Martin Berzins School of Computing

CPU and COMMUNICATIONS PERFORMANCE ISSUES

The most constant difficulty in contriving the engine has arisen from the desire to reduce the time in which the calculations were executed to the shortest which is possible.”

Charles Babbage

1791-1871

High performance requires understanding of modern computer architecture

- Modern CPUs are starved for memory bandwidth
- Main memory is slow but cheap
- Cache is expensive, made of SRAM (static ram)

Memory hierarchy consists of multiple levels of memory

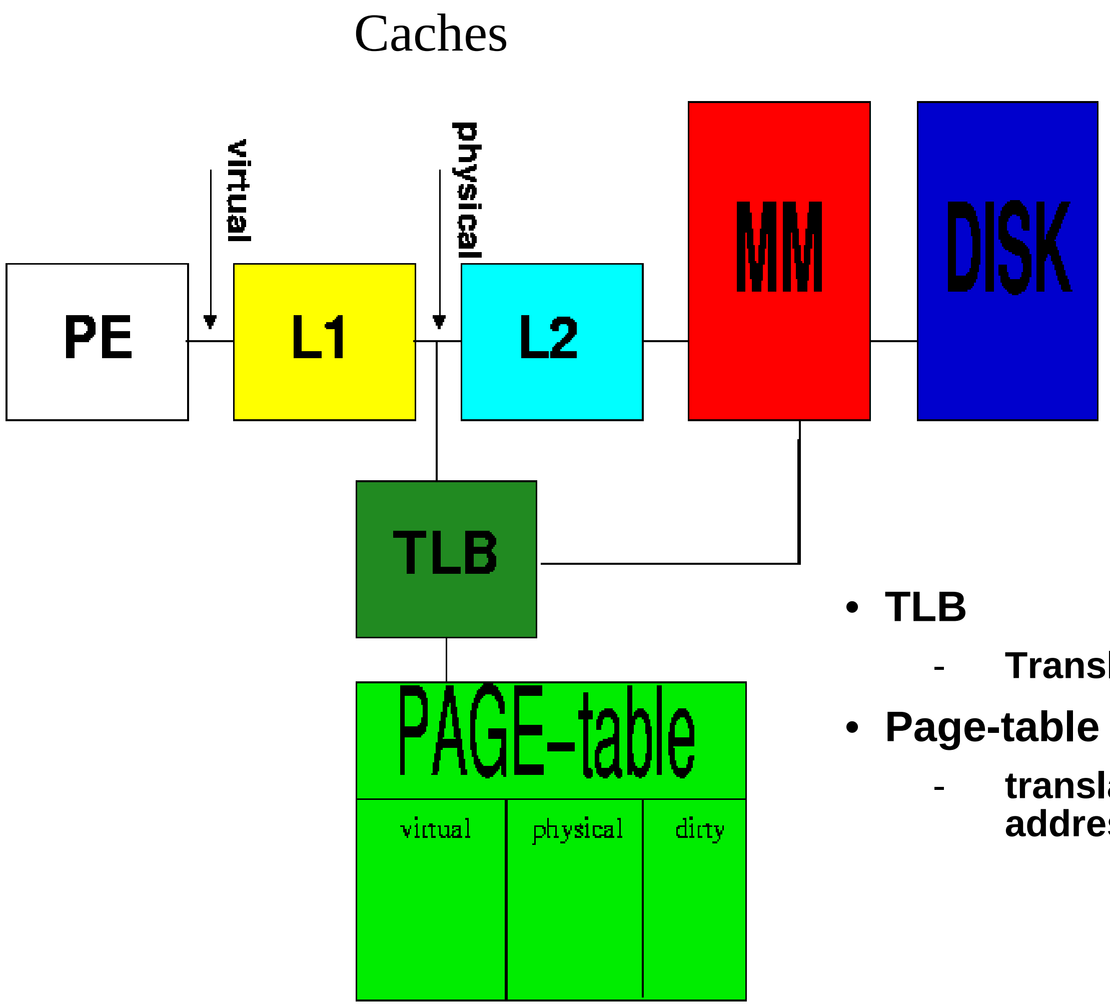
Network and memory speeds have not increase as fast

Present bus and network speeds are slow x MBs and y microseconds

Basics of CPU architecture

- CPUs are
  - superscalar, execute more than one instruction per clock cycle
    - 4 integer, 2 floating-points or 2 multiply-add
  - Piplined:
    - Floating point operation take O(10) cycles to complete
    - Operations can be started every clock cycle
  - Load-Store: Operations are done in registers
  - Now have more than one core.
- Code performance dependent on optimization

Memory Hierarchy



- ♦ Multiple levels of memory
- ♦ Fastest memory closest to CPU
- ♦ Each layer keeps a copy of previous one
- ♦ L1 fastest,smallest
- ♦ L2 second level
- ♦ RAM main memory

- **TLB**
  - **Translation Lookup Buffer**
- **Page-table**
  - **translation between virtual and physical addresses**

Caches are SRAM main memory is slower but less expensive DRAM

Cache Definitions

- **Cache hit**
  - **CPU gets data directly from cache**
- **Cache miss**
  - **CPU doesn’t get the data directly from cache**
- **Hit rate**
  - **average percentage of times that the processor will get a cache hit**
- **Locality of reference**
  - **Programs reuse data and instructions**
  - **Rule of thumb: 90% of time in about 10% of the code**

Latency of memory access

Latency: time for task

to be accomplished

- **CPU Registers: 0 cycles**
- **L1 hit:2 or 3 cycles**
- **L1 miss satisfied by L2: 8-10 cycles**
- **L2 miss, no TLB miss: 75-250 cycles**
- **TLB miss, reload memory: 2000 cycles**
- **TLB miss, reload from disk: Millions cycles**
- **Network, communication dependent**

SERIOUS ISSUE FOR PERFORMANCE

Memory Hierarchy

Memory: the larger it gets, the slower it gets

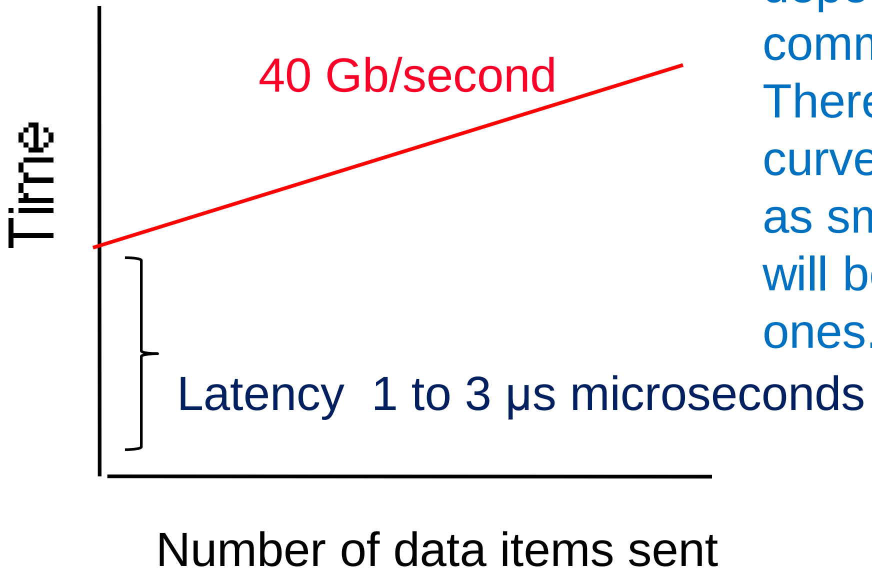
•Rough numbers:

	Latency	**Bandwidth**	**Size**
SRAM (L1, L2, L3)	1-2ns	200GBps	1-20MB
DRAM (memory)	70ns	20GBps	1-20GB
Flash (disk)	70-90µs	200MBps	100-1000GB
HDD (disk)	10ms	1-150MBps	500-3000GB

SRAM $2K to 5K Per GB

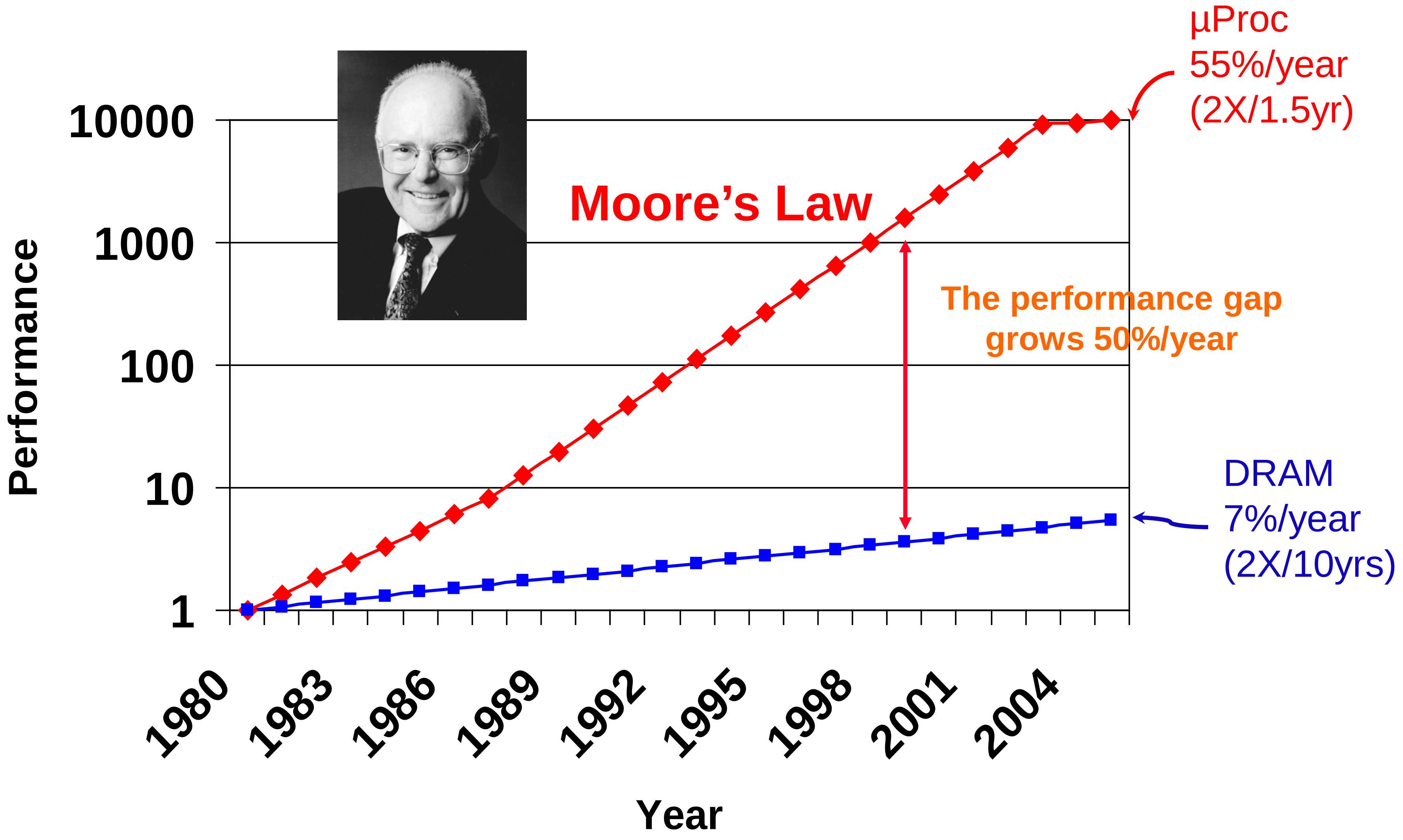
DRAM $20-75 per GB

Disk $0.20 – $2 per GB



The startup time will vary depending on the type of communication.
There will also be a slight curve at the start of the line as small messages
will be sent faster than large ones.

