Multiprocessor OS

COMP9242 – Advanced Operating Systems
Ihor Kuz (Kry10, UNSW) Ihor.kuz@unsw.edu.au
2022 T2 Week 09

Overview

Multiprocessor OS (Background and Review)

- How does it work? (Background)
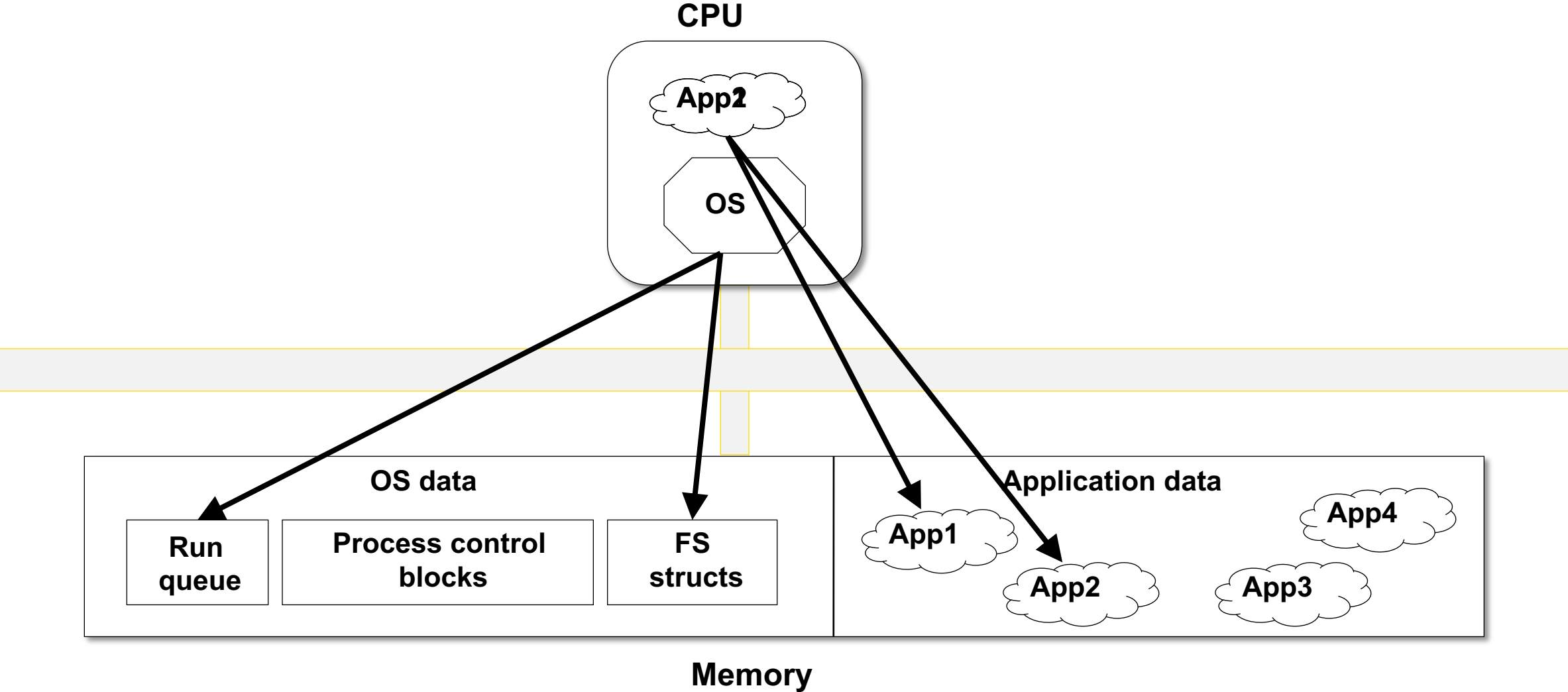- Scalability (Review)

Multiprocessor Hardware

- Contemporary systems (Intel, AMD, ARM, Oracle/Sun)
- Experimental (Intel, MS, Polaris)

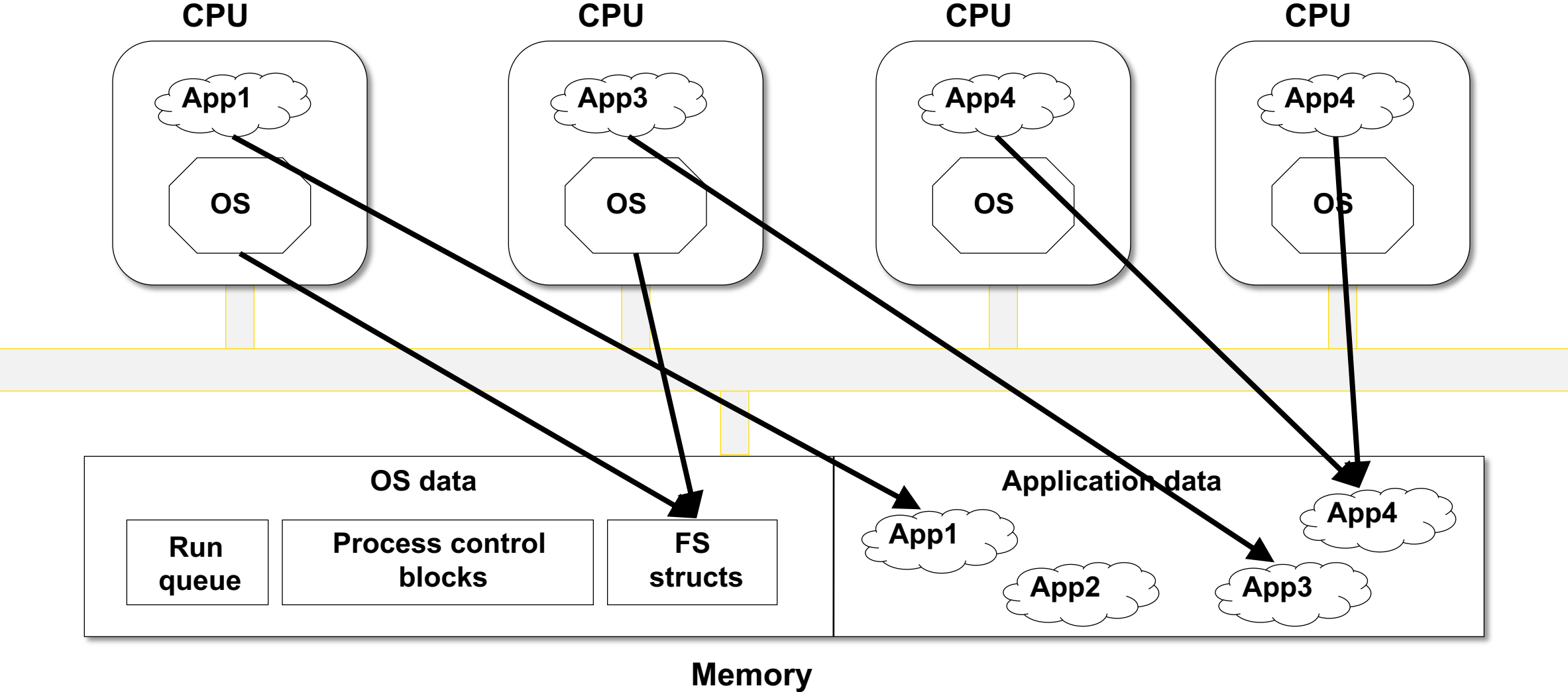## OS Design for Multiprocessors

- Guidelines
- Design approaches
  - Divide and Conquer (Disco, Tesselation)
  - Reduce Sharing (K42, Corey, Linux, FlexSC, scalable commutativity)
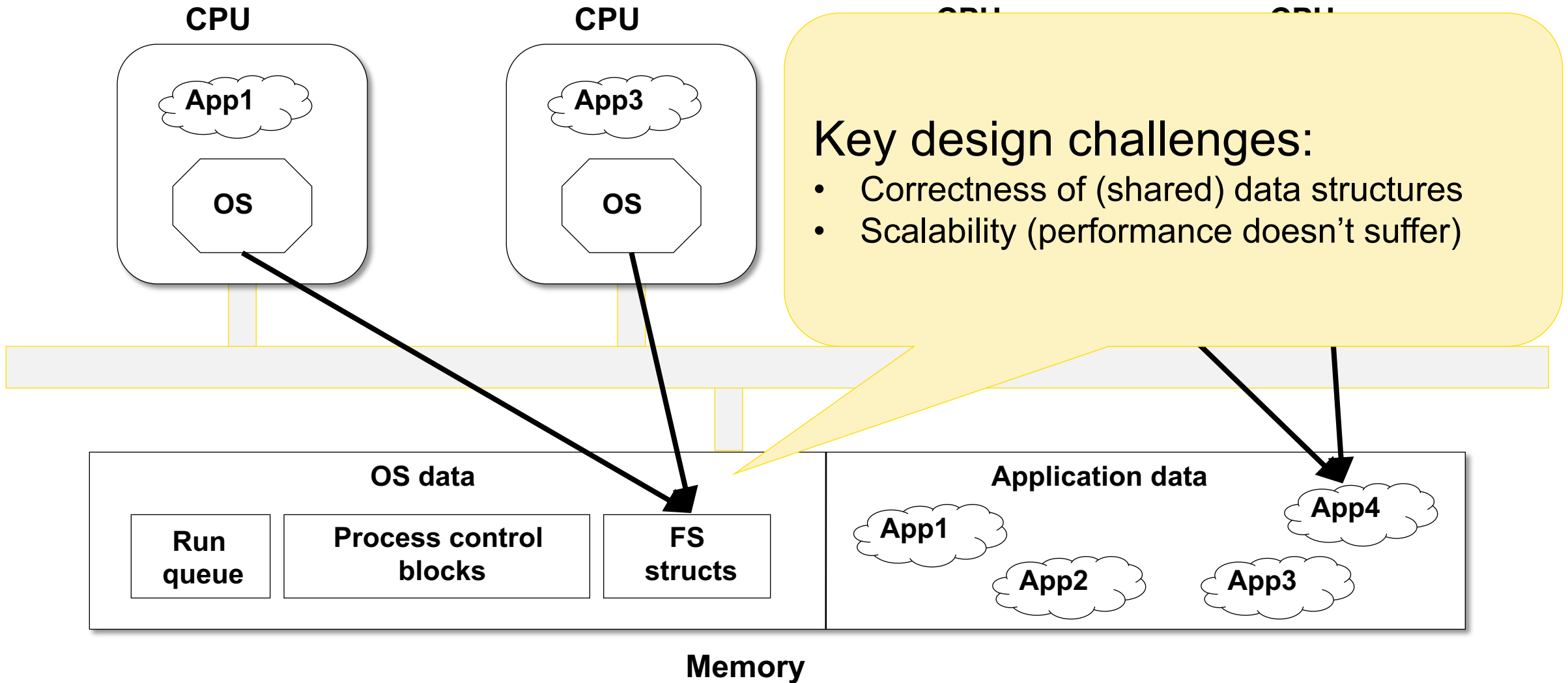 - No Sharing (Barrelfish, fos)

Multiprocessor OS

Uniprocessor OS

CPU

App2

OS

OS data

Run queue | **Process control blocks** | **FS structs**

Application data

App1

App2

App4

App3

Memory

Multiprocessor OS

Multiprocessor OS

CPU

App1

OS

CPU

App3

OS

Key design challenges:
- Correctness of (shared) data structures
- Scalability (performance doesn't suffer)

OS data

| Run queue | Process control blocks | FS structs |

Application data

App1

App2

App4

App3

Memory

Correctness of Shared Data

Concurrency control

- Locks
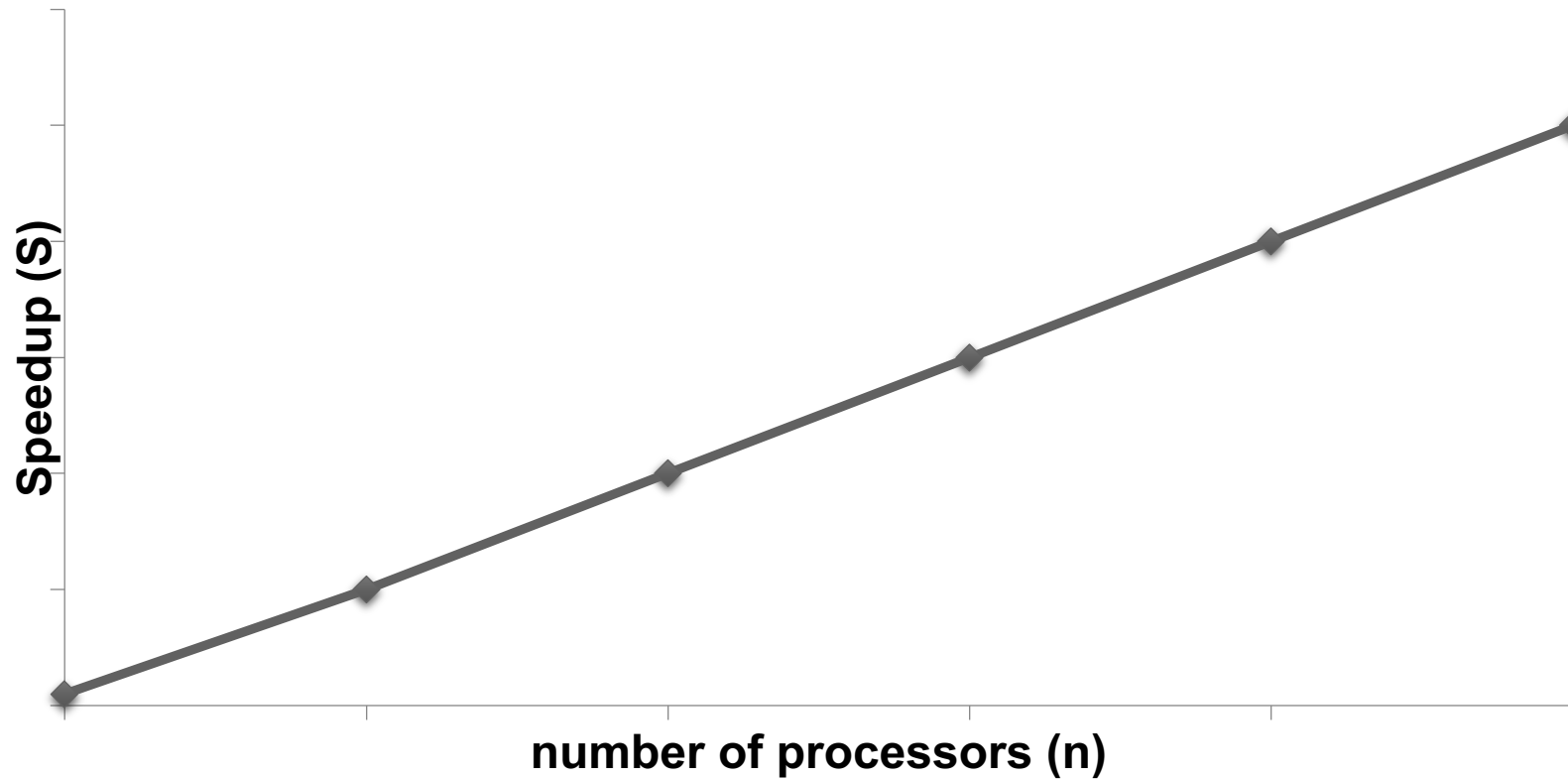- Semaphores
- Transactions
- Lock-free data structures

