缓存端口连接到内存总线。
4. 创建内存控制器：创建一个 DDR3 内存控制器，并将其连接到内存总线。
5. 创建进程并设置 CPU：创建一个进程对象并将其命令设置为要运行的程序的路径。然后将 CPU 的工作负载设置为该进程，并在 CPU 中创建功能执行上下文。
6. 实例化系统并开始模拟：创建 Root 对象并实例化模拟器，然后开始执行模拟。实例化过程会遍历在 Python 中创建的所有 SimObject，并创建它们的 C++ 等效对象。在执行过程中，gem5 将模拟系统的行为并输出模拟的结果。

将该文件命名为 simple.py，放在 /gem5/labs/lab1 下，运行时**需要在 gem5 根目录下。**

运行结果：

```
xxa@ubuntu:~/Desktop/gem5$ sudo build/X86/gem5.opt labs/lab1/simple.py
gem5 Simulator System.  https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 22.1.0.0
gem5 compiled Mar 31 2023 20:53:57
gem5 started Apr  2 2023 02:57:49
gem5 executing on ubuntu, pid 7092
command line: build/X86/gem5.opt labs/lab1/simple.py

Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and p
df.
build/X86/mem/dram_interface.cc:690: warn: DRAM device capacity (8192 Mbytes) do
es not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation!
build/X86/sim/simulate.cc:192: info: Entering event queue @ 0.  Starting simulat
ion...
Hello world!
Exiting @ tick 454646000 because exiting with last active thread context
```

2.4 将 Cache 添加到配置脚本

在之间的配置脚本的基础山，添加缓存层级结构。

- gem5 目前有两个完全不同的子系统来模拟系统中的片上缓存，“经典缓存”和“Ruby”

在 caches.py 中：

1. 创建 L1 Cache 类，设置一些参数。创建两个函数，分别是将 Cache 连接到 CPU 和将 Cache 连接到 BUS
2. 创建 L1ICache 和 L1DCache，继承 L1 Cache，设置大小以及重写 connectCPU 函数
3. 创建 L2 Cache 类，设置一些参数。创建两个函数，分别是将 Cache 连接到 CPU 总线和将 Cache 连接到 Memory 总线

将创建的 Cache 添加到配置文件中。

新脚本名为 two_level.py，首先包括之前脚本的全部内容。并添加以下内容：

1. 创建 L1 Cache，使用辅助函数将缓存连接到 CPU 端口，删除原本的将缓存端口直接连接到内存总线的线
2. 创建一个 L2 总线来将我们的 L1 Cache 连接到 L2 Cache
3. 创建 L2 Cache 并将其连接到 L2 总线和内存总线

将 two_level.py 和 caches.py 均放在 /gem5/labs/lab1 下，运行时**需要在 gem5 根目录下**

运行结果：

```
xxa@ubuntu:~/Desktop/gem5$ sudo build/X86/gem5.opt labs/lab1/two_level.py
[sudo] password for xxa:
gem5 Simulator System.  https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 22.1.0.0
gem5 compiled Mar 31 2023 20:53:57
gem5 started Apr  2 2023 04:34:29
gem5 executing on ubuntu, pid 2811
command line: build/X86/gem5.opt labs/lab1/two_level.py

Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and p
df.
build/X86/mem/dram_interface.cc:690: warn: DRAM device capacity (8192 Mbytes) do
es not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation!
build/X86/sim/simulate.cc:192: info: Entering event queue @ 0.  Starting simulat
ion...
Hello world!
Exiting @ tick 56435000 because exiting with last active thread context
```

为脚本添加可选参数，实现在运行脚本时配置参数。

使用 `argparse` 模块解析命令行参数，包括一个位置参数 `binary` 和三个可选参数 `--l1i_size`、`--l1d_size` 和 `--l2_size`

在 binary 参数上，我设置 default 为 `tests/test-progs/hello/bin/x86/linux/hello`.

需要删除原本的 binary 一行，修改引用到 binary 的地方为引用 options.binary

需要为 L1 Cache 和 L2 Cache 添加构造函数，如果在命令行参数中制定了大小，则进行相应的配置，否则采用默认值。

运行结果：在 **gem5 根目录下**

`build/X86/gem5.opt labs/lab1/two_level.py -h`

```
xxa@ubuntu:~/Desktop/gem5$ sudo build/X86/gem5.opt labs/lab1/two_level.py -h
gem5 Simulator System.  https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 22.1.0.0
gem5 compiled Mar 31 2023 20:53:57
gem5 started Apr  2 2023 05:31:23
gem5 executing on ubuntu, pid 2811
command line: build/X86/gem5.opt labs/lab1/two_level.py -h

usage: two_level.py [-h] [--l1i_size L1I_SIZE] [--l1d_size L1D_SIZE]
                    [--l2_size L2_SIZE]
                    [binary]

A simple system with 2-level cache.

positional arguments:
  binary               Path to the binary to execute.

optional arguments:
  -h, --help           show this help message and exit
  --l1i_size L1I_SIZE  L1 instruction cache size. Default: 16kB.
  --l1d_size L1D_SIZE  L1 data cache size. Default: Default: 64kB.
  --l2_size L2_SIZE    L2 cache size. Default: 256kB.
```

`build/X86/gem5.opt labs/lab1/two_level.py --l2_size='1MB' --l1d_size='128kB'`

```
xxa@ubuntu:~/Desktop/gem5$ sudo build/X86/gem5.opt labs/lab1/two_level.py --l2_s
ize='1MB' --l1d_size='128kB'
gem5 Simulator System.  https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 22.1.0.0
gem5 compiled Mar 31 2023 20:53:57
gem5 started Apr  2 2023 05:38:24
gem5 executing on ubuntu, pid 2826
command line: build/X86/gem5.opt labs/lab1/two_level.py --l2_size=1MB --l1d_size
=128kB

Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and p
df.
build/X86/mem/dram_interface.cc:690: warn: DRAM device capacity (8192 Mbytes) do
es not match the address range assigned (512 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation!
build/X86/sim/simulate.cc:192: info: Entering event queue @ 0.  Starting simulat
ion...
Hello world!
Exiting @ tick 56435000 because exiting with last active thread context
```

2.5 阅读 Getting started 的后续章节

> 阅读 Getting started 的后续章节，学习和了解 gem5 输出的统计文件内容，如何使用 gem5 自带的配置文件进行模拟

除了模拟脚本打印出的任何信息外，在运行 gem5 之后，还会在名为 `m5out` 的目录中生成三个文件：

config.ini：包含为模拟创建的每个 SimObject 的列表及其参数值。

config.json：与 config.ini 相同，但采用 json 格式。

stats.txt：模拟注册的所有 gem5 统计信息的文本表示。每个统计数据都有一个名称（第一列）、一个值（第二列）和一个描述（最后一列以# 开头），后跟统计数据的单位。

gem5 附带的配置脚本（部分）：

boot/：这些是在全系统模式下使用的 rcS 文件。这些文件在 Linux 启动后由模拟器加载并由 shell 执行。

common/：该目录包含许多用于创建模拟系统的帮助脚本和函数。

dram/：包含用于测试 DRAM 的脚本。

example/：包含一些示例 gem5 配置脚本，可以开箱即用地运行 gem5

ruby/：包含 Ruby 的配置脚本及其包含的缓存一致性协议。

3. 参考

gem5: Learning gem5

4. 附录

- `simple.py`
- `caches.py`
- `two_level.py`
- `PB20061343_徐奥_lab1.pdf`