Analysis and modeling of

Computational Performance

Latency and throughput of memory

Memory latency

- Latency:
  - in general (recall):
    - time between the stimulation and the response, between the cause and the effect, between the beginning of operation and its end
  - for memory accesses the time between issuing a memory request and its finalization
    - for reading: arrival of data
    - for writing: storing data in memory
      - complicated by cache coherence protocols (see Lecture 6)
  - theoretical values based on hardware characteristics
  - experimental estimates:
    - very short times, impossible to measure individually
    - there must be multiple accesses to measure time
    - how to arrange multiple accesses so that the average access time corresponds to a single separate memory access
      - how to eliminate the effects of all latency hiding techniques

Memory latency

- Typical loop:

```
.L5:
addsd     0(%rbp,%rdx), %xmm7
addq      $8, %rdx
cmpq      $80000000, %rdx
movsd     %xmm7, (%rsp)
jne    .L5
```

  - in each iteration:
    - time for arithmetic operations: several cycles
    - time for DRAM memory accesses: hundreds of cycles
- Two mechanisms for latency hiding:
  - cache memories
  - prefetching (hardware and software)
    - e.g. for the loop above

```
addsd 0(%rbp,%rdx), %xmm7
[prefetch data for next iteration]
```

    - critical hardware ability to process many memory requests concurrently

Memory latency

- How to measure latency experimentally:
  - different types of accesses (depends also whether inclusive or exclusive caches, shared or separate per core caches etc.)
    - L1 (L1 hit)
    - L2 (L1 miss)
    - L2 from a different core (L1, L2 miss, cache coherence protocol)
    - L3 (L1, L2 miss)
    - L3 from a different processor (L1, L2, L3 miss, cache coherence protocol)
    - DRAM (L1, L2, L3 miss)
    - other? (NUMA?)
  - organization of accesses
    - should not have data locality
      - temporal – single data element accesses separated by accesses to many other elements (to force eviction from caches)
      - spatial – no accesses to the same cache line

Memory latency

- How to measure latency experimentally:
  - several simple strategies:
    - only one array accessed
    - read only accesses
      - e.g. `sum += tab[index];`
    - write only accesses
      - e.g. `tab[index] = data;`
    - read-modify-write accesses:
      - e.g. `tab[index]++;`
    - strided accesses:
      - e.g. `tab[index]++; index+= stride;`
    - random accesses:
      - e.g. `index = random_cache_line*cache_line_size; tab[index]++;`
    - pointer chasing:
      - e.g. `index = tab[index];`

Memory throughput (bandwidth)

- The maximal transfer rate between processor and a given level of memory hierarchy
- Should use all available latency hiding mechanisms (except caches closer to pipelines and temporal locality):
  - prefetching (hardware and may be software)
  - concurrency (including multithreading)
    - at all levels – memory controller, buses, DRAM modules
      - pipelining, multi-banking, non-blocking, etc.
- Theoretical throughput (bandwidth)
  - based on hardware characteristics
- Experimental estimates:
  - massive transfers
  - many independent memory requests
    - maximizing concurrency
    - multithreading for accesses to shared resources

Memory throughput (bandwidth)

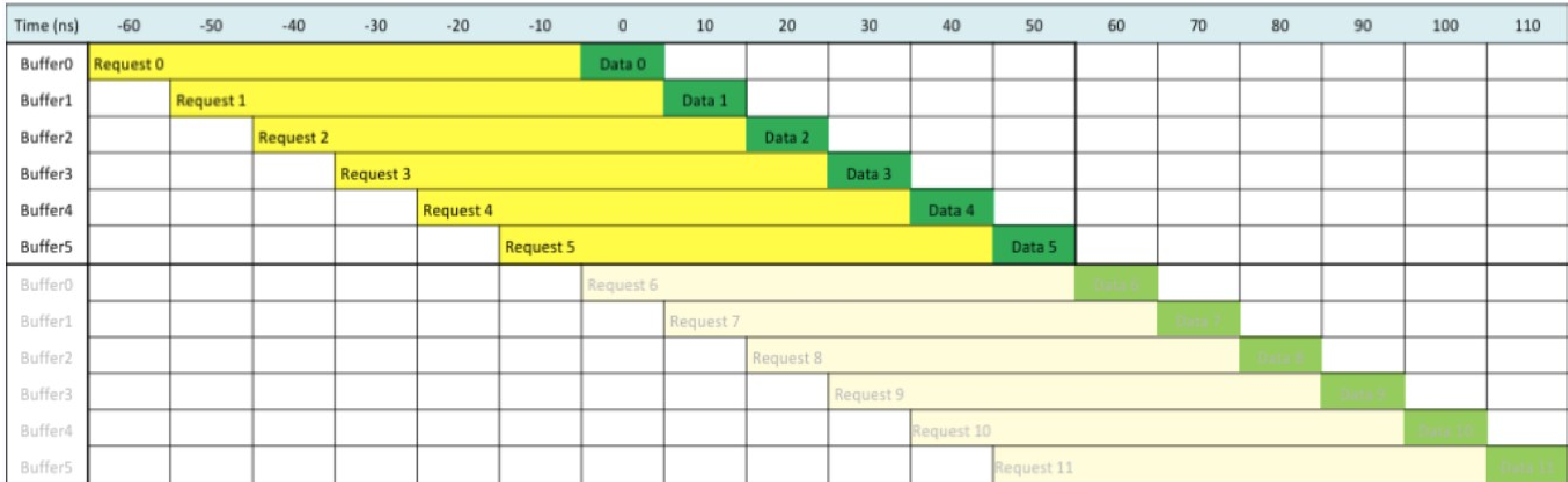
- How to measure throughput experimentally:
  - massive transfer
    - array(s) fitting in the given memory level
    - multiple repetitions
      - accesses to the same element must be from the tested memory level – separated by sufficient number of accesses to different elements to evict from levels of memory closer to the core
  - spatial locality
    - stride 1, full exploitation of the content of cache lines
  - many independent memory requests
    - for different cache lines
  - number_of_accesses * sizeof(data) / execution_time
    - number of accesses from source code (checked with assembly code)
      - effective accesses – data used in the code
        - » not the data transferred by hardware, due e.g. to prefetching
        - » the use of hardware counters can be misleading