We know how to do this:

- In the application
- In the OS

Scalability

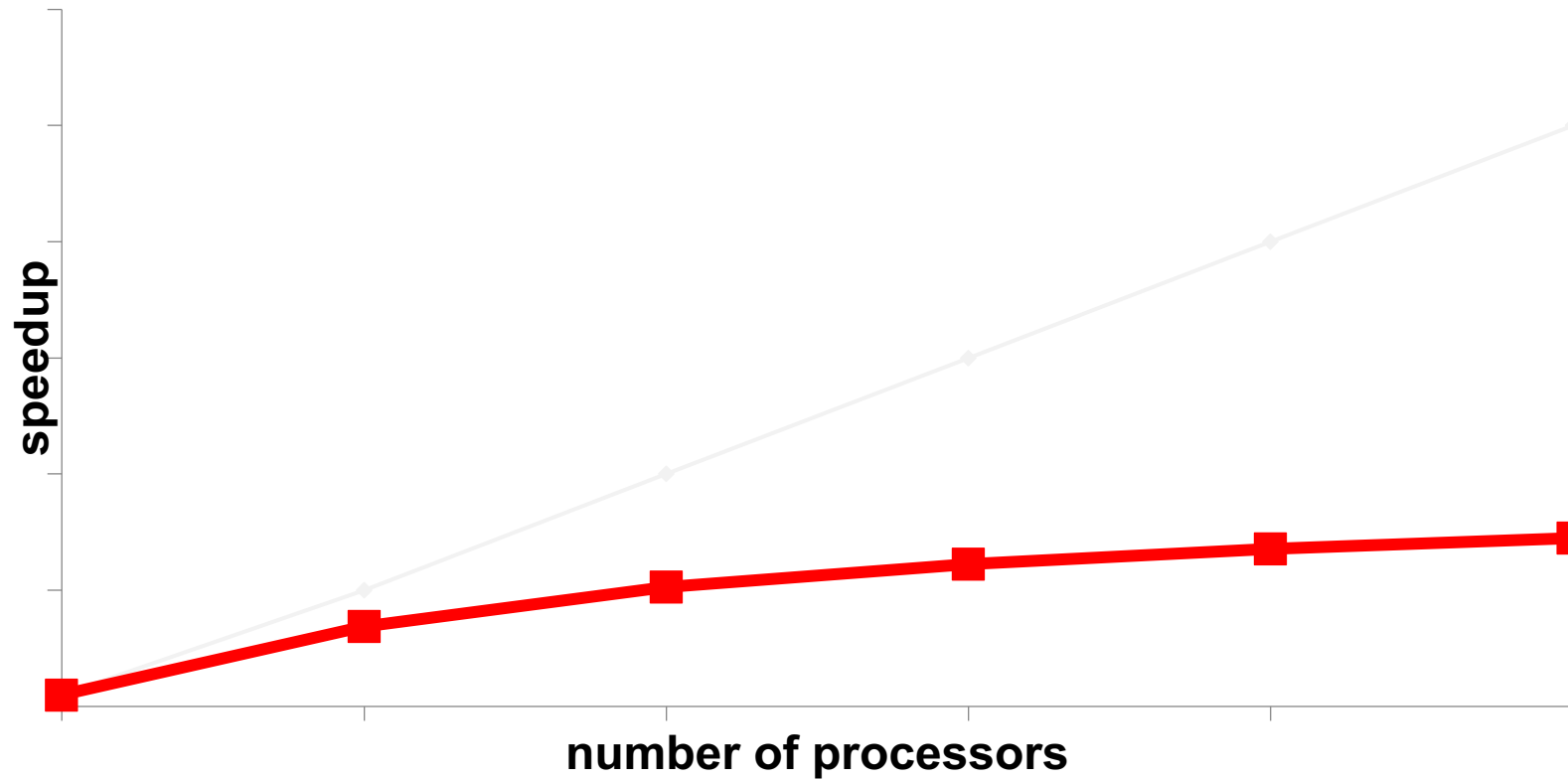Speedup as more processors added

Ideal

$$S(N) = \frac{T_1}{T_N}$$

(y-axis) **Speedup (S)**
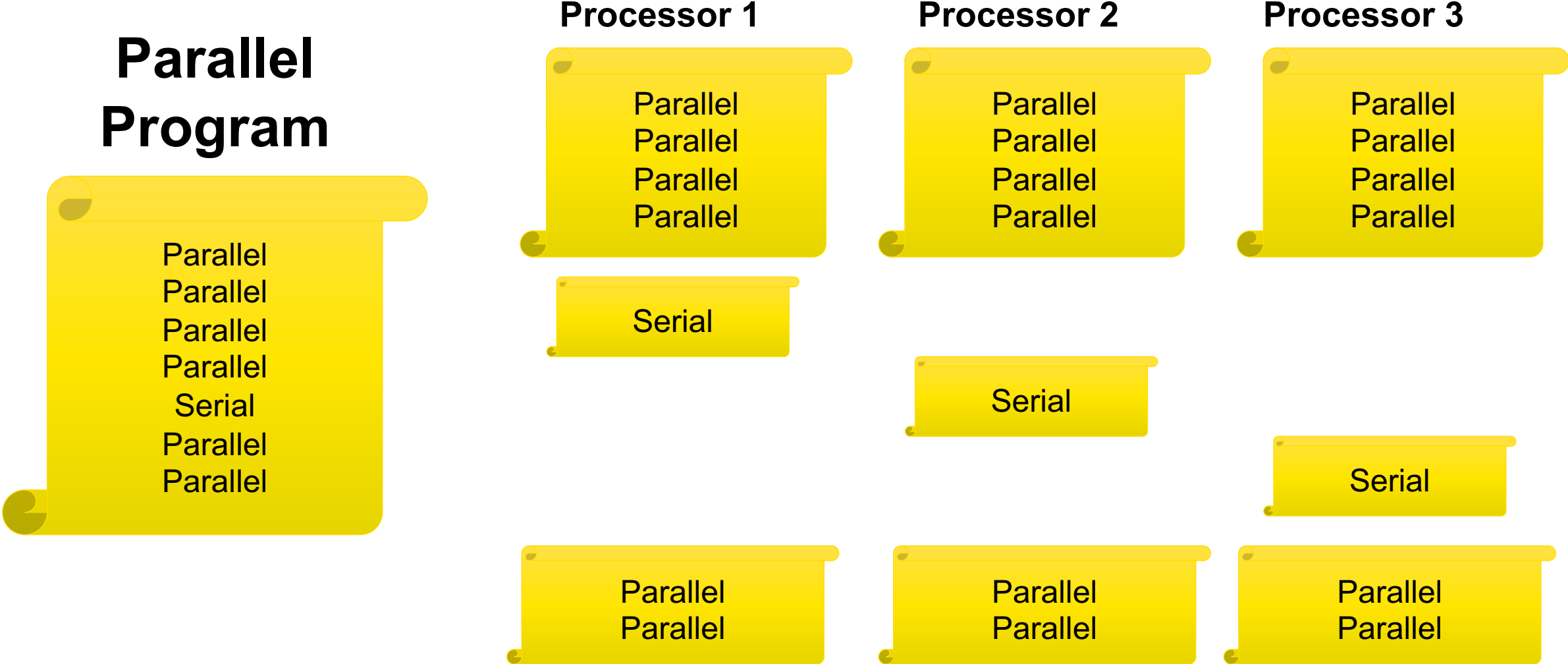
(x-axis) **number of processors (n)**
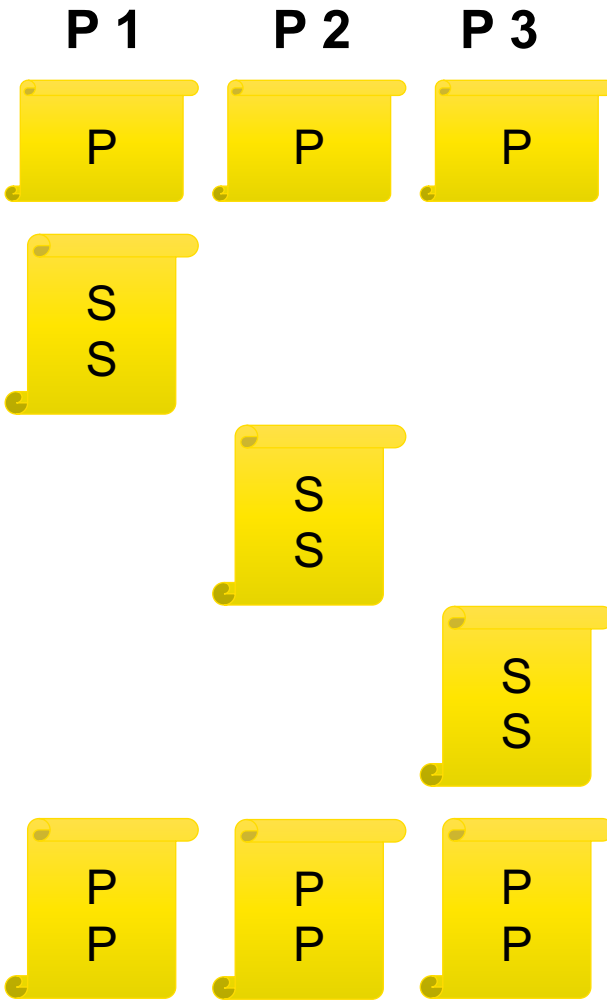
Scalability

Speedup as more processors added

Reality

$$S(N) = \frac{T_1}{T_N}$$

Scalability and Serialisation

Parallel Program

Parallel
Parallel
Parallel
Parallel
Serial
Parallel
Parallel

Processor 1

Parallel
Parallel
Parallel
Parallel

Serial

Parallel
Parallel

Processor 2

Parallel
Parallel
Parallel
Parallel

Serial

Parallel
Parallel

Processor 3

Parallel
Parallel
Parallel
Parallel

Serial

Parallel
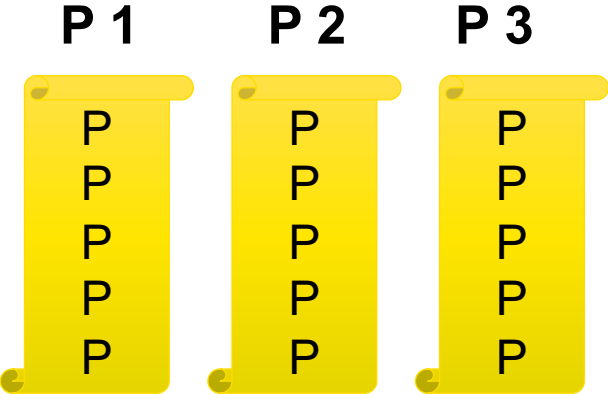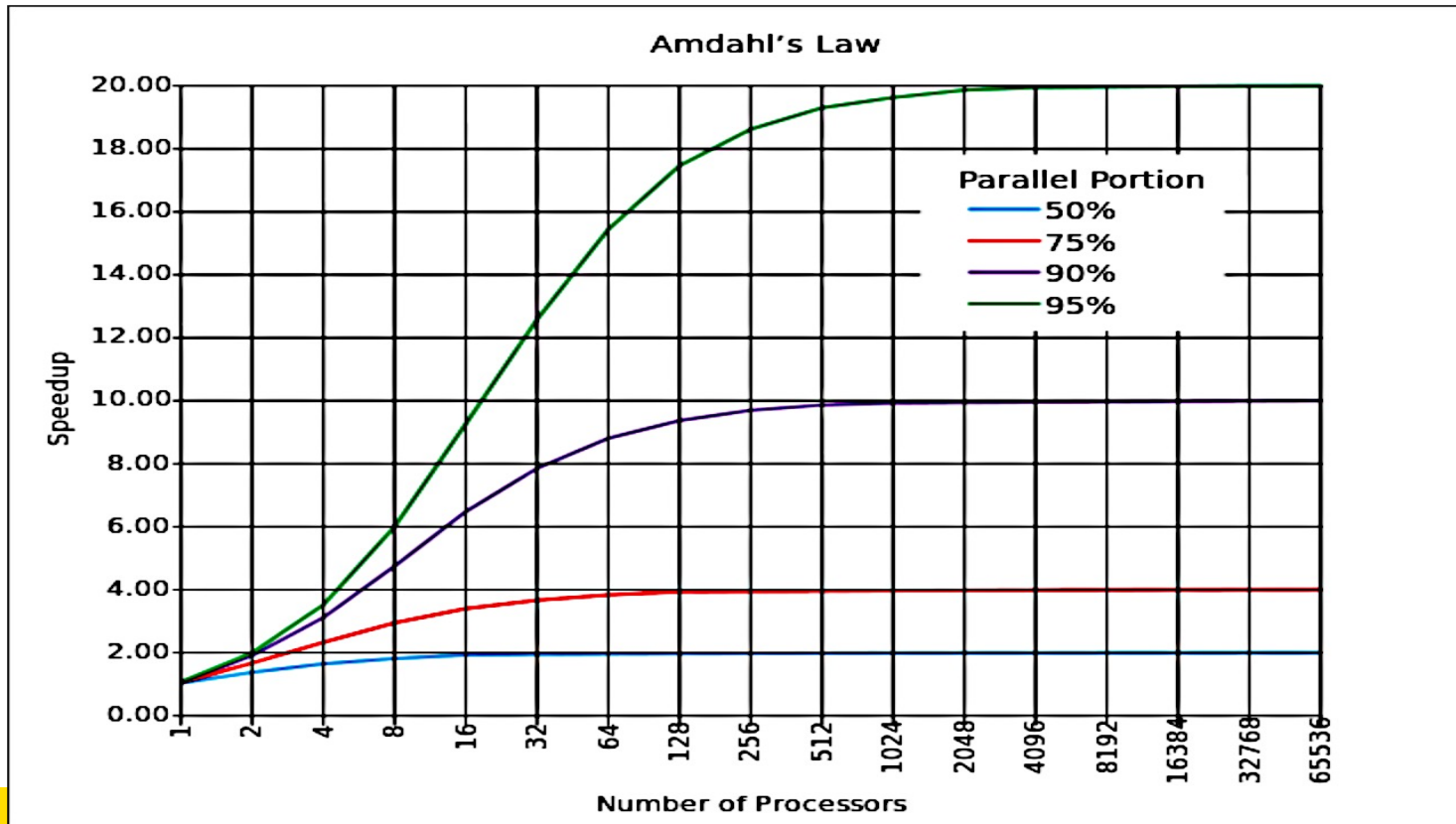Parallel

