THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

Project 2 – Cache

CS6304 Computer Architecture

Chih-An Chang , William Chang,
CXC210017, CXC200006
Department of Computer Science

Outline

1. Cache Introduction
2. Gem5 Setup
3. CPI Result
4. Optimize CPI Result
5. Define Cost Function

Cache Introduction

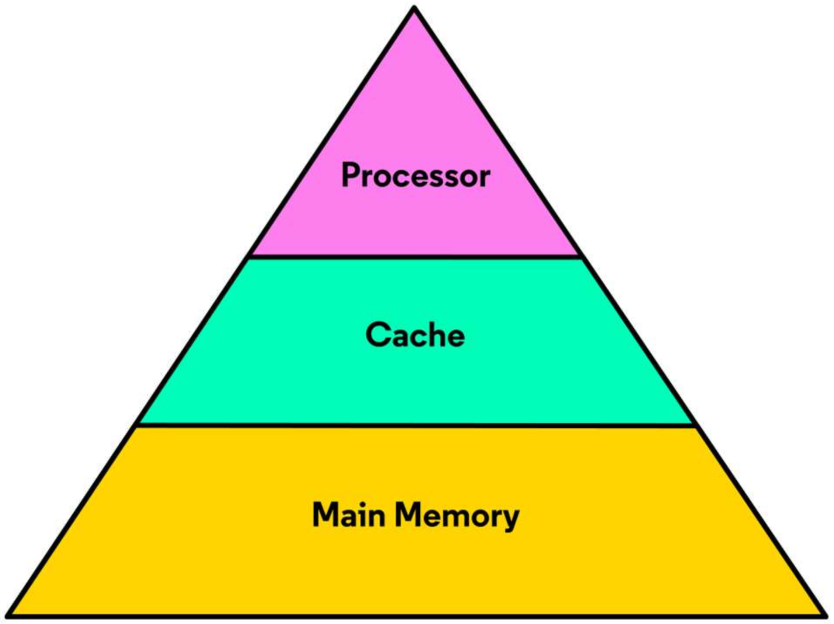
Overview

What is Cache?

Why it is important?

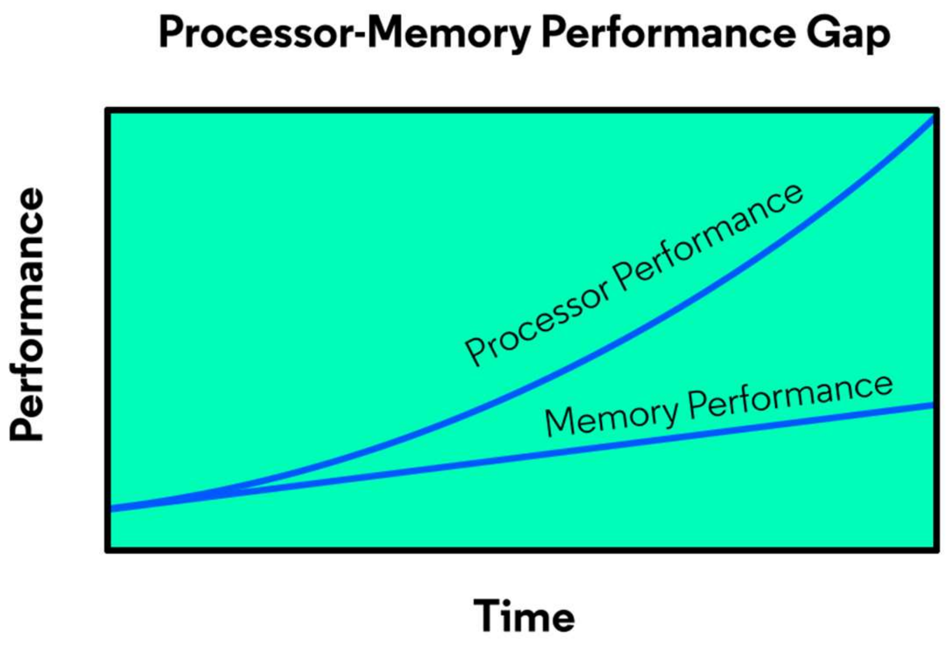
What is Cache?

