CS510 Computer Architecture

Lecture 15: Improving Cache Performance

Soontae Kim

Spring 2019

School of Computing, KAIST

Term project

- **Recommended topic list is posted.**
- **Brief explanation of popular simulators is posted.**
- **Send me your team member names by tomorrow so that I can arrange your proposal presentation appropriately.**
- **If your team has more members, I will expect more efforts from your team.**
- **You have to spend some time to learn your experimental tools and environment in order to avoid potential problems and difficulties in performing your projects.**

Improving Cache Performance

1. Reduce the miss rate,

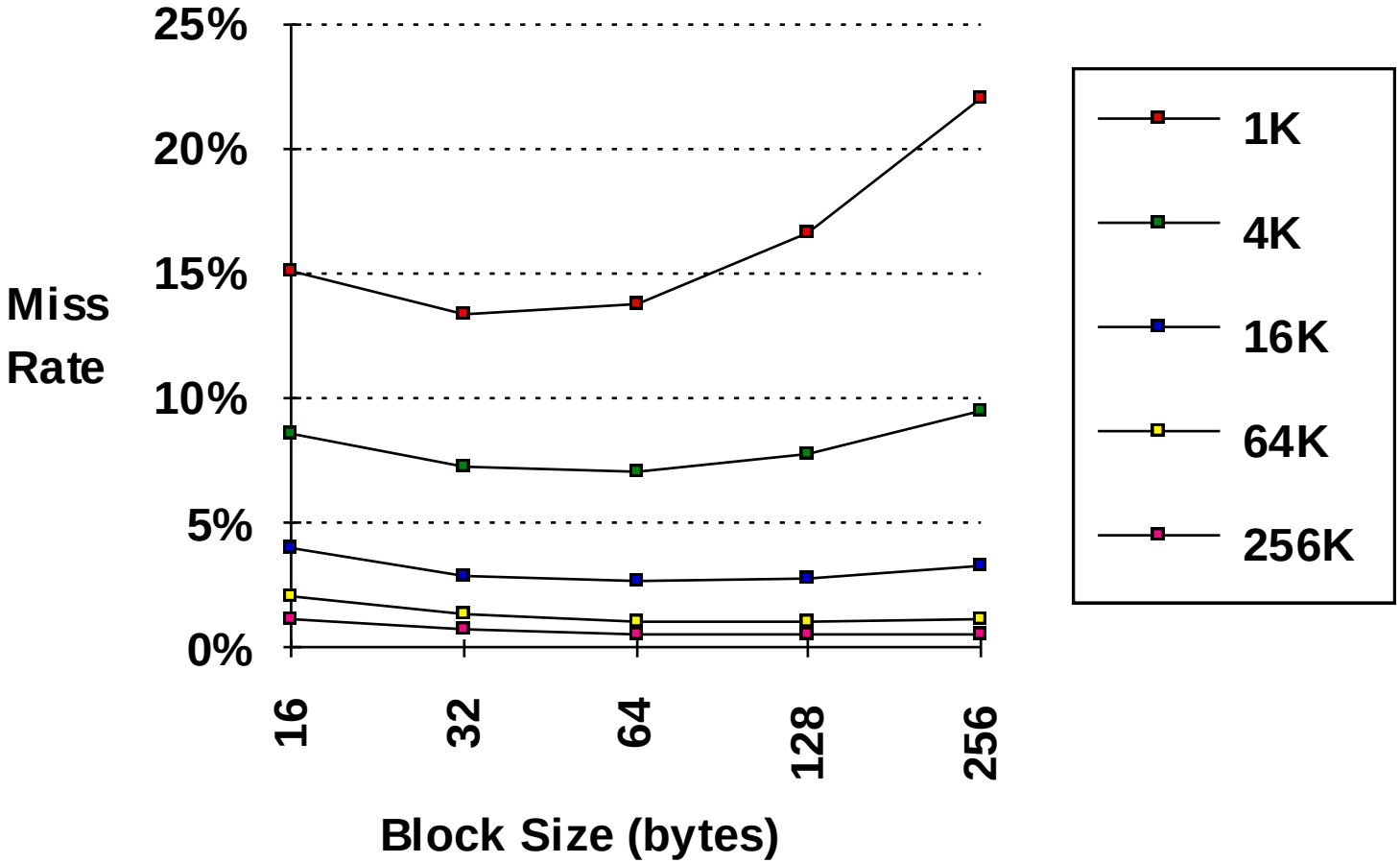
2. Reduce the miss penalty, or

3. Reduce the time to hit in the cache.

$$AMAT = HitTime + MissRate \times MissPenalty$$

1. Reduce Misses via Larger Block Size

Larger blocks increase miss penalty



2. Reduce Misses via Higher Associativity

- **2:1 Cache Rule:**
  - Miss Rate DM cache size N = Miss Rate 2-way cache size N/2
- **Beware: Execution time is only final measure!**
  - Will Clock Cycle time increase?