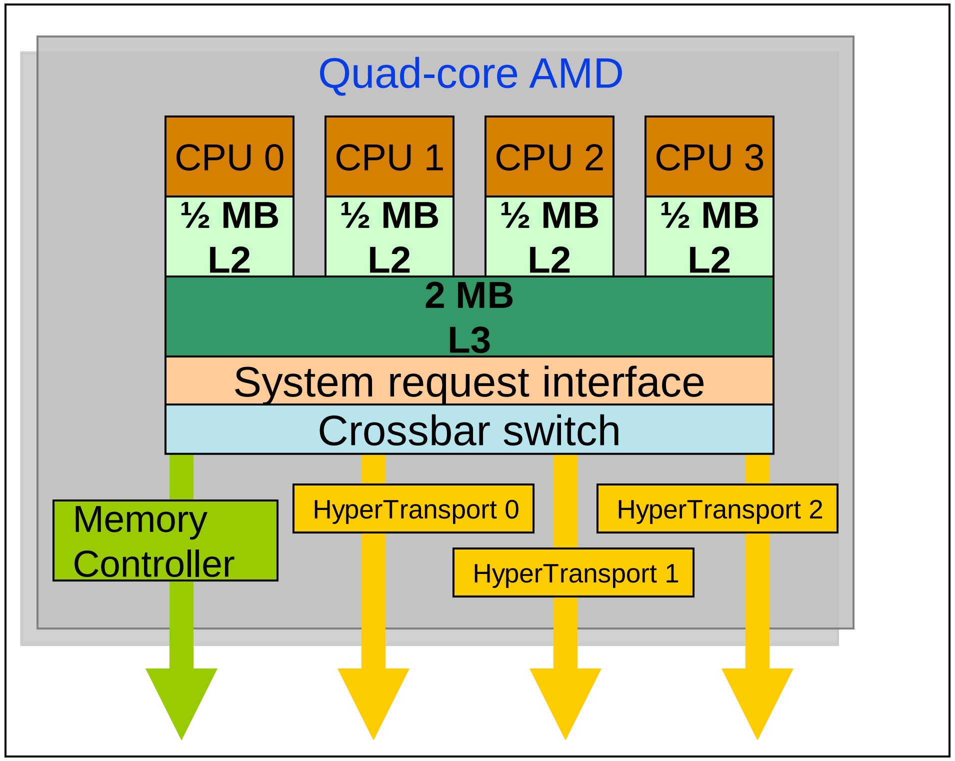
CPU vs Memory Performance



Prof. Sean Lee's Slide

CPU PROCESSOR TECHNOLOGY AMD Barcelona Chip

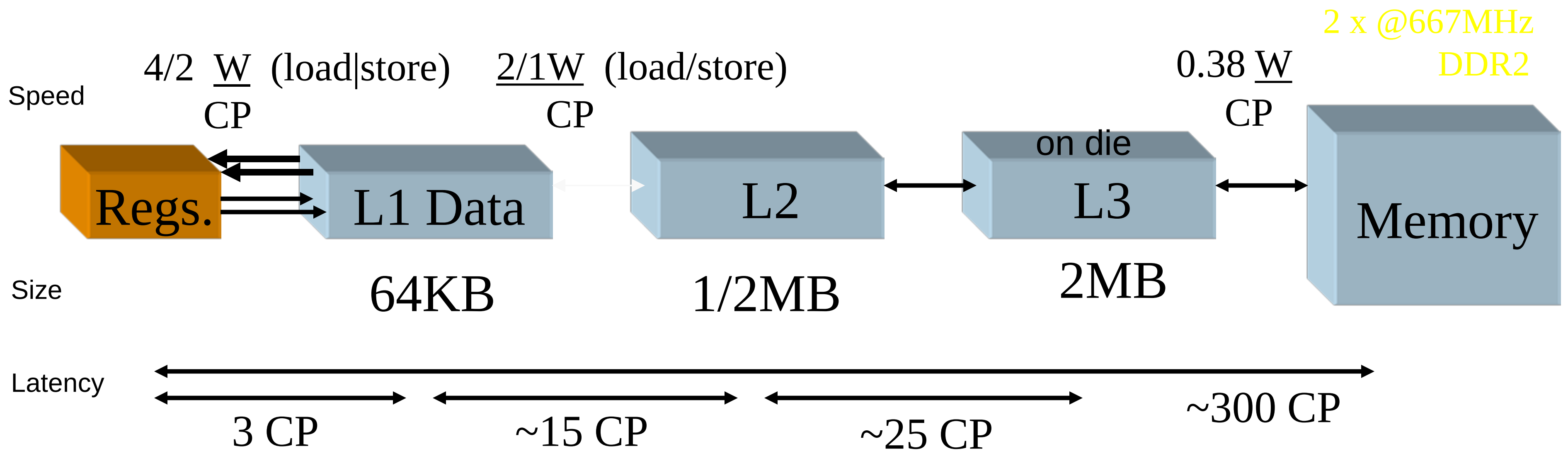
Each CPU has a 64K level 1 cache too



http://arstechnica.com/news.ars/post/20061206-8363.html

Source: Kent Milfeld

Speeds & Feeds (Barcelona)



Rate	50GB/s	25GB/s	12 GB/s	8GB/s

DISTANT Memory 15k CP

Approx 1000 CP per μsec

4 FLOPS/CP

Cache Line size L1/L2 = 8W/8W

W	PF Word (64 bit)
CP	Clock Period

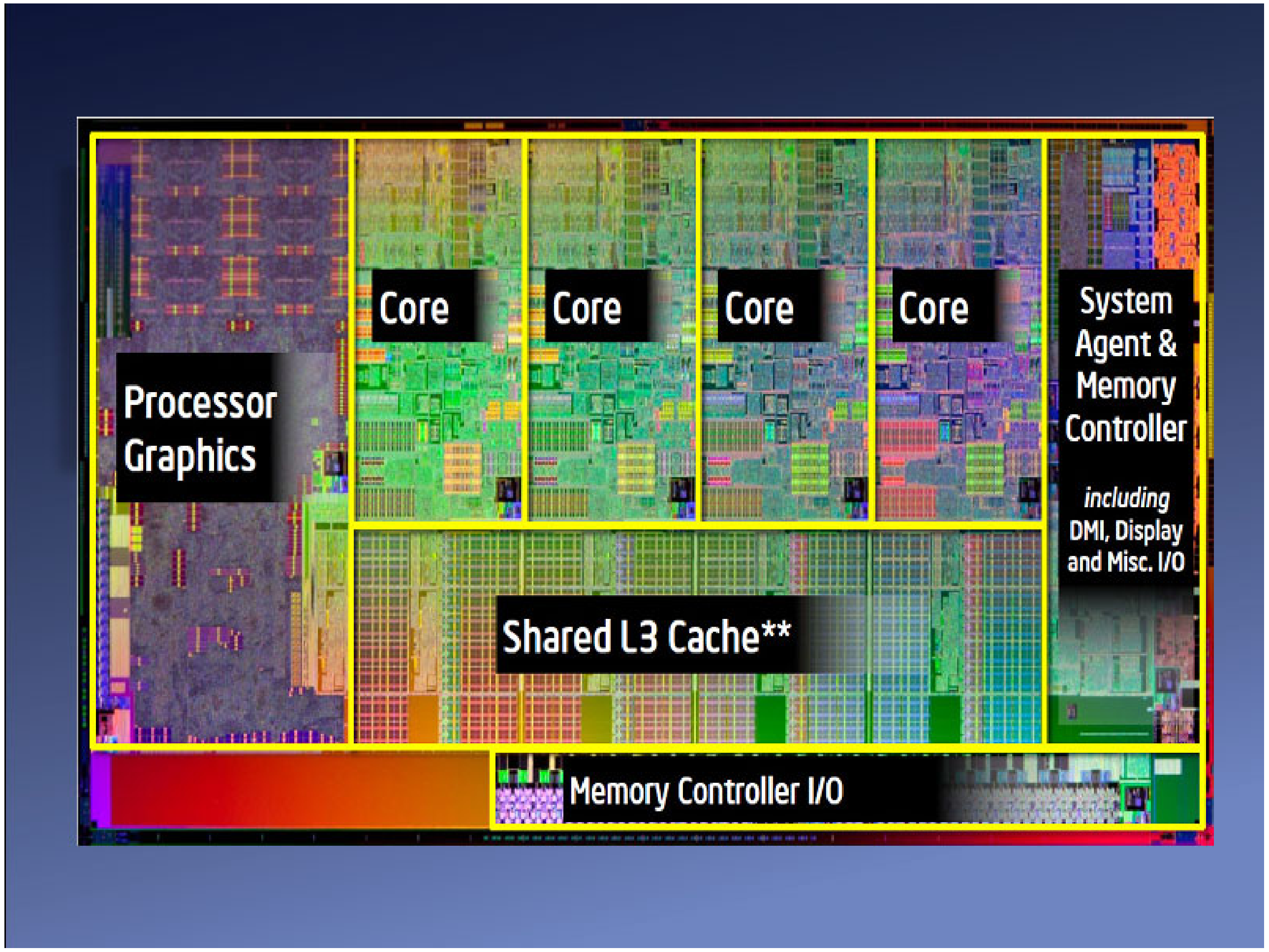
Source: Kent Milfeld

Core i7 (2nd Gen.)

2nd Generation Core i7

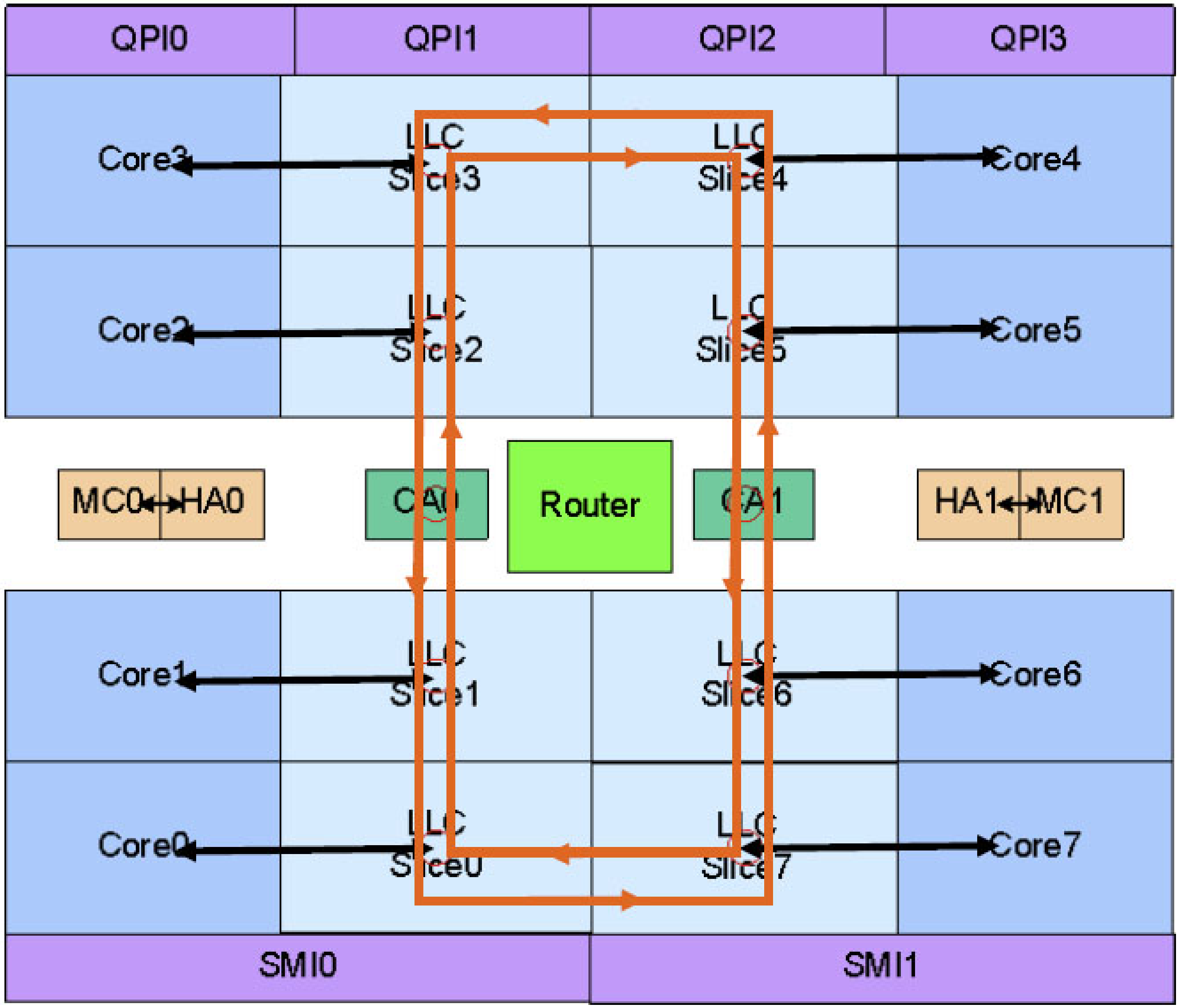
Sandy Bridge

L1	32 KB
L2	256 KB
L3	8MB



995 million transistors in 216 mm^2 with 32nm technology

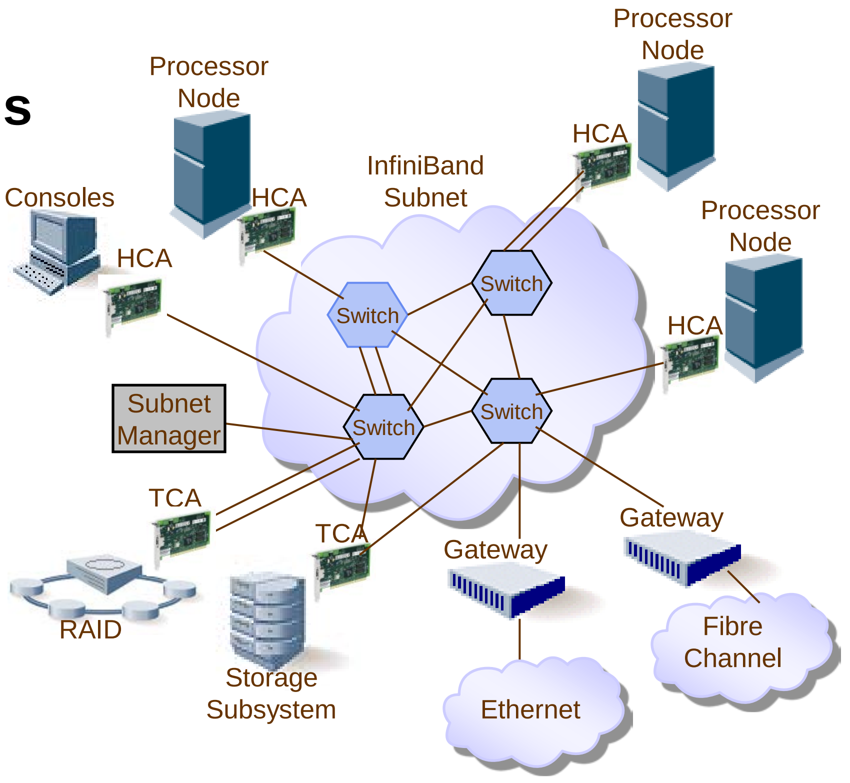
SANDY BRIDGE RING BUS



The InfiniBand Architecture

- Industry standard defined by the InfiniBand Trade Association
- Defines System Area Network architecture
  - Comprehensive specification:

    from physical to applications
- Architecture supports
  - Host Channel Adapters (HCA)
  - Target Channel Adapters (TCA)
  - Switches
  - Routers
- Facilitated HW design for
  - Low latency / high bandwidth
  - Transport offload



Infiniband Highest Performance

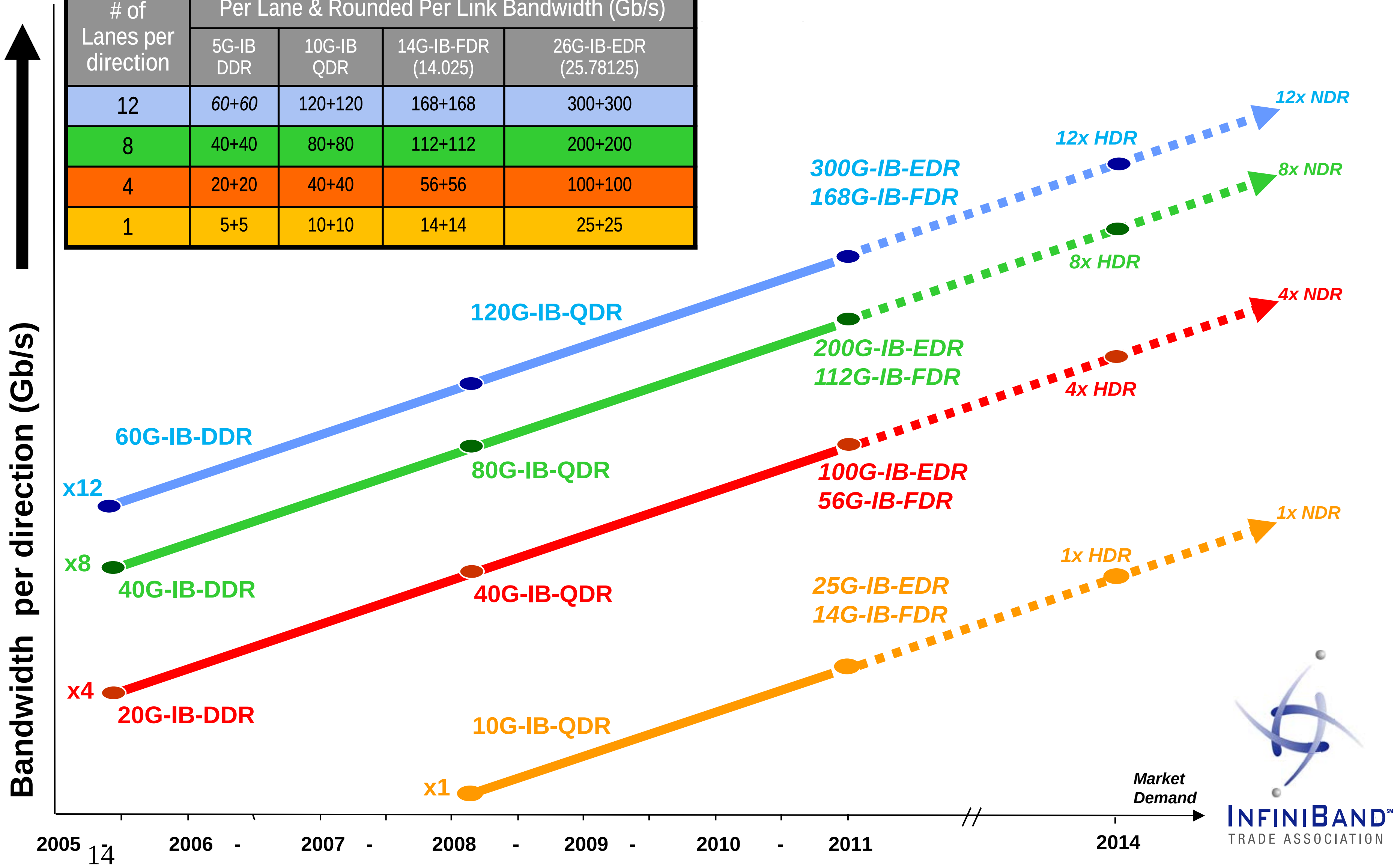
- **Highest throughput**
  - **40Gb/s node to node**
  - **Nearly 90M MPI messages per second**
  - **Send/receive and RDMA operations with zero-copy**

- **Lowest latency**
  - **1-1.3usec MPI end-to-end**
  - **0.9-1us InfiniBand latency for RDMA operations**
  - **100ns switch latency at 100% load**
  - **Lowest latency 648-port switch – 25% to 45% faster vs other solutions**

- **Lowest CPU overhead**
  - **Full transport offload maximizes CPU availability for user applications**

InfiniBand Link Speed Roadmap

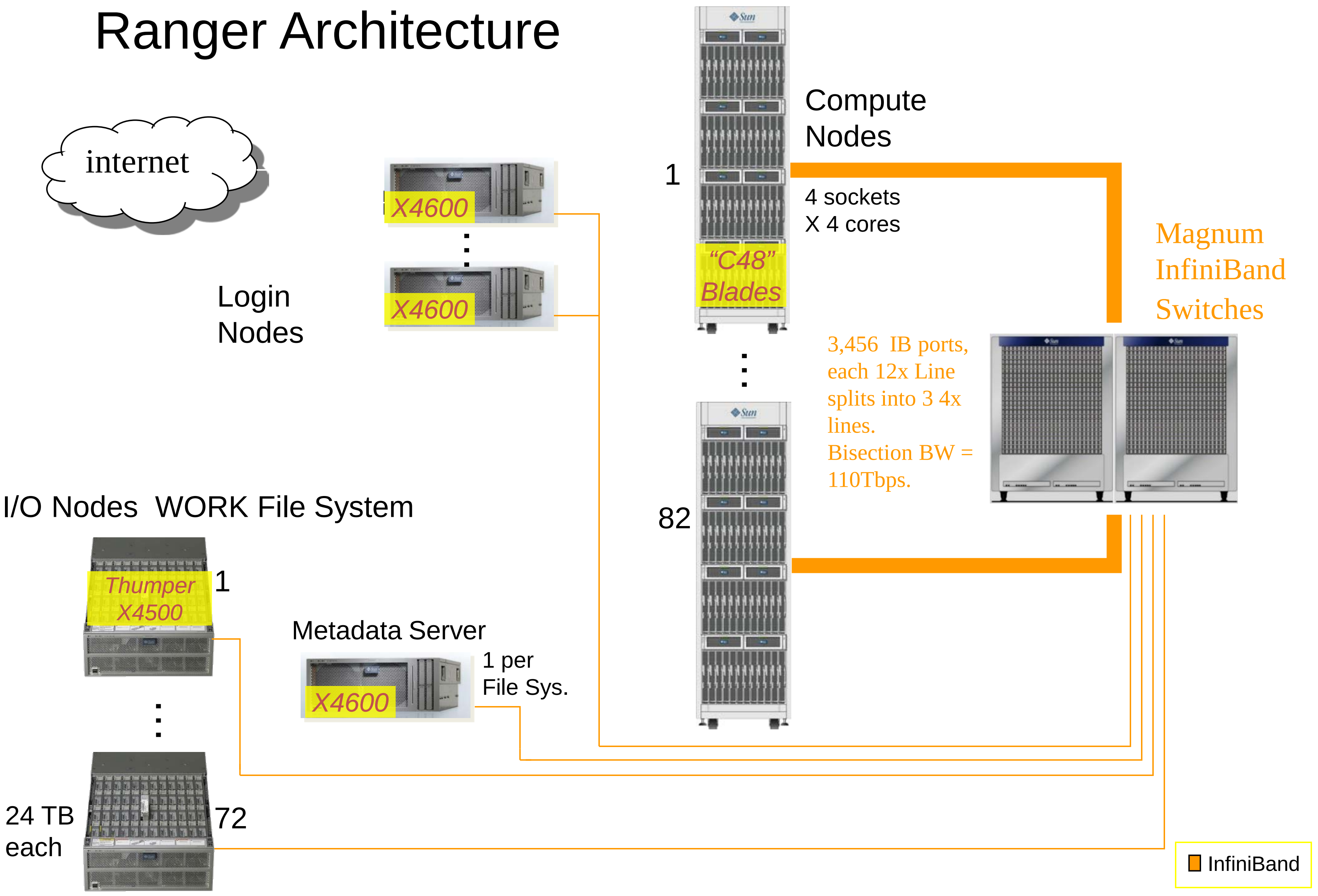
# of Lanes per direction	Per Lane & Rounded Per Link Bandwidth (Gb/s)			
	5G-IB DDR	10G-IB QDR	14G-IB-FDR (14.025)	26G-IB-EDR (25.78125)
12	*60+60*	120+120	168+168	300+300
8	40+40	80+80	112+112	200+200
4	20+20	40+40	56+56	100+100
1	5+5	10+10	14+14	25+25