Little's law for memory accesses

- Little's law (recall)
  - the average number L of customers in a stationary system is equal to the average effective arrival rate λ multiplied by the average time W that a customer spends in the system: $L = \lambda W$
  - for memory access requests:
    - L – the number of requests processed concurrently [B]
      - should be measured by the number of cache lines
    - λ – the throughput [GB/s]
    - W – the time to process each of the memory requests [ns]
  - **in order to maximize the throughput**, $\lambda = L / W$, i.e. to keep it as close as possible to the theoretical maximum, given the time W that depends on hardware and operating system:
    - **maximize the number of requests processed concurrently** L
      - sufficient number of independent requests in the code

Little's law for memory accesses

60 ns latency, 6.4 GB/s (=10ns per 64B cache line)

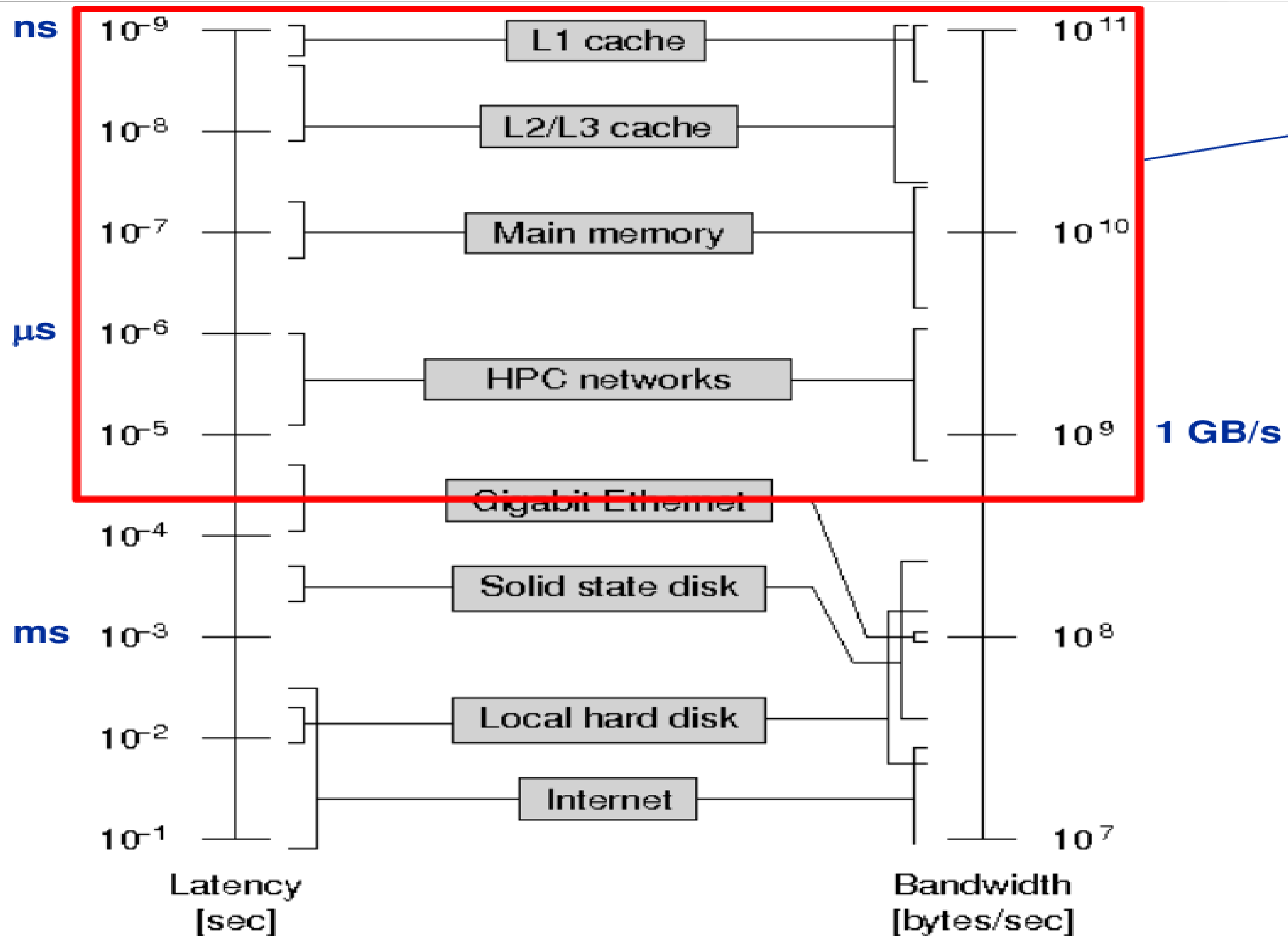


- 60 ns * 6.4 GB/s = 384 Bytes = 6 cache lines
