Anthony Danalis, Piotr Luszczek, Gabriel Marin,
Jeffrey S. Vetter, and Jack Dongarra

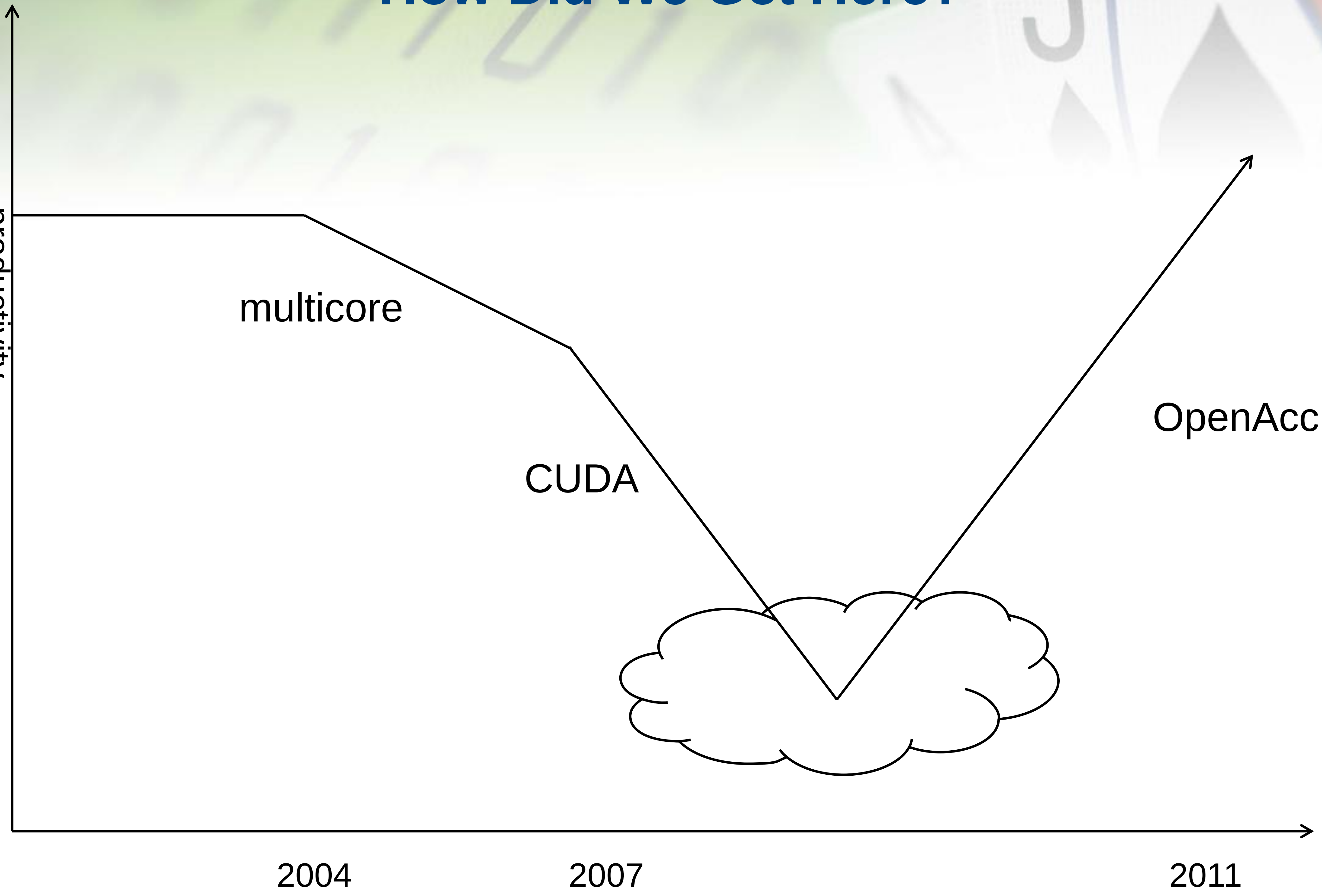
BlackjackBench:

Portable Hardware Characterization

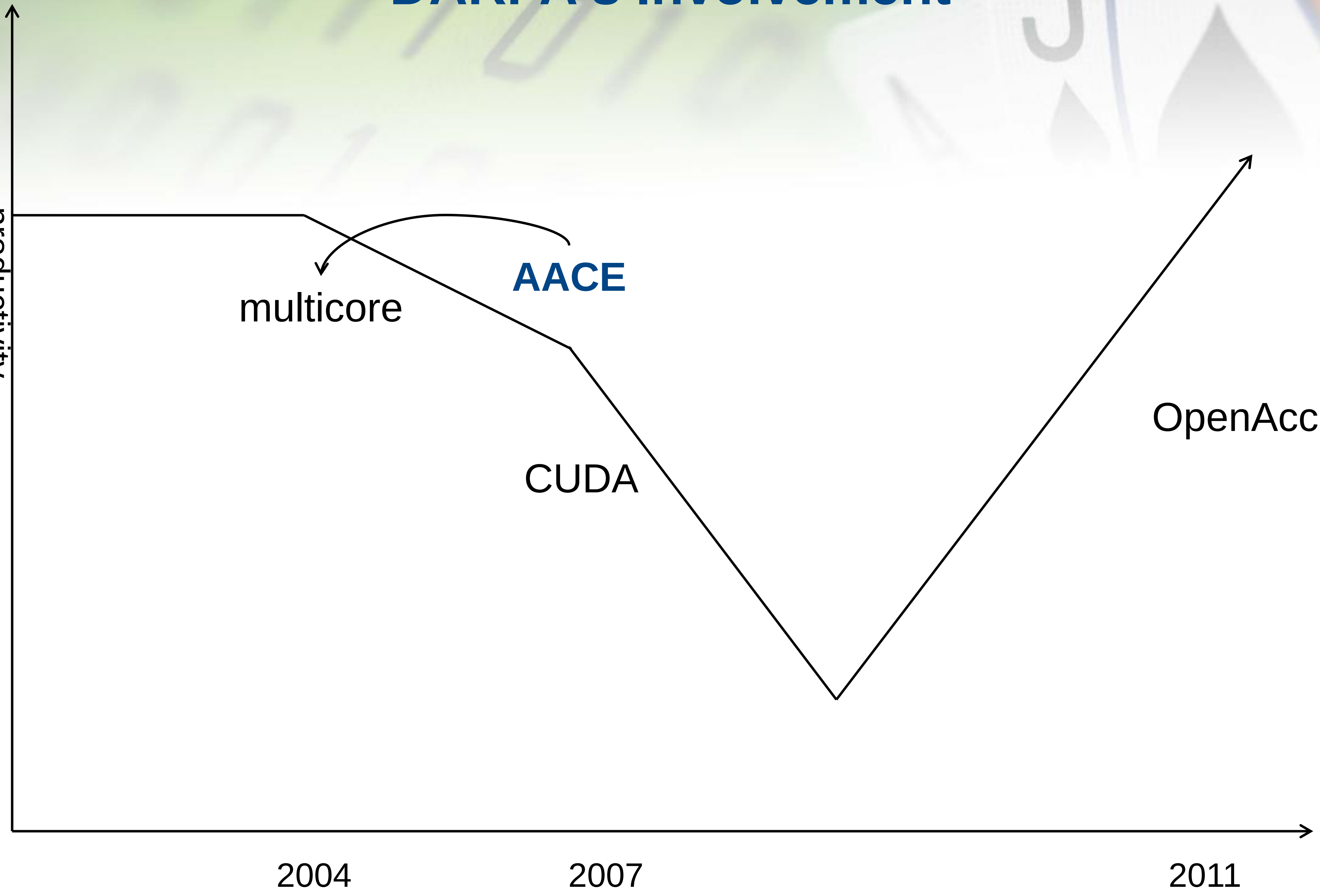
presented by

Piotr Luszczek

How Did We Get Here?



