COMP 633 - Parallel Computing

Lecture 6
Tue Sep 7, 2021

SMM (1)
Memory Hierarchies and Shared Memory

Topics

- Memory systems
  - organization
  - caches and the memory hierarchy
  - influence of the memory hierarchy on algorithms
- Shared memory systems
  - Taxonomy of actual shared memory systems
    - UMA, NUMA, cc-NUMA

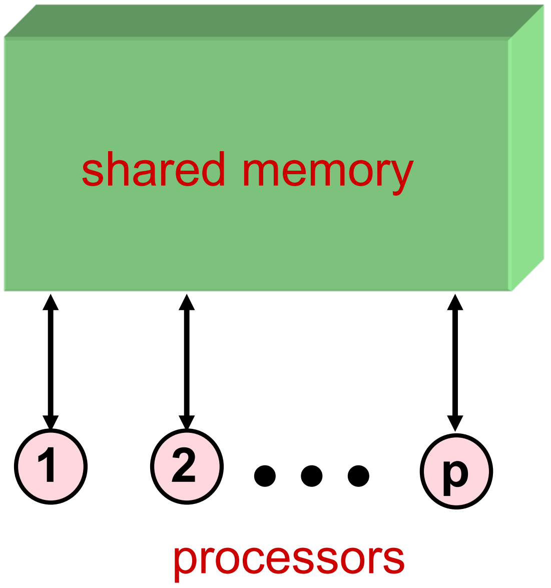
Recall PRAM shared memory system

- PRAM model
  - assumes access latency is constant, regardless of value of *p* or the size of memory
  - simultaneous reads permitted under CR model and simultaneous writes permitted under CW model

- Physically impossible to realize
  - processors and memory occupy physical space
    - speed of light limitations