- Cache is memory placed in between the processor and main memory.
- Cache is responsible for holding copies of main memory data for faster retrieval by the processor.



Why it so important?

- The processor-memory performance gap and results in a computer that can process data much faster than it can retrieve data from memory.
- Cache memory stores data for faster access times to help bridge the processor-memory performance gap.



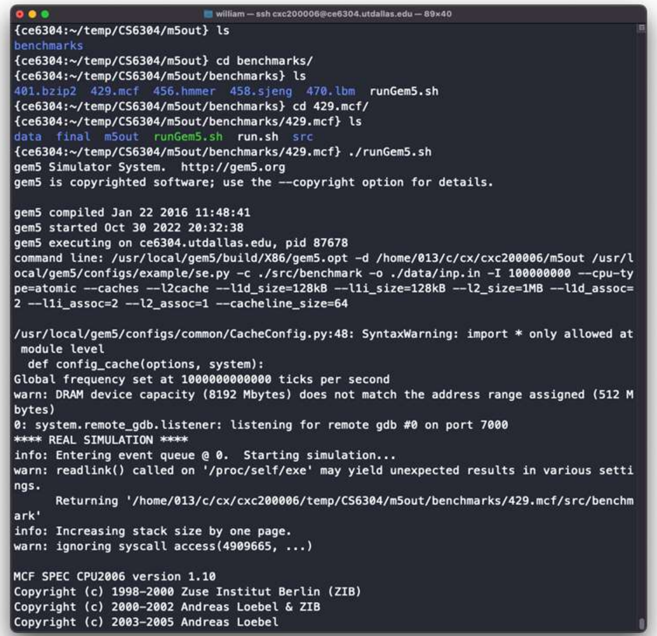
Gem5 Cache

Overview

Setup

Configuration

Gem5 Setup



william — ssh cxc200006@ce6304.utdallas.edu — 89×40

```
{ce6304:~/temp/CS6304/m5out} ls
benchmarks
{ce6304:~/temp/CS6304/m5out} cd benchmarks/
{ce6304:~/temp/CS6304/m5out/benchmarks} ls
401.bzip2  429.mcf  456.hmmer  458.sjeng  470.lbm  runGem5.sh
{ce6304:~/temp/CS6304/m5out/benchmarks} cd 429.mcf/
{ce6304:~/temp/CS6304/m5out/benchmarks/429.mcf} ls
data  final  m5out  runGem5.sh  run.sh  src
{ce6304:~/temp/CS6304/m5out/benchmarks/429.mcf} ./runGem5.sh
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 compiled Jan 22 2016 11:48:41
gem5 started Oct 30 2022 20:32:38
gem5 executing on ce6304.utdallas.edu, pid 87678
command line: /usr/local/gem5/build/X86/gem5.opt -d /home/013/c/cx/cxc200006/m5out /usr/l
ocal/gem5/configs/example/se.py -c ./src/benchmark -o ./data/inp.in -I 100000000 --cpu-ty
pe=atomic --caches --l2cache --l1d_size=128kB --l1i_size=128kB --l2_size=1MB --l1d_assoc=
2 --l1i_assoc=2 --l2_assoc=1 --cacheline_size=64

/usr/local/gem5/configs/common/CacheConfig.py:48: SyntaxWarning: import * only allowed at
 module level
  def config_cache(options, system):
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 M
bytes)
0: system.remote_gdb.listener: listening for remote gdb #0 on port 7000
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
warn: readlink() called on '/proc/self/exe' may yield unexpected results in various setti
ngs.
      Returning '/home/013/c/cx/cxc200006/temp/CS6304/m5out/benchmarks/429.mcf/src/benchm
ark'
info: Increasing stack size by one page.
warn: ignoring syscall access(4909665, ...)

MCF SPEC CPU2006 version 1.10
Copyright (c) 1998-2000 Zuse Institut Berlin (ZIB)
Copyright (c) 2000-2002 Andreas Loebel & ZIB
Copyright (c) 2003-2005 Andreas Loebel
```