DARPA's Involvement



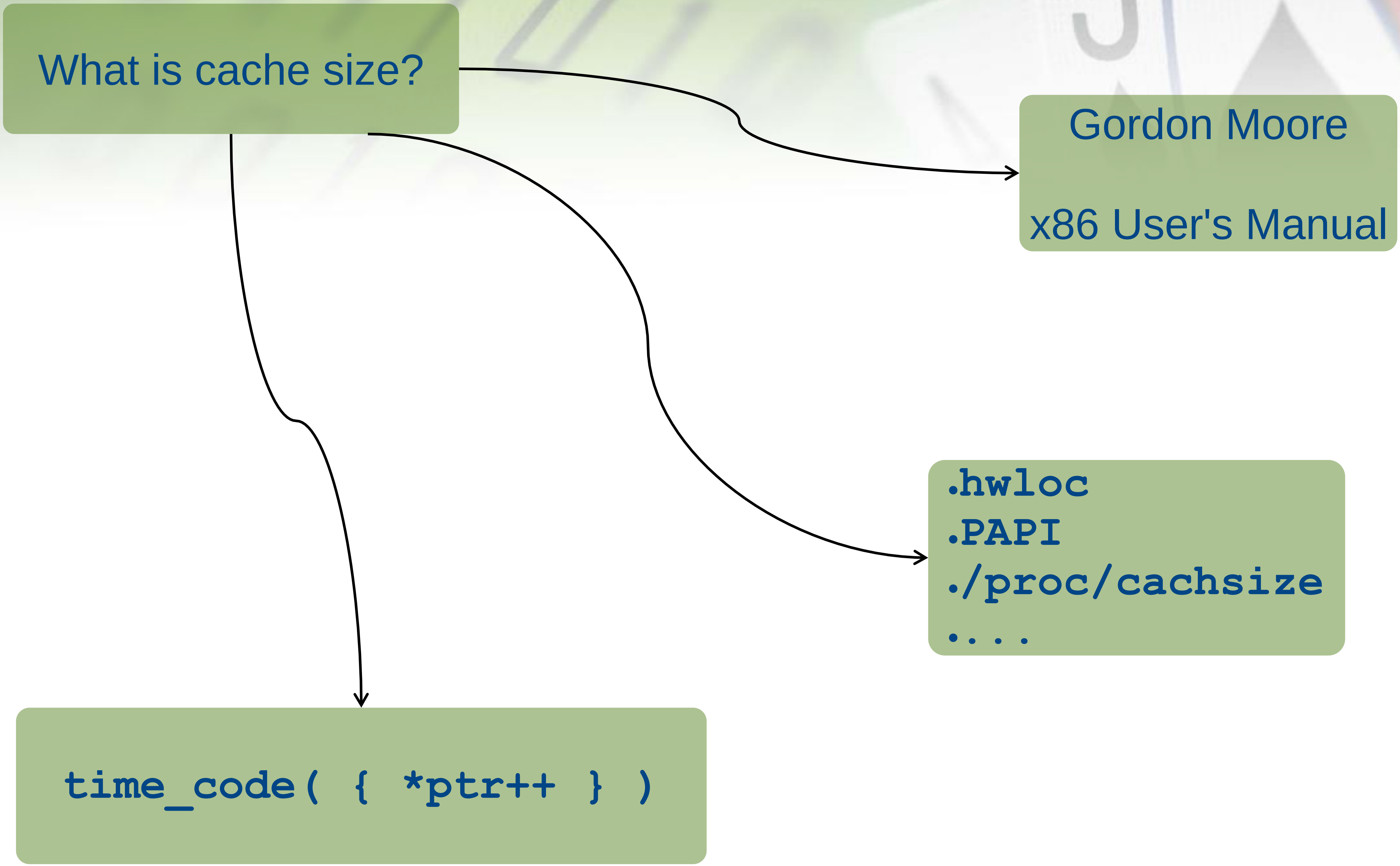
Goal:

Discover Hardware Related Parameters

with

Standard Software Tools

on Multicore Processors



What is cache size?

Gordon Moore

x86 User's Manual

- hwloc
- PAPI
- /proc/cachsize
- ...

```
time_code( { *ptr++ } )
```

Fundamental Concepts

The Canonical Micro-Benchmark Loop

Pointer Chaser

```
for (i = 0; i < N; ++i) {
  p[i] = ...; // create memory access pattern
}
```

Initialization

```
for (i = repetitions; i; ++i) {
  p = *p;
}

return *p; // compiler must see we're using the value
```

Read-only walk

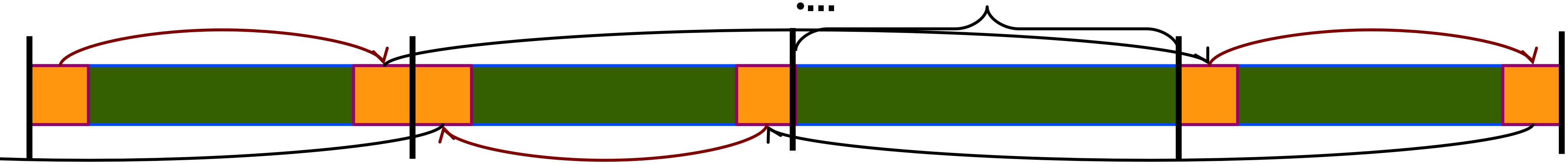
```
for (i = repetitions; i; ++i) {
  p ^= global_var_set_to_zero;
  p = *p;
}

return *p;
```

Read-write walk

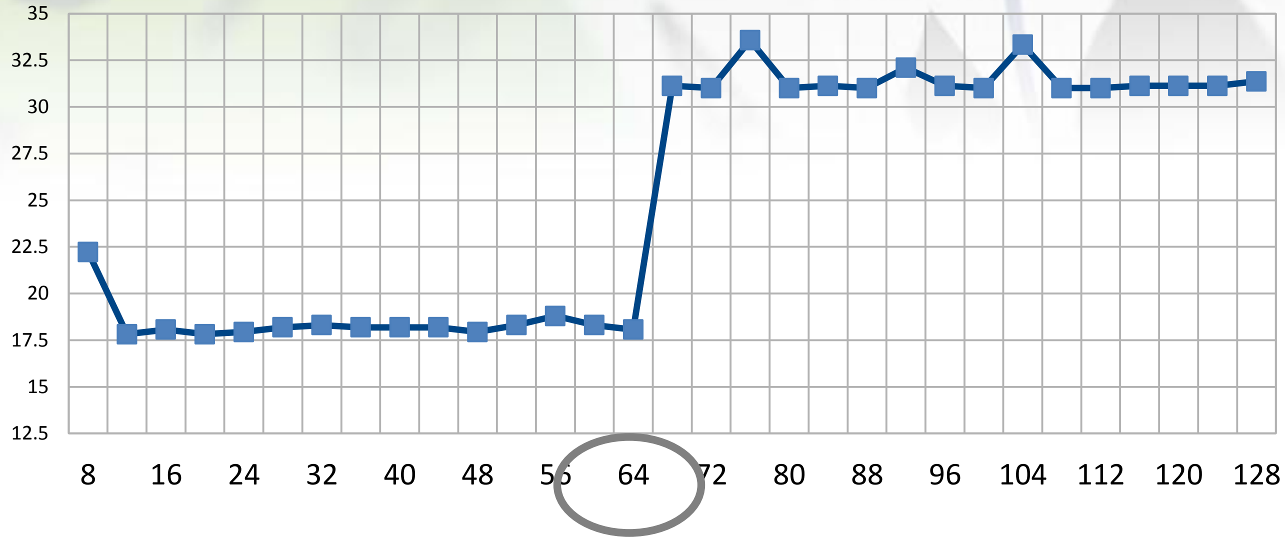
Visual View of a Pointer Chain

- Cache line
- TLB page
- One level of cache
- ...

