CPU Caches

Jamie Allen

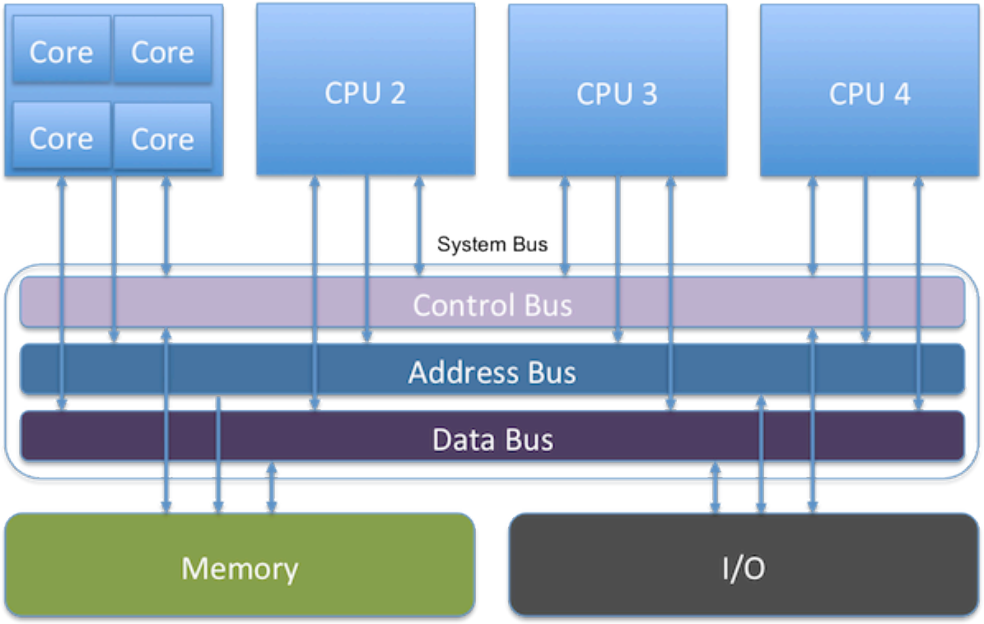
JavaZone 2012

Why?

- Increased virtualization
  - Runtime (JVM, RVM)
  - Platforms/Environments (cloud)
- Disruptor, 2011

SMP

- Symmetric Multiprocessor (SMP) Architecture
- Shared main memory, controlled by single OS
- No more Northbridge



Memory Controller

- Handles communication between the CPU and RAM
- Contain the logic to read to and write from RAM
- Integrated Memory Controller on die

NUMA

- Non-Uniform Memory Access
- Access time is dependent on the memory locality to a processor
- Memory local to a processor can be accessed faster than memory farther away
- The organization of processors reflect the time to access data in RAM, called the NUMA factor
- Shared memory space (as opposed to multiple commodity machines)

Cache Lines

- Most commonly 64 contiguous bytes, can be 32-256
- Look out for false sharing
- Padding can be used to ensure unshared line
- Transferred in 64-bit blocks (8x for 64 byte lines), arriving every ~4 cycles
- Position in the line of the "critical word" matters, but not if pre-fetched
- @Contended annotation coming to JVM?