Scalability and Serialisation
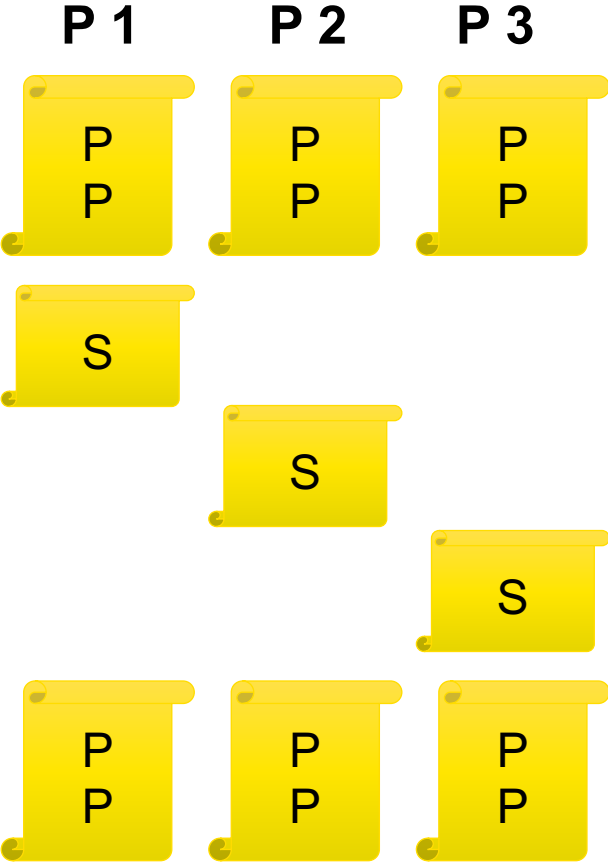
Scalability and Serialisation

Remember Amdahl's law

- Serial (non-parallel) portion: when application not running on all cores
- Serialisation prevents scalability

$$T_1 = 1 = (1 - P) + P$$

$$T_N = (1 - P) + \frac{P}{N}$$

$$S(N) = \frac{T_1}{T_N} = \frac{1}{(1 - P) + \frac{P}{N}}$$

$$S(\infty) \rightarrow \frac{1}{(1 - P)}$$

From http://en.wikipedia.org/wiki/File:AmdahlsLaw.svg

Serialisation

Where does serialisation show up?

- Application (e.g. access shared app data)
- OS (e.g. performing syscall for app) How much time is spent in OS?

Sources of Serialisation

Locking (explicit serialisation)

- Waiting for a lock ➔ stalls self
- Lock implementation:
  - Atomic operations lock bus ➔ stalls everyone waiting for memory
  - Cache coherence traffic loads bus ➔ stalls others waiting for memory

Memory access (implicit)

- Relatively high latency to memory ➔ stalls self

Cache (implicit)

- Processor stalled while cache line is fetched or invalidated
- Affected by latency of interconnect
- Performance depends on data size (cache lines) and contention (number of cores)

More Cache-related Serialisation

False sharing

- Unrelated data structs share the same cache line
- Accessed from different processors
- ➔ Cache coherence traffic and delay

Cache line bouncing

- Shared R/W on many processors
- E.g: bouncing due to locks: each processor spinning on a lock brings it into its own cache
- ➔ Cache coherence traffic and delay

Cache misses

- Potentially direct memory access ➔ stalls self
- When does cache miss occur?
  - Application accesses data for the first time, Application runs on new core
  - Cached memory has been evicted
    - Cache footprint too big, another app ran, OS ran

Multiprocessor Hardware

Multi-What?

Terminology:
- core, die (chip), package (module, processor, CPU)

Multiprocessor, SMP
- >1 separate processors, connected by off-processor interconnect

Multithread, SMT
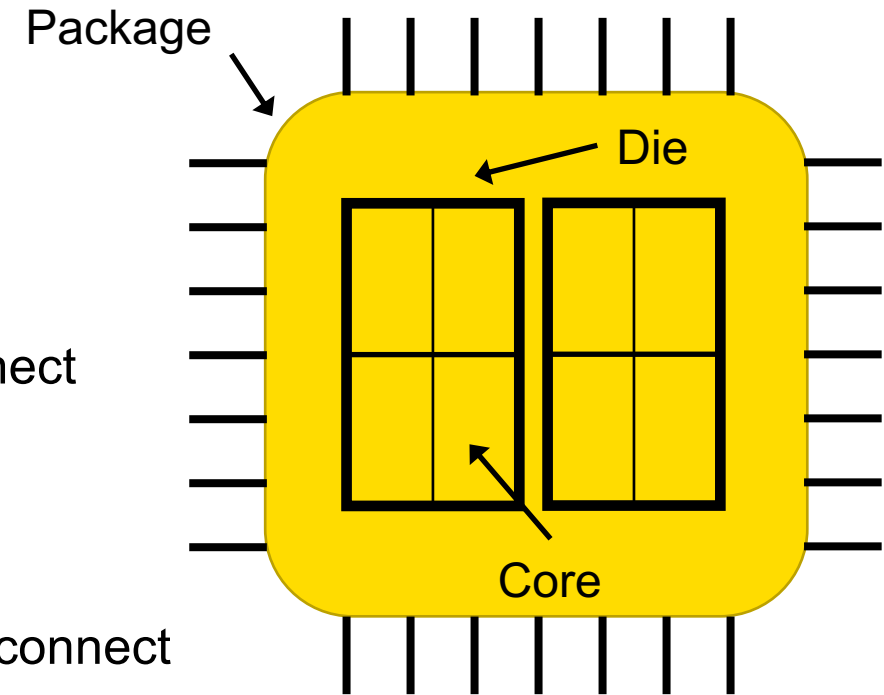- >1 hardware threads in a single processing core

Multicore, CMP
- >1 processing cores in a single die, connected by on-die interconnect

Multicore + Multiprocessor
- >1 multicore dies in a package (multi-chip module), on-processor interconnect
- >1 multicore processors, off-processor interconnect

Manycore
- Lots (>100) of cores

Package

Die

Core

Contemporary Multiprocessor Hardware

Intel:
- Nehalem, Westmere: 10 core, QPI
- Sandy Bridge, Ivy Bridge: 5 core, ring bus, integrated GPU, L3, IO
- Haswell (Broadwell): 18+ core, ring bus, transactional memory, slices (EP)
- Skylake (SP): mesh architecture

AMD:
- K10 (Opteron: Barcelona, Magny Cours): 12 core, Hypertransport
- Bulldozer, Piledriver, Steamroller (Opteron, FX)
  - 16 core, Clustered Multithread: module with 2 integer cores
- Zen: on die NUMA: CPU Complex (CCX) (4 core, private L3)
- Zen 2: chiplets (2xCCX) chiplets, IO die (incl mem controller)

Oracle (Sun) UltraSparc T1,T2,T3,T4,T5 (Niagara), M5,M7
- T5: 16 cores, 8 threads/core (2 simultaneous), crossbar, 8 sockets,
- M8: 32 core, 8 threads, on chip network, 8 sockets, 5GHz

ARM Cortex A9, A15 MPCore, big.LITTLE, DynamIQ
- 4 -8 cores, big.LITTLE: A7 + A15, dynamIQ: A75 + A55

Experimental/Non-mainstream Multiprocessor Hardware

Microsoft Beehive

- Ring bus, no cache coherence

Tilera (now Mellanox) Tile64, Tile-Gx

- 100 cores, mesh network

Intel Polaris

- 80 cores, mesh network

Intel SCC

- 48 cores, mesh network, no cache coherency

Intel MIC (Multi Integrated Core)

- Knight's Corner/Landing - Xeon Phi
- 60+ cores, ring bus/mesh

Interesting Properties of Multiprocessors

Scale and Structure

- How many cores and processors are there
- What kinds of cores and processors are there (homogeneous vs heterogeneous)

Memory Locality

- Where is the memory

Caches

- What is the cache architecture

Interconnect

- How are the cores and processors connected
- Access to IO, etc.

Interprocessor Communication

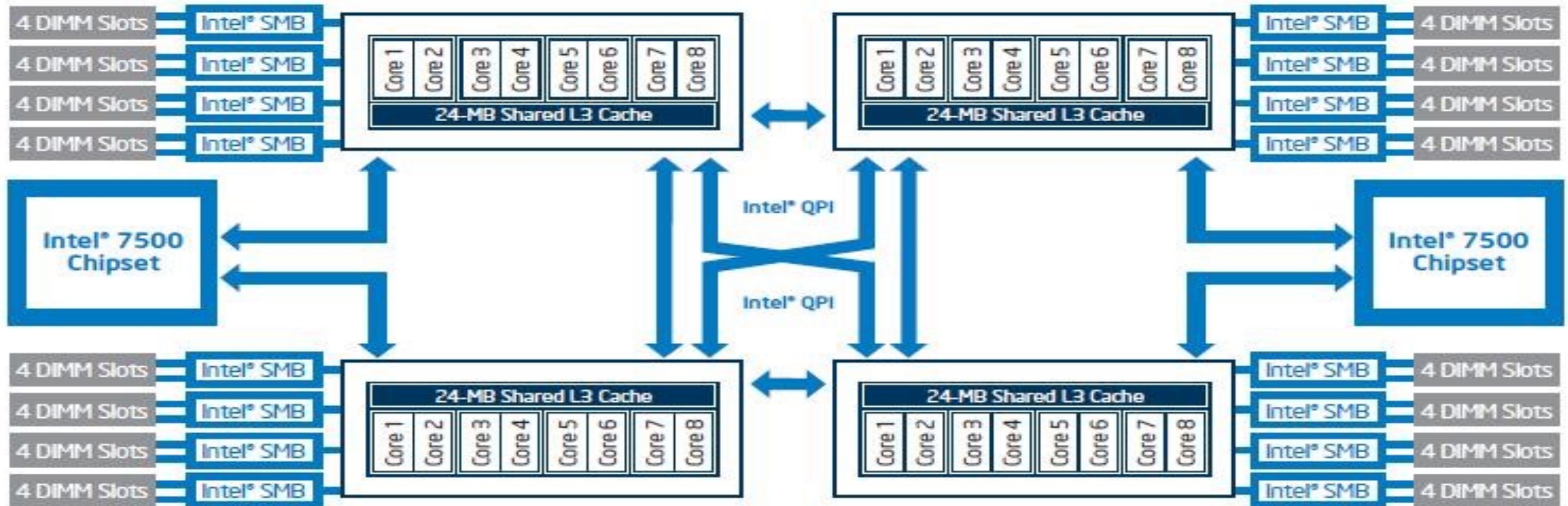- How do cores and processors send messages to each other

Scale and Structure
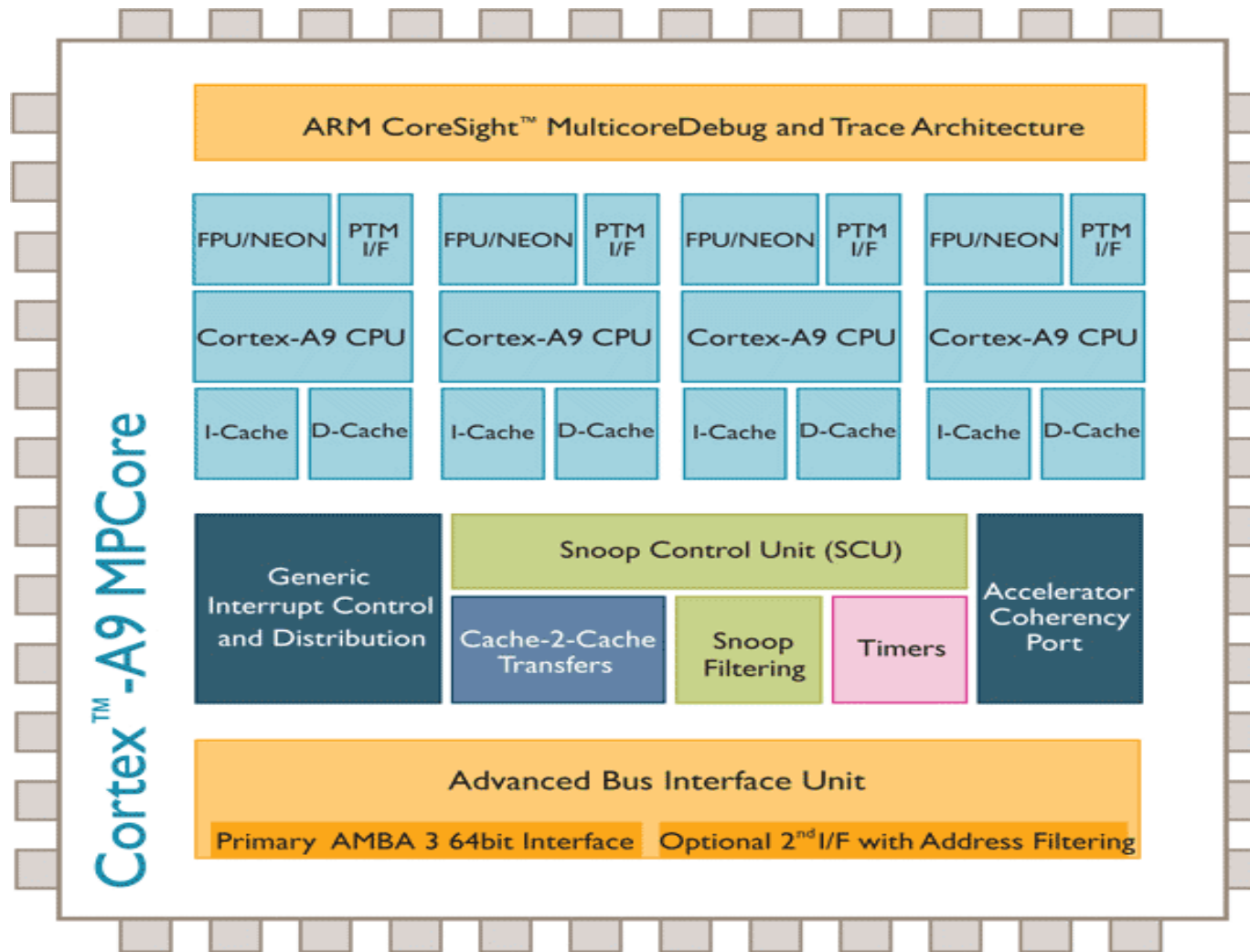
ARM Cortex A9 MPCore