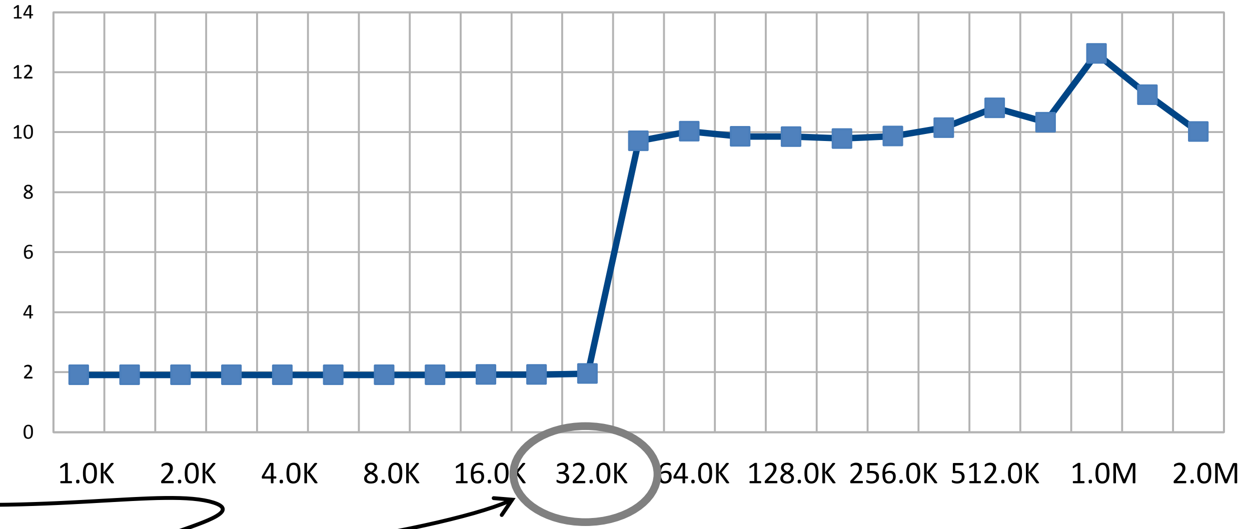


Cache Characterization

Level 1 Cache on x86

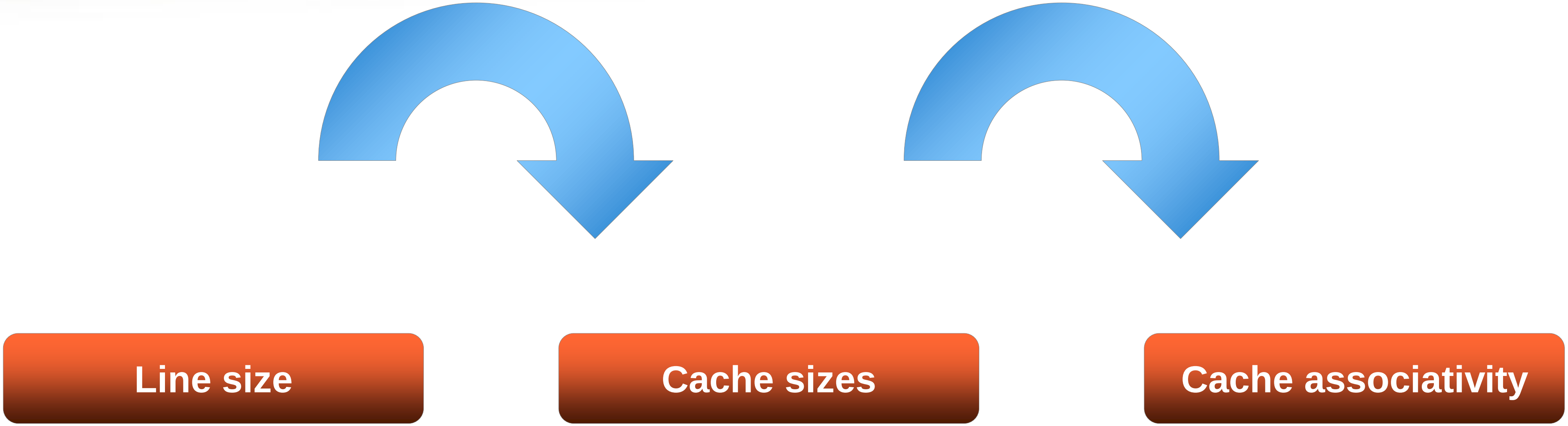


Line Size: 64 Bytes



L1 Size: 32 KBytes

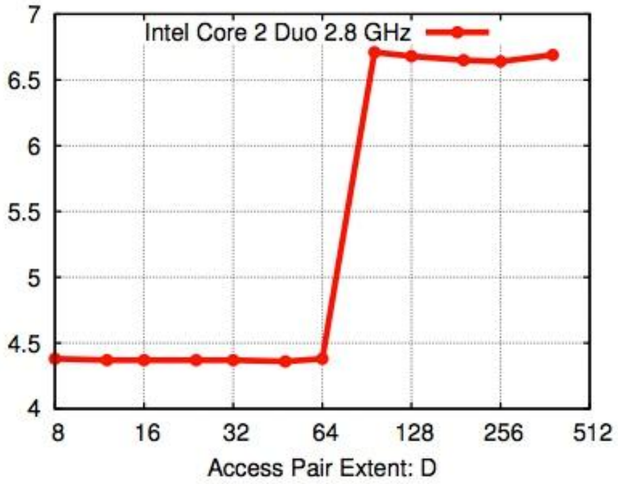
Bootstrapping of Cache Characterization