1. **Login to CE6304 linux server**

   `ssh <net-id>@ce6304.utdallas.edu`

2. **Copy gem5 folder to your local directory**

   `cp –rf /usr/local/gem5 /home/<netid>/<proj_folder>`

3. **Downloads the Benchmarks**

   `git clone https://github.com/timberjack/Project1_SPEC`

4. **Building Gem5 X86 with Scons**

   `scons build/X86/gem5.opt –j5`

5. **Give permission to run the script**

   `chmod +x ./runGem5.sh`

Cache Configuration

CPU Models	**TimingSimpleCPU**
Cache levels	**2**
L1 Cache Category	**Separate**
L2 Cache Category	**Unified**
L1 Data Cache Size	**128 KB**
L1 Instruction Cache Size	**128 KB**
Unified L2 Cache Size	**1 MB**
L1 Data Cache Associativity	**2-Way**
L1 Instruction Cache Associativity	**2-Way**
Unified L2 cache	**Direct-mapped**
Block size	**64 bytes**
Block replacement policy	**Least Recent Used (LRU)**

Command Line Parameters

```
$GEM5_DIR/build/X86/gem5.opt -d ./m5out
$GEM5_DIR/configs/example/se.py -c $BENCHMARK -o $ARGUMENT -I 500000000

## CPU types
--cpu-type=timing

## Cache Level setting
--caches --l2cache

## Setting Cache size
--l1d_size=128kB
--l1i_size=128kB
--l2_size=1MB

## Setting Associativity
--l1d_assoc=2
--l1i_assoc=2
--l2_assoc=1

## Setting cache block size
--cacheline_size=64
```

CPI Result

Overview

429.mcf

401.bzip2

CPI Calculation

- Using “system.cpu.icache.overall_miss_rate::total” × total number of instruction (500,000,000) to get the total miss count is incorrect. We realized that the total number of instructions for D-Cache and L2 Cache can not be 500,000,000.
- Find “system.cpu.icache.overall_misses::total” in “stats.txt”
- Substitute the parameters into the formula below

$$\mathrm{CPI} = 1 + \frac{\left(\mathrm{IL1}_{\mathrm{num}_{\mathrm{miss}}} + \mathrm{DL1}_{\mathrm{num}_{\mathrm{miss}}}\right) \times 6 + \mathrm{L2num}_{\mathrm{miss}} \times 50}{\text{Total num instruction}}$$

CPI for 429.mcf

$$\frac{\left[\underset{676}{\text{L1 I-cache Number of miss}} + \underset{10225985}{\text{L1 D-cache Number of miss}}\right] \times \underset{6}{\text{L1 miss penalty}} + \underset{6552122}{\text{L2 cache Number of miss}} \times \underset{50}{\text{L2 miss penalty}}}{\underset{500{,}000{,}000}{\text{Total number of Inst}}} + 1 = \underset{1.777932132}{\text{CPI}}$$

CPI for 401.bzip2

$$\frac{\left[\begin{array}{c}\text{L1 I-cache}\\ \text{Number of miss}\\ \mathbf{387}\end{array} + \begin{array}{c}\text{L1 D-cache}\\ \text{Number of miss}\\ \mathbf{4{,}027{,}321}\end{array}\right] \times \begin{array}{c}\text{L1 miss penalty}\\ \mathbf{6}\end{array} + \begin{array}{c}\text{L2 cache Number of miss}\\ \mathbf{2{,}743{,}240}\end{array} \times \begin{array}{c}\text{L2 miss penalty}\\ \mathbf{50}\end{array}}{\begin{array}{c}\text{Total number of Inst}\\ \mathbf{500{,}000{,}000}\end{array}} + 1 = \begin{array}{c}\text{CPI}\\ \mathbf{1.3226633}\end{array}$$