- basic structure
- single die
- homogeneous cores

From http://www.arm.com/images/Cortex-A9-MP-core_Big.gif

Scale and Structure

Intel Nehalem – multiprocessor & multicore

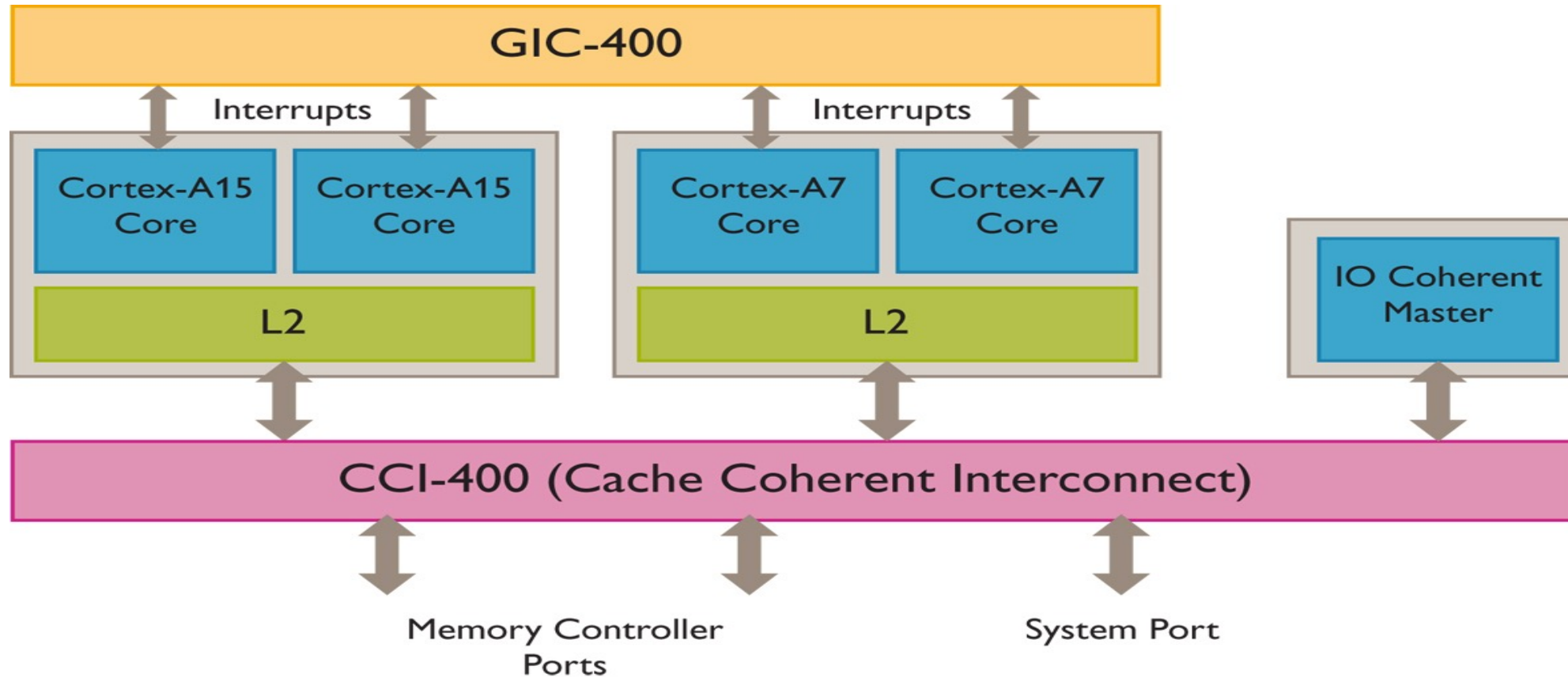From www.dawnofthered.net/wp-content/uploads/2011/02/Nehalem-EX-architecture-detailed.jpg

Scale and Structure

Tilera Tile64, Intel Polaris: manycore – lots of core
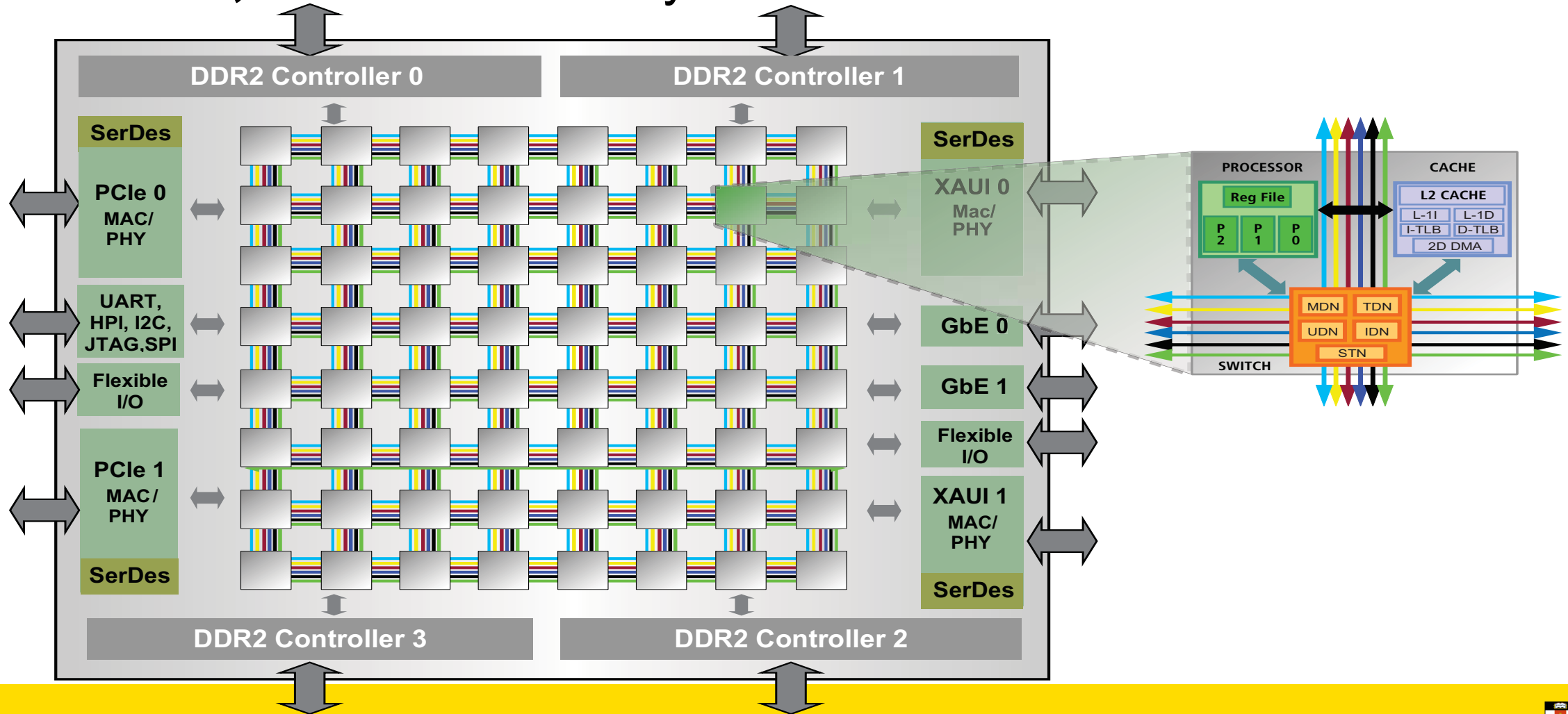
Scale and Structure

ARM big.LITTLE

From http://www.arm.com/images/Fig_1_Cortex-A15_CCI_Cortex-A7_System.jpg

Scale and Structure

Conventional big.LITTLE

Quad Cortex-A53

Octa Cortex-A53

DynamIQ big.LITTLE

1b+2L

1b+3L

1b+4L

1b+7L

Scale and Structure

NVIDIA Parker (Tegra X2) SOC

- Heterogeneity
- Application CPUs
- GPUs
- Management CPUs

From: https://elinux.org/Jetson_TX2
https://www.usenix.org/conference/osdi21/presentation/fri-keynote
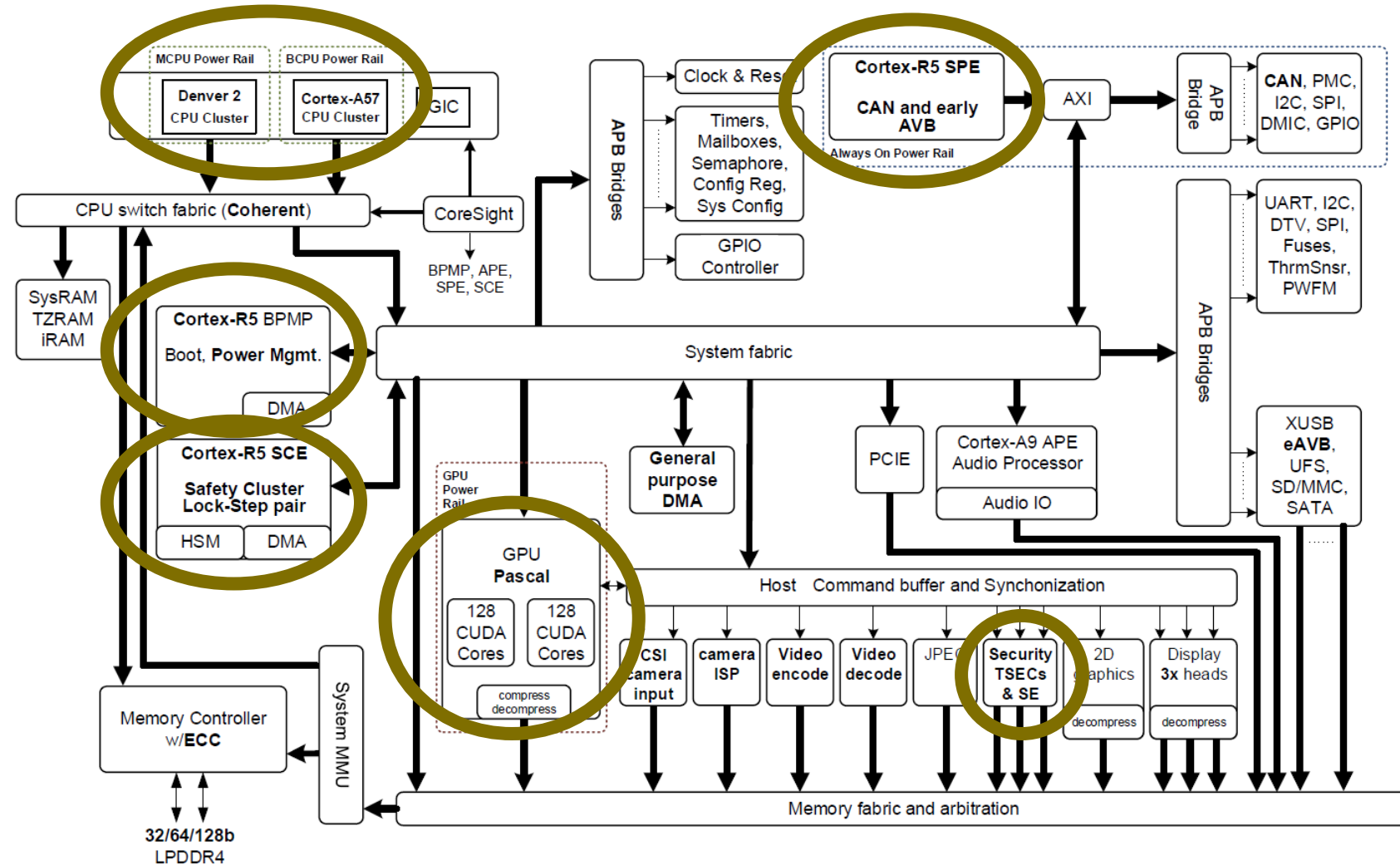
Memory Locality

NUMA (Non-Uniform Memory Access)