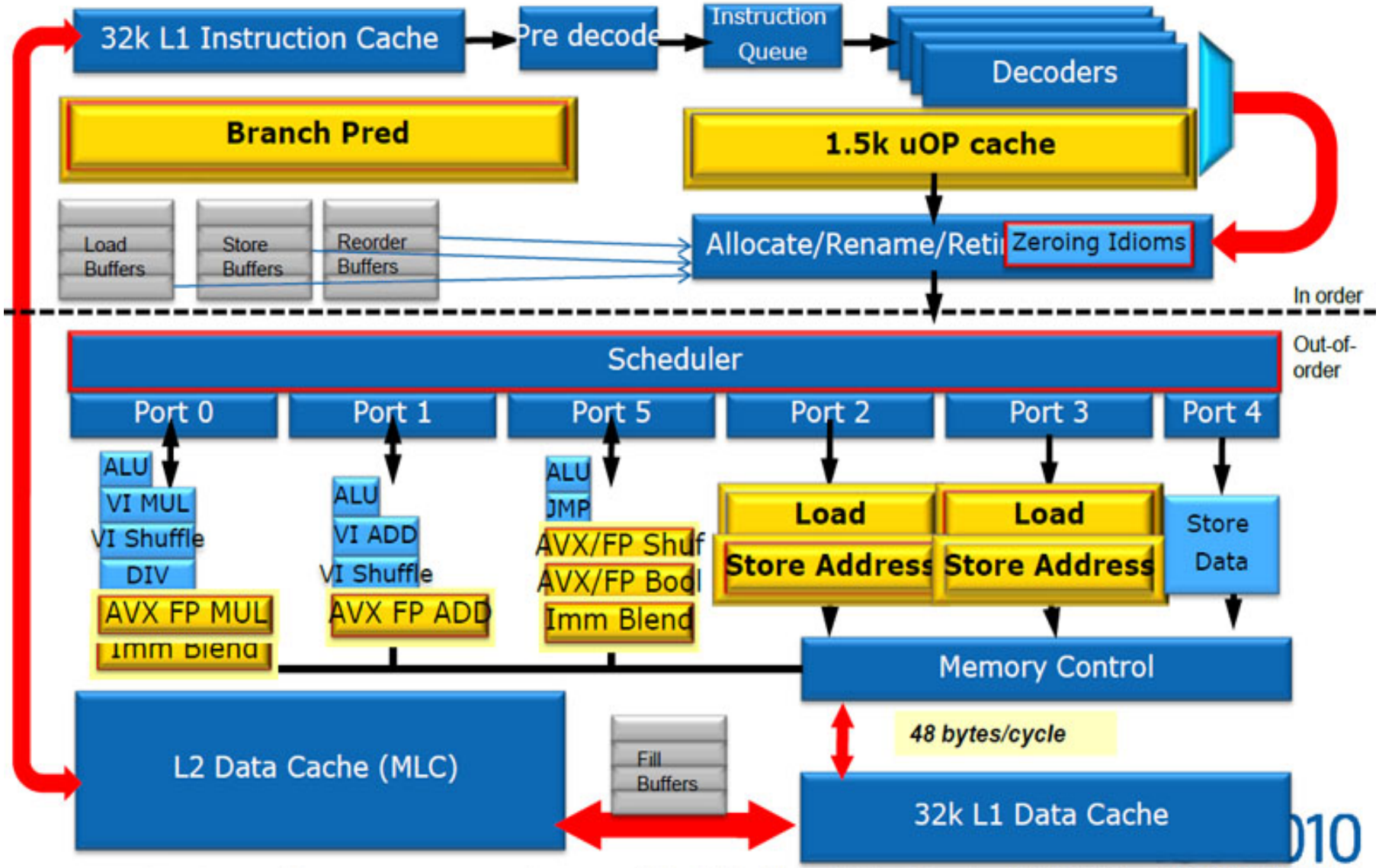
Cache Write Strategies

- Write through - changed line immediately goes back to main memory
- Write back - line is marked when dirty, eviction sends back to main memory
- Write combining - grouped writes of cache lines back to main memory
- Uncachable - dynamic values that can change without warning

