15-213 Recitation 6: 10/14/02

Outline

• Optimization
  – Amdahl’s Law
• Cache
  – Performance Metrics
  – Access Patterns

Reminder

• L4 due: Thursday 10/24, 11:59pm
Amdahl’s Law

Old program

\[ T_1 \quad T_2 \]

Old time: \( T = T_1 + T_2 \)

New program (improved)

\[ T_1' = T_1 \quad T_2' \leq T_2 \]

New time: \( T' = T_1' + T_2' \)

Speedup = \( T / T' \)

- Amdahl’s Law describes a general principle for improving any process, not only for speeding up computer systems.

\( T_1 = \) time that can NOT be improved.

\( T_2 = \) time that can be improved.

\( T_2' = \) time after the improvement.
Amdahl’s Law: Example

- Planning a trip PGH -> NY -> Paris -> Metz
- Suppose both PGH -> NY and Paris -> Metz take 4 hours
- For NY -> Paris take 8.5 hours by a Boeing 747
- Total travel time: What if we choose faster methods?

<table>
<thead>
<tr>
<th></th>
<th>NY- &gt;Paris</th>
<th>Total time</th>
<th>Speedup over 747</th>
</tr>
</thead>
<tbody>
<tr>
<td>747</td>
<td>8.5 hours</td>
<td>16.5 hours</td>
<td>1</td>
</tr>
<tr>
<td>SST</td>
<td>3.75 hours</td>
<td>11.75 hours</td>
<td>1.4</td>
</tr>
<tr>
<td>rocket</td>
<td>0.25 hours</td>
<td>8.25 hours</td>
<td>2.0</td>
</tr>
<tr>
<td>rip</td>
<td>0.0 hours</td>
<td>8.0 hours</td>
<td>2.1</td>
</tr>
</tbody>
</table>

- It’s hard to gain significant improvement.
- Larger speedup comes from improving larger fraction of the whole system.
Cache Performance Metrics

- **Miss Rate**
  - Fraction of memory references not found in cache (misses/references)

- **Hit Time**
  - Time to deliver a line in the cache to the processor (including determining time)

- **Miss Penalty**
  - Additional time required because of a miss
Locality

- **Temporal locality:**
  - a memory location that is referenced once is likely to be *reference again multiple times* in the near future

- **Spatial locality:**
  - if a memory location is referenced once, then the program is likely to *reference a nearby memory location* in the near future
Practice Problem 6.4

• Permute the loops so that it scans the 3-dimensional array $a$ with a stride-1 reference pattern:

```c
int summary3d(int a[N][N][N])
{
    int i, j, k, sum = 0;
    for (i = 0; i < N; i++) {
        for (j = 0; k < N; j++ ) {
            for (k = 0; k < N; k++ ) {
                sum += a[k][i][j];
            }
        }
    }
    return sum;
}
```
Array Organization in Memory

\[ a[0][0][0], a[0][0][1], \ldots, a[0][0][N-1], \]
\[ a[0][1][0], a[0][1][1], \ldots, a[0][1][N-1], \]
\[ a[0][2][0], a[0][2][1], \ldots, a[0][2][N], \]
\[ \ldots \]
\[ a[1][0][0], a[1][0][1], \ldots, a[1][0][N], \]
\[ \ldots \]
\[ a[N-1][N-1][0], a[N-1][N-1][1], \ldots, a[N-1][N-1][N-1] \]
Solution

```c
int summary3d(int a[N][N][N])
{
    int i, j, k, sum = 0;
    for (k = 0; k < N; k++) {
        for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++) {
                sum += a[k][i][j];
            }
        }
    }
    return sum;
}
```
Cache Organization (review)

Cache is an array of sets.
Each set contains one or more lines.
Each line holds a block of data.

<table>
<thead>
<tr>
<th>set 0:</th>
<th>set 1:</th>
<th>set S-1:</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid tag 0 1 ... B-1</td>
<td>valid tag 0 1 ... B-1</td>
<td>valid tag 0 1 ... B-1</td>
</tr>
</tbody>
</table>

1 valid bit per line, t tag bits per line, $B = 2^b$ bytes per cache block

$S = 2^s$ sets

$E$ lines per set
Cache Access Patterns

• Now it’s your turn to spend 15 minutes working on Practice Problems 6.15-6.17 😊
• Handout is a photocopy from the text book

• Note that:
  - The size of struct `algae_position` is 8 bytes
  - Each cache block (16 bytes) holds two algae_position structs
  - The 16×16 array requires 2048 bytes of memory
  - Twice the size of the 1024 byte cache
Practice Problem 6.15-17

- Each row: 16 struct items, 8 cache blocks, 128 bytes
- Each column: 16 struct items
- Yellow area: 1024 bytes, green area: 1024 bytes
6.15 - Row Major Access Pattern
6.15 - Stride of two words

- First loop, accessing all x’s
- When a cache miss happens, load a block from memory

<table>
<thead>
<tr>
<th>miss</th>
<th>hit</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="table.png" alt="Table" /></td>
<td></td>
</tr>
</tbody>
</table>
6.15 - Stride of two words

- First loop, accessing all x’s
- When a cache miss happens, load a block from memory
### 6.15 - Stride of two words

- Second loop, accessing all y’s
- Same missing pattern, the green area flushes blocks from the yellow area

<table>
<thead>
<tr>
<th>x y x y</th>
<th>x y x y</th>
<th>x y x y</th>
<th>x y</th>
<th>x y x y</th>
</tr>
</thead>
<tbody>
<tr>
<td>x y x y</td>
<td>x y x y</td>
<td>x y x y</td>
<td>x y</td>
<td>x y x y</td>
</tr>
<tr>
<td>x y x y</td>
<td>x y x y</td>
<td>x y x y</td>
<td>x y</td>
<td>x y x y</td>
</tr>
<tr>
<td>x y x y</td>
<td>x y x y</td>
<td>x y x y</td>
<td>x y</td>
<td>x y x y</td>
</tr>
<tr>
<td>x y x y</td>
<td>x y x y</td>
<td>x y x y</td>
<td>x y</td>
<td>x y x y</td>
</tr>
<tr>
<td>x y x y</td>
<td>x y x y</td>
<td>x y x y</td>
<td>x y</td>
<td>x y x y</td>
</tr>
<tr>
<td>x y x y</td>
<td>x y x y</td>
<td>x y x y</td>
<td>x y</td>
<td>x y x y</td>
</tr>
</tbody>
</table>

*Green area flushes blocks from the yellow area.*
6.15 - Stride of two words

- Second loop, accessing all y’s
- Same missing pattern, the green area flushes blocks from the yellow area
Answer to Problem 6.15

• A: 512
  - 16x16 = 256 array elements in total
  - twice for each element

• B: 256
  - every other array element experiences a miss

• C: 50%
### 6.16 - Column Major Access Pattern

- New access removes first cache line contents before it is used
6.16 - Column Major Access Pattern

- New access removes first cache line contents before it is used
Answer to Problem 6.16

- A: 512
- B: 256
- C: 50%
6.16 - Column Major Access Pattern

- What if the cache was 2048 bytes?
- No misses on second access to each block, since the entire array fits in the cache

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>
Answer to Problem 6.16

• A: 512
• B: 256
• C: 50%
• D: 25%
6.17 - Stride of One Word

- Access both x and y in row major order
6.17 - Stride of One Word

- Access both x and y in row major order
Answer to Problem 6.17

• A: 512

• B: 128
  - All are compulsory misses

• C: 25%

• D: 25%
  - Cache size doesn’t matter since all misses are *must*
  - The block size does matter though