From www.dawnofthered.net/wp-content/uploads/2011/02/Nehalem-EX-architecture-detailed.jpg
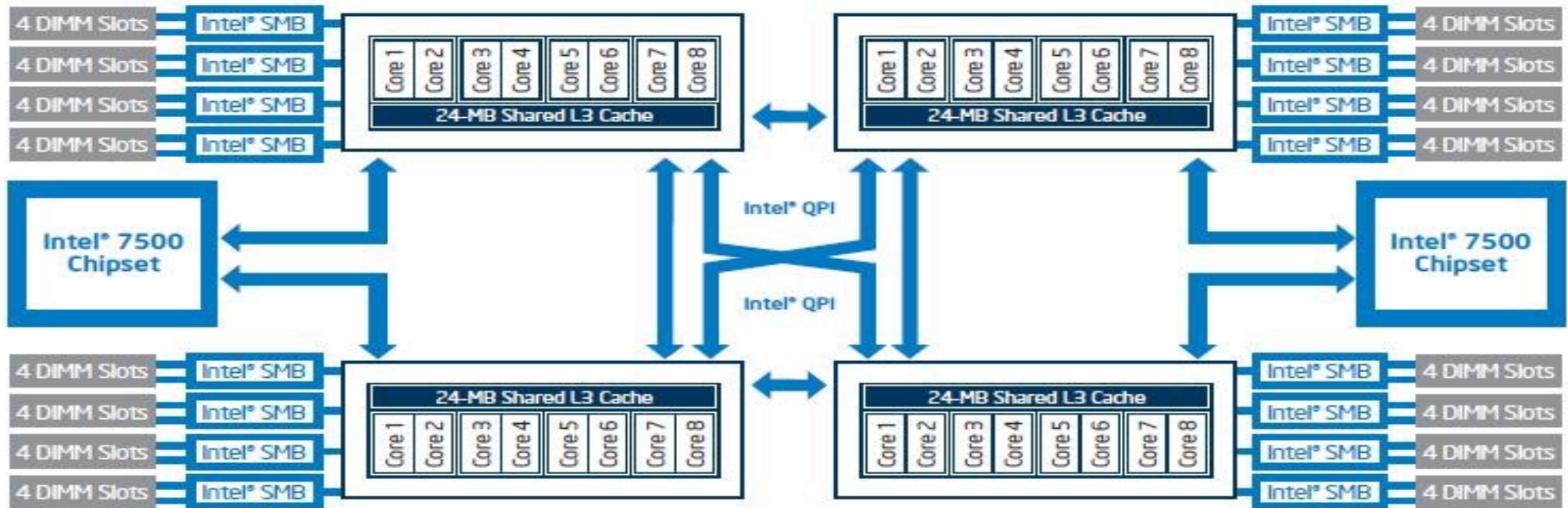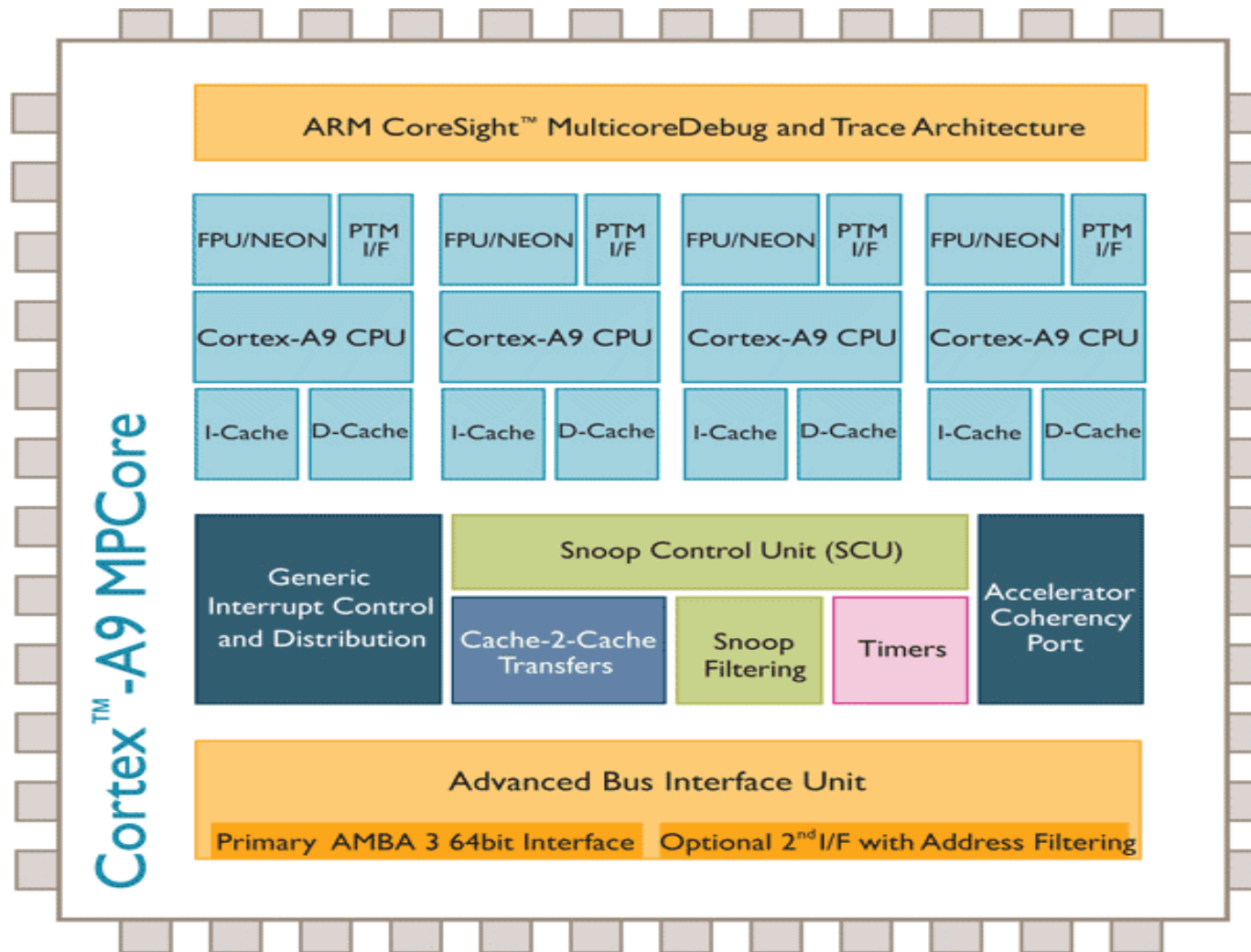
Cache

Hierarchy
- L1, L2, L3, …

Sharing
- Private – per core
- Shared – all/some cores
- Partitioned – distributed and shared

Coherence
- No inconsistent values in caches
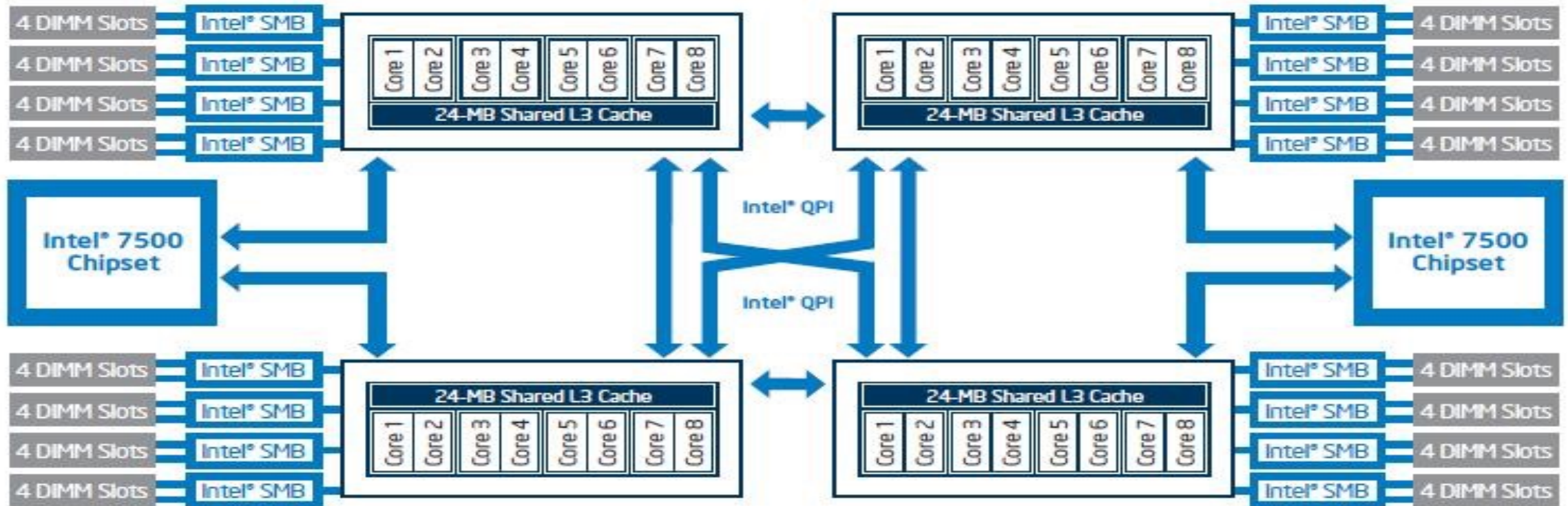- At same level, at different levels
- Snooping, directory-based

| CPU |
| L1-I | L1-D |
| L2 |
| ... |
| LLC |
| Main Memory |
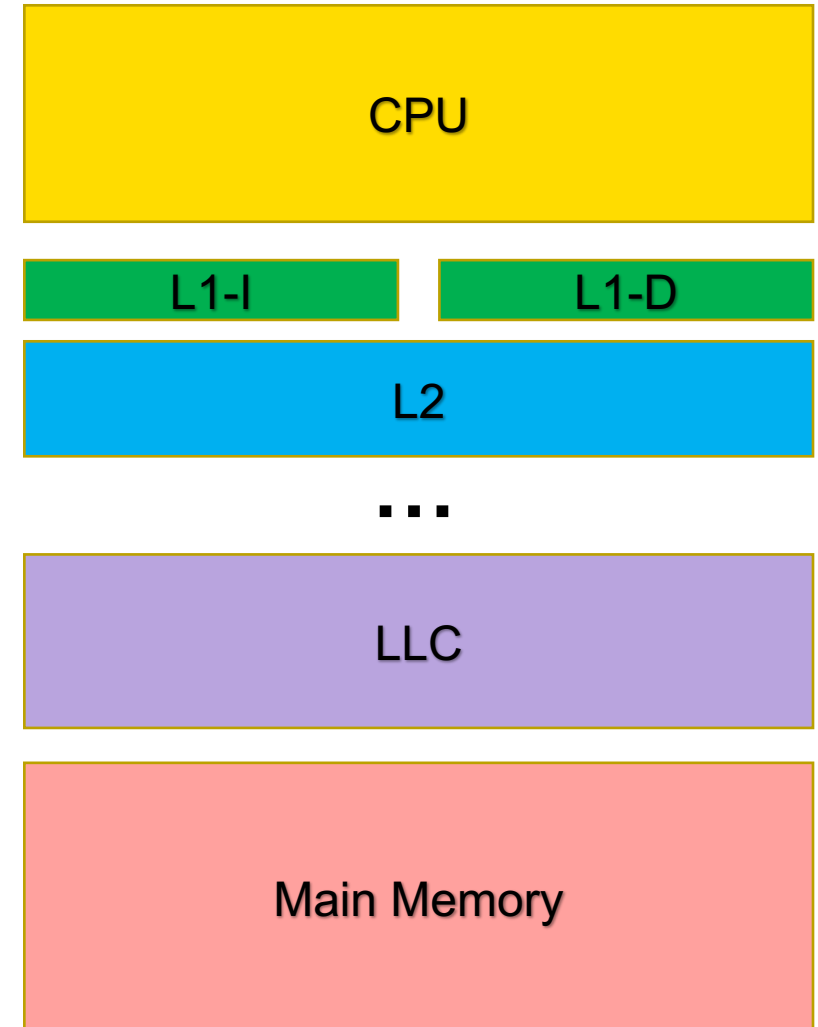
Cache

ARM Cortex A9 MPCore

- L1 – private, split, coherent, optimised MESI

- Optional L2 – shared

- DMA cache coherent with L1 (ACP)

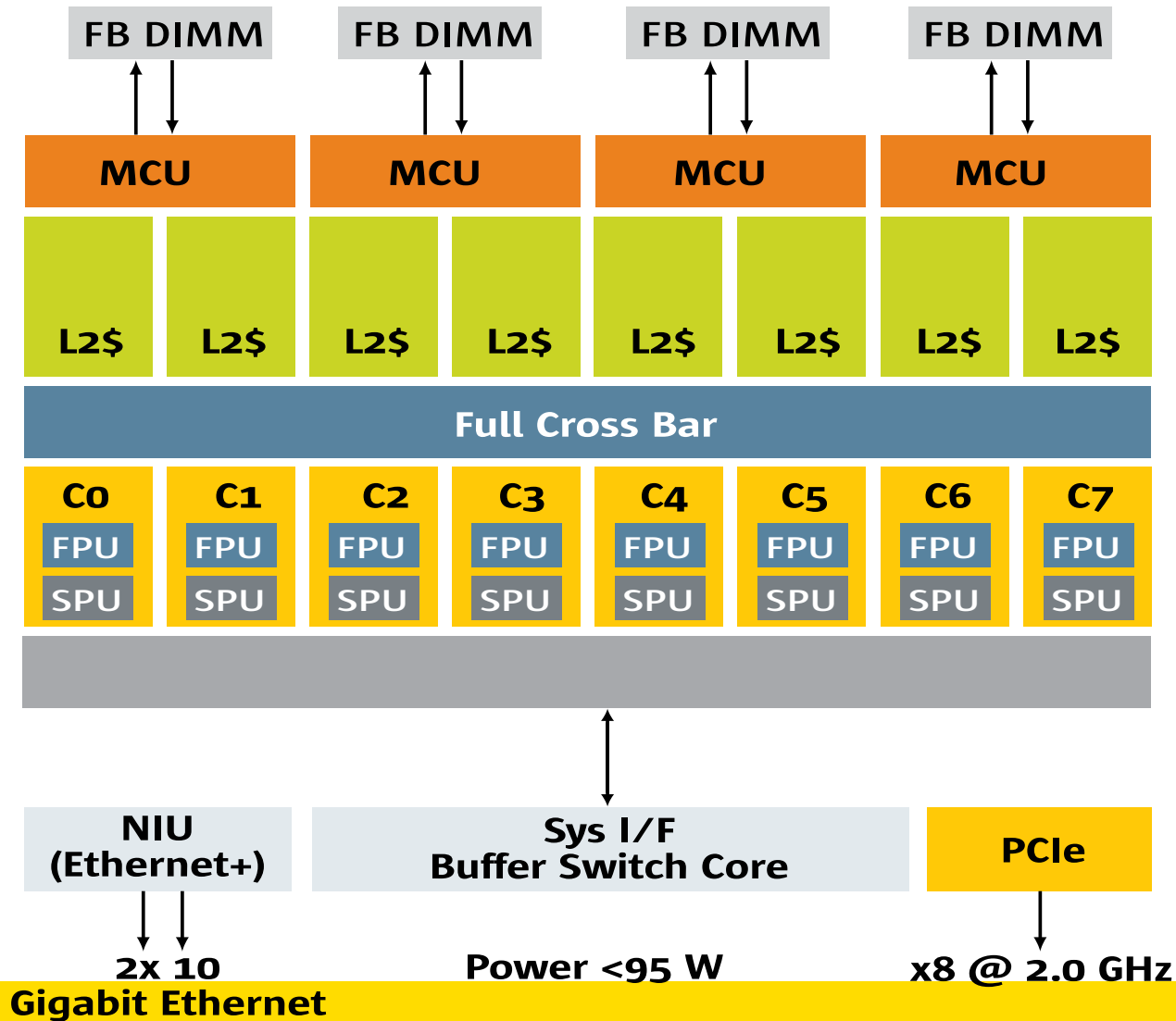From http://www.arm.com/images/Cortex-A9-MP-core_Big.gif