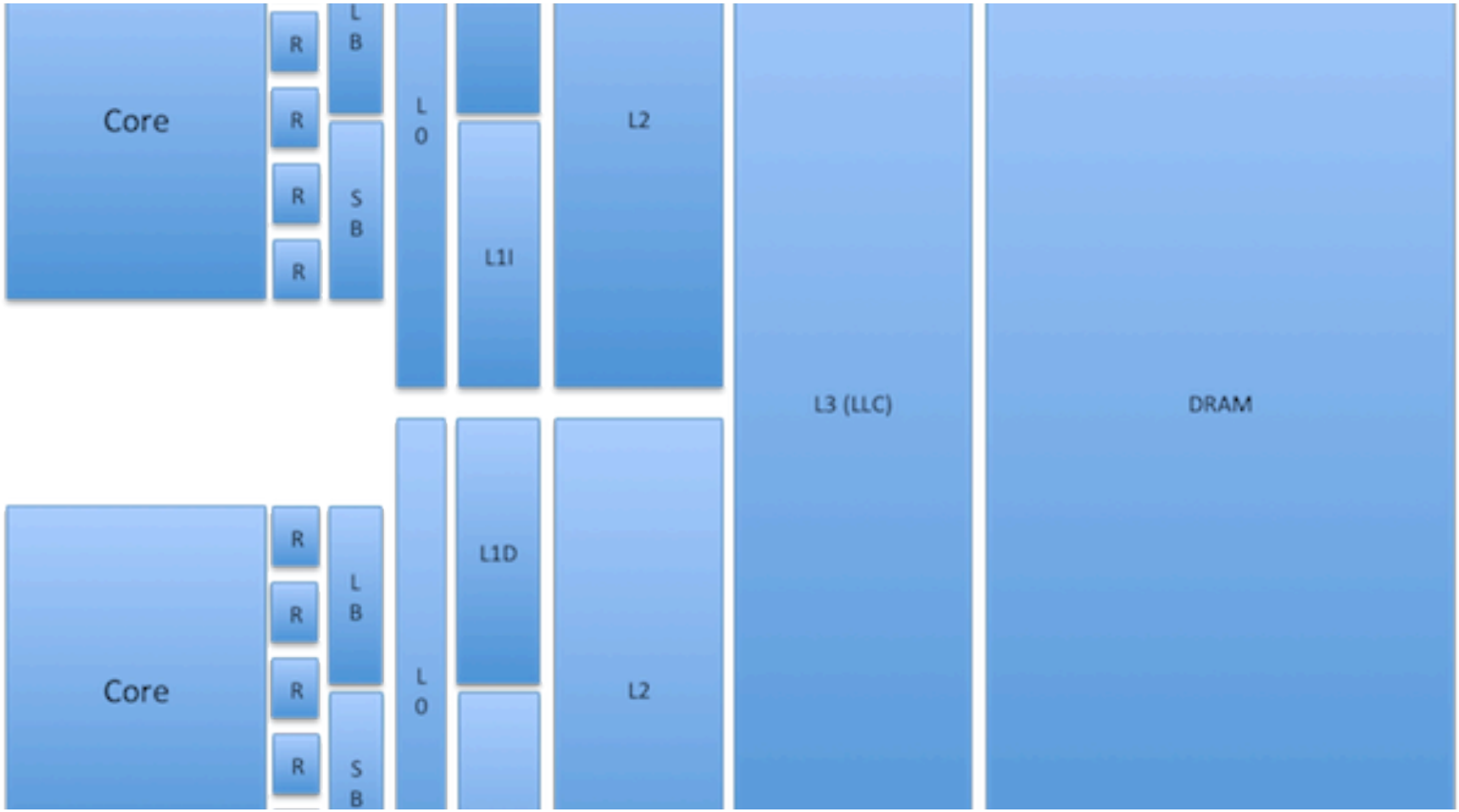
Current Processors

- Intel
  - Nehalem/Westmere
  - Sandy Bridge
  - Ivy Bridge
- AMD
  - Bulldozer
- Oracle
  - UltraSPARC isn't dead

Sandy Bridge Microarchitecture



CPU Caches



Data Locality

- The most critical factor in performance
- **Spatial** - reused over and over in a loop, data accessed in small regions
- **Temporal** - high probability it will be reused before long

CPU or Instruction Cycle

- aka Fetch and Execute Cycle
- aka Fetch-Decode-Execute Cycle (FDX)
- Retrieve instruction from memory, determine actions required and carry them out
- Equates to roughly 1/3 of a nanosecond