Example: Avg. Memory Access Time vs. Miss Rate

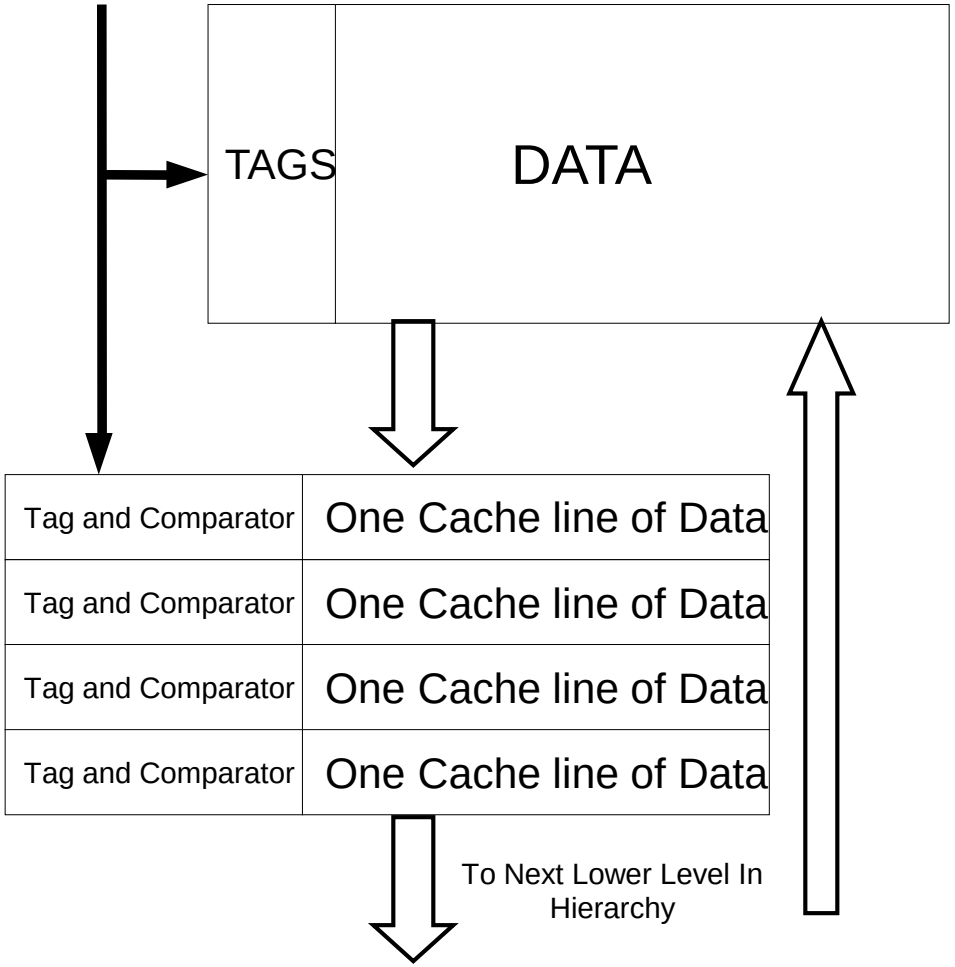
- **Example: assume CCT = 1.10 for 2-way, 1.12 for 4-way, 1.14 for 8-way vs. CCT= 1 for direct mapped**

Cache Size (KB)	Associativity 1-way	2-way	4-way	8-way
1	2.33	2.15	2.07	2.01
2	1.98	1.86	1.76	1.68
4	1.72	1.67	1.61	1.53
8	1.46	1.48	1.47	1.43
16	1.29	1.32	1.32	1.32
32	1.20	1.24	1.25	1.27
64	1.14	1.20	1.21	1.23
128	1.10	1.17	1.18	1.20

(Red means A.M.A.T. not improved by higher associativity)