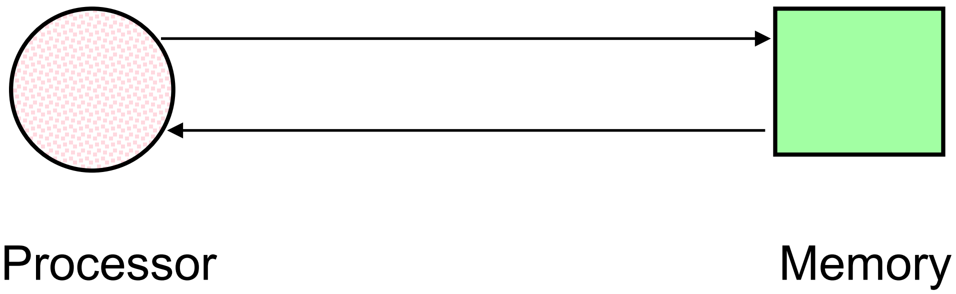
$$L = \Omega\left((p+m)^{1/3}\right)$$

  - CR / CW must be reduced to ER / EW
    - requires Ω(lg p) time in general case



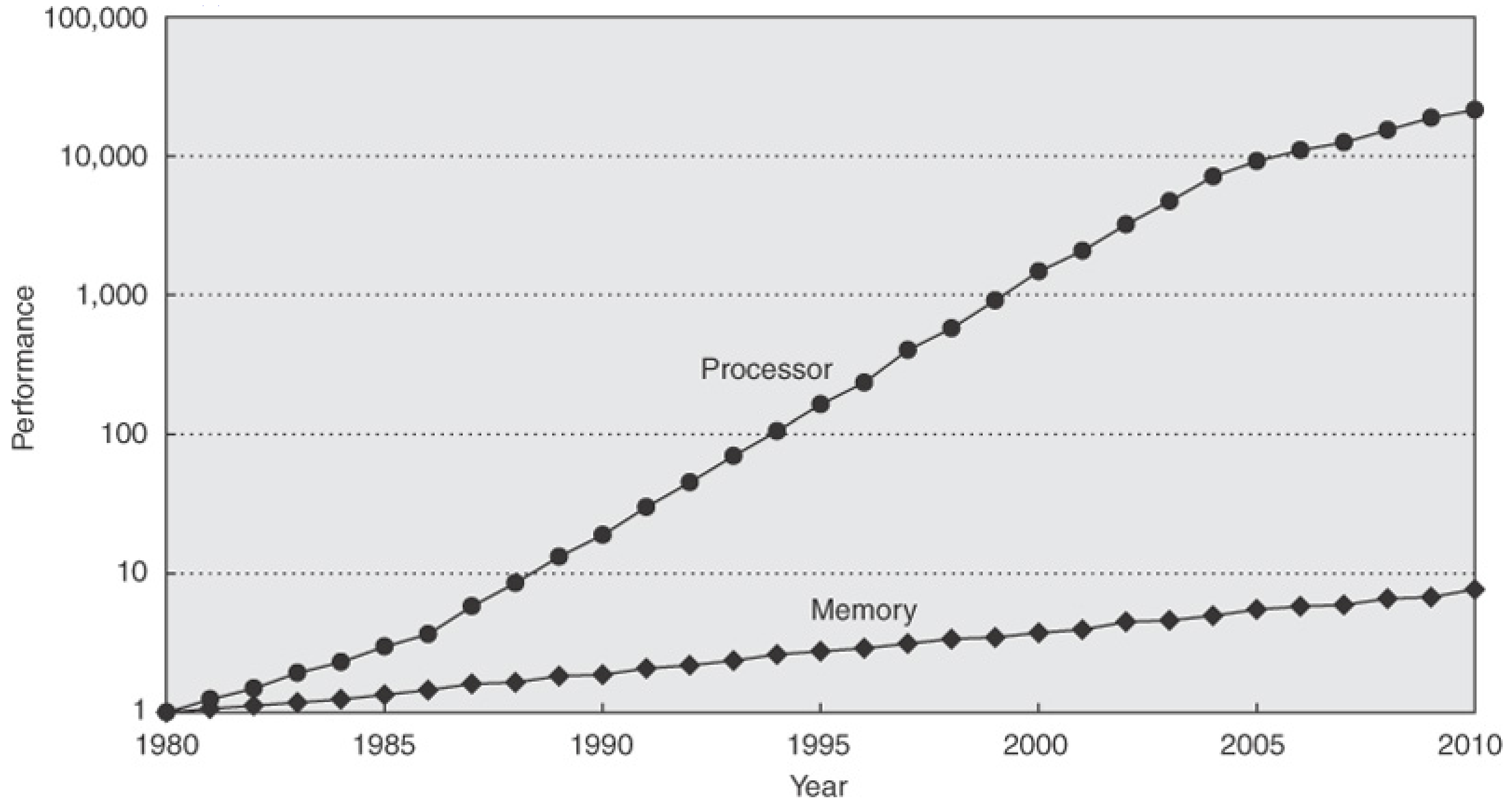
Anatomy of a processor $\leftrightarrow$ memory system

- Performance parameters of Random Access Memory (RAM)
  - latency L
    - elapsed time from presentation of memory address to arrival of data
      - address transit time
      - memory access time t_{mem}
      - data transit time
  - bandwidth W
    - number of values (e.g. 64 bit words) delivered to processor per unit time
      - simple implementation W ~ 1/L



Processor vs. memory performance

- The memory "wall"
  - Processors compute faster than memory delivers data
    - increasing imbalance $t_{\text{arith}} \ll t_{\text{mem}}$