Optimize CPI Result

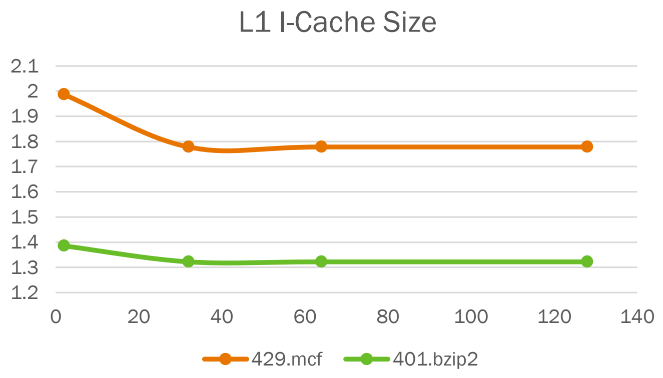
Overview

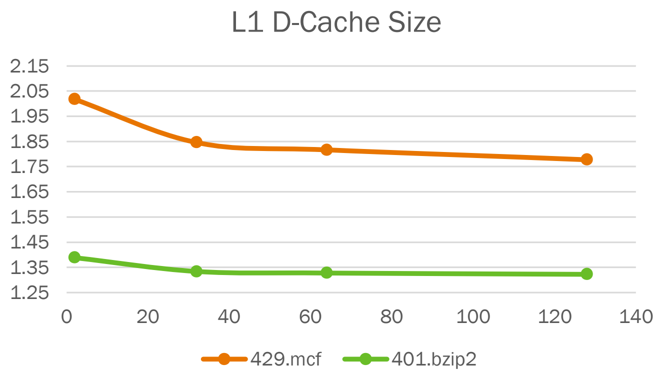
How Do We Get Lowest Configuration?

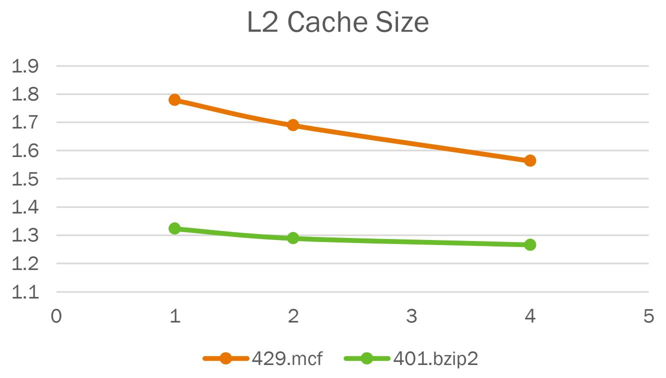
Lowest CPI Configuration

How Do We Get Lowest CPI Configuration?

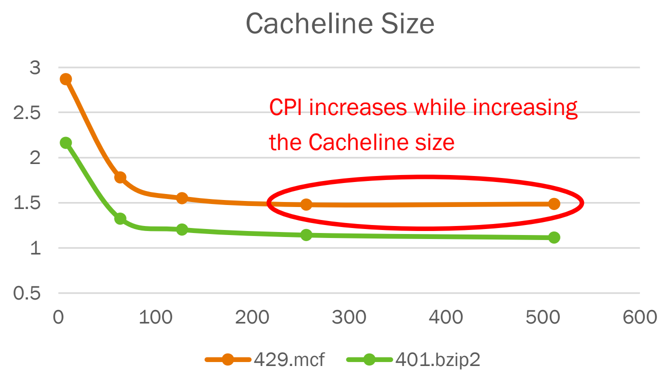
- We changed each parameter one at a time, and select the lowest CPI setting
- Except for Cacheline size and associativity, the rest of the parameters match the trend that the bigger the size the lower the CPI.