Ranger Cluster Overview ranger.tacc.utexas.edu

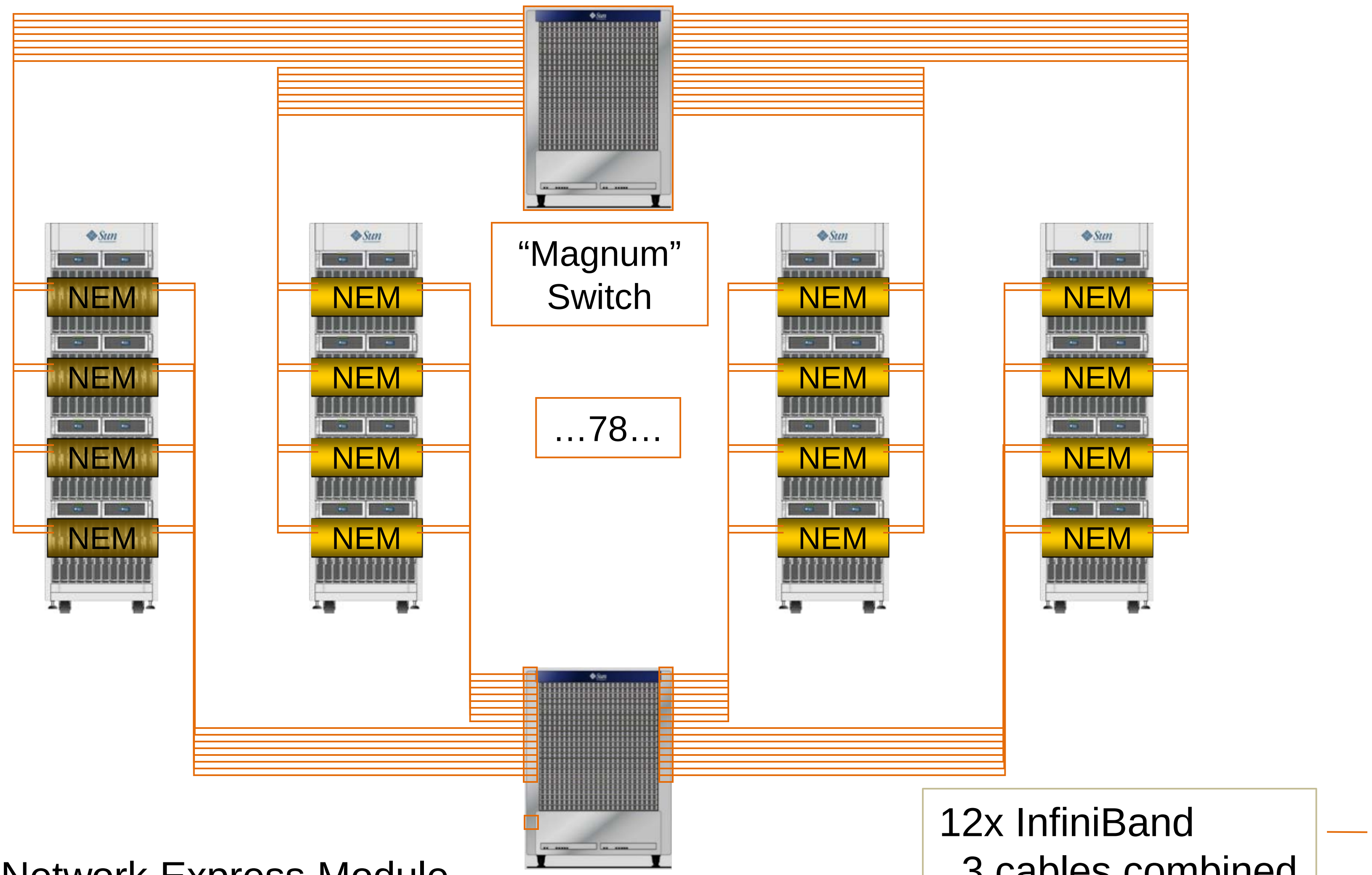
Hardware	Components	Characteristics
Compute Nodes Sun AMD Barcelona 4flops per cycle	3,936 Nodes 62,976 Cores 4x4core sockets/node	2.3 GHz 4MB/Cache 32GB Mem/node
Sun x4500 “Thumper” I/O Servers	72 I/O Nodes Lustre File System	24 TB each 1.7PB (raw)
Login Development	2 logins: ranger 24 Nodes (dev. queue)	2.2 GHz, 32GB Mem 2.3 GHz, 32GB/node
Interconnect (MPI) InfiniBand	NEM – Magnum two tier switch	1GB/sec P-2-P Fat Tree Topology

Ranger Architecture



Source: Kent Milfeld

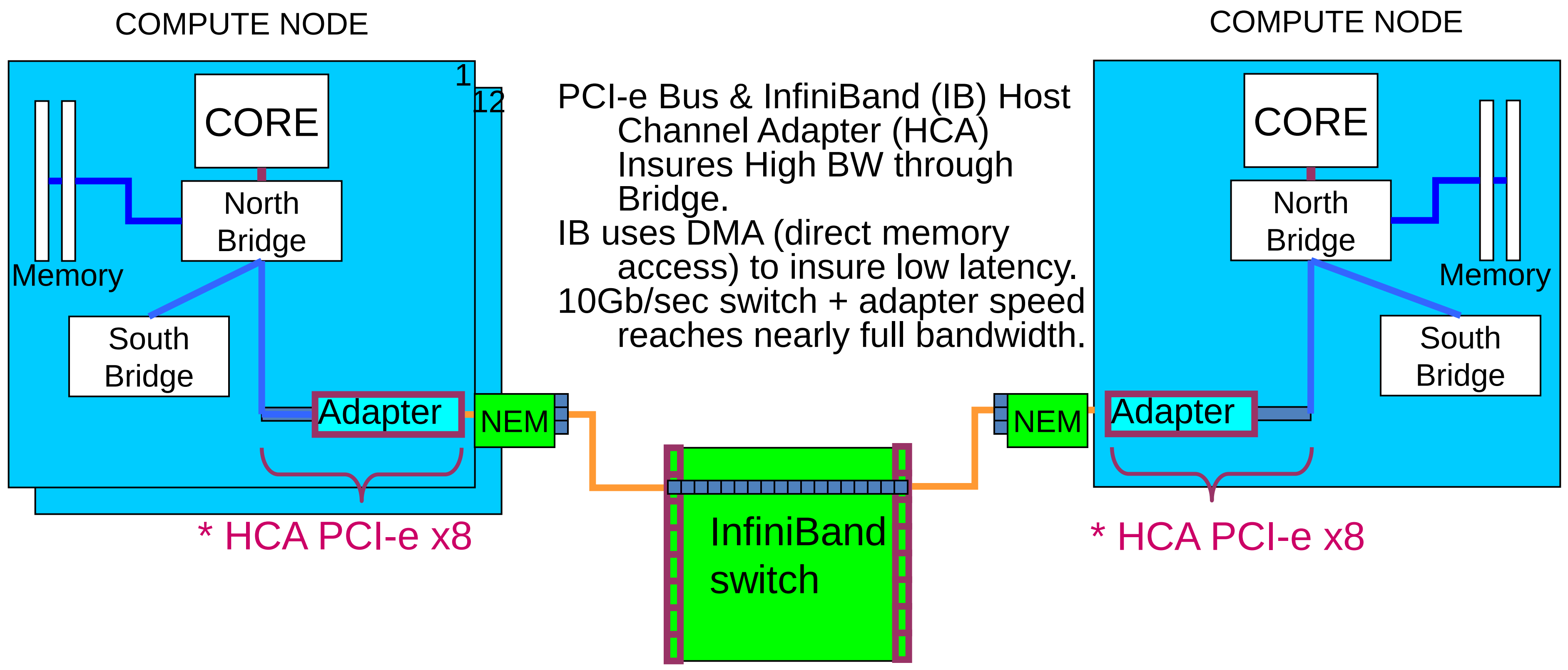
Ranger 2 level Infiniband Interconnect Architecture



NEM: Network Express Module

Source: Kent Milfeld

Interconnect Architecture



PCI-e Bus & InfiniBand (IB) Host Channel Adapter (HCA) Insures High BW through Bridge.
IB uses DMA (direct memory access) to insure low latency.
10Gb/sec switch + adapter speed reaches nearly full bandwidth.

Latency ~ 2 μsec Bandwidth ~1GB/sec DMA

4 x 4 cores on a compute node

* 1x = 250MB/s in 1 direction

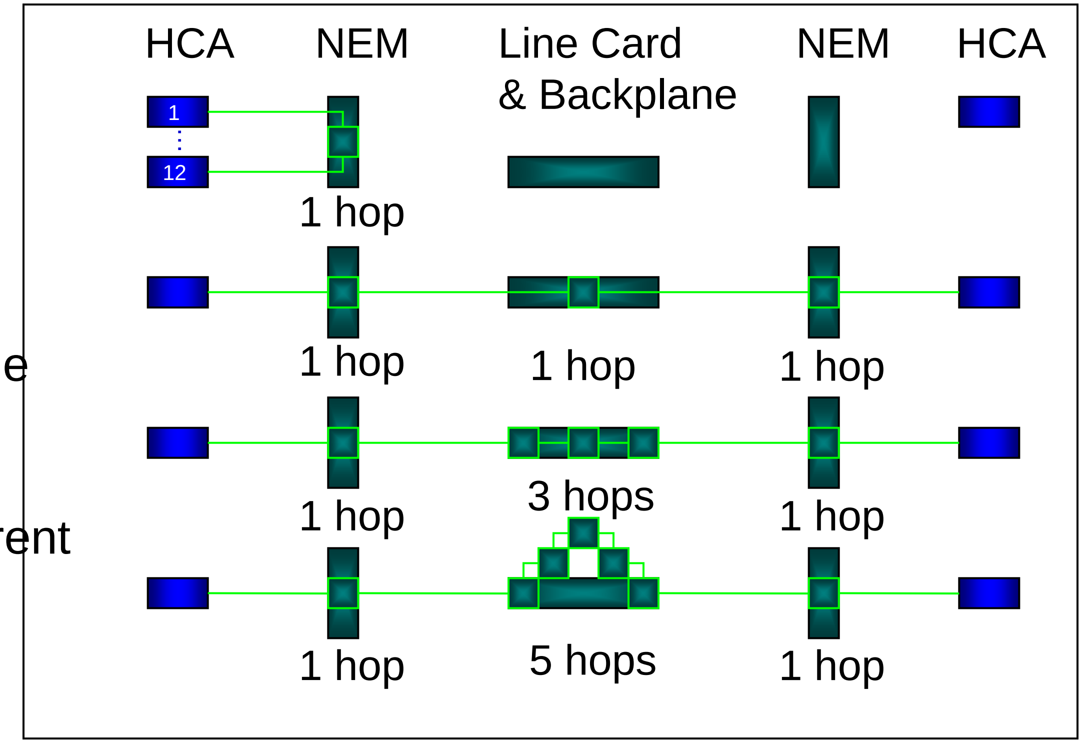
Source: Kent Milfeld

Ranger Non- Uniform Communications times
Switch Hops

Tasks on same chassis stay in NEM

If 2 chassis on the same line card

2 chasis on different Line cards and Backplane used



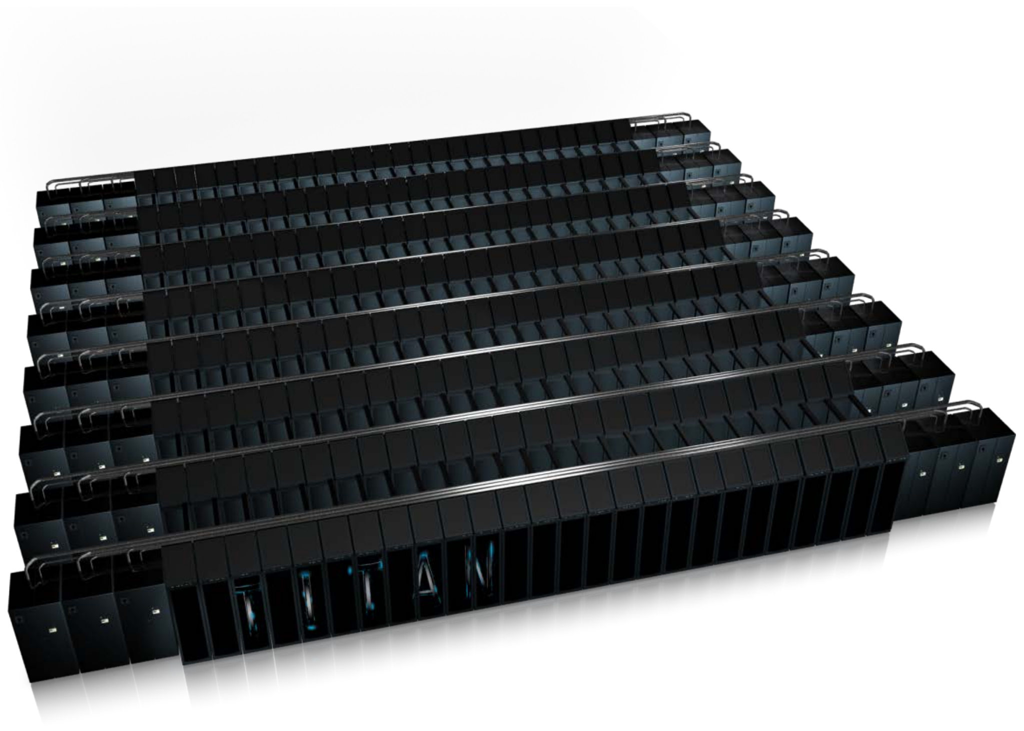
HCA Host Channel adapter
NEM Network Express Model