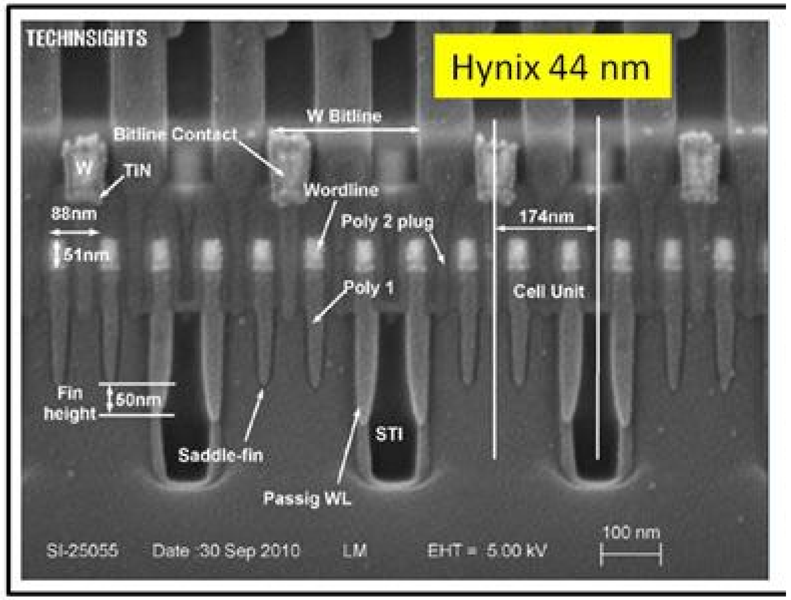
© 2007 Elsevier, Inc. All rights reserved.

Improving memory system performance

- Decrease latency L to memory
  - speed of light is a limiting factor
    - bring memory closer to processor

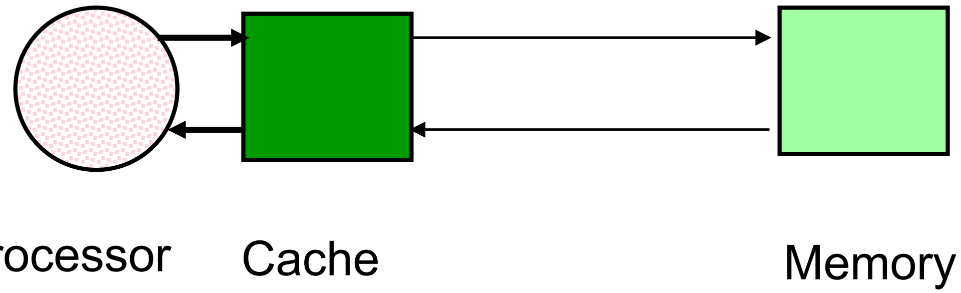
- Decrease memory access time by using 2D memory layout
  - access time $\propto s^{½}$ (VLSI)

- Use different memory technologies
  - DRAM (Dynamic RAM) 1 transistor per stored bit
    - High density, low power, low cost, but long access time
  - SRAM (Static RAM) 6 transistors per stored bit
    - Short access time, but low density, high power, and high cost.



Improving memory system performance (1)

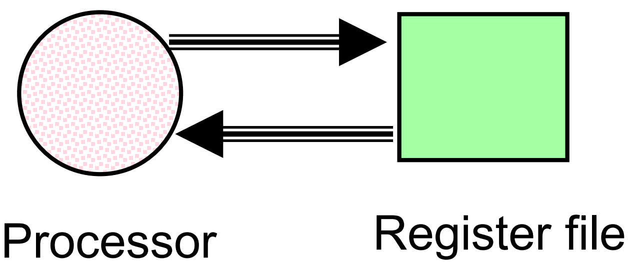
- Decrease latency using cache memory
  - low latency access to frequently used values, high latency for the remaining values



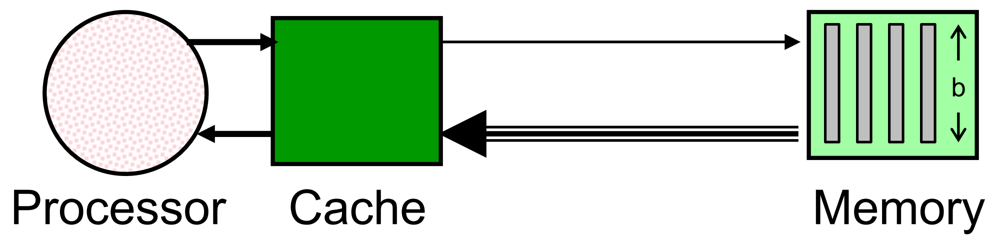
  - Example
    - 90% of references are to cache with latency L_1
    - 10% of references are to memory with latency L_2
    - average latency is $0.9L_1 + 0.1L_2$

Improving memory system performance (2)

- Increase bandwidth W
  - multiport (parallel access) memory
    - multiple reads, multiple exclusive writes per memory cycle
      - High cost, very limited scalability