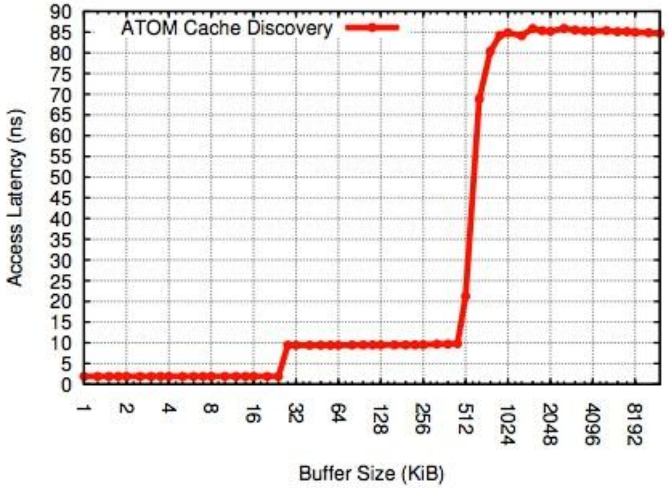


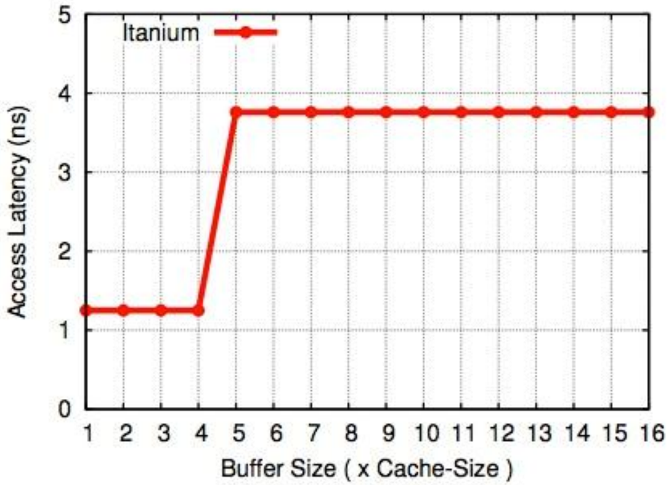
Line size

Cache sizes

Cache associativity

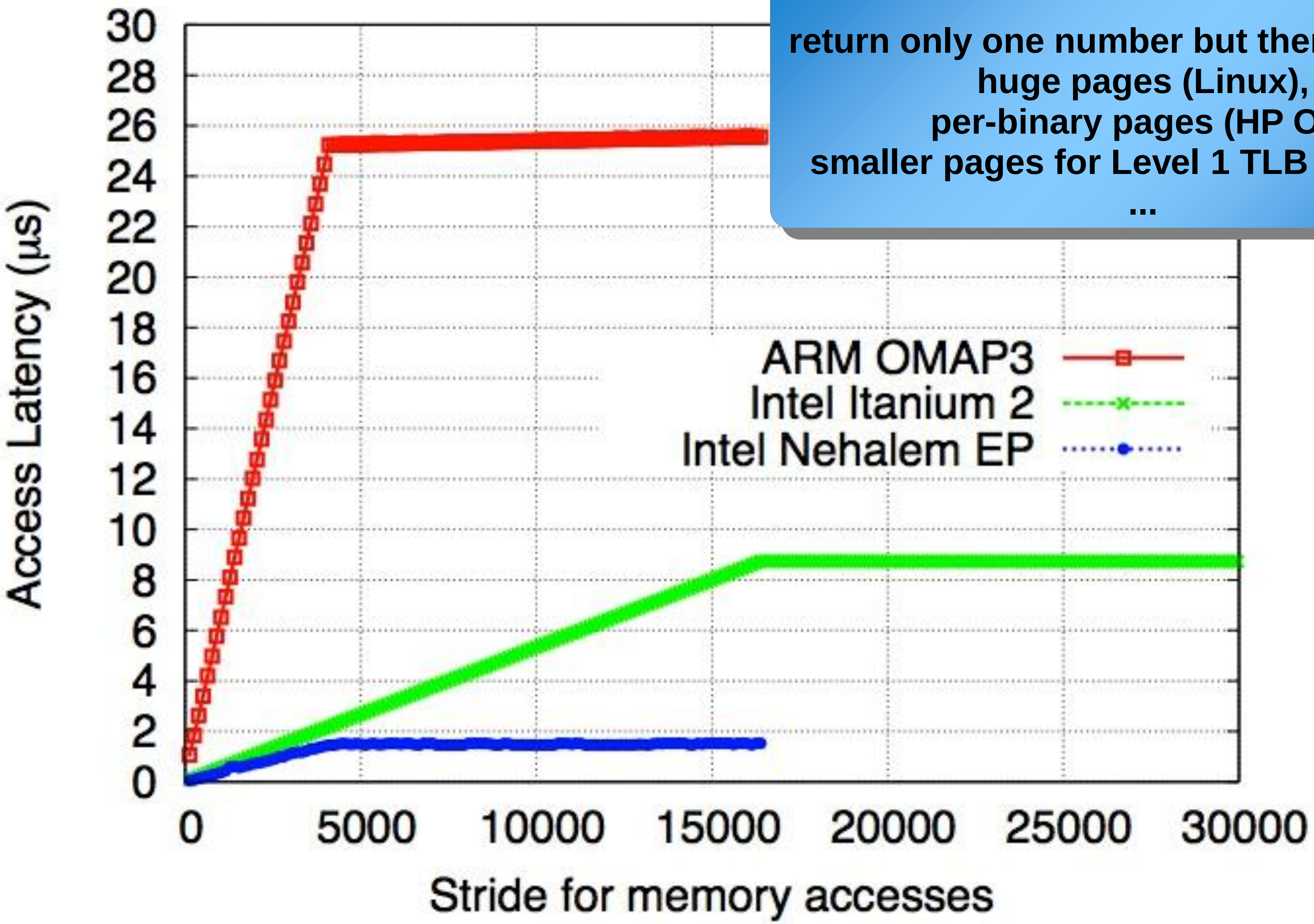






TLB Characterization

TLB Page Size



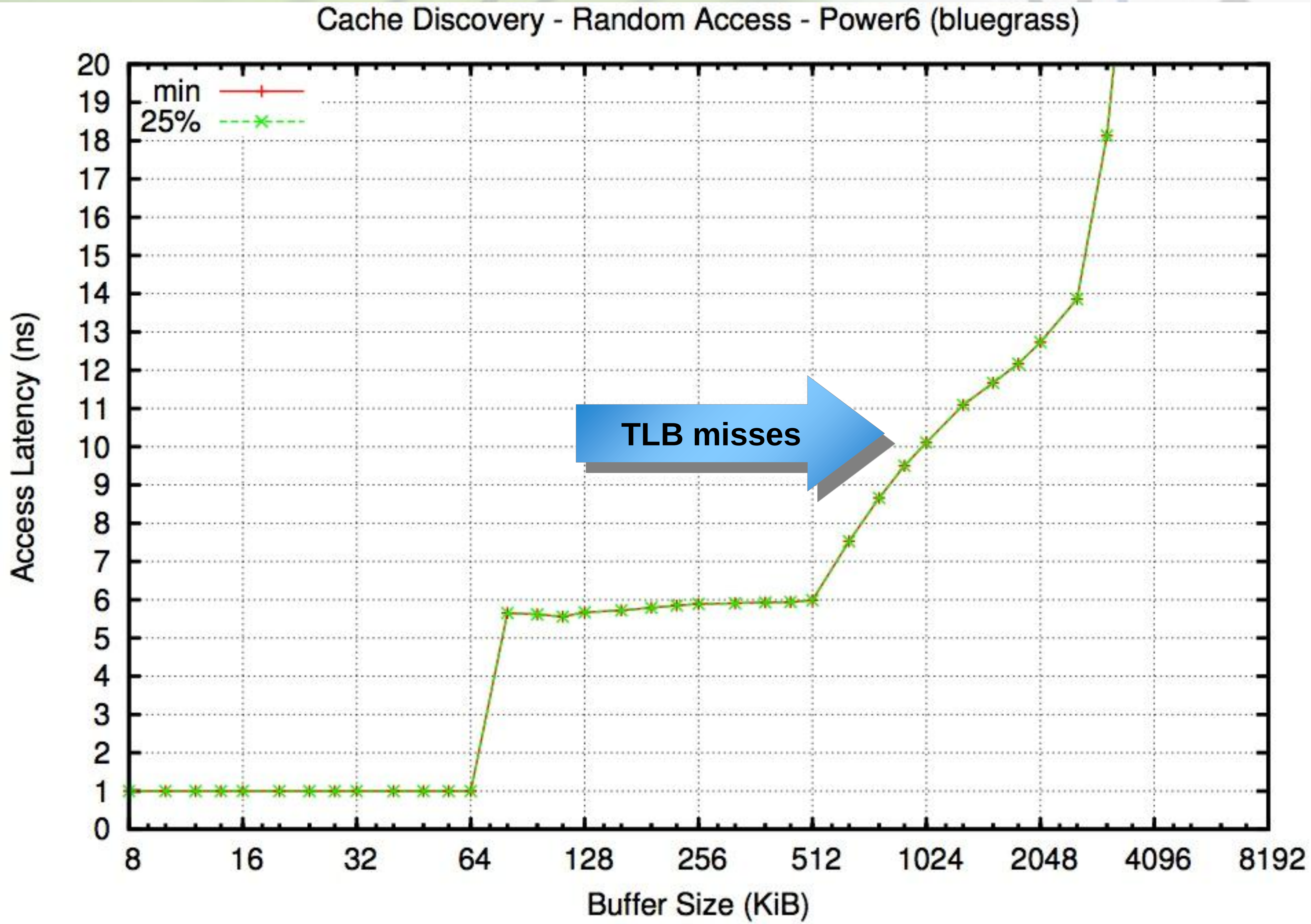
getpagesize() and sysconf(_SC_PAGE_SIZE)