MPI Latencies

1 Hop	2 Hops	5 Hops	7 Hops
1.7 µsec	2.2 µsec	2.8 µsec	3.2 µsec

Source: Kent Milfeld

OLCF
OAK RIDGE LEADERSHIP COMPUTING FACILITY



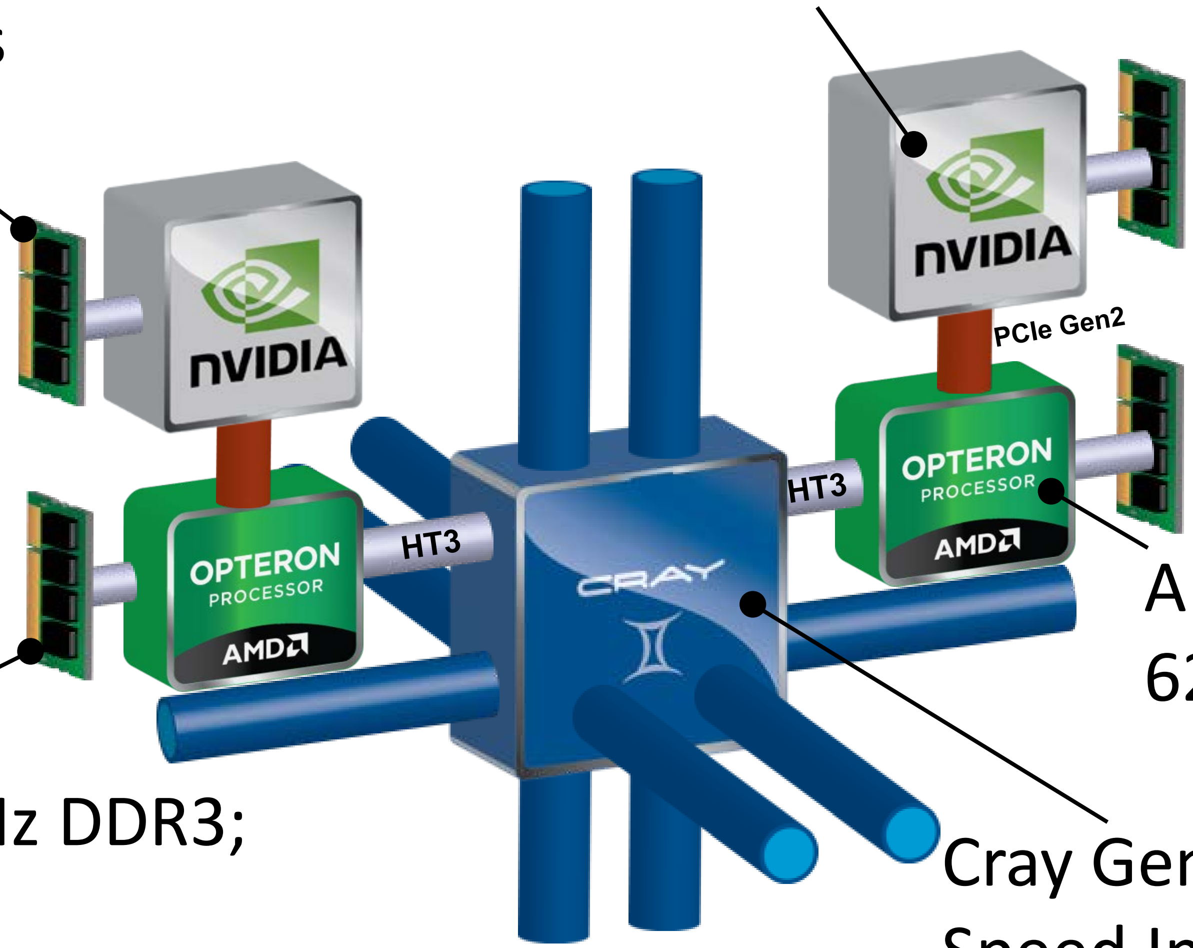
Titan Configuration

Name	Titan
Architecture	XK7
Processor	AMD Interlagos
Cabinets	200
Nodes	18,688
CPU Memory/Node	32 GB
GPU Memory/Node	6 GB
Interconnect	Gemini
GPUs	Nvidia Kepler

Cray XK7 Architecture

NVIDIA Kepler GPU

6GB GDDR5;
138 GB/s

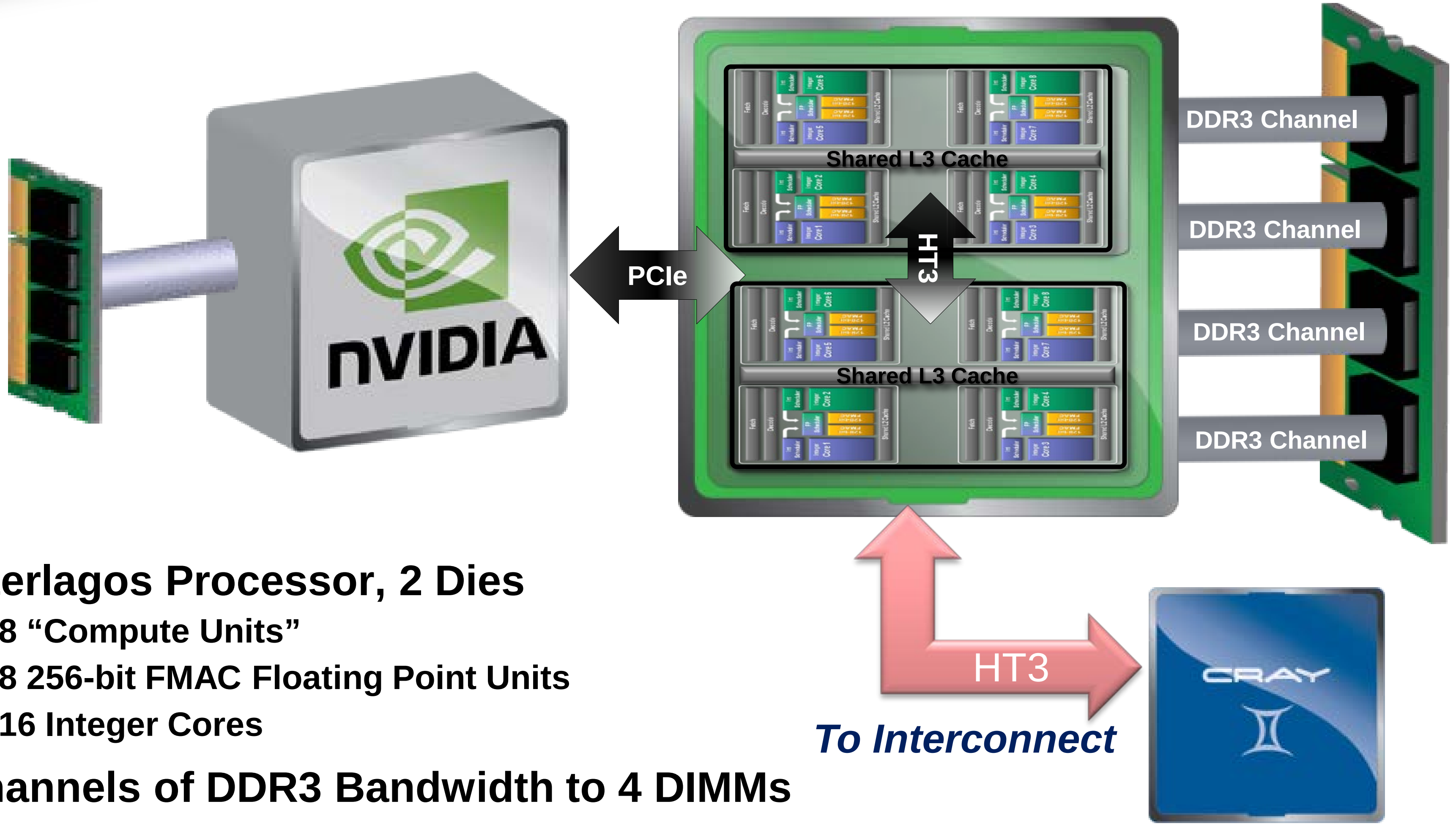


AMD Series
6200 CPU

1600 MHz DDR3;
32 GB

Cray Gemini High
Speed Interconnect

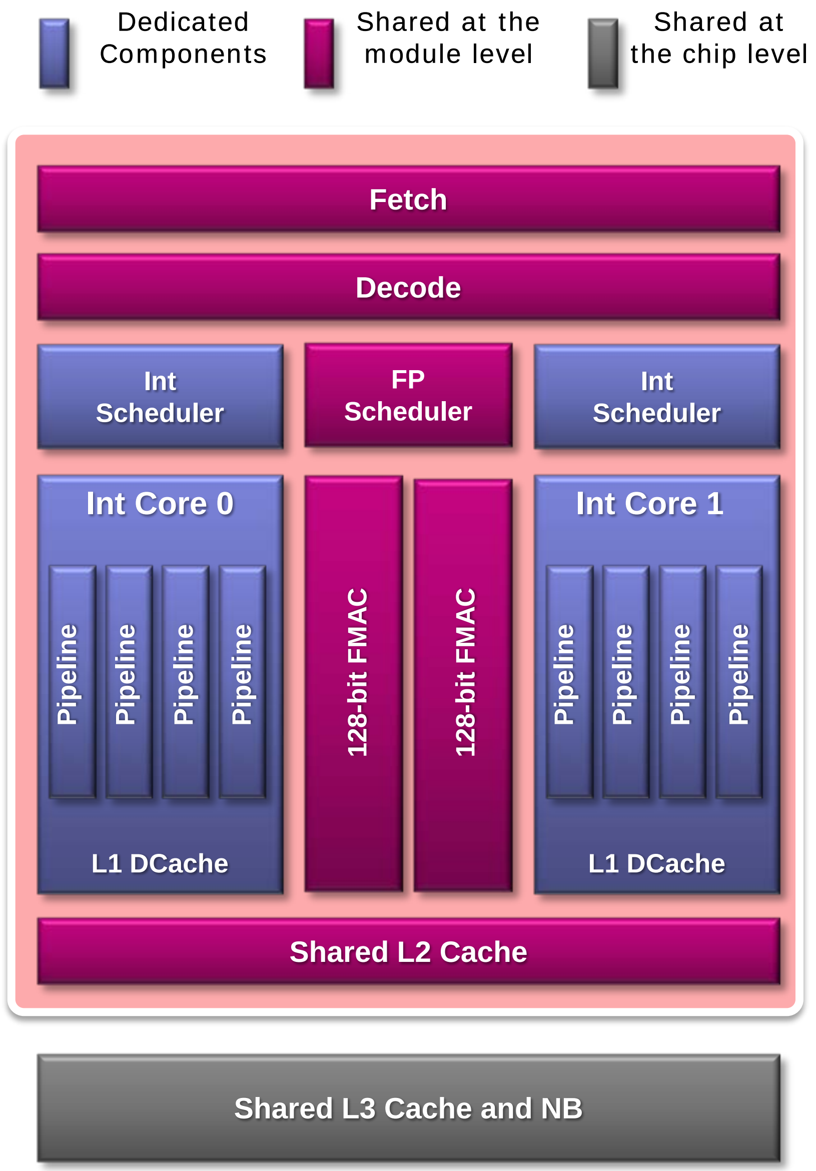
XK7 Node Details



- 1 Interlagos Processor, 2 Dies
  - 8 “Compute Units”
  - 8 256-bit FMAC Floating Point Units
  - 16 Integer Cores
- 4 Channels of DDR3 Bandwidth to 4 DIMMs
- 1 Nvidia Kepler Accelerator
  - Connected via PCIe Gen 2

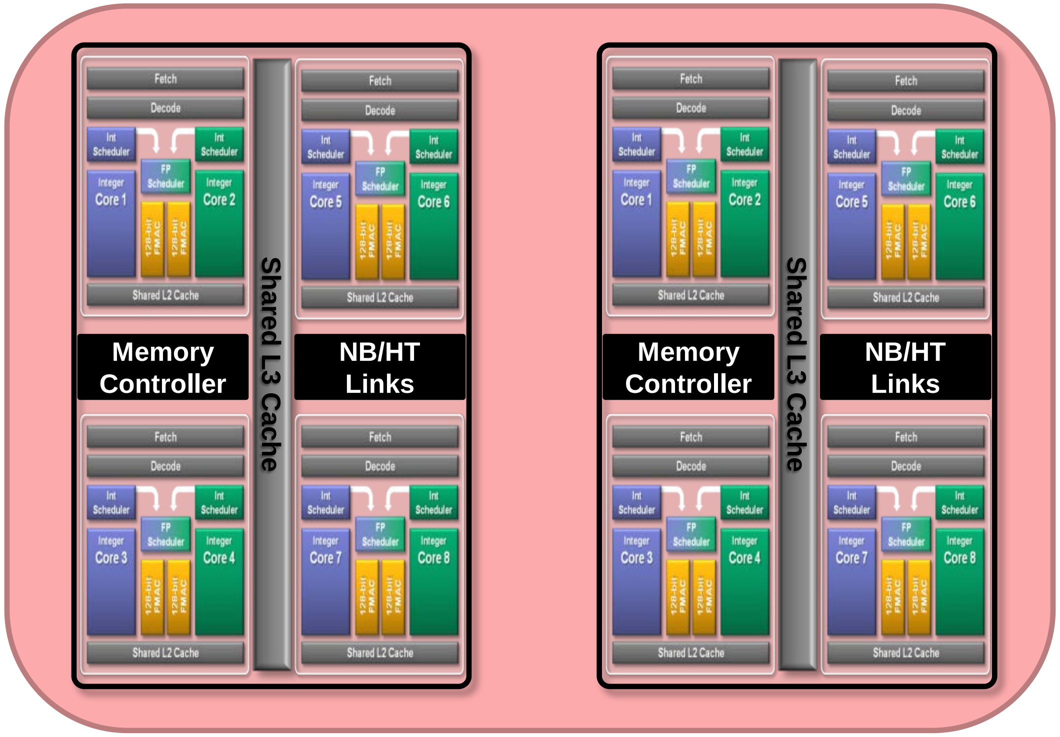
Interlagos Processor Architecture

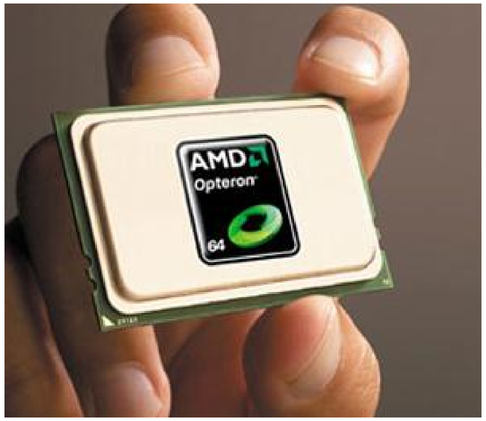
- Interlagos is composed of a number of "Bulldozer modules" or "Compute Unit"
  - A compute unit has shared and dedicated components
    - There are two independent integer units; shared L2 cache, instruction fetch, Icache; and a *shared*, 256-bit Floating Point resource
  - A single Integer unit can make use of the entire Floating Point resource with 256-bit AVX instructions
    - Vector Length
      - 32 bit operands, VL = 8
      - 64 bit operands, VL = 4