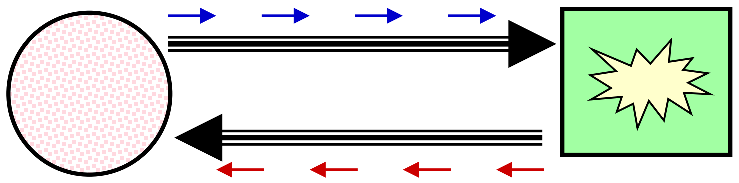


  - “blocked” memory
    - memory supplies block of size b containing requested word
      - supports *spatial locality* in cache access

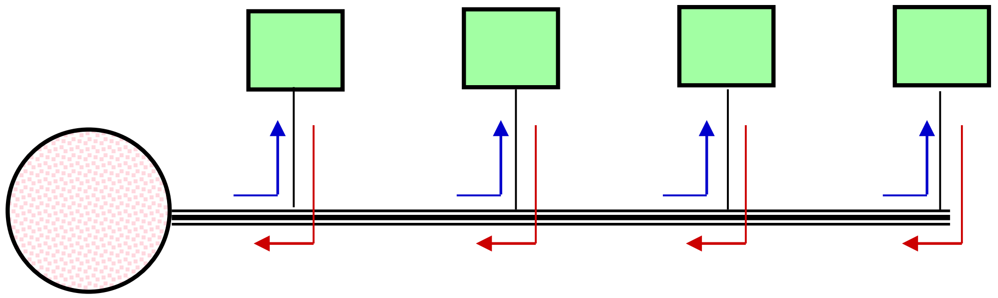


Improving memory system performance (2)

- Increase bandwidth W (contd)
  - pipeline memory requests
    - requires *independent* memory references



  - interleave memory
    - problem: memory access is limited by t_{mem}
    - use m separate memories (or memory banks)
    - W ~ m / L if references *distribute* over memory banks



Latency hiding

- *Amortize* latency using a pipelined interleaved memory system
  - k independent references in $\Omega(L + k \cdot t_{proc})$ time
    - O(L/k) amortized (expected) latency per reference

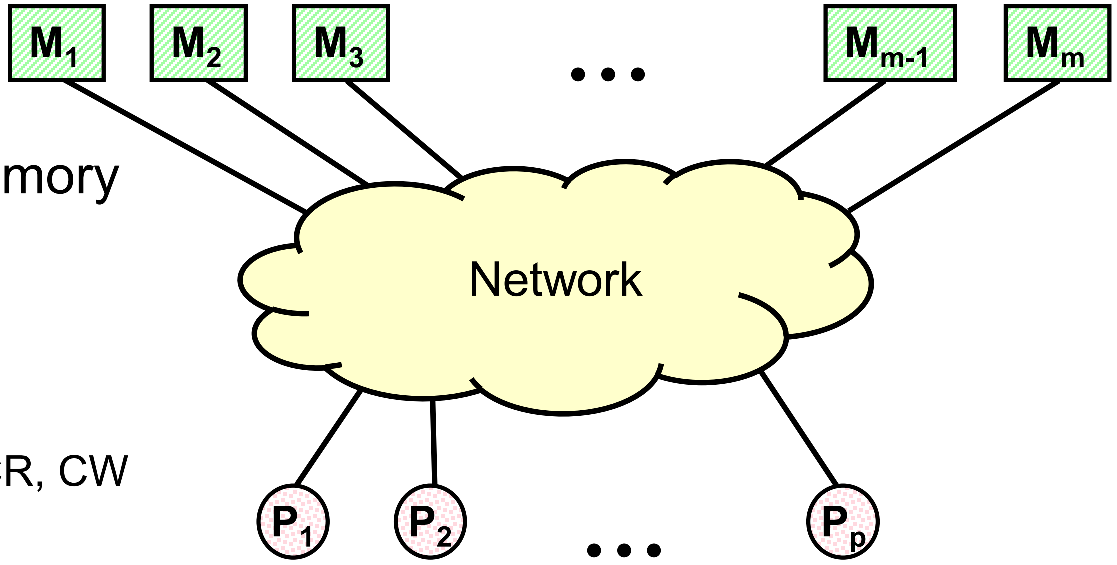
- Where do we get independent references?
  - out-of-order execution of independent load/store operations
    - found in most modern performance-oriented processors
    - partial latency hiding: k ~ 2 - 10 references outstanding