Cache

Core: L1, L2. Socket: L3. Cache coherent between sockets
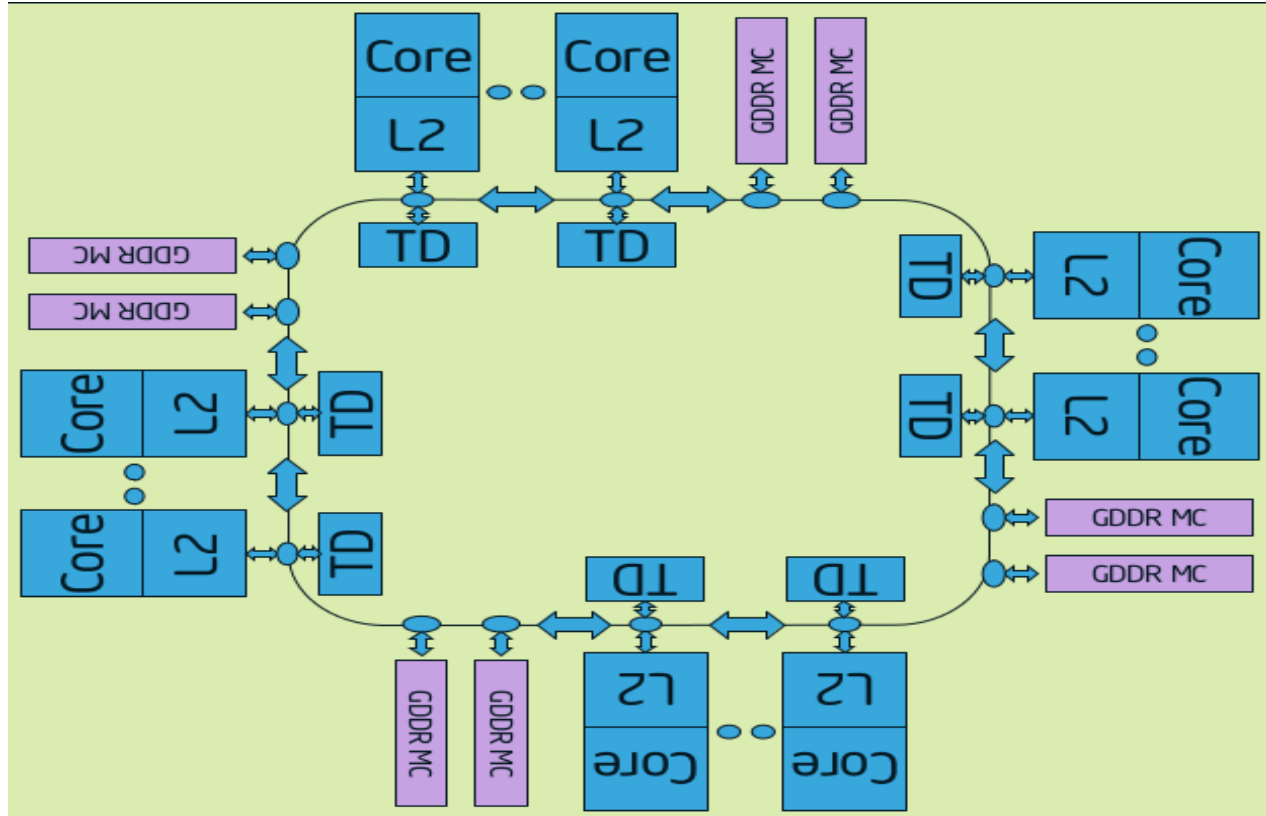
Cache

Oracle Sparc T2
(Niagara 2)

- private L1

- partitioned L2

- all cores equal
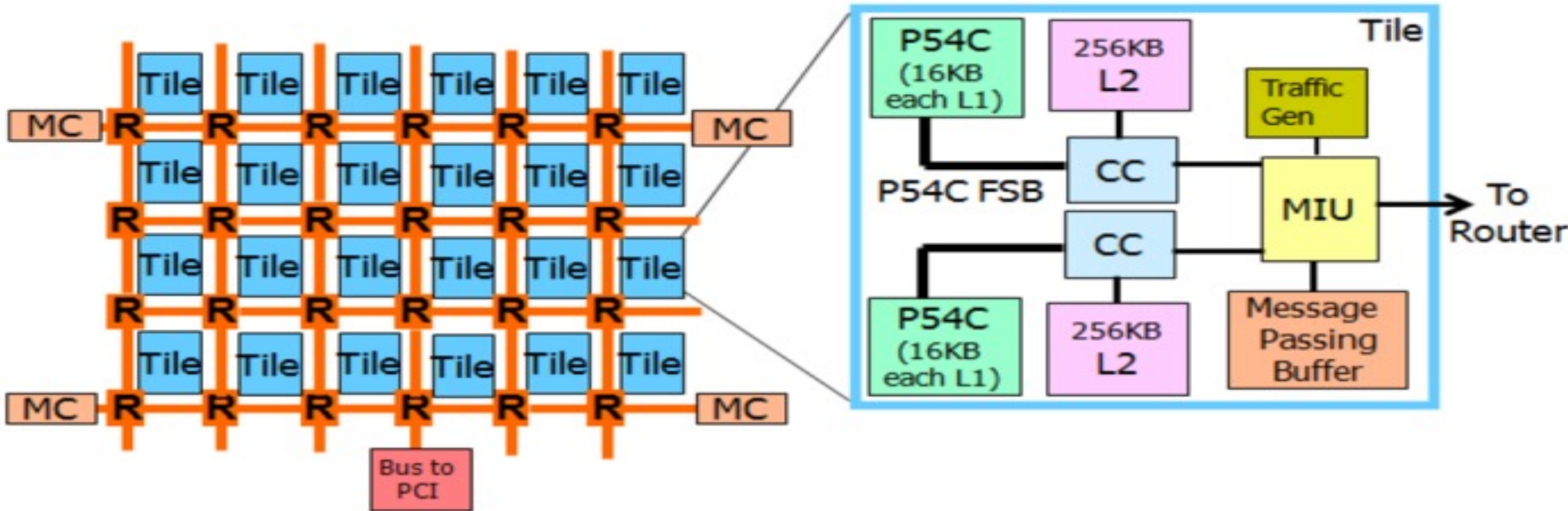access to L2s

From Sun/Oracle

Cache

Intel MIC (Multi Integrated Core) (Knight's Corner/Landing - Xeon Phi)

- Private L2

  - Tag Directory – info about addresses in other L2s

  - Send messages to other cores to access their L2

From http://semiaccurate.com/2012/08/28/intel-details-knights-corner-architecture-at-long-last/

Cache

Intel SCC – no hardware cache coherence

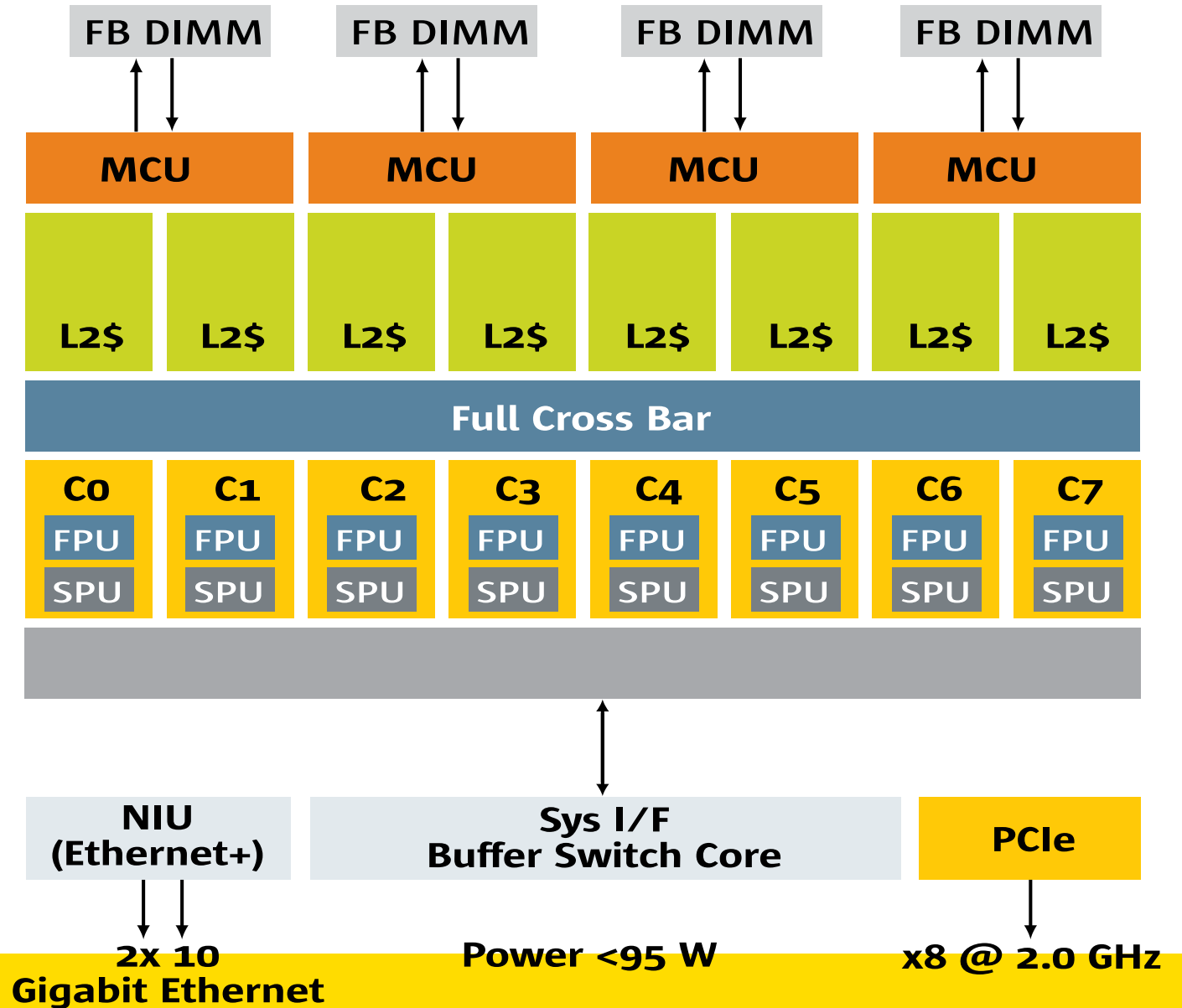From techresearch.intel.com/spaw2/uploads/files/SCC_Platform_Overview.pdf