Interlagos Processor

- Two die are packaged on a multi-chip module to form an Interlagos processor
  - Processor socket is called G34 and is compatible with Magny Cours
  - Package contains
    - 8 compute units
    - 16 MB L3 Cache
    - 4 DDR3 1333 or 1600 memory channels





Cray Network Evolution

SeaStar

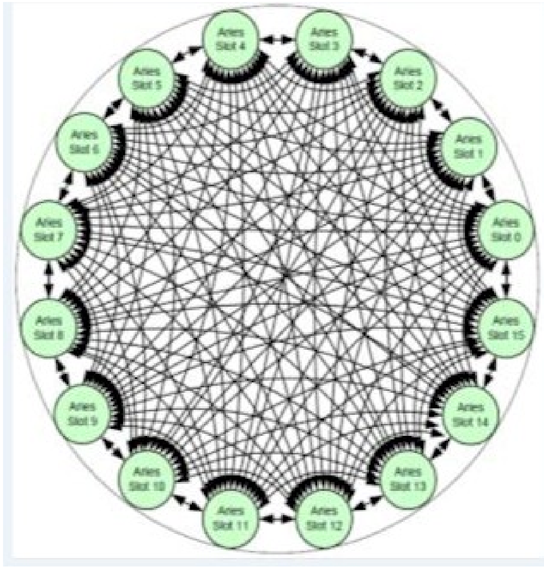
- Built for scalability to 250K+ cores
- Very effective routing and low contention switch

Gemini

- 100x improvement in message throughput
- 3x improvement in latency
- PGAS Support, Global Address Space
- Scalability to 1M+ cores

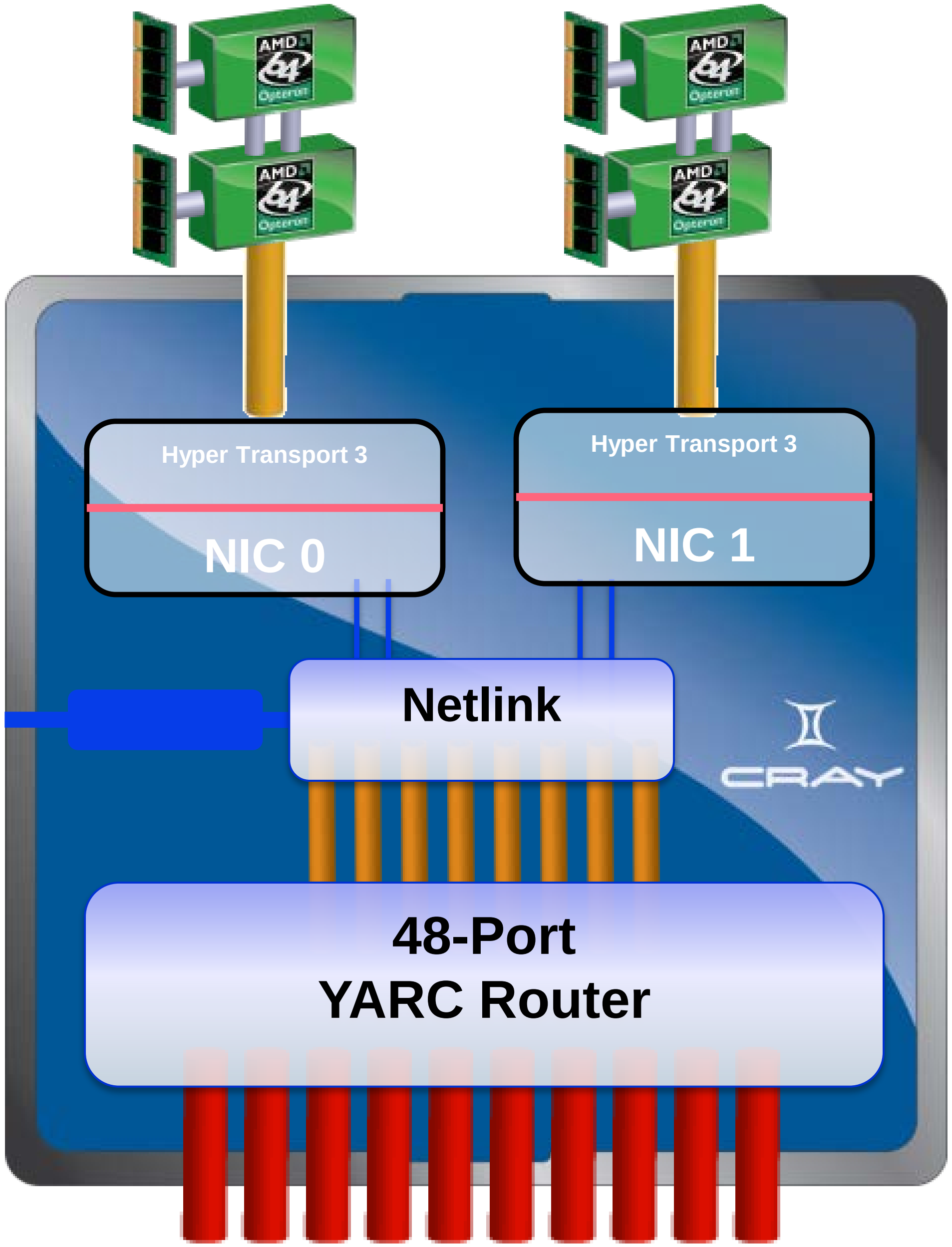
Aries

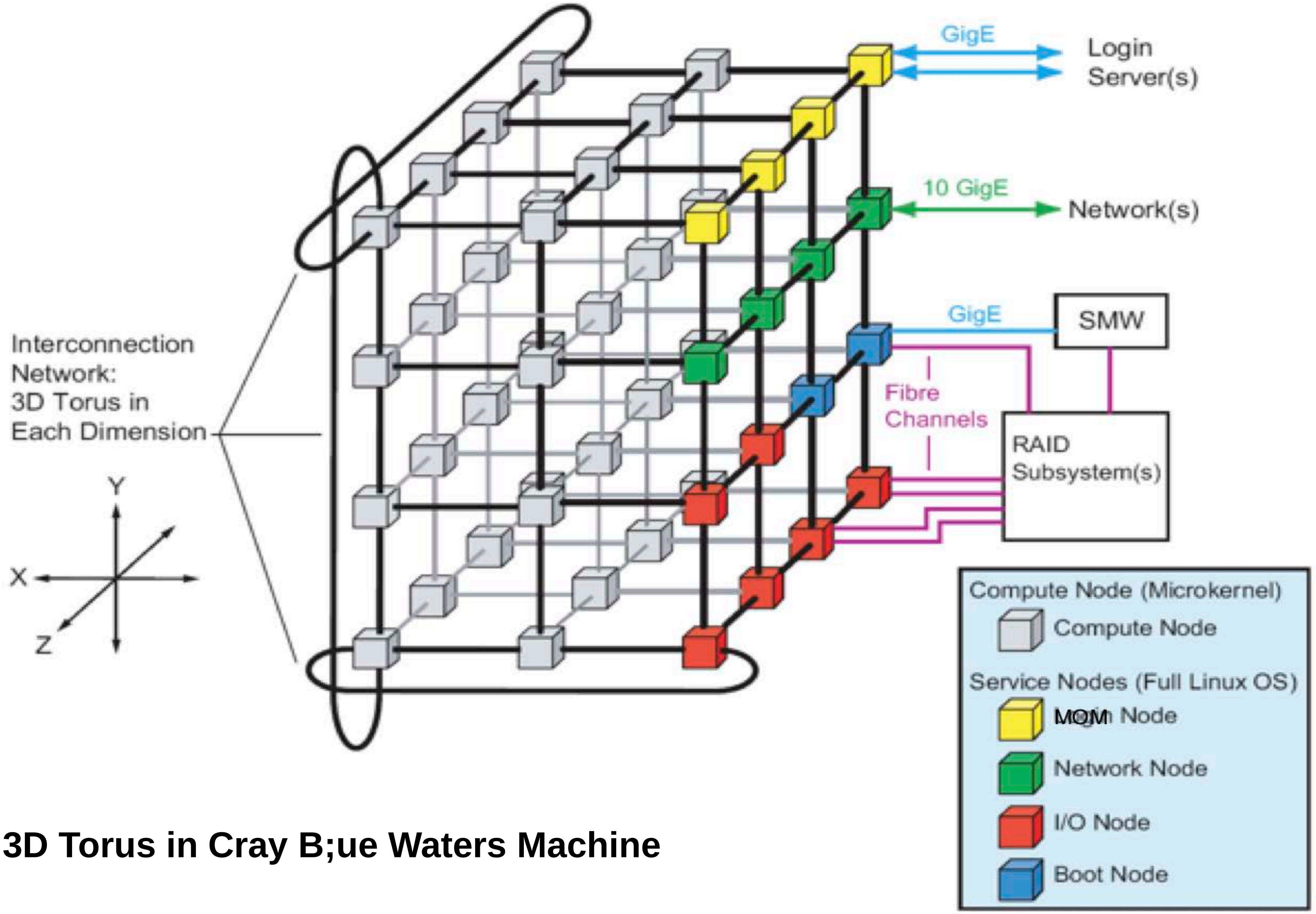
- Cray “Cascade” Systems
- Funded through DARPA program
- 4X improvement over Gemini < 1.0µ second latency



Cray Gemini

- 3D Torus network
- Supports 2 Nodes per ASIC
- 168 GB/sec routing capacity
- Scales to over 100,000 network endpoints
  - Link Level Reliability and Adaptive Routing
  - Advanced Resiliency Features
- Provides global address space
- Advanced NIC designed to efficiently support
  - MPIMillions of messages/second
  - One-sided MPI
  - UPC, FORTRAN 2008 with coarrays, shmem