3. Reducing Misses or Miss Penalty via a Victim Cache

- **How to combine fast hit time of DM and low miss rate of SA?**
- **Add buffer to place data discarded from cache**
- **Jouppi [1990]: 4-entry victim cache removed 20% to 95% of conflicts for a 4 KB direct mapped data cache**
- **Used in Alpha, HP machines**



4. Reducing Misses via Way Prediction

- **How to combine fast hit time of Direct Mapped and have the lower conflict misses of 2-way SA cache?**
- **Access a predicted way. On a miss, check other half of cache. If it hits there, we have a <u>pseudo-hit</u> (slow hit)**

- Used in the MIPS R10000

5. Reducing Misses by Hardware Prefetching of Instructions & Data

- **Instruction Prefetching**
  - Alpha 21064 fetches 2 consecutive blocks on a miss
  - Extra block placed in "stream buffer"
  - On a miss, check stream buffer
- **Works with data blocks too:**
  - Jouppi [1990] 1 data stream buffer got 25% misses from 4KB DM cache; 4 streams got 43%
  - Palacharla & Kessler [1994] for scientific programs, 8 streams got 50% to 70% of misses from 2 64KB, 4-way set associative caches
  - UltraSPARC III : stride-based prefetching
- **Prefetching relies on having extra memory bandwidth that can be used without penalty**

6. Reducing Misses by Software Prefetching of Data

- **Compiler inserts prefetch instructions**
- **Prefetching comes in two flavors:**
  - Binding prefetch: load data directly into register.
    - **Must be correct address and register!**
  - Non-Binding prefetch: Load into cache.
    - **Can be incorrect.**
  - Special prefetching instructions cannot cause faults
- **Issuing Prefetch Instructions takes time**
  - Is cost of prefetch issues < savings in reduced misses?

7. Reducing Misses by Compiler Optimizations

- **McFarling [1989] reduced cache misses by 75% on 8KB direct mapped cache, 4 byte blocks in software**
- **Instructions**
  - Reorder procedures in memory so as to reduce conflict misses
  - Profiling to look at conflicts(using tools they developed)
- **Data**
  - *Merging Arrays*: improve spatial locality by single array of compound elements vs. 2 arrays
  - *Loop Interchange*: change nesting of loops to access data in order stored in memory
  - *Loop Fusion*: Combine 2 independent loops that have same looping and some variables overlap
  - *Blocking*: Improve temporal locality by accessing “blocks” of data repeatedly vs. going down whole columns or rows

Merging Arrays Example

```
/* Before: 2 sequential arrays */
int val[SIZE];
int key[SIZE];

/* After: 1 array of stuctures */
struct merge {
  int val;
  int key;
};
struct merge merged_array[SIZE];
```

Reducing conflicts between val & key; improve spatial locality

Loop Interchange Example

```
/* Before */
for (k = 0; k < 100; k = k+1)
  for (j = 0; j < 100; j = j+1)
      for (i = 0; i < 5000; i = i+1)
            x[i][j] = 2 * x[i][j];
/* After */
for (k = 0; k < 100; k = k+1)
  for (i = 0; i < 5000; i = i+1)
      for (j = 0; j < 100; j = j+1)
            x[i][j] = 2 * x[i][j];
```

Sequential accesses instead of striding through memory every 100 words; improve spatial locality

Loop Fusion Example

```
/* Before */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
      a[i][j] = 1/b[i][j] * c[i][j];
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
      d[i][j] = a[i][j] + c[i][j];
/* After */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
  {   a[i][j] = 1/b[i][j] * c[i][j];
      d[i][j] = a[i][j] + c[i][j];}
```

2 misses per access to `a` & `c` vs. one miss per access; improve temporal locality

Summary: Miss Rate Reduction

$$CPUtime = IC \times \left(CPI_{Execution} + \frac{Memory\ \ accesses}{Instruction} \times \textbf{Miss rate} \times Miss\ \ penalty \right) \times Clock\ \ cycle\ \ time$$

- **3 Cs: Compulsory, Capacity, Conflict**
  1. Reduce Misses via Larger Block Size
  2. Reduce Misses via Higher Associativity
  3. Reducing Misses via Victim Cache
  4. Reducing Misses via Pseudo-Associativity
  5. Reducing Misses by HW Prefetching Instr, Data
  6. Reducing Misses by SW Prefetching Data
  7. Reducing Misses by Compiler Optimizations