Interconnect

Oracle Sparc T2

- Crossbar switch between cores L2

- Cores have independent access to L2

From Sun/Oracle

Interconnect

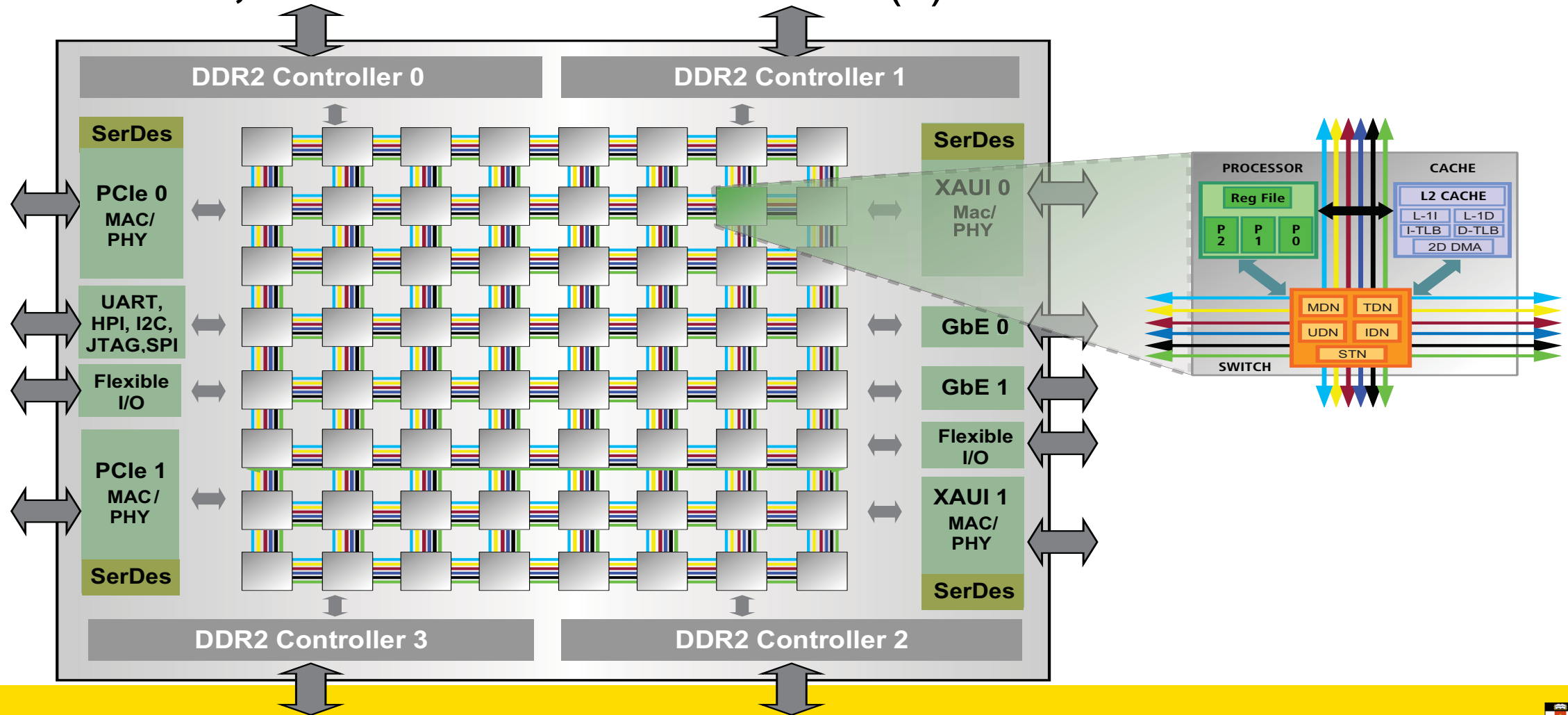
AMD Barcelona

From www.sigops.org/sosp/sosp09/slides/baumann-slides-sosp09.pdf
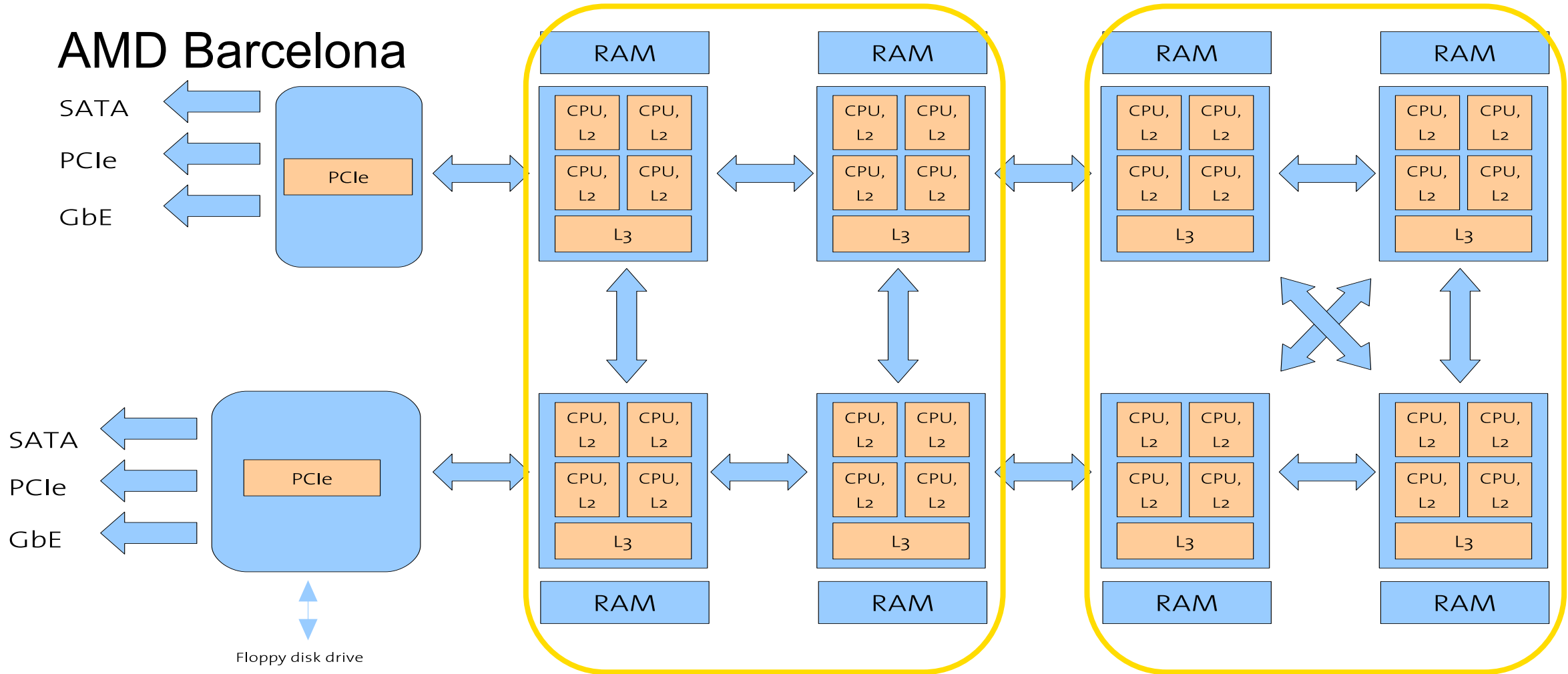
Interconnect (Latency)

Interconnect (Bandwidth)

From https://www.usenix.org/conference/atc15/technical-session/presentation/lepers

36 |
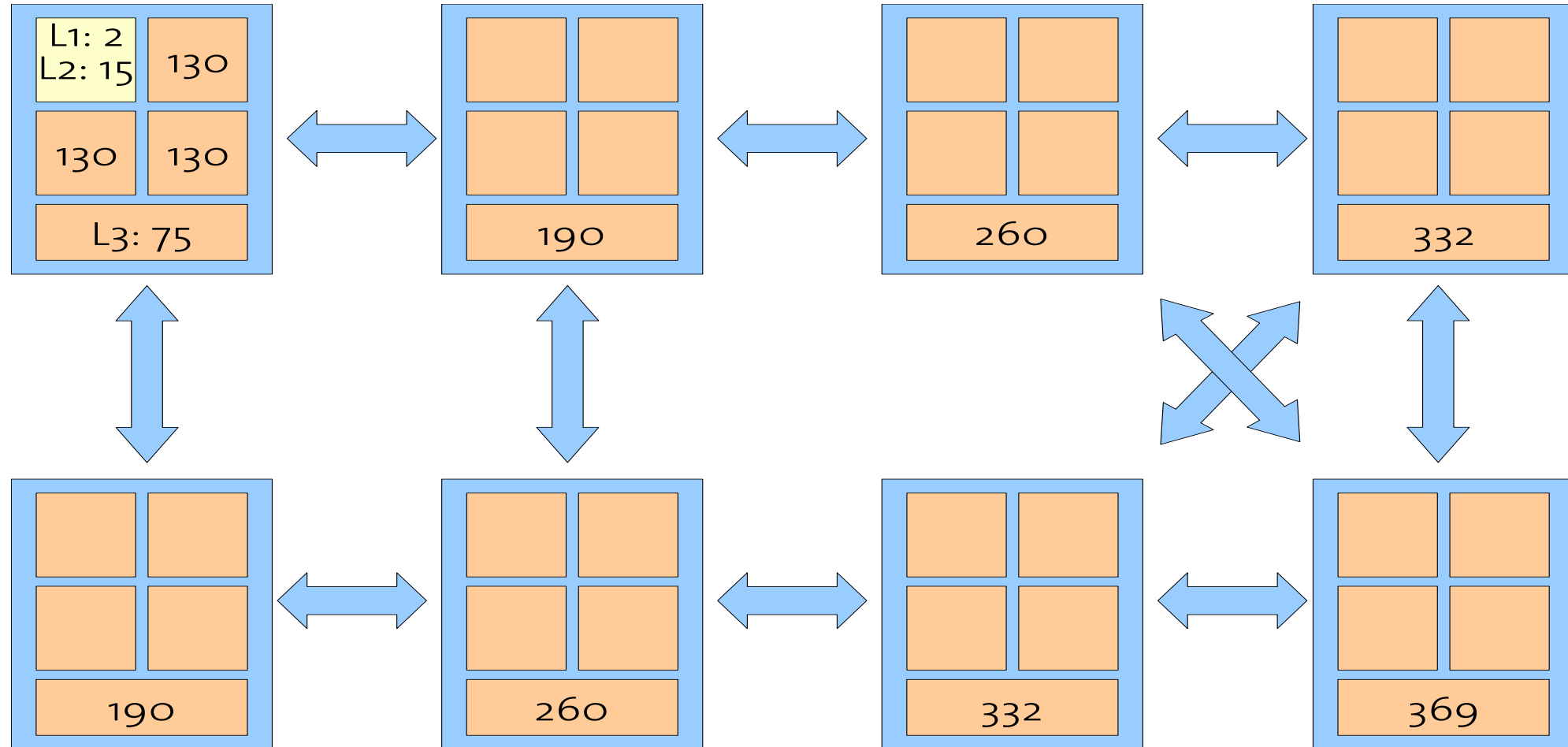
Interconnect
Tilera Tile64, Intel Polaris: Mesh network(s)

From www.tilera.com/products/processors/TILE64

Interconnect

Beehive

- Ring

- No hardware cache coherence

From projects.csail.mit.edu/beehive/BeehiveV5.pdf
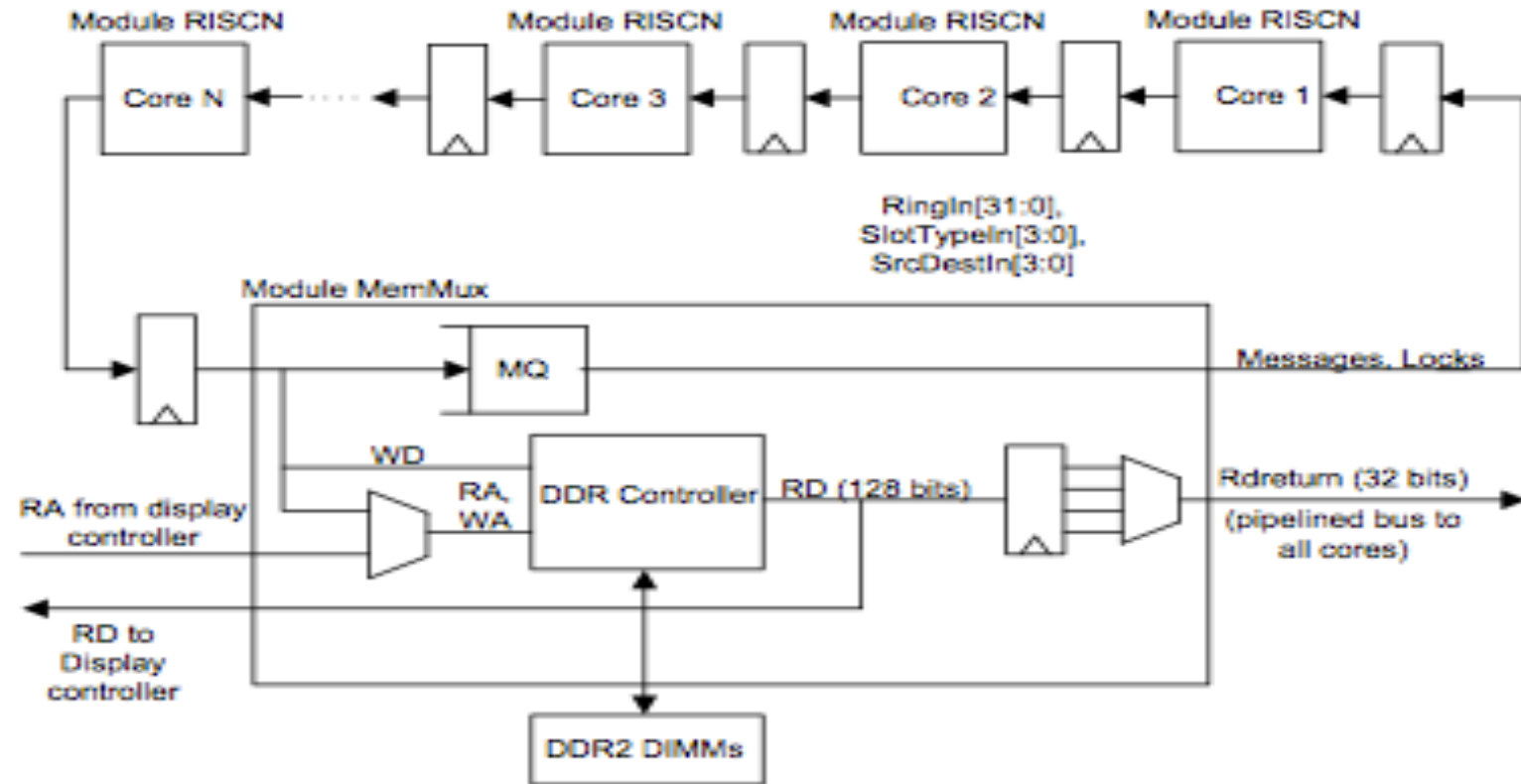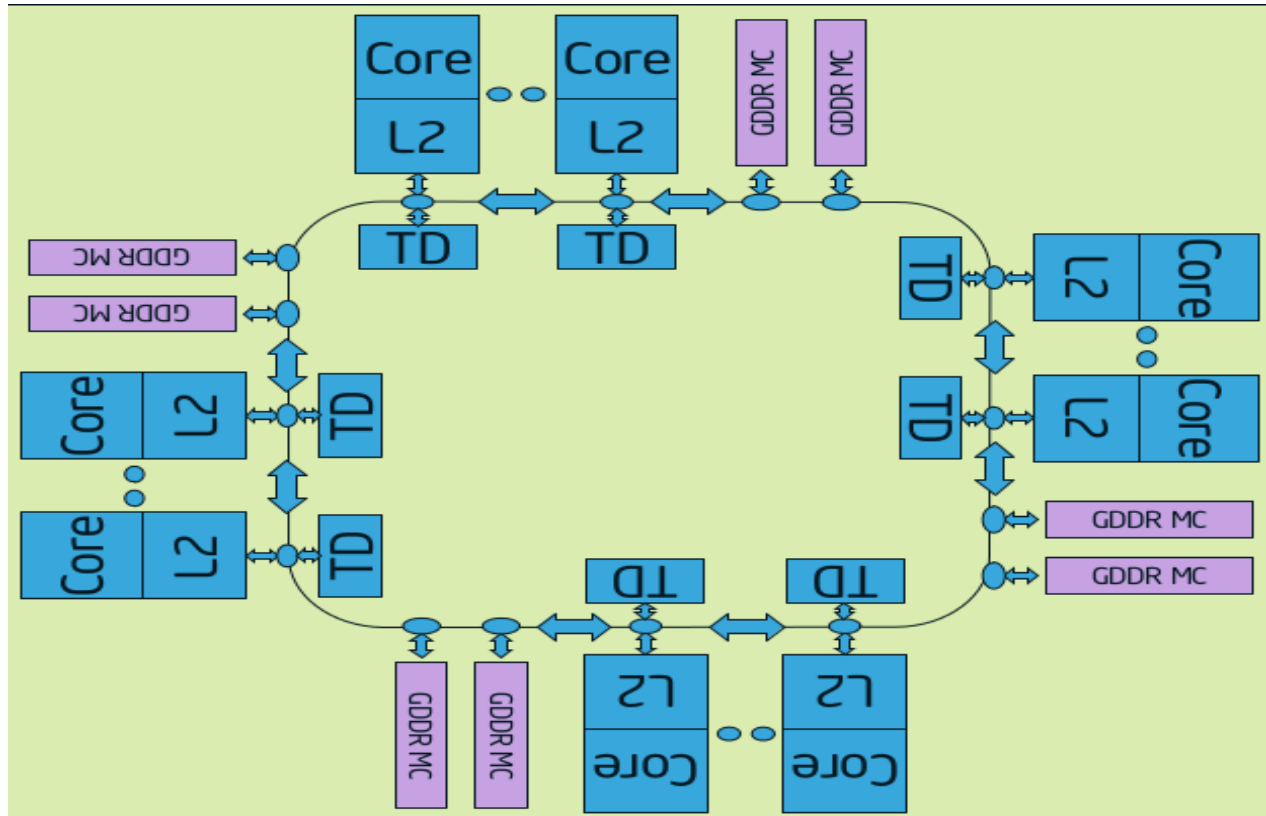
Interconnect

Intel MIC (Multi Integrated Core) (Knight's Corner/Landing - Xeon Phi)