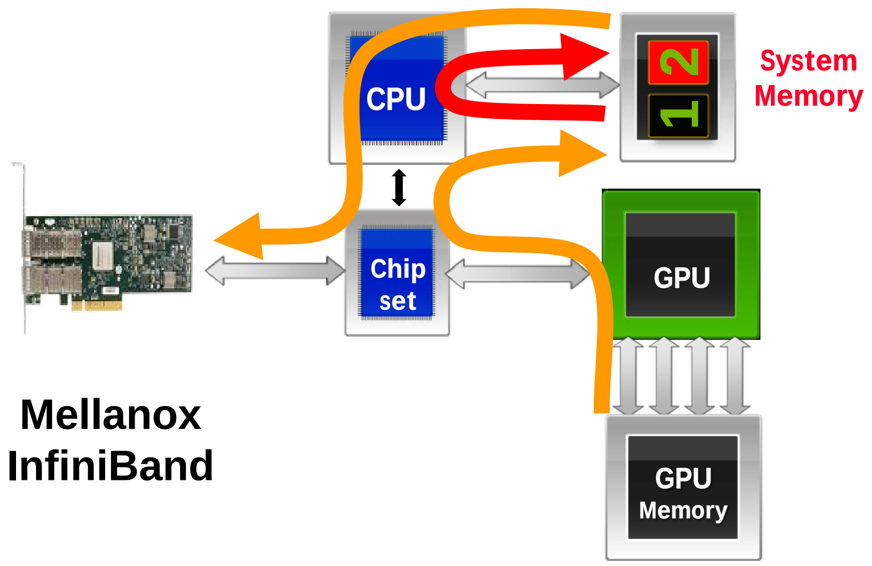


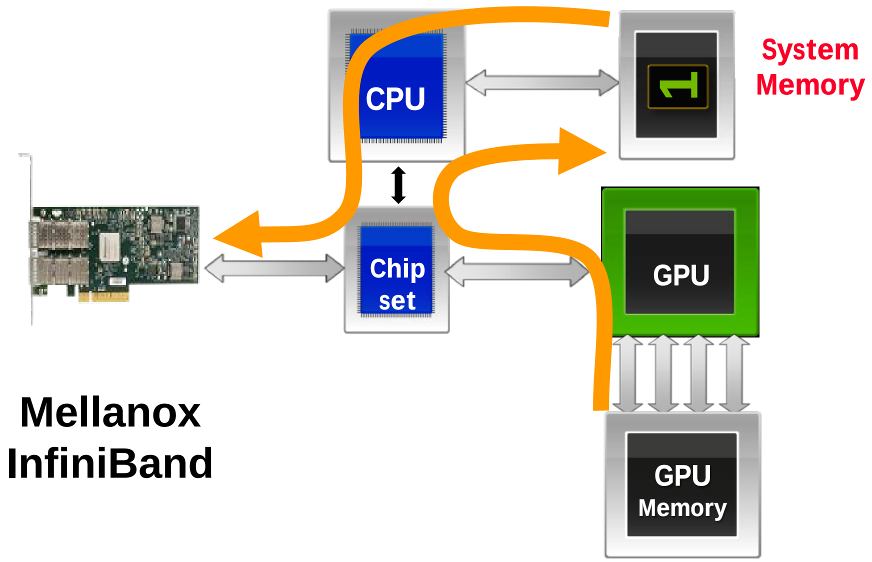


3D Torus in Cray B;ue Waters Machine

MELLANOX Efficient use of CPUs and GPUs

- GPU-direct
  - Works with existing NVIDIA Tesla and Fermi products
  - Enables fastest GPU-to-GPU communications
  - Eliminates CPU copy and write process in system memory
  - Reduces 30% of the GPU-to-GPU communication time

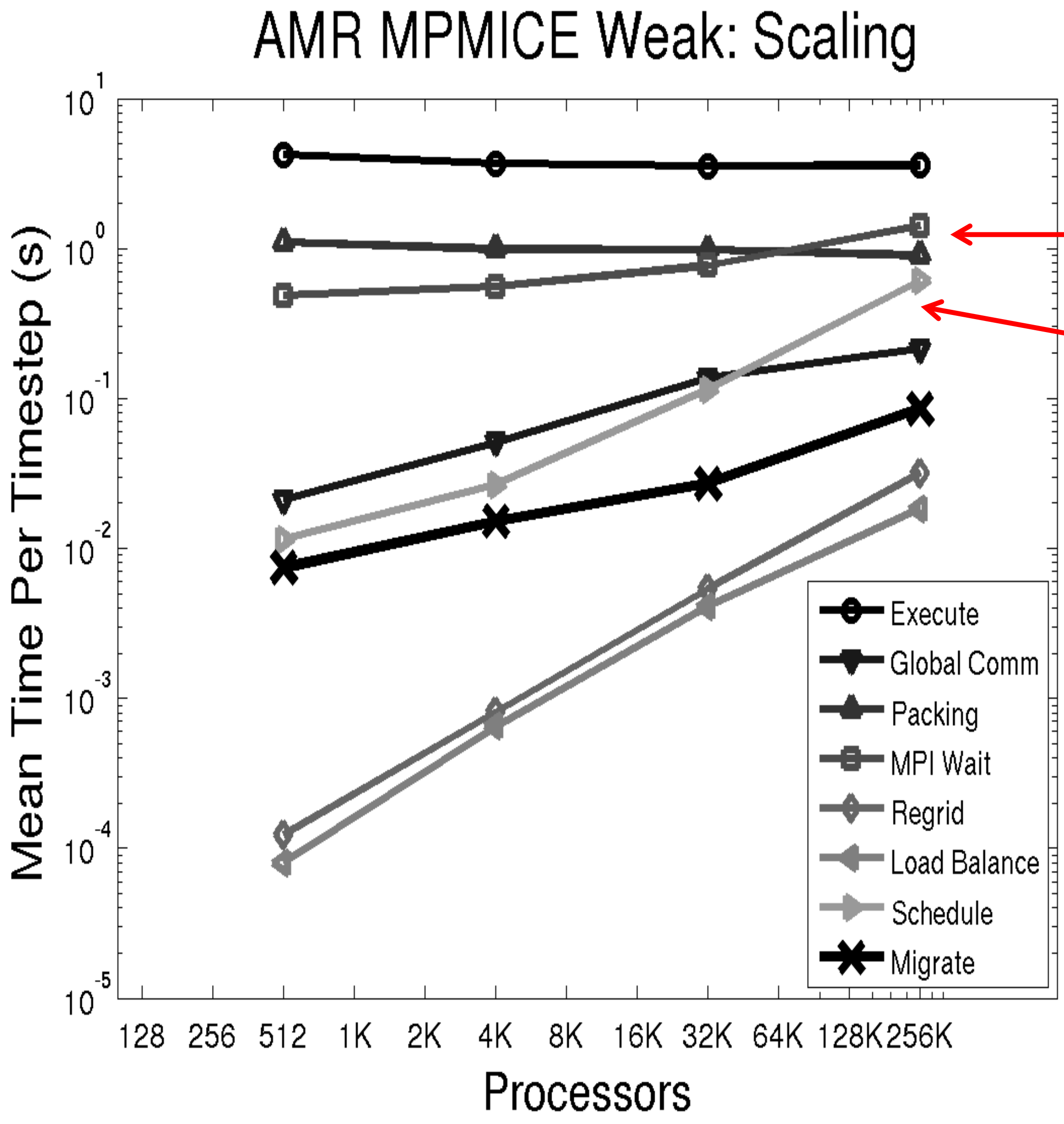




Latest Mellanox products have a latency of 1 micro second

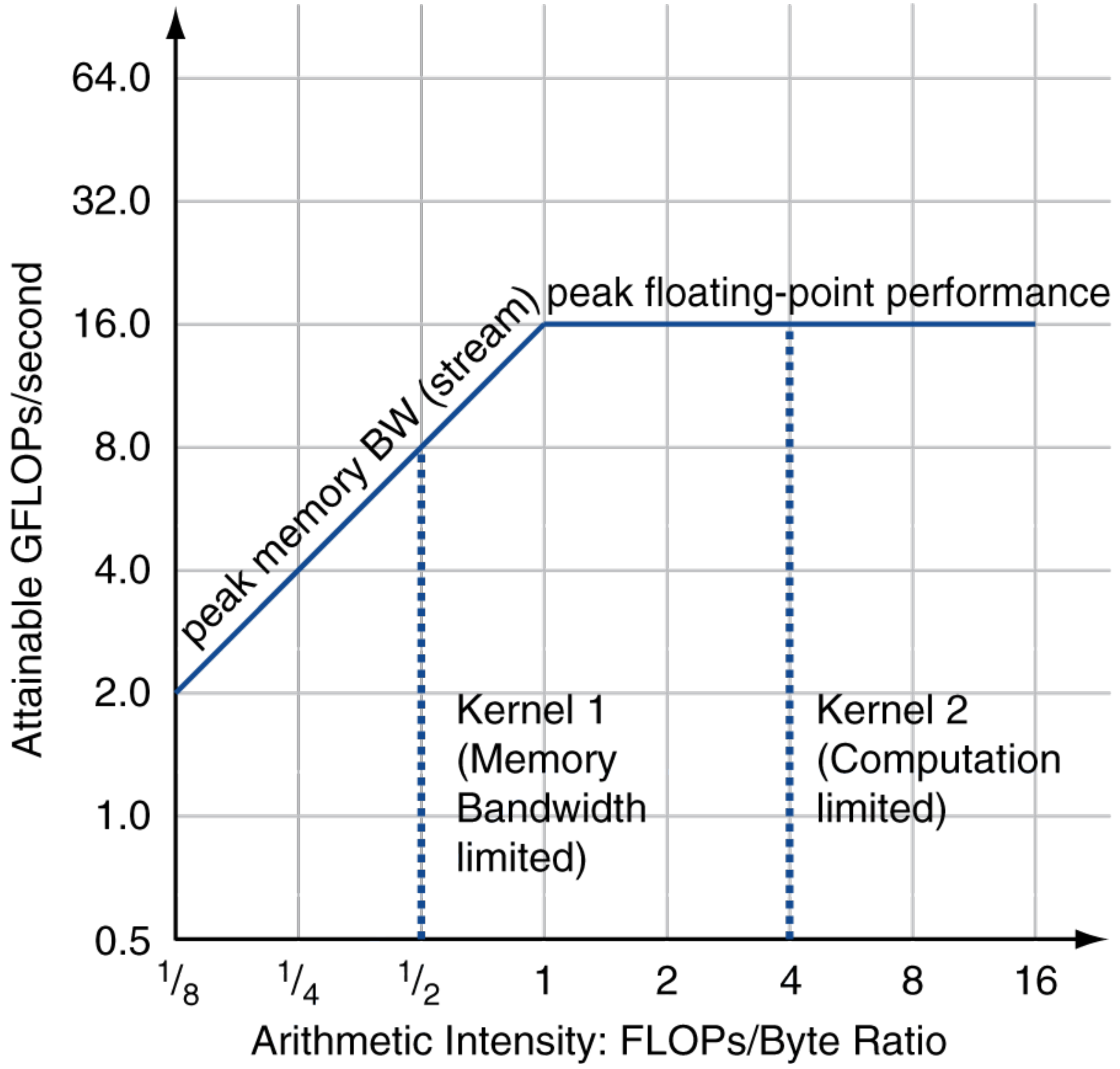
Architecture Effects on Performance

(i) Network delays or slow communications leads to MPI wait time and possibly scalability problems
(ii) Inability to move data through cache quickly enough leads to processors waiting
(iii) Inability to use advanced a arithmetic features of cores and/or gpus leads to slower than possible execution.



Effect of slow network communications on scalability – weak scalability break down due to growing overheads

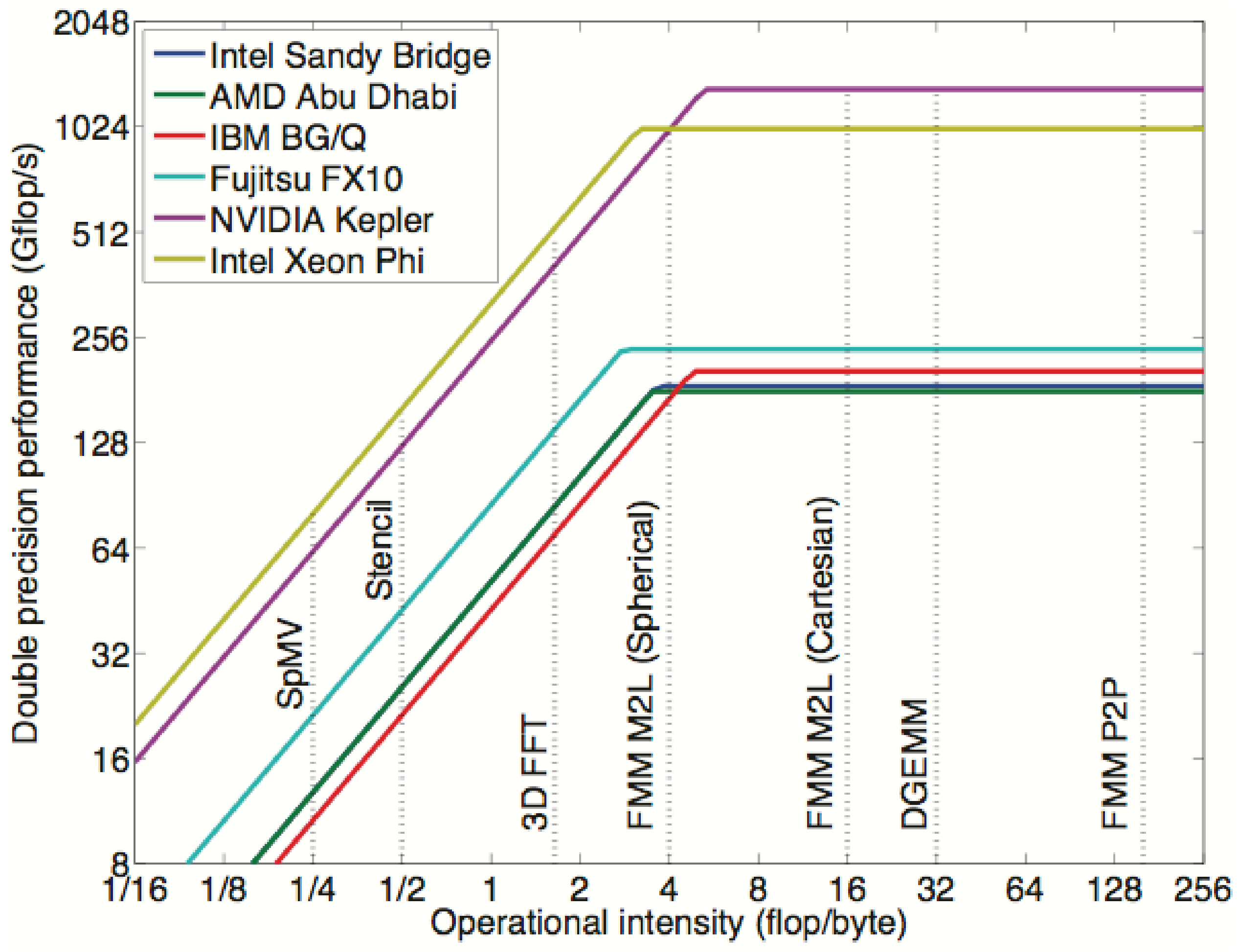
Roofline Diagram of Processor Performance