**return only one number but there could be huge pages (Linux),
per-binary pages (HP OS),
smaller pages for Level 1 TLB (Itanium),
...**

Cache and TLB

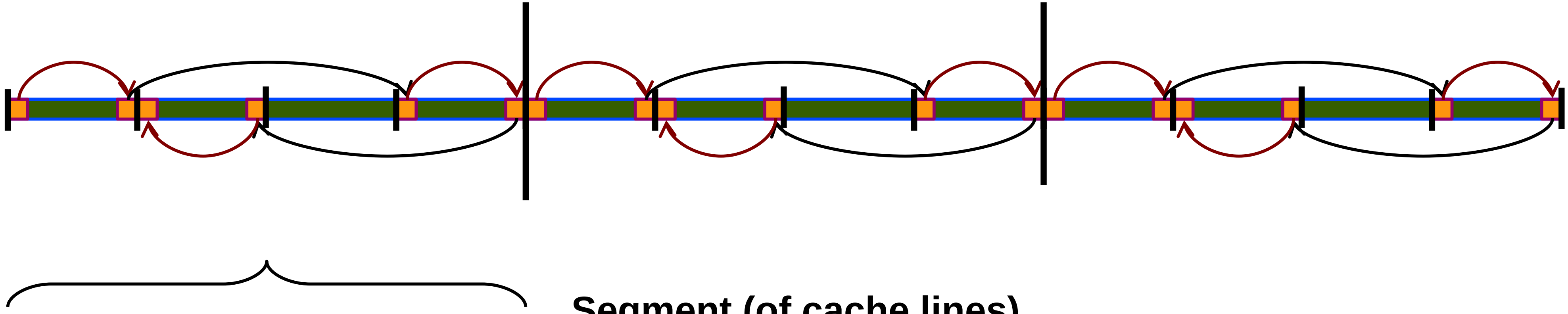
Affect Each Other

POWER6 – Cache Discovery – Random Access



