Understanding CPU Caches

Ulrich Drepper
Introduction

Discrepancy main CPU and main memory speed

• Intel lists for Pentium M nowadays:
  - ~240 cycles to access main memory

• The gap is widening

• Faster memory is too expensive
The Solution for Now

CPU caches: additional set(s) of memory added between CPU and main memory

• Designed to not change the programs' semantics
• Controlled by the CPU/chipset
• Can have multiple layers with different speed (i.e., cost) and size
Cache Usage Factors

Numerous factors decide cache performance:

• Cache size

• Cacheline handling
  - associativity

• Replacement strategy

• Automatic prefetching
Cache Addressing

- Address (32/64 Bits)
- M Bits
  - Cacheline Size
- H Bits
  - Hash Bucket Address
- N-way Buckets
- Aliases!
Observing the Effects

Test Program to see the effects:

• Walks single linked list
  - Sequential in memory
  - Randomly distributed

• Write to list elements

```c
struct l {
    struct l *n;
    long pad[NPAD];
};
```
Sequential Access (NPAD=0)

Cycles / List Element vs Working Set Size (Bytes)
Sequential vs Random Access (NPAD=0)

- Cycles / List Element
- Working Set Size (in bytes)

Sequential
Random

Sequential: 
Random: 

Cycles:
0 50 100 150 200 250 300 350 400 450 500

Working Set Size (in bytes):
$2^{10}$ $2^{12}$ $2^{14}$ $2^{16}$ $2^{18}$ $2^{20}$ $2^{22}$ $2^{24}$ $2^{26}$ $2^{28}$
Sequential Access (NPAD=1)

Cycles / List Element vs. Working Set Size (Bytes)

- Follow
- Inc
- Addnext0
Optimizing for Caches I

- Use memory sequentially
  - For data, use arrays instead of lists
  - For instructions, avoid indirect calls

- Chose data structures as small as possible

- Prefetch memory
More Fun: Multithreading

1. CPU Core #1 and #3 read from a memory location; L2 the relevant L1 contain the data

2. CPU Core #2 writes to the memory location
   a) Notify L1 of core #1 that content is obsolete
   b) Notify L2 and L1 of second proc that content is obsolete
More Fun: Multithreading

3. Core #4 writes to the memory location
   a) Wait for core #2's cache content to land in main memory
   b) Notify core #2's L1 and L2 that content is obsolete
Optimizing for Caches II

Cacheline ping-pong is deadly for performance

• If possible, write always on the same CPU
• Use per-CPU memory; lock thread to specific CPU
• Avoid placing often independently read and written-to data in the same cacheline
Questions?