Effect of relatively slow core to cpu communications through the cache hierarchy

Attainable GPLOPs/sec
= Max (Peak Memory BW × Arithmetic Intensity, Peak FP Performance)

ROOFLINE MODEL FOR SOME MODEL ARCHITECTURES



3D FFT is Fast Fourier Transform in three space dimensions (see later in course)
DGEMM is matrix by matrix multiplication
FMM is Fast Multipole Method
SPMV is sparse matrix vector multiplication
Stencil is Laplace type finite difference calculations

Source Barba

BLAS

- **Basic Linear Algebra System**
- **Fundamental level of linear algebra libraries**
- **Many other libraries built on top of BLAS**
- **Three levels:**

Level 1- Vector-vector operations –

O(N) operations $\mathbf{y} \leftarrow \alpha\mathbf{x} + \mathbf{y}$

Level 2- Matrix-vector operations –

O(N*N) operations $\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$

Level 3- Matrix-matrix operations –

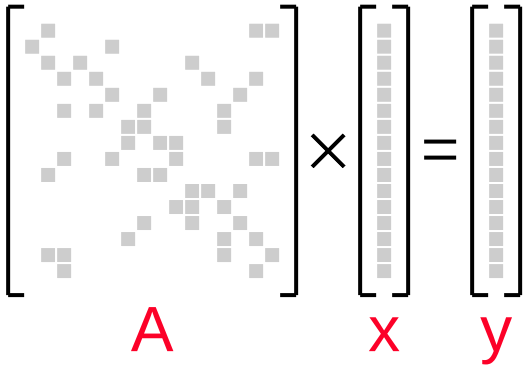
O(N*N*N) operations $C \leftarrow \alpha AB + \beta C$

A,B,C **are NxN matrices,** *x* **and** y **are N vectors**

α, β are constants

Sparse Matrix Vector Multiplication SPMV

- Sparse Matrix
  - Most entries are zero maybe only <5% are nonzero
  - Performance advantage in only storing/operating on the nonzeros
- Evaluate y=Ax
  - A is a sparse matrix
  - x & y are dense vectors



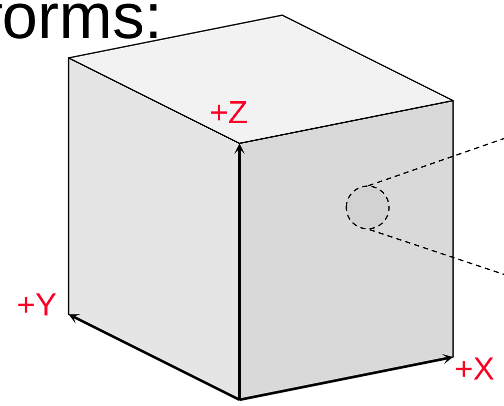
- Challenges
  - Irregular memory access to source vector
  - Difficult to load balance
  - Very low arithmetic intensity (often <0.166 flops/byte)
  - Compexity is O(N) only with complex irregular data structures = likely memory bound

- Simplest derivation of the Heat Equation with the Laplace operator (see later) results in a constant coefficient 7-point stencil

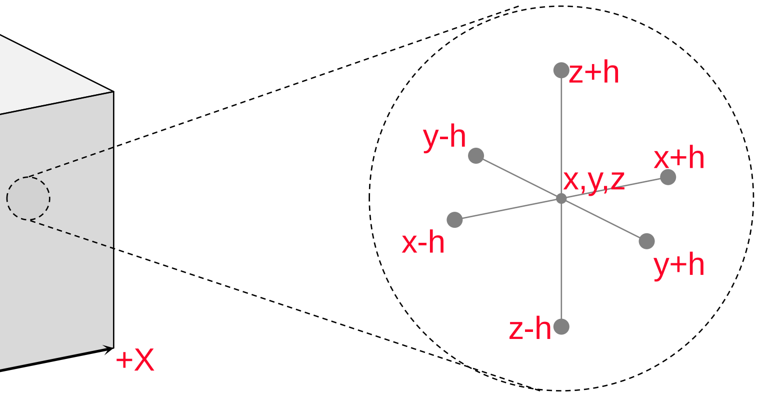
```
   for all x,y,z:
     u(x,y,z,t+dt) = alpha*u(x,y,z,t) + beta*(
                                u(x,y,z-h,t) + u(x,y-h,z,t) + u(x-h,y,z,t)
     +
                                u(x+h,y,z,t) + u(x,y+h,z,t) + u(x,y,z+h,t)
                                )
```

- **dt is time step and h is the stencil spacing**

- Clearly each stencil performs:
  - 8 floating-point operations
  - 8 memory references



PDE grid



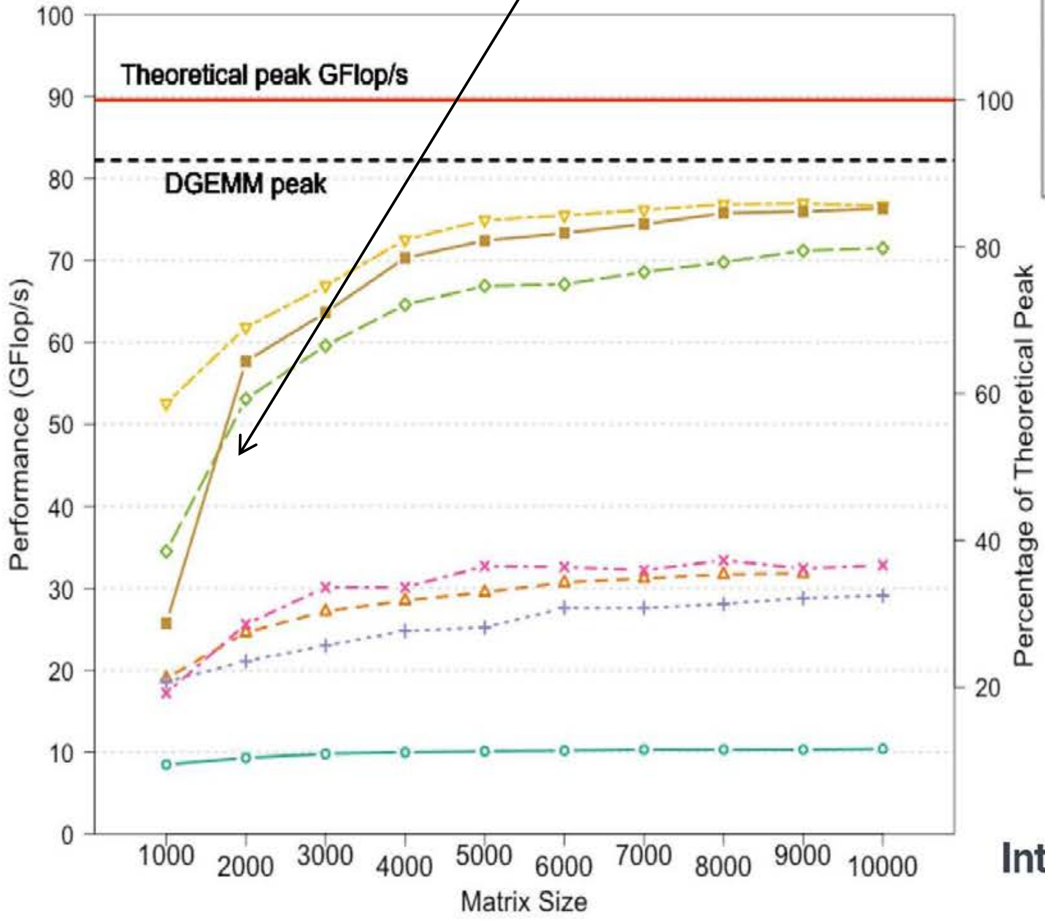
stencil for heat equation PDE

Stencil Calculations

Regularity of matrix access makes possible very efficient code

Examples of Roofline Performance

Cholesky Performance



Aparna Chandramowlishwaran

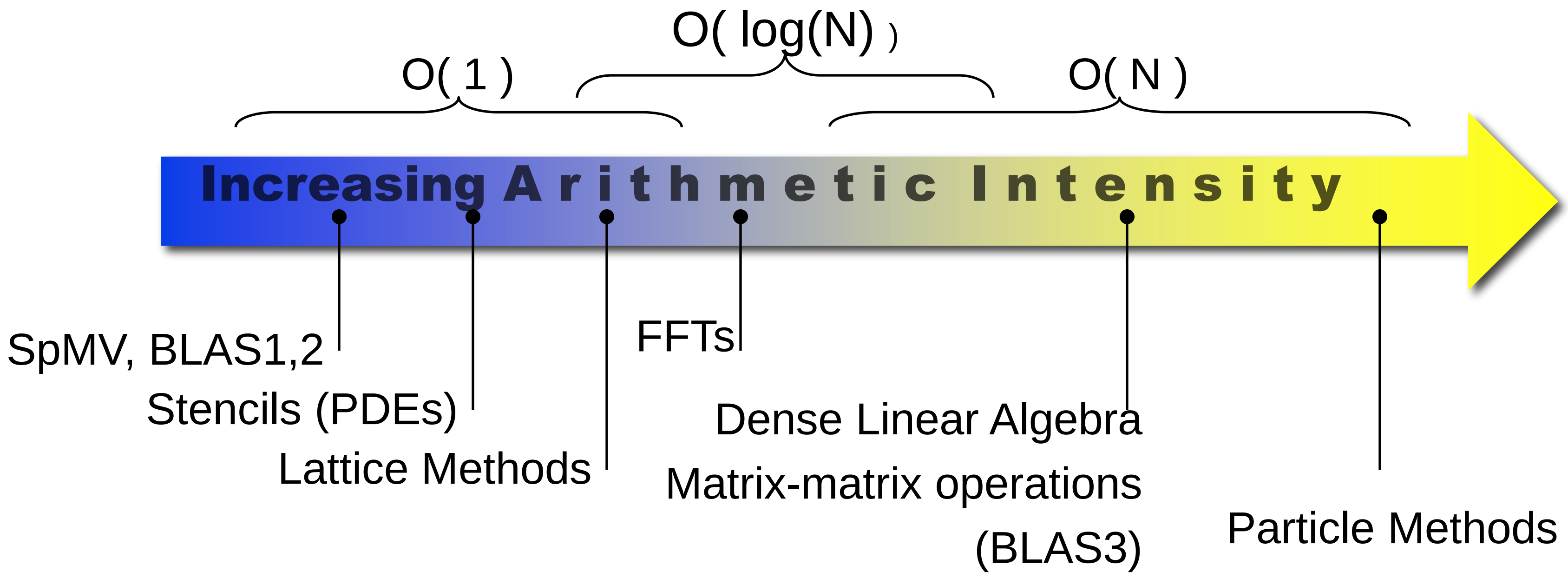
Rich Vuduc

(Georgia Tech)

Intel 2-socket x 4-core Nehalem @ 2.8 GHz + Intel MKL 10.2

CnC

Arithmetic Intensity per word



- **Arithmetic Intensity ~ Total Flops / Total DRAM Bytes**

SUMMARY

The rationale for using parallel computers is to apply multiple processors to solve larger problems faster.
Even on a simple serial processor :

- Performance of a program can be a complicated function of the architecture
- Slight changes in the architecture or program change the performance significantly
- To write even fast serial programs, need to consider architecture
- To write fast parallel programs need to pay even more attention to architecture and algorithms
- Even simple models of computation can help us design efficient serial and parallel algorithms