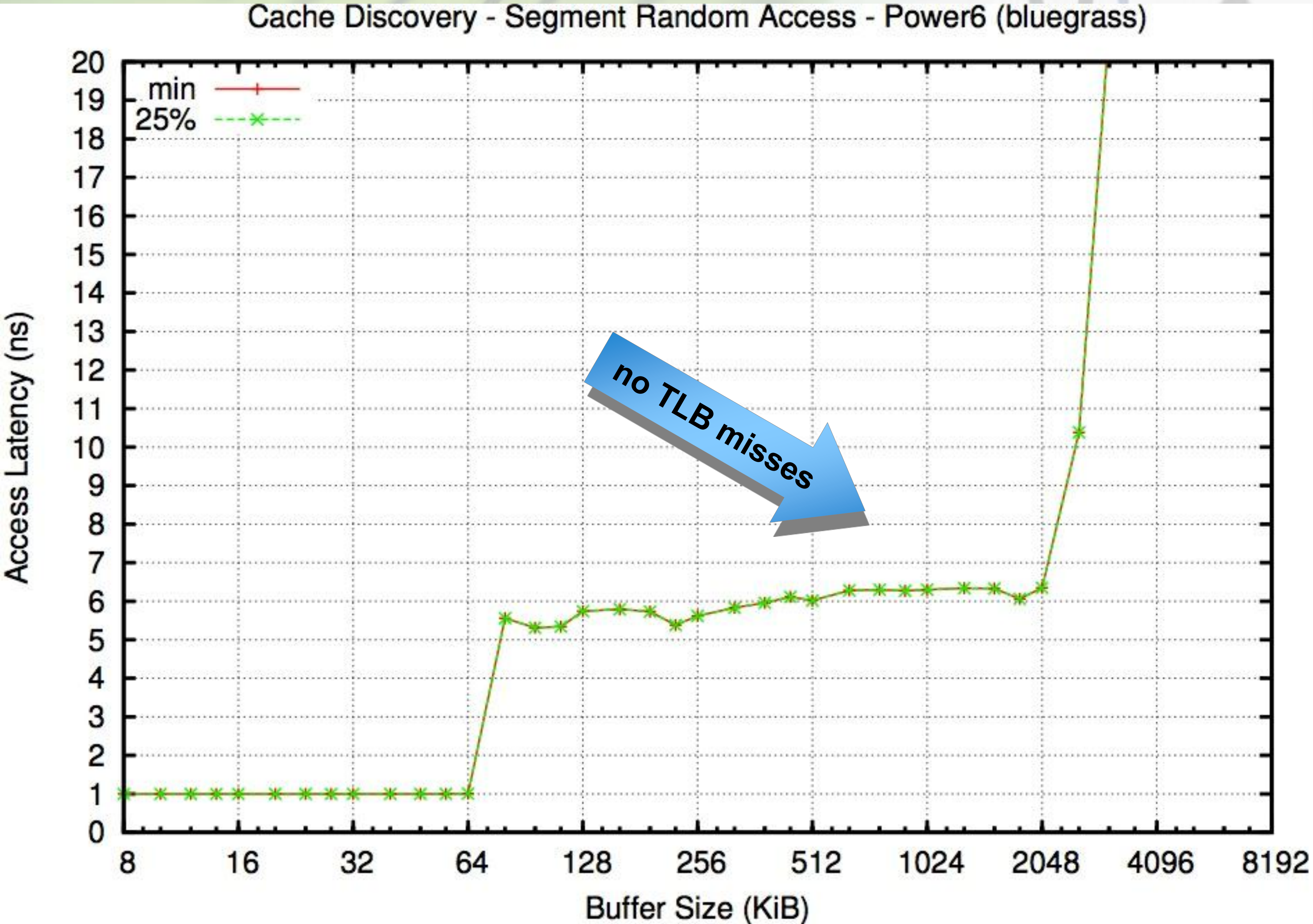
TLB misses

Segmented Random Access



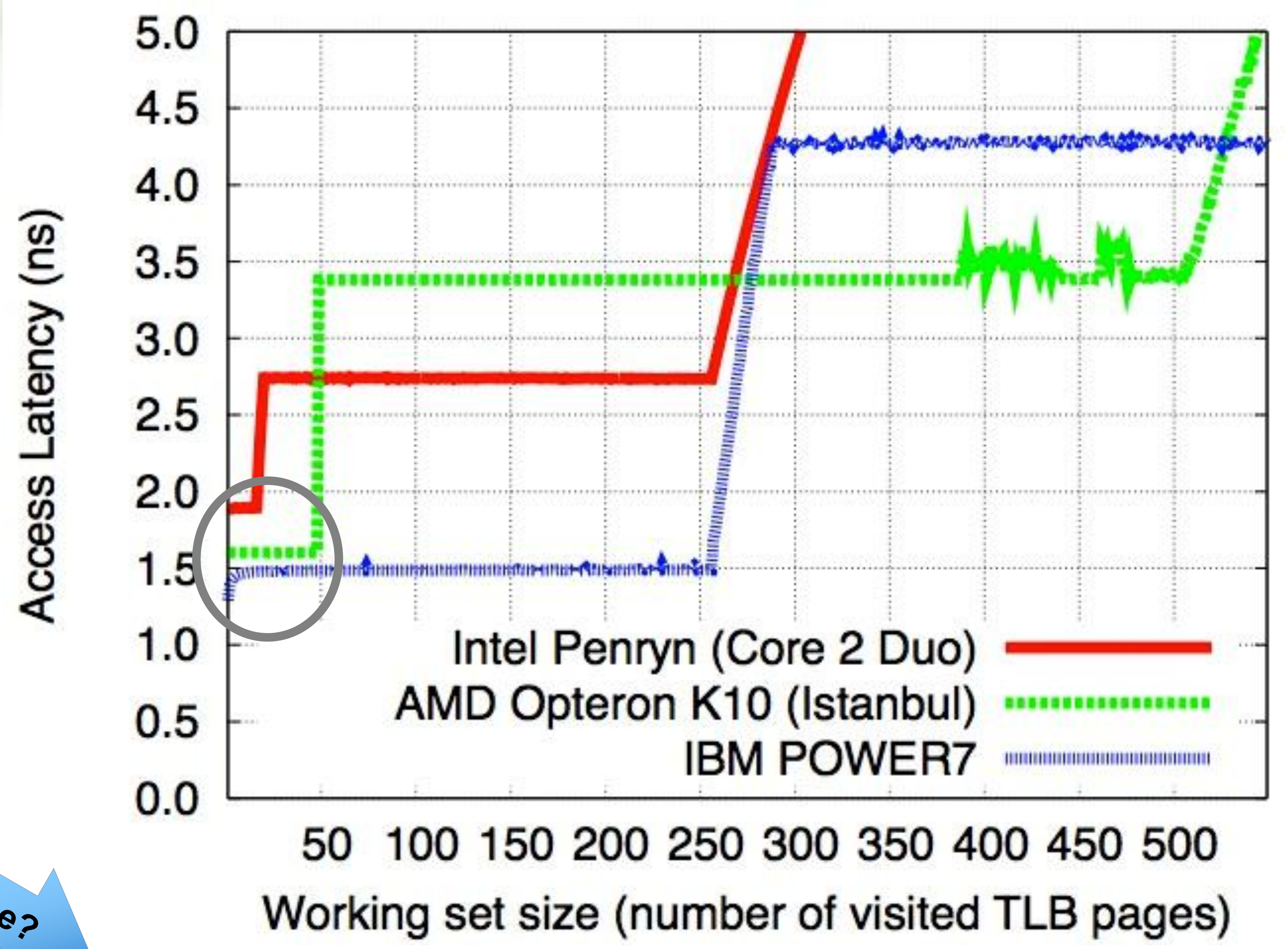
Segment (of cache lines)

POWER6 – Cache Discovery – Segment-Random



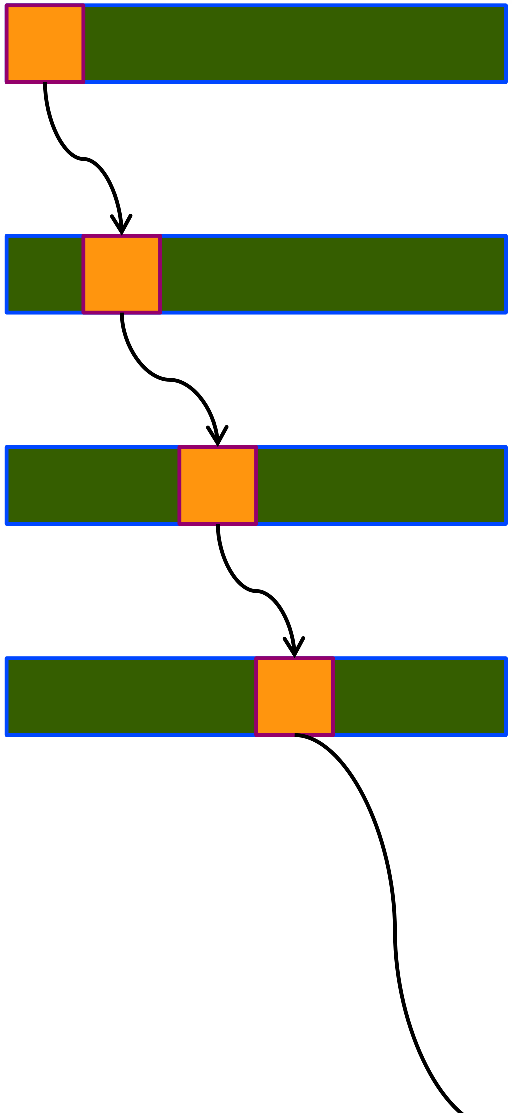
no TLB misses

TLB Level 1



TLB or cache?