  - vector load/store operations
    - small vector units (AVX512)
      - vector length 2-8 words (Intel Xeon)
      - partial latency hiding
    - high-performance vector units (NEC SX-9, SX-Aurora)
      - vector length $k = L / t_{proc}$ (128 - 256 words)
      - crossbar network to highly interleaved memory (~ 16,000 banks)
      - full latency hiding: amortized memory access at processor speed

  - multithreaded operation
    - independent execution threads with individual hardware contexts
      - partial latency hiding: 2-way hyperthreading (Intel)
      - full latency hiding: 128-way threading with high-performance memory (Cray MTA)

Implementing the PRAM

- How close can we come to O(1) latency PRAM memory in practice?

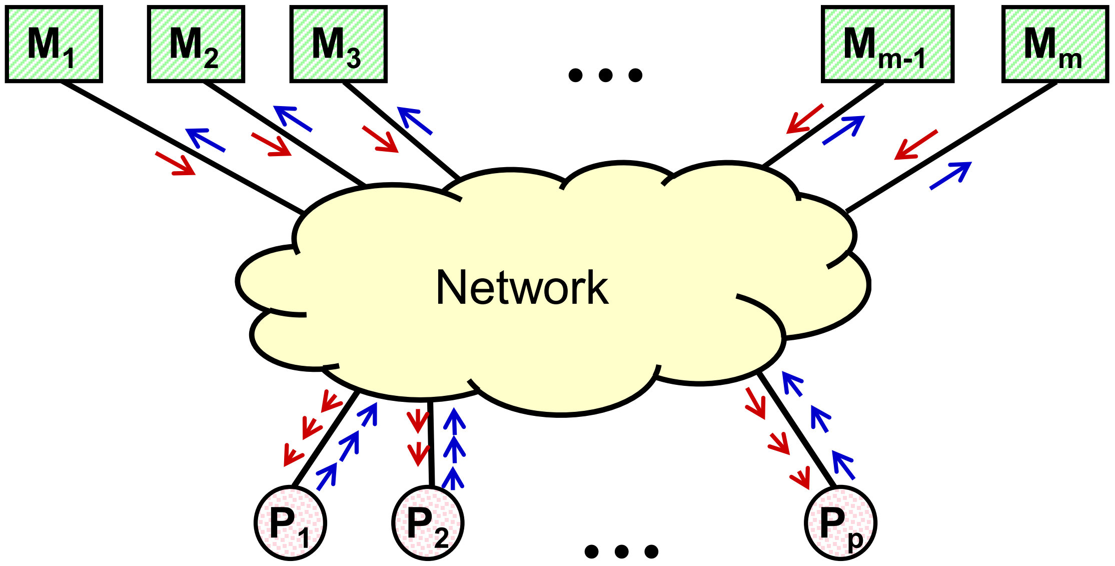


– requires processor to memory network
  - latency L = sum of
    – twice network latency
    – memory cycle time
    – serialization time for CR, CW
  - L increases with m, p
    – L too large with current technology

– examples
  - NYU Ultracomputer (1987), IBM RP3 (1991), SBPRAM (1999)
    – logarithmic depth combining network eliminates memory contention time for CR, CW
      » Ω(lg p) latency in network is prohibitive

Implementing PRAM – a compromise

- Using latency hiding with a high-performance memory system
  - implements *p·k* processor EREW PRAM slowed down by a factor of *k*
    - use $m \geq p\ (t_{mem}\ /\ t_{proc})$ memory banks to match memory reference rate of p processors
    - total latency 2L for $k = L / t_{proc}$ *independent random* references at each processor
    - $O(t_{proc})$ amortized latency per reference at each processor
  - unit latency degrades in the presence of concurrent reads/writes



