mp_design_space

University Of Illinois

Fall 2024

1 Introduction

GEM5 is a widely used architectural simulator that models many different microarchitectures. Architechural simulators allow for rapid design space exploration, by modelling key performance attributes of the hardware being simulated. They work at a higher level of abstraction than RTL models and are described as "Cycle Accurate", but unlike RTL, do not actually model all signals, gates and flops in a design.

They have a few key advantages,

- Faster simulation time: Usually, a GEM5 simulation will be 2-3 orders of magnitude faster than a corresponding RTL simulation
- Ability to model large systems: Simulators like GEM5 can model full systems, along with IO devices, which is not feasible in a RTL model.
- Ease of development: Working at the higher abstraction level, simulators like GEM5 are easier to extend, modify and verify, compared to writing full RTL models

Such simulators are often used in early stages of CPU design, to evaluate the performance of microarchitechtural parameters, like cache sizes, branch predictor types, branch predictor history table sizes, number of reservation stations etc.

2 Preparing Benchmarks

We will use a subset of the SPEC 2006 Benchmarks for this mp. Please download the benchmarks from here. We will first compile and run these benchmarks on the machine you are using. You can do this on EWS or a local linux environment (recommended). (Example: WSL on Windows or Multipass on MacOS).

We will use the benchmarks `429.mcf` and `433.milc` with test input sizes. We are using a subset due to simulation time limitations.

To compile and run these two benchmarks, first, untar the speccpu tarball

```
mkdir spec
tar -xf cpu2006.tar.bz2 -C spec
cd spec
./install.sh
```

Then in the config folder, make a new file *ece.cfg* and paste in the config from Listing 1. This part is for configuring the compilation flags etc for the benchmarks.

Then finally, from the `spec` dir, run the following command to build and run the two benchamrks with the test input sizes.

```
    runspec --config=ece.cfg --tune=base --size=test --noreportable 429.mcf 433.milc
```

Please note the build directory in the output, as you will need the path to the binaries and data files to run the simulation under gem5. The example paths for the mcf binary and data are listed below

```
    benchspec/CPU2006/429.mcf/run/build_base_amd64-m64-gcc42-nn.0000/mcf
    benchspec/CPU2006/429.mcf/data/test/input/inp.in
```

runspec前还需要：
1.配置环境变量：export PATH=$PATH:/home/px/Desktop/Linux_mp_share/GEM5/spec/bin
2.加载配置文件：source shrc

Listing 1: Compilation Config

```
iterations=1

ignore_errors = yes
tune          = base
ext           = amd64-m64-gcc42-nn
output_format = txt
reportable    = 1
teeout        = no
teerunout     = no
hw_avail = Dec-9999
license_num = 9999
test_sponsor = Turbo Computers
prepared_by =
tester      =
test_date = Dec-9999

default=default=default=default:
#############################################################################
#
# Compiler selection
#
#############################################################################
CC           = gcc
CXX          = g++
FC           = gfortran

sw_other = None
sw_auto_parallel = No
sw_base_ptrsize = 64-bit
sw_peak_ptrsize = Not Applicable


#############################################################################
# Optimization
#############################################################################
## Base is low opt
default=base=default=default:
COPTIMIZE     = -O2
CXXOPTIMIZE  = -O2
FOPTIMIZE    = -O2
```

3 Getting Started with gem5

3.1 Installation

Using the guide here, clone and build a copy of gem5.

3.2 Running your first Simulation

To start with, we will use a given configuration file to run a simple simulation. While the gem5 simulator is written in C++, it takes as input a configuration file written in Python. For this MP, you will only be creating a configuration file that varies cache parameters.

The given file can be found at `configs/learning_gem5/part1/two_level.py`.

To run the simulation,

```
build/X86/gem5.opt configs/learning_gem5/part1/two_level.py
```

After this, you should see the simultion output in a folder named `m5out`. `stats.txt` will contain the releavant outputs.

We will use this file as a starting point. Your task is to modify the given configuration to run the two SPEC benchamrks, with the configuration defined below. You should use the SimpleOpts.add_option to add more command line arguments to your script as needed.

You should modify the simulation script to only simulate a maximum of 10000000000000 ticks. You can do this by passing a argument to the `m5.simulate()` call.

4 Exploring Cache Configurations

We want to explore various memory hierarchies for our design. For this, we will use the `X86O3CPU` which is a detailed out of order CPU model.

We want to test the follwoing configurations for the L1D Cache.

Size	**Assoc**	**Response Latency**
32KB	4	2
64KB	4	3
64KB	8	5
128KB	8	8
128KB	16	12

5 Deliverables

Please submit a report that contains your modified configuration script, and list of commands to run your simulations. Your report should also contain a table that lists IPC for all 5 cache configurations, along with L1D cache Hit/Miss statistics for both benchmarks.