- To keep the pipeline full, there must always be 6 cache lines “in flight”
- Each request must be launched at least 60 ns before the data is needed

Memory throughput

- Memory throughput factors:
  - hardware
    - memory technology (e.g. DDR4)
    - number of banks, ranks etc.
    - number of channels
    - the width of a single channel (bus, usually 64 bits)
    - processor's memory system capabilities (often expressed as the number of (usually 64-bit) transactions per second)
  - software
    - number of generated cache line accesses
      - several arrays or proper loop unrolling for a single array
    - spatial locality of accesses
      - full use of the whole cache hierarchy
    - vectorization of accesses (e.g. *-march=core-avx2*)
    - alignment of arrays in memory (e.g. posix_memalign(...))

Latency and throughput



Example theoretical cache parameters

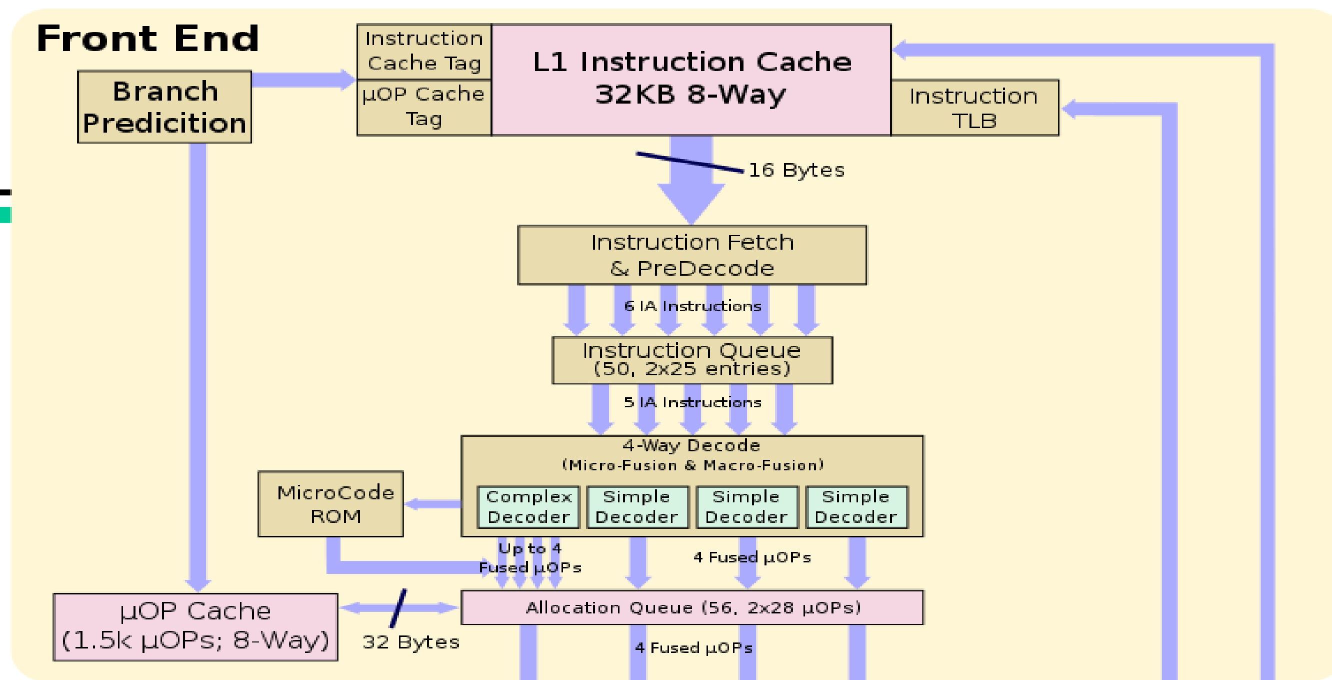
Table 2-1. Cache Comparison Between Skylake Microarchitecture and Broadwell Microarchitecture