  - Bottom line: doable but very expensive and only limited scaling in p

Memory systems summary

- Memory performance
  - Latency is limited by physics
  - Bandwidth is limited by cost

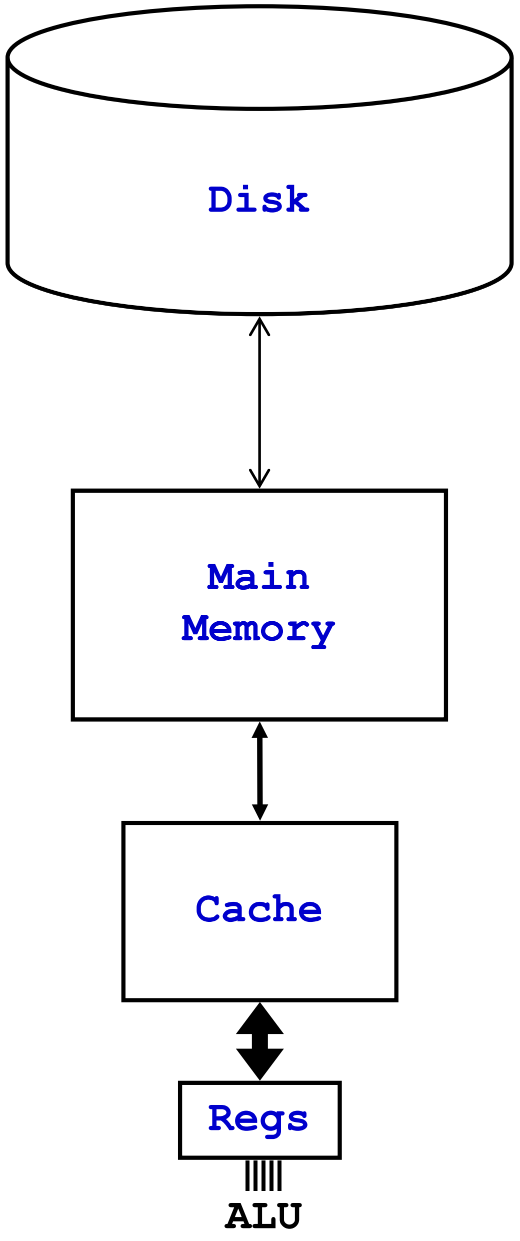
- Cache memory: low latency access to some values
  - caching frequently used values
    - rewards *temporal locality* of reference
  - caching consecutive values
    - rewards *spatial locality* of reference
  - decrease *average* latency
    - 90 fast references, 10 slow references: effective latency = $0.9L_1 + 0.1L_2$

- Parallel memories
  - 100 *independent* references ≈ 100 fast references
  - relatively expensive
  - requires parallel processing

Simple uniprocessor memory hierarchy

- Each component is characterized by
  - capacity
  - block size
  - (associativity)
- Traffic between components is characterized by
  - access latency
  - transfer rate (bandwidth)
- Example:
  - IBM RS6000/320H (ca. 1991)

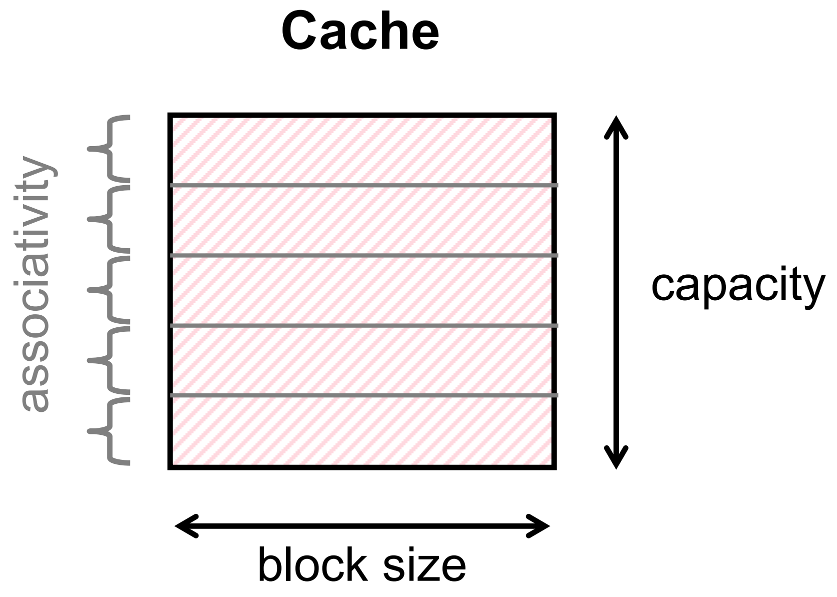
Storage component	Latency (cycles)	Transfer Rate (words [8B] / cycle)
Disk	1,000,000	0.001
Main memory	60	0.1
Cache	2	1
Registers	0	3