Finding Level 1 TLB by Avoiding Cache Misses



Random TLB page 1

Random TLB page 2

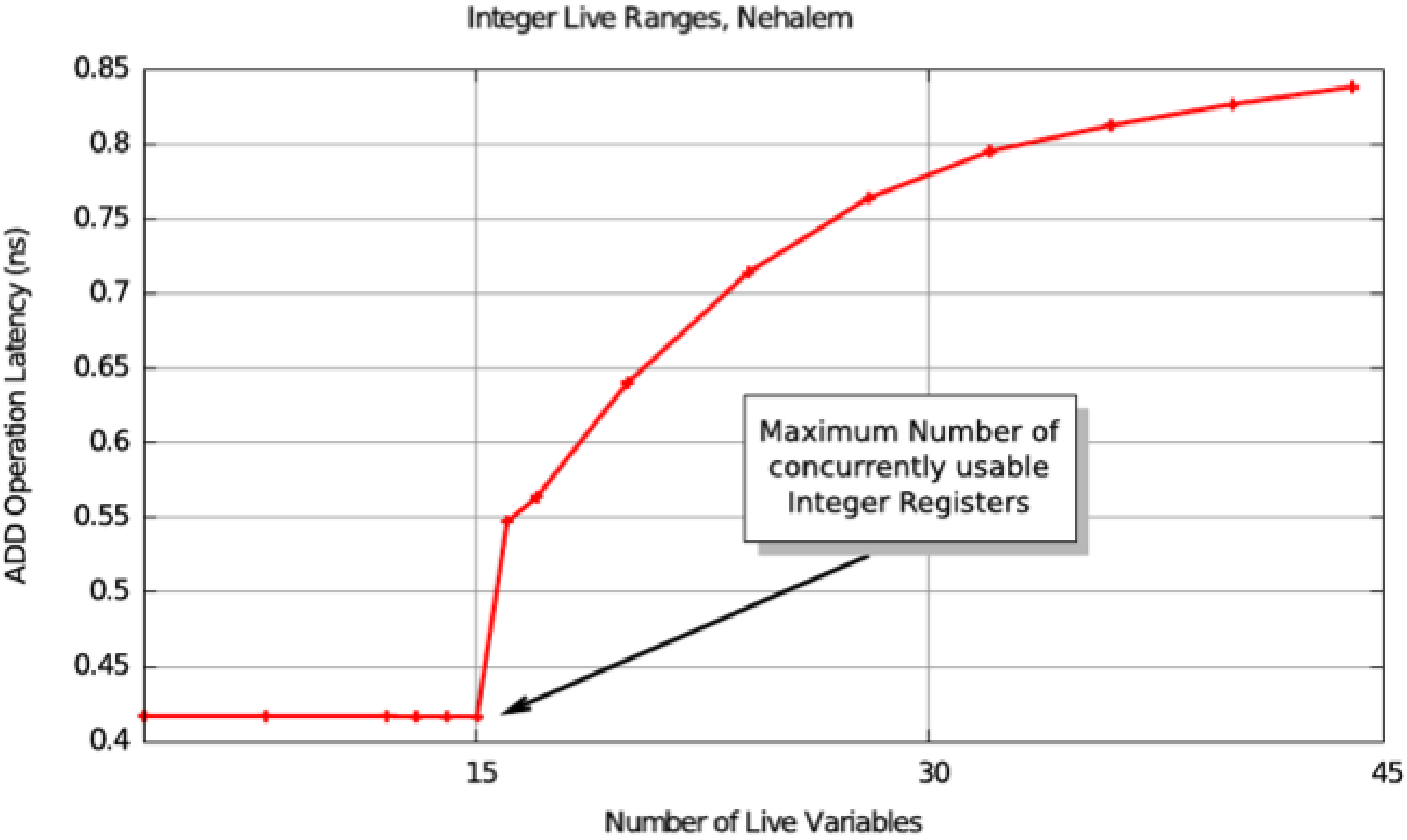
Random TLB page 3

Random TLB page 4

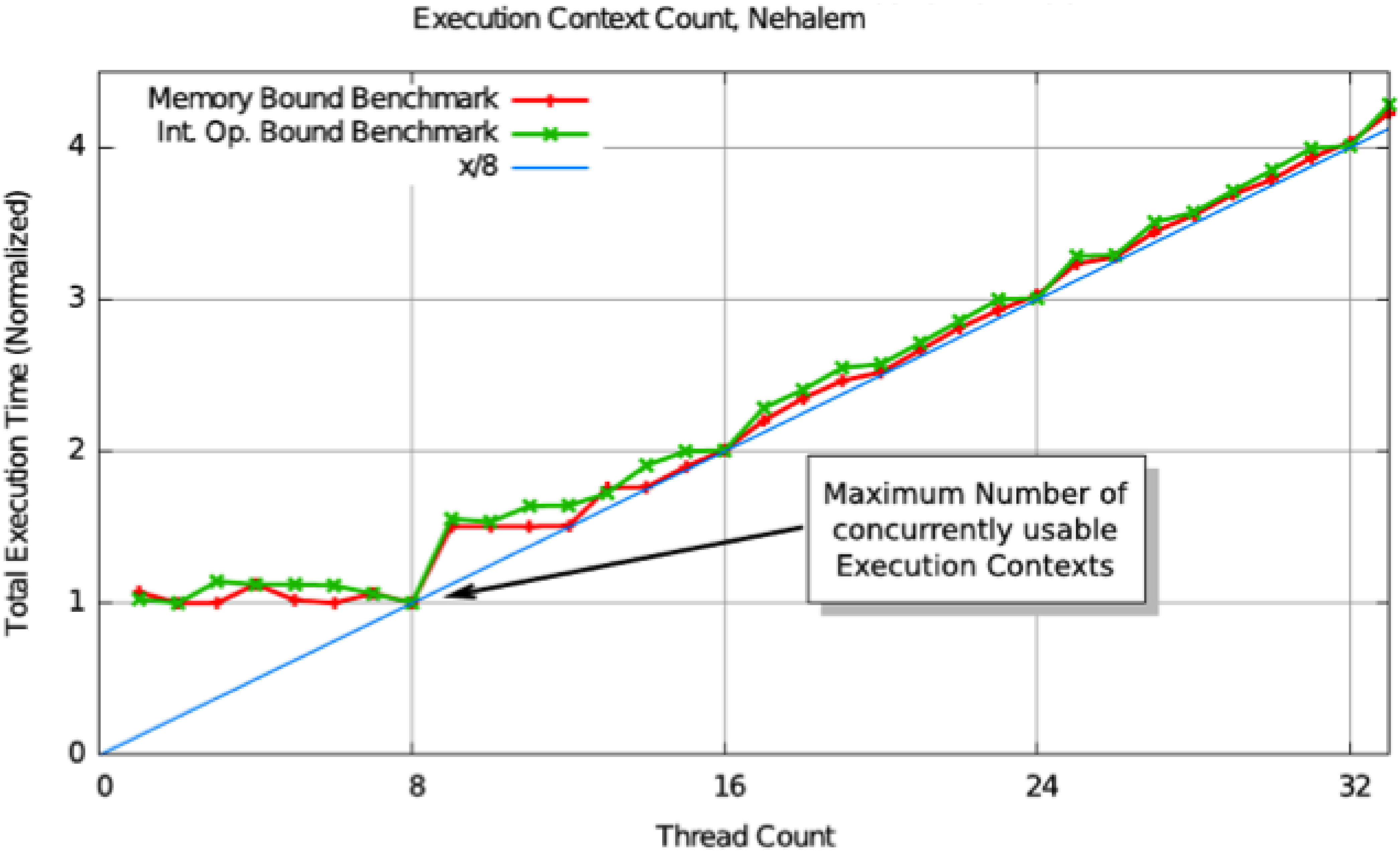
Characterization of Other Hardware Parameters

Execution Contexts, Live Ranges, ...