“Latency Numbers Everyone Should Know”

```
L1 cache reference ......................... 0.5 ns
Branch mispredict ............................ 5 ns
L2 cache reference ........................... 7 ns
Mutex lock/unlock ........................... 25 ns
Main memory reference ...................... 100 ns
Compress 1K bytes with Zippy ............. 3,000 ns  =   3 µs
Send 2K bytes over 1 Gbps network ....... 20,000 ns  =  20 µs
SSD random read ........................ 150,000 ns  = 150 µs
Read 1 MB sequentially from memory ..... 250,000 ns  = 250 µs
Round trip within same datacenter ...... 500,000 ns  = 0.5 ms
Read 1 MB sequentially from SSD* ..... 1,000,000 ns  =   1 ms
Disk seek ........................... 10,000,000 ns  =  10 ms
Read 1 MB sequentially from disk .... 20,000,000 ns  =  20 ms
Send packet CA->Netherlands->CA .... 150,000,000 ns  = 150 ms
```

- Shamelessly cribbed from this gist: https://gist.github.com/2843375

Measured Cache Latencies

Sandy Bridge-E	L1d	L2	L3	Main
Sequential Access	3 clk	11 clk	14 clk	6ns
Full Random Access	3 clk	11 clk	38 clk	65.8ns

SI Software's benchmarks: http://www.sisoftware.net/?d=qa&f=ben_mem_latency

Registers

- On-core for instructions being executed and their operands
- Can be accessed in a single cycle
- There are many different types
- A 64-bit Intel Nehalem CPU had 128 Integer & 128 floating point registers

Load/Store Buffers

- Important for holding instructions for Out of Order (OoO) execution
- Can be snooped by other cores to preserve program order
- ~1 cycle

Cache Associativity

- Replacement policy determines where in the cache an entry of main memory will go
- Fully Associative: Put it anywhere
- Somewhere in the middle: 2-4 way set/ skewed associative
- Direct Mapped: Each entry can only go in one specific place

Static RAM (SRAM)

- Requires 6-8 pieces of circuitry per datum, cannot be dense
- Runs at a cycle rate, not quite measurable in time
- Uses a relatively large amount of power for what it does
- Data does not fade or leak, does not need to be refreshed/recharged

L0 (zero)

- New to Sandy Bridge
- A cache of the last 1536 uops (~6kB) decoded
- Well-suited for hot loops
- Not the same as the older "trace" cache

L1

- Divided into data and instructions
- 32K data, 32K instructions per core on a Sandy Bridge
- Sandy Bridge loads data at 256 bits per cycle, double that of Nehalem
- 3-4 cycles to get to L1d

L2

- 256K per core on a Sandy Bridge
- 2MB per "module" on AMD's Bulldozer architecture
- 11 cycles to reach
- Unified data and instruction caches from here up
- If the working set size is larger than L2, misses grow

L3

- Was a unified cache up until Sandy Bridge, shared between cores
- Became known as the Last Level Cache (LLC)
- Where concurrency takes place
- Since no longer unified, any core could access data from any of the LLCs (until Sandy Bridge)
- Varies in size with different processors and versions of an architecture
- 14-38 cycles to reach

Exclusive versus Inclusive

- Only relevant below L3
- AMD is exclusive
  - Progressively more costly due to eviction
  - Can hold more data
  - Bulldozer uses "write through" from L1d back to L2
- Intel is inclusive
  - Can be better for inter-processor memory sharing

Intra-Socket Communication

- Sandy Bridge introduced a ring architecture so that components inside of a CPU can communicate directly
- L3 is no longer unified, each core has a region
- The ring architecture in the Sandy Bridge has 4 rings in it, for data, request, ACK and snooping
- The ring can select the shortest path between components for speed

Inter-Socket Communication

- Uses Quick Path Interconnect links (QPI, Intel) or HyperTransport (AMD) for:
  - cache coherency across sockets
  - snooping
- QPI is proprietary to Intel. Maximum speed of 8 GT/s
- Each QPI message takes ~20NS
- HyperTransport was developed by a consortium of AMD, NVidia, Apple, etc. Maximum speed of 6.4 GT/s (that I know of)
- Both are DDR and point to point interfaces to other CPUs in a NUMA setup
- Both transfer 16 bits per transmission in practice, but Sandy Bridge is really 32