Cache operation

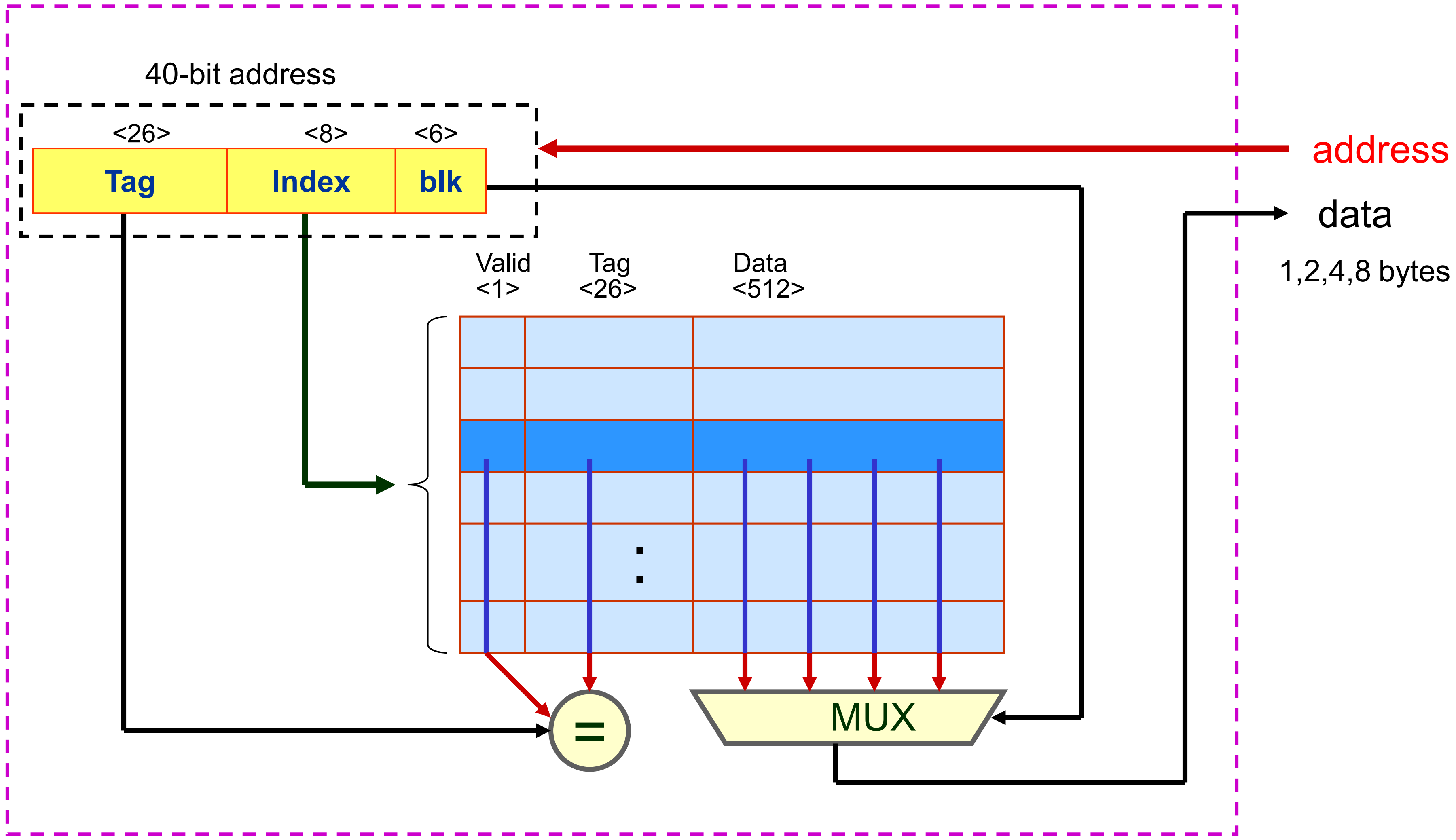
- ABC cache parameters
  - associativity
  - block size
  - capacity