Live Range: No. of Concurrently Usable Registers



Execution Contexts



Maximum Number of concurrently usable Execution Contexts

POWER7 Characterization

Execution Context

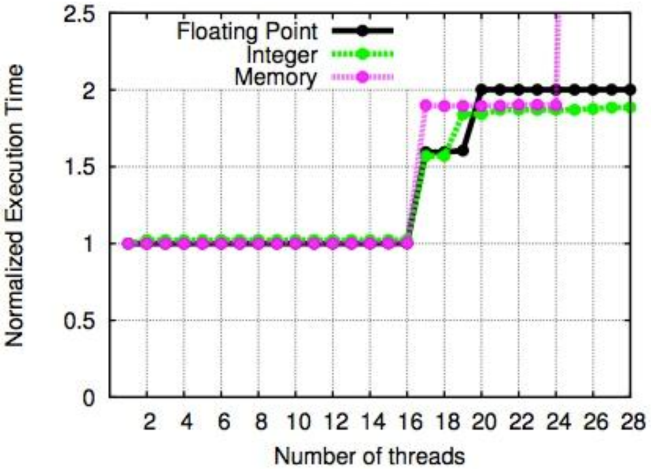
And

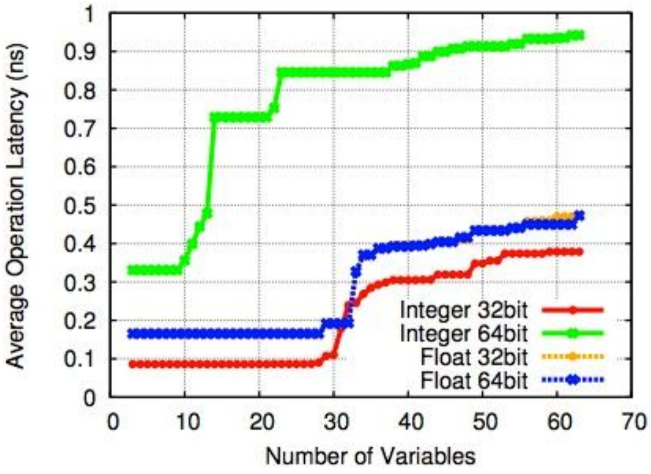
IBM's SMT

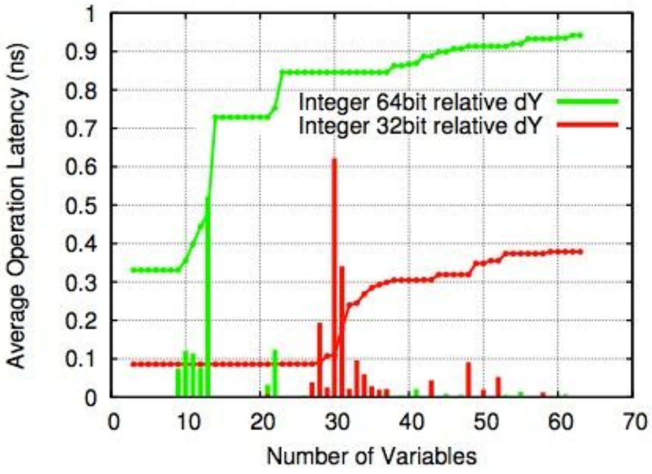
Live Ranges

Live Ranges

Analysis

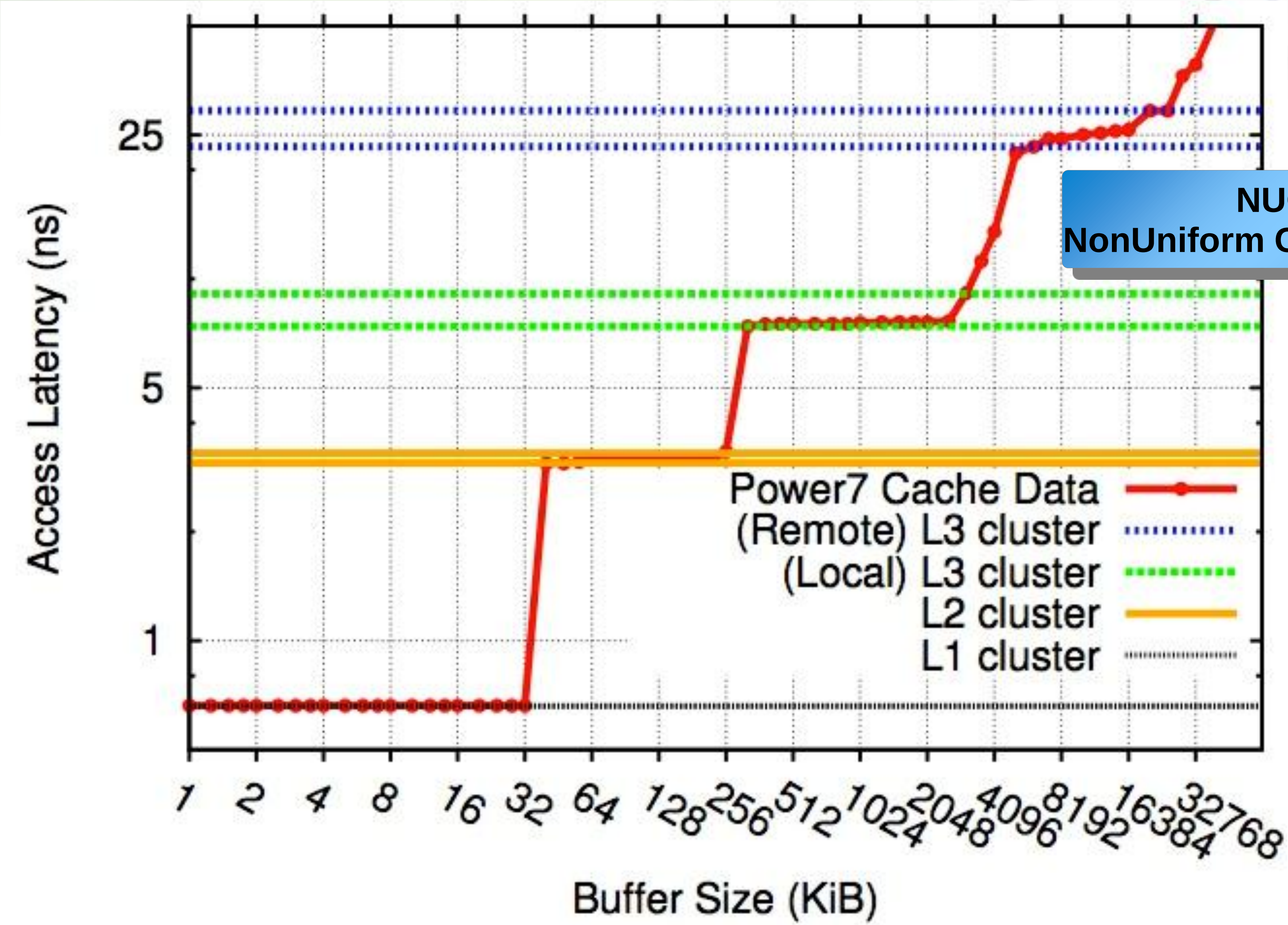






Data Analysis

Data Analysis with QT Clustering



NUCA
NonUniform Cache Access

Conclusions and Future Work

- The most interesting architecture was POWER7
  - ARM is almost as interesting
    - Locked-down TLB entries
    - Optional hardware features that differ between foundries
    - But the cache hierarchy is simple for now
- x86 prefetcher borders AI
- Hardware threading challenges
  - Intel's HyperThreading
  - IBM's SMT
  - AMD Bulldozer Architecture
- Core comm. matrix complexity

- Multicore are here to stay
  - Even if for limited number of workloads
- Accelerators will provide a rich environment for characterization
  - Internal structure not always known
  - Drastic changes between generations
  - Many levels of indirection:
    - Programming Language
    - Virtual ISA
    - Hardware