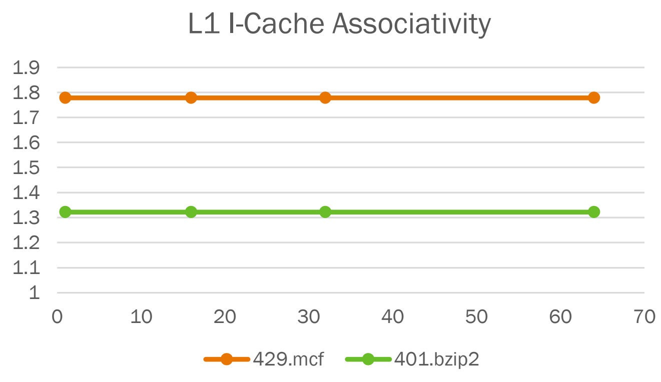


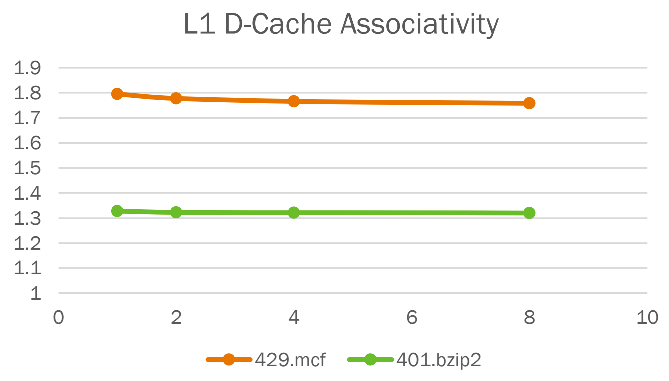


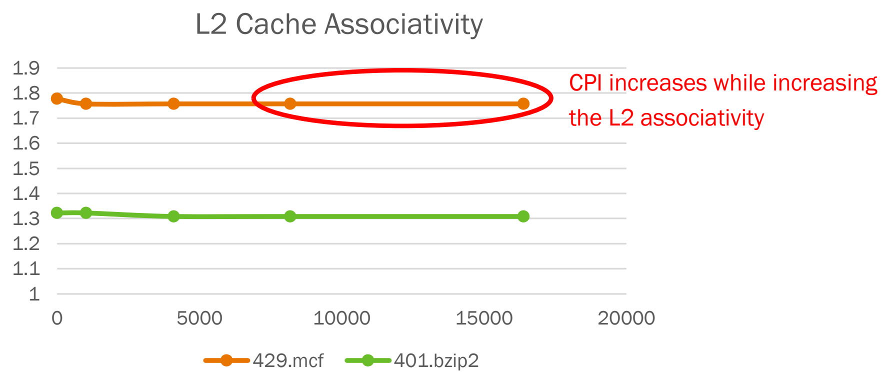


How Do We Get Lowest CPI Configuration? – cont.









Lowest CPI Configuration

	L1 I-Cache Size	L1 D-Cache Size	L2 Cache	Cacheline Size	L1 I-Cache Associativity	L1 D-Cache Associativity	L2 Cache Associativity	CPI	Cost
401.bzip2	128 KB	128 KB	4 MB	512 KB	16	8	4096	1.07636	260.8
429.mcf	128 KB	128 KB	4 MB	256 KB	16	8	4096	1.25182	260.8

Define Cost Function

Cost Function

Unit price

L1 unit price per KB	L2 unit price per MB	Associativity price per way
$0.05	$0.2	$0.06

Total Cost = **L1-Cache Size (KB) × L1 unit price** + **L2-Cache Size (MB) × L2 unit price** + **Total number of associativity × Associativity price**