- Multiple rings
  - Directional
  - Data rings
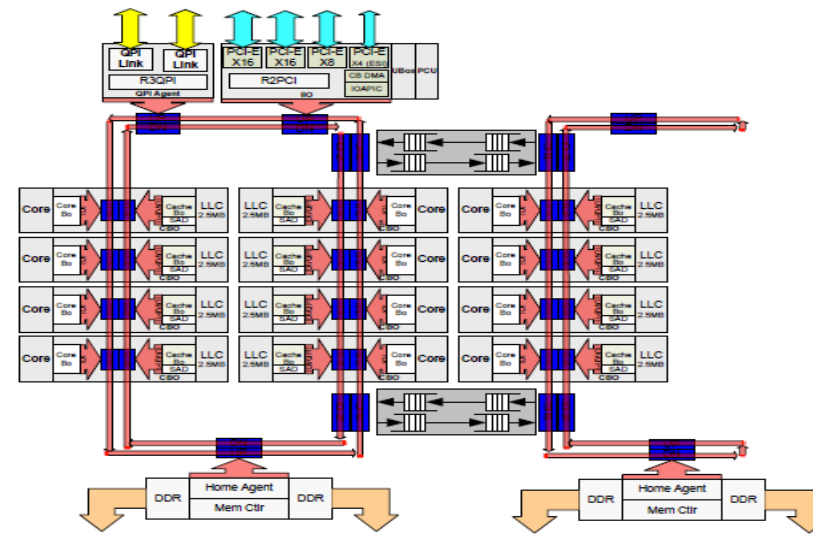 - Address rings
  - Coherence rings

From http://semiaccurate.com/2012/08/28/intel-details-knights-corner-architecture-at-long-last/

Interconnect

Haswell EP Die Configurations
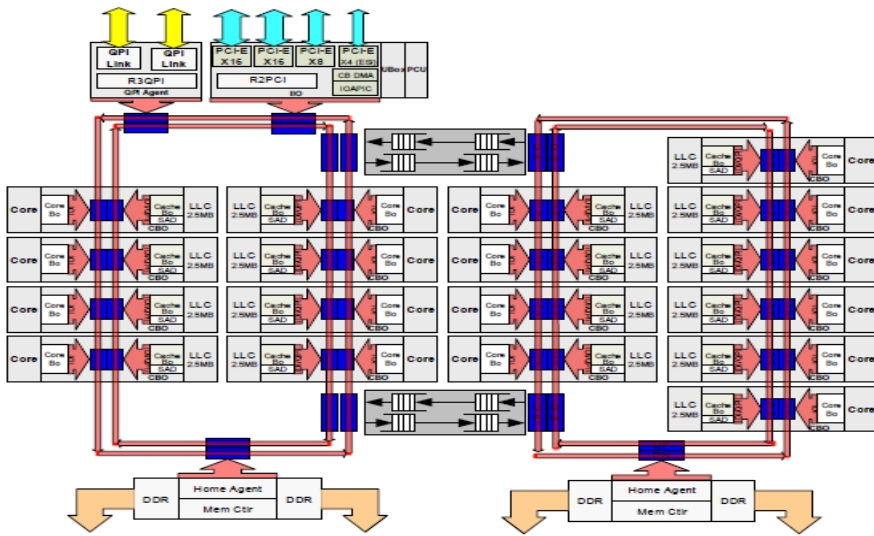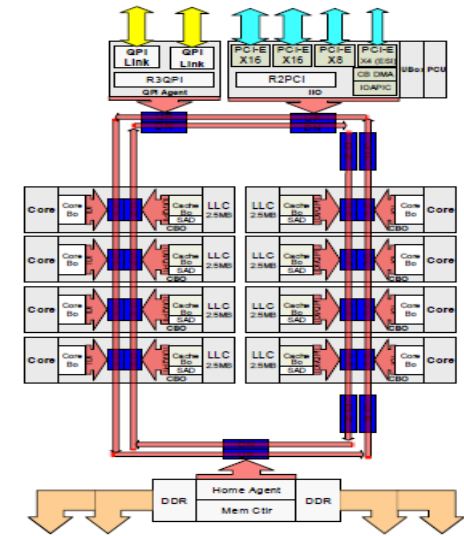
Not representative of actual die-sizes, orientation and layouts – for informational use only.

Chop	Columns	Home Agents	Cores	Power (W)	Transitors (B)	Die Area (mm²)
HCC	4	2	14-18	110-145	5.69	662
MCC	3	2	6-12	65-160	3.84	492
LCC	2	1	4-8	55-140	2.60	354

From http://www.anandtech.com/show/8423/intel-xeon-e5-version-3-up-to-18-haswell-ep-cores-/4

Interconnect/Structure/Memory

Cluster on Die (COD) Mode

- Supported on 1S & 2S SKUs with 2 Home Agents (10+ cores)

- In memory directory bits & directory cache used on 2S to reduce coherence traffic and cache-to-cache transfer latencies

- Targeted at NUMA optimized workloads where latency is more important than sharing across Caching Agents
  - Reduces average LLC hit and local memory latencies
  - HA sees most requests from reduced set of threads potentially offering higher effective memory bandwidth

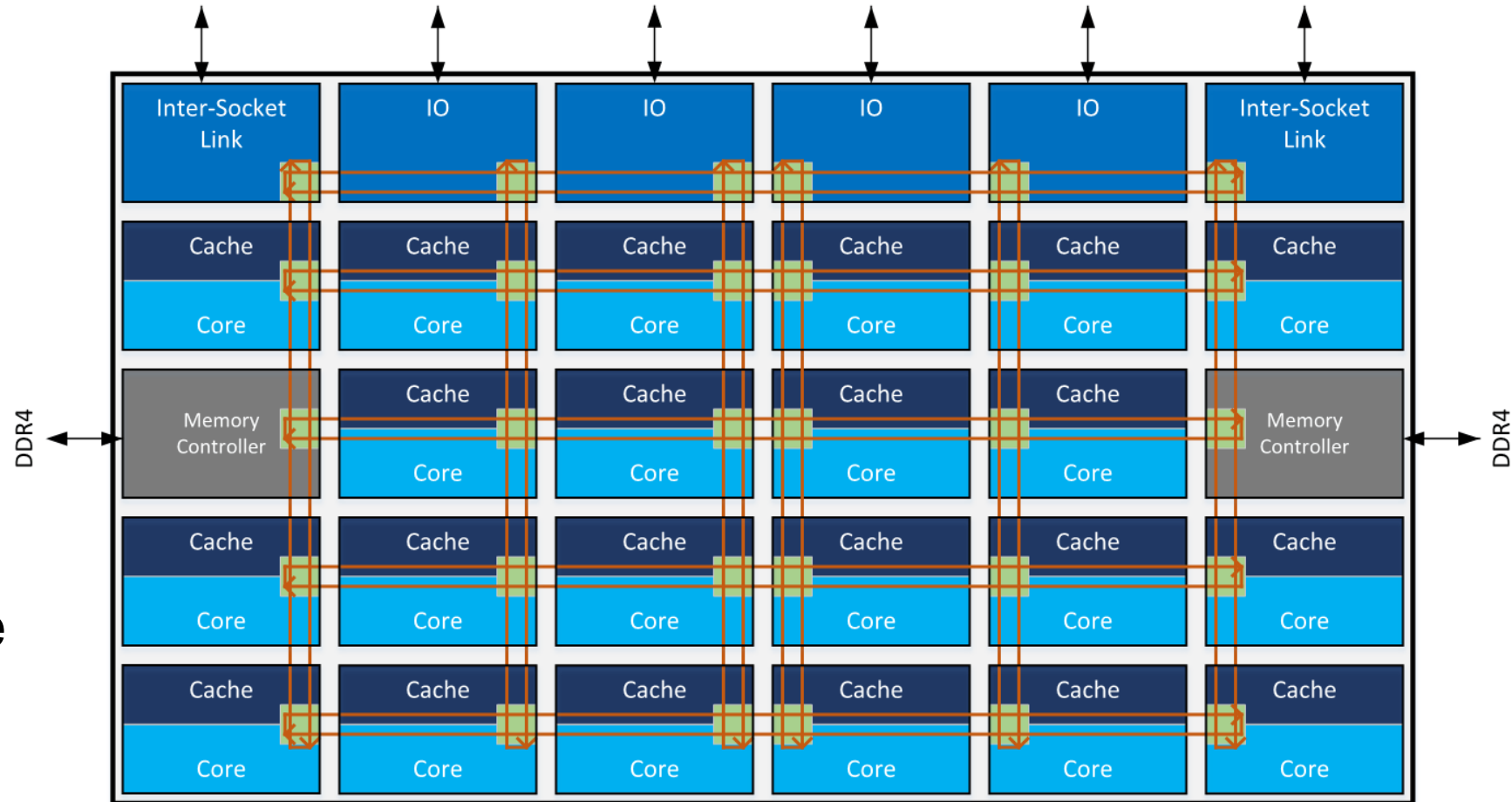- OS/VMM own NUMA and process affinity decisions
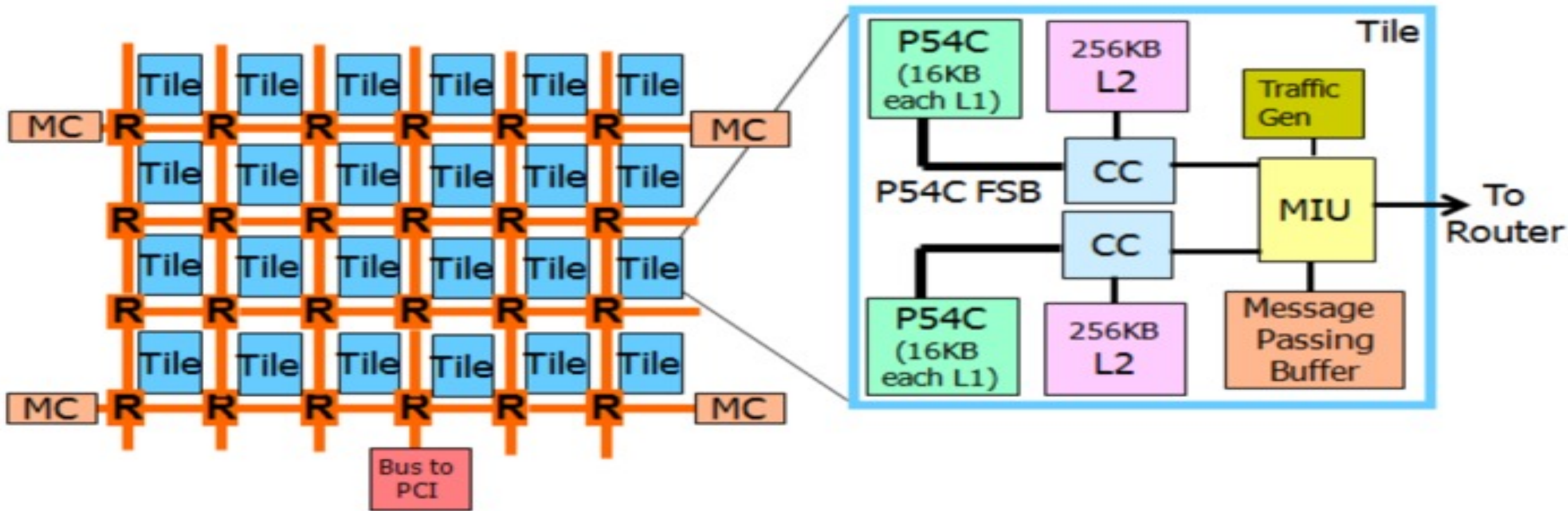

COD Mode for 18C E5-2600 v3
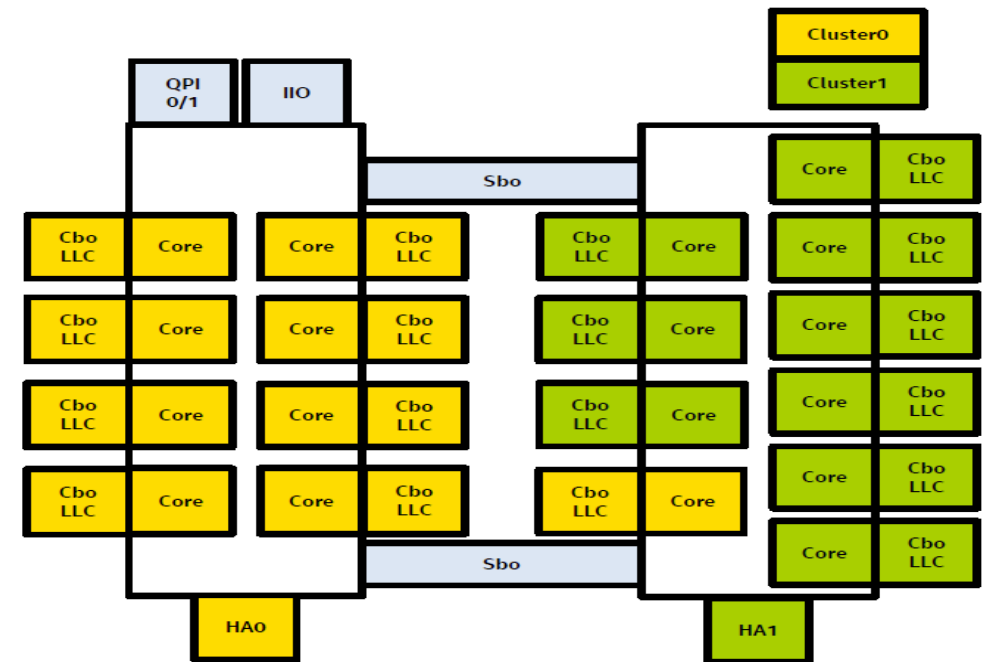
Interconnect

Skylake SP

- Server

- Mesh
  - Array of half-rings

- Sub-NUMA clustering (replacing CoD)
  - separate memory domains

- Per core LLC slice
  - Directory based coherency

From https://itpeernetwork.intel.com/intel-mesh-architecture-data-center/

Interprocessor Communication

Intel SCC – explicit message passing buffer

From techresearch.intel.com/spaw2/uploads/files/SCC_Platform_Overview.pdf

Summary

Scalability
- 100+ cores
- Amdahl's law really kicks in

Heterogeneity
- Heterogeneous cores, memory, etc.
- Properties of similar systems may vary wildly (e.g. interconnect topology and latencies between different AMD platforms)

NUMA
- Also variable latencies due to topology and cache coherence

Cache coherence may not be possible
- Can't use it for locking
- Shared data structures require explicit work

Computer is a distributed system
- Message passing
- Consistency and Synchronisation
- Fault tolerance