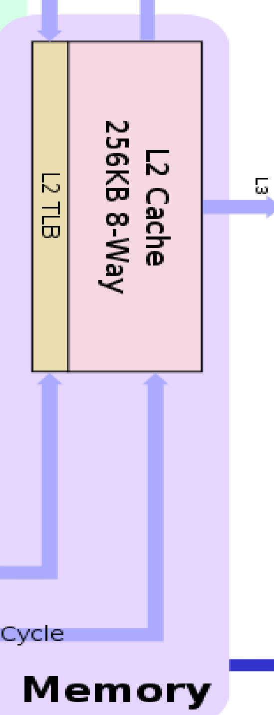
Cache level	Category	Broadwell Microarchitecture	Skylake Server Microarchitecture
L1 Data Cache Unit (DCU)	Size [KB]	32	32
	Latency [cycles]	4-6	4-6
	Max bandwidth [bytes/cycles]	96	192
	Sustained bandwidth [bytes/cycles]	93	133
	Associativity [ways]	8	8
L2 Mid-level Cache (MLC)	Size [KB]	256	1024 (1MB)
	Latency [cycles]	12	14
	Max bandwidth [bytes/cycles]	32	64
	Sustained bandwidth [bytes/cycles]	25	52
	Associativity [ways]	8	16
L3 Last-level Cache (LLC)	Size [MB]	Up to 2.5 per core	up to 1.375[1] per core
	Latency [cycles]	50-60	50-70
	Max bandwidth [bytes/cycles]	16	16
	Sustained bandwidth [bytes/cycles]	14	15

Paged virtual memory and caches

TLB	Page table	Cache	Possible? If so, under what circumstance?
Hit	Hit	Miss	Possible, although the page table is never really checked if TLB hits.
Miss	Hit	Hit	TLB misses, but entry found in page table; after retry, data is found in cache.
Miss	Hit	Miss	TLB misses, but entry found in page table; after retry, data misses in cache.
Miss	Miss	Miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
Hit	Miss	Miss	Impossible: cannot have a translation in TLB if page is not present in memory.
Hit	Miss	Hit	Impossible: cannot have a translation in TLB if page is not present in memory.
Miss	Miss	Hit	Impossible: data cannot be allowed in cache if the page is not in memory.

Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	15,000–50,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	500–4000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%


Front End
Branch Predicition
Instruction Cache Tag
µOP Cache Tag
L1 Instruction Cache 32KB 8-Way
Instruction TLB
16 Bytes
Instruction Fetch & PreDecode
6 IA Instructions
Instruction Queue (50, 2x25 entries)
5 IA Instructions
MicroCode ROM
4-Way Decode (Micro-Fusion & Macro-Fusion)
Complex Decoder
Simple Decoder
Simple Decoder
Simple Decoder
Up to 4 Fused µOPs
4 Fused µOPs
µOP Cache (1.5k µOPs; 8-Way)
32 Bytes
Allocation Queue (56, 2x28 µOPs)
4 Fused µOPs
Execution Engine
Rename / Allocate / Retirement
ReOrder Buffer (192 entries)
Zeroing Idioms
Scheduler
Unified Reservation Station (RS)
(64 entries)
Integer Physical Register File (168 Registers)
Vector Physical Register File (168 Registers)
Port 0
Port 1
Port 5
Port 6
Port 2
Port 3
Port 4
Port 7
ALU & Shift
Branch
Divide
256b FMA
256b FP Mul
Vect Int Mul
Vector Logic
Vector Shift
ALU
LEA
Multiply
256b FMA
256b FP Add
Vect Int ALU
Vector Logic
ALU
LEA
Vect Shuffle
Vect Int ALU
Vector Logic
ALU & Shift
Branch
Load Addrs
Store Addrs
Load Addrs
Store Addrs
Store Data
Store Addrs
Store Buffer & Forwarding (42 entries)
32B/Cycle store
L1 Data Cache 32KB 8-Way
Data TLB
Load Buffer (72 entries)
2x32B/Cycle load
64B/Cycle
Memory


L2 TLB
L2 Cache 256KB 8-Way
L3