MESI+F Cache Coherency Protocol

- Specific to data cache lines
- Request for Ownership (RFO), when a processor tries to write to a cache line
- Modified, the local processor has changed the cache line, implies only one who has it
- Exclusive, only one processor is using the cache line, not modified
- Shared, multiple processors are using the cache line, not modified
- Invalid, the cache line is invalid, must be refetched
- Forward, to another socket via QPI
- All processors MUST acknowledge a message for it to be valid

Dynamic RAM (DRAM)

- Very dense, only 2 pieces of circuitry per datum
- DRAM "leaks" its charge, but not sooner than 64 milliseconds
- Every read depletes the charge, requires a subsequent recharge
- Memory Controllers can "refresh" DRAM by sending a charge through the entire device
- Takes 240 cycles (100 NS) to retrieve from here
- Intel's Nehalem architecture - each CPU socket controls a portion of RAM, no other socket has direct access to it

DDR3 SDRAM

- Double Data Rate, Synchronous Dynamic
- Has a high-bandwidth three-channel interface
- Also reduces power consumption over DDR2 by 30%
- Data is transferred on the rising and falling edges of a 400-1066 MHz I/O clock of the system

Striding & Pre-fetching

- Predictable memory access is really important
- Hardware prefetcher on the core looks for patterns of memory access
- Can be counter-productive if the access pattern is not predictable

Cache Misses

- Cost hunderds of cycles
- Keep your code simple
- Instruction read misses are most expensive
- Data read miss are less so, but still hurt performance
- Write misses aren't so bad unless the policy is "write through"

Programming Optimizations

- Stack allocated data is cheap
- Pointer interaction - you have to retrieve data being pointed to, even in registers
- Avoid locking and resultant kernel arbitration
- CAS is better and all on-thread, but algorithms become more complex
- Match workload to the size of the last level cache (LLC, L3)

What about Functional Programming?

- Have to allocate more and more space for your data structures, leads to eviction
- When you cycle back around, you get cache misses
- Choose immutability by default, profile to find poor performance
- Use mutable data in targeted locations

Hyperthreading

- Great for I/O-bound applications
- If you have lots of cache misses
- Doesn't do much for CPU-bound applications
- You have half of the cache resources per core

Data Structures

- **BAD**: Linked list structures and tree structures
- **BAD**: Java's HashMap uses chained buckets!
- **BAD**: Standard Java collections generate lots of garbage
- **GOOD**: Array-based and contiguous in memory is much faster
- **GOOD**: Write your own that are lock-free and contiguous
- **GOOD**: Fastutil library, but note that it's additive

Application Memory Wall & GC

- Tremendous amounts of RAM at low cost
- GC will kill you with compaction
- Use pauseless GC
  - IBM's Metronome, very predictable
  - Azul's C4, very performant

Using GPUs

- Locality matters!
- Need to be able to export a task with data that does not need to update

ManyCore

- David Ungar says > 24 cores, generally many 10s of cores
- Really trying to think about it with 1000 or more
- Cache coherency won't be possible
- Non-deterministic

Memristor

- Non-volatile, static RAM, same write endurance as Flash
- 200-300 MB on chip
- Sub-nanosecond writes
- Able to perform processing?
- Multistate, not binary

Phase Change Memory (PRAM)

- Higher performance than today's DRAM
- Intel seems more fascinated by this, just released its "neuromorphic" chip design
- Not able to perform processing
- Write degradation is supposedly much slower
- Was considered susceptible to unintentional change, maybe fixed?

Credits!

- What Every Programmer Should Know About Memory, Ulrich Drepper of RedHat, 2007
- Java Performance, Charlie Hunt
- Wikipedia/Wikimedia Commons
- AnandTech
- The Microarchitecture of AMD, Intel and VIA CPUs
- Everything You Need to Know about the Quick Path Interconnect, Gabriel Torres/Hardware Secrets
- Inside the Sandy Bridge Architecture, Gabriel Torres/Hardware Secrets
- Martin Thompson's Mechanical Sympathy blog and Disruptor presentations
- The Application Memory Wall, Gil Tene, CTO of Azul Systems
- AMD Bulldozer/Intel Sandy Bridge Comparison, Gionatan Danti
- SI Software's Memory Latency Benchmarks
- Martin Thompson and Cliff Click provided feedback &additional content