Project Title

Comparison of Cache Replacement Policies using GEM5 Simulator

***Team Members* - Bhagyashree Borate, Nivin Anton, Sridivya Basavaraju.**

***Team Number* - 1**

** This file contains the steps to follow to run and evaluate our implementation on the GEM5 simulator. Also, the description for each file and steps.

Gem5 Simulator steps –

1. **How to use EC2 instance to access GEM5 simulator for result analysis -**

- Download "putty" and download .pem file for gem5.

  putty.exe : http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

- Download "puttygen" for generating the key.

  puttygen.exe http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

1) **Open puttygen** -> Load -> browse .pem file -> save private key
2) **Open putty ->**

   Enter hostname =
   ubuntu@ec2-52-43-19-255.us-west-2.compute.amazonaws.com

   Enter session name in saved session

   In left tabs goto

   Connection -> Data - > Auto-login username = ubuntu

   Connection -> SSH -> Auth -> browse and enter .ppk file which you

generated through puttygen

Session -> press save button

Session -> Load.

2. Once you are i EC2 -
   cd gem5 to enter into GEM5.

3. **cd gem5/m5out**
   This folder gives config.ini, stats.txt files, which are output files.
   config.ini = Have the configuration like clock rate, cache size, cache algorithms etc.
   stats.txt = This file have parameter results like miss rate, overall access, CPU cycles etc.

4. **Three-level-cache.py :**

   This file contains the python code to run 3 level caches simulation.
   Path - cd gem5/configs/learning-gem5/part1/three-level-cache.py

5. **cache_level.py :**

   This file contains the main parameters like associativity, size etc. Here you can change associativity with assoc=8, change the cache size with size="256kB" etc.

   This file is referenced by three_level_cache.py in order to check how many caches and what kind of parameters are used for the given program.

   Path – cd gem5/configs/learning-gem5/part1/cache_level.py

6. Different algorithms at different levels :

   In **cd gem5/configs/learning-gem5/part1/cache_level.py**
   In each cache level you can change algorithm as below –

```
class L1Cache(Cache):
        tags: LRU ()
class L12Cache(Cache):
        tags: RandomRepl()
class L3Cache(Cache):
        tags: Fifo()
```

   This Fifo, LRU, RandomRepl are the main functions used from main c++ file.

7. **Cache.py :**

   This is a main file where you will declare which "Algorithm" to use at all cache levels to evaluate the performance.

   tags = Param.BaseTags(**LRU(),** "Tag store (replacement policy)")

In order to run with different algorithm change the LRU() tag with =

a. Fifo() – for Fifo.
b. RandomRepl() – for random replacement
c. LRU () – for least recently used

Path – cd gem5/build/X86/mem/cache/Cache.py

8. **Adding new algorithm files to gem5 –**

In order to add new algorithm, you should create two files –
file_name.hh and file_name.cc
where, .hh is header file & .cc is main c++ file
You can refer existing file in order to write the new algorithm.

Both files can be found and placed at below path:
Path: cd gem5/build/X86/mem/cache/tags/<file-name>.hh
cd gem5/build/X86/mem/cache/tags/<file-name>.cc

9. **Once the algorithm files .cc and .hh are done, include them in the main base.cc file -**

In base.cc file add below line –
#include "mem/cache/tags/<algorithm_file_name>.hh"

#include "mem/cache/tags/lru.hh"
#include "mem/cache/tags/fifo.hh"
#include "mem/cache/tags/random_repl.hh"

Path: cd gem5/build/X86/mem/cache/base.cc

10. **Running Benchmarks –**

We have 4 different workload benchmarks to run and evaluate cache performance.
You can learn about this files in detail at -
https://www.cs.virginia.edu/~cr4bd/6354/F2016/homework2.html

a. **Queens –**

Basic Run -./queens -c 10

How to Run in GEM5 –

- Goto **cd gem5/configs/learning-gem5/part1/three-level-cache.py**
- In this file there is a parameter / line –

binary = 'configs/learning_gem5/part1/benchmarks/queens

So here you have to give a path to your benchmark workload you are going to use. As in this example queens has been used as a benchmark. Queens is a .out file generated after running the c++ program.

In the same file cd gem5/configs/learning-gem5/part1/three-level-cache.py
Update the parameter / line =

process.cmd = [binary,'-c', 10]
Here process.cmd works as passing the arguments to the queens.out program which we mentioned above.

b. SHA –

Basic Run -./sha path/to/example-sha-input.txt

How to Run in GEM5 –

- Goto **cd gem5/configs/learning-gem5/part1/three-level-cache.py**
- In this file there is a parameter / line –

  binary = 'configs/learning_gem5/part1/benchmarks/sha'

  So here you have to give a path to your benchmark workload you are going to use. As in this example **sha** has been used as a benchmark. sha is a .out file generated after running the c++ program.

  In the same file cd gem5/configs/learning-gem5/part1/three-level-cache.py
  Update the parameter / line =

  process.cmd = [binary,'configs/learning_gem5/benchmarks/inputs/example-sha-input.txt']
  Here process.cmd works as passing the arguments to the sha.out program which we mentioned above.

c. BFS –

Basic Run -./bfs path/to/RL3k.graph

How to Run in GEM5 –

- Goto **cd gem5/configs/learning-gem5/part1/three-level-cache.py**
- In this file there is a parameter / line –

binary = 'configs/learning_gem5/part1/benchmarks/bfs'

So here you have to give a path to your benchmark workload you are going to use. As in this example **bfs** has been used as a benchmark. bfs is a .out file generated after running the c++ program.

In the same file cd gem5/configs/learning-gem5/part1/three-level-cache.py
Update the parameter / line =

process.cmd = [binary,'configs/learning_gem5/benchmarks/inputs/RL3K.graph']
Here process.cmd works as passing the arguments to the bfs.out program which we mentioned above.

d. Matmul –

Basic Run -./blocked-matmul

How to Run in GEM5 –

- Goto **cd gem5/configs/learning-gem5/part1/three-level-cache.py**
- In this file there is a parameter / line –

  binary = 'configs/learning_gem5/part1/benchmarks/blocked-matmul'

  So here you have to give a path to your benchmark workload you are going to use. As in this example **bfs** has been used as a benchmark. bfs is a .out file generated after running the c++ program.

  In the same file cd gem5/configs/learning-gem5/part1/three-level-cache.py
  Update the parameter / line =

  process.cmd = [binary]
  Here process.cmd works as passing the arguments to the bfs.out program which we mentioned above.

11. Now, go back to cd gem5 directory

First build the gem5 again in order to make the changes to effect –

scons build/X86/gem5.opt -j2

where, scons - is a build command
build/X86/gem5.opt – is build file
-j2 – number of processors to run the build.

12. Run the simulation with main program
- cd gem5/

- build/X86/gem5.opt configs/learning-gem5/part1/three-level-cache.py

Once this is done the output files will be generated in the cd gem5/m5out.