Smallest Cost Configuration

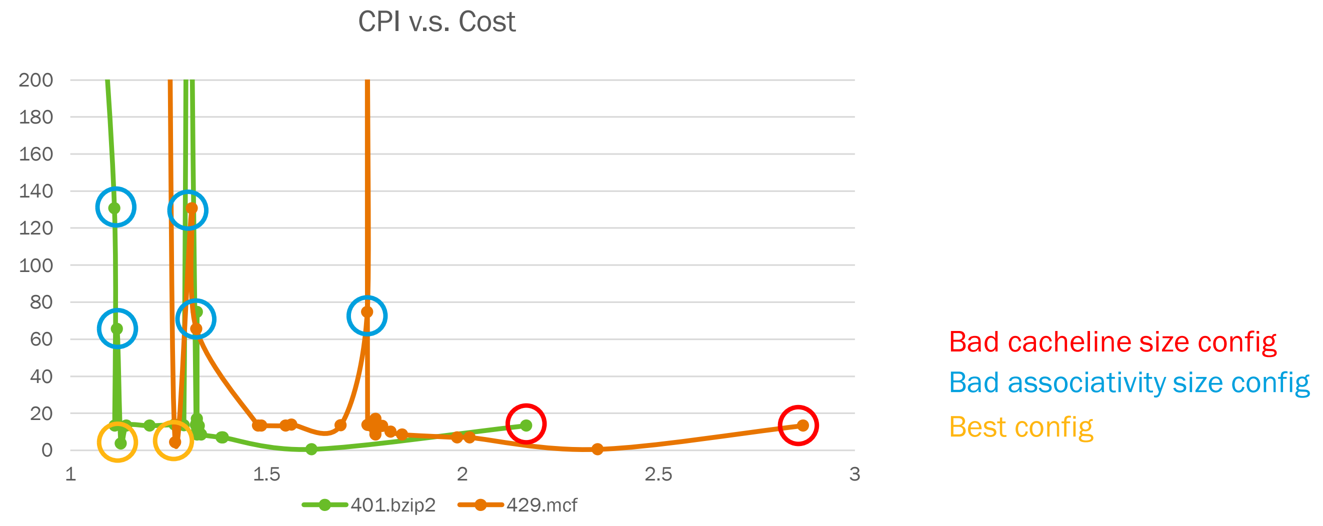
	L1 I-Cache Size	L1 D-Cache Size	L2 Cache	Cacheline Size	L1 I-Cache Associativity	L1 D-Cache Associativity	L2 Cache Associativity	CPI	Cost
401.bzip2	2 KB	2 KB	1 MB	512 KB	1	1	1	1.61455	0.58
429.mcf	2 KB	2 KB	1 MB	512 KB	1	1	1	2.34459	0.58

Best Cost-Performance Ratio Configuration

Overview

CPI vs. Price

- Based on the plotted chart, we can choose the best cost performance ratio configuration



Analysis and Conclusion

- Associativity does not affect much for both benchmarks. In 429.mcf, an extremely high L2 associativity might even increase overall miss count.
- Choosing an appropriate Cacheline Size is important. It can reduce the compulsory misses, but if Cacheline Size is set too big, it starts increasing the overall miss count.
- For both L1 and L2 cache size, the rule is the bigger the better. However, increasing the size of L1 I-Cache is not wise because instruction misses in both benchmarks are low even when the L1 I-Cache size is small, and the cost of L1 cache is high.
- Besides anomalies like extremely high associativity and inappropriate cache line size, investing more cost will give better performance.

Best Cost-Performance Ratio Configuration

	L1 I-Cache Size	L1 D-Cache Size	L2 Cache	Cacheline Size	L1 I-Cache Associativity	L1 D-Cache Associativity	L2 Cache Associativity	CPI	Cost
401.bzip2	32 KB	32 KB	1 MB	512 KB	2	2	1	1.12874	3.7
429.mcf	32 KB	32 KB	4 MB	256 KB	2	2	1	1.26722	4.3

Thank you

THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science