- CCC performance model
  - cache misses can be
    - compulsory
    - capacity
    - conflict



Cache operation: read

associativity = 256-way
block size = 64 bytes (512b)



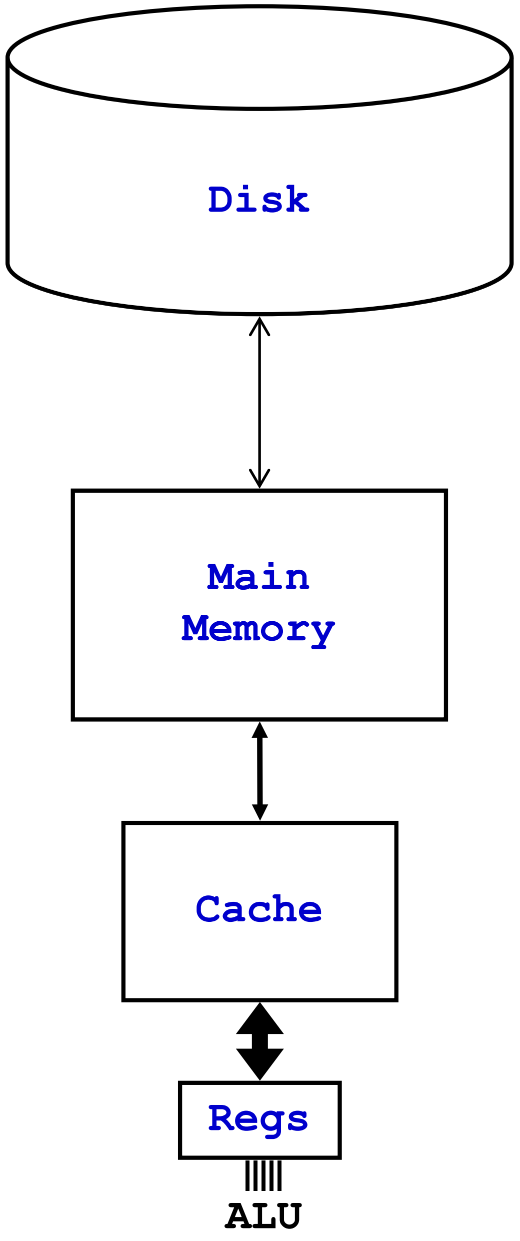
The changing memory hierarchy

- IBM RS6000 320H - 25 MHz (1991)

Storage component	Latency (cycles)	Transfer Rate (words [8B] / cycle)
Disk	1,000,000	0.001
Main memory	60	0.1
Cache	2	1
Registers	1	3

- Intel Xeon 61xx [per core @3GHz] (2017)

Storage component	Latency (cycles)	Transfer Rate (words [8B] / cycle)
HDD	18,000,000	0.00007
SSD	300,000	0.02
Main memory	250	0.2
L3 Cache	48	0.5
L2 Cache	12	1
L1 Cache	4	2
Registers	1	6



Computational Intensity: a key metric limiting performance

- Computational intensity of a problem

$$I = \frac{\text{(total \# of arithmetic operations required)}}{\text{(size of input + size of result)}} \quad \begin{array}{l}\text{in flops} \\ \text{in 64-bit words}\end{array}$$

- BLAS - Basic Linear Algebra Subroutines
  - Asymptotic performance limited by computational intensity
    - $A, B, C \in \Re^{n \times n}$ $\quad x, y \in \Re^{n}$ $\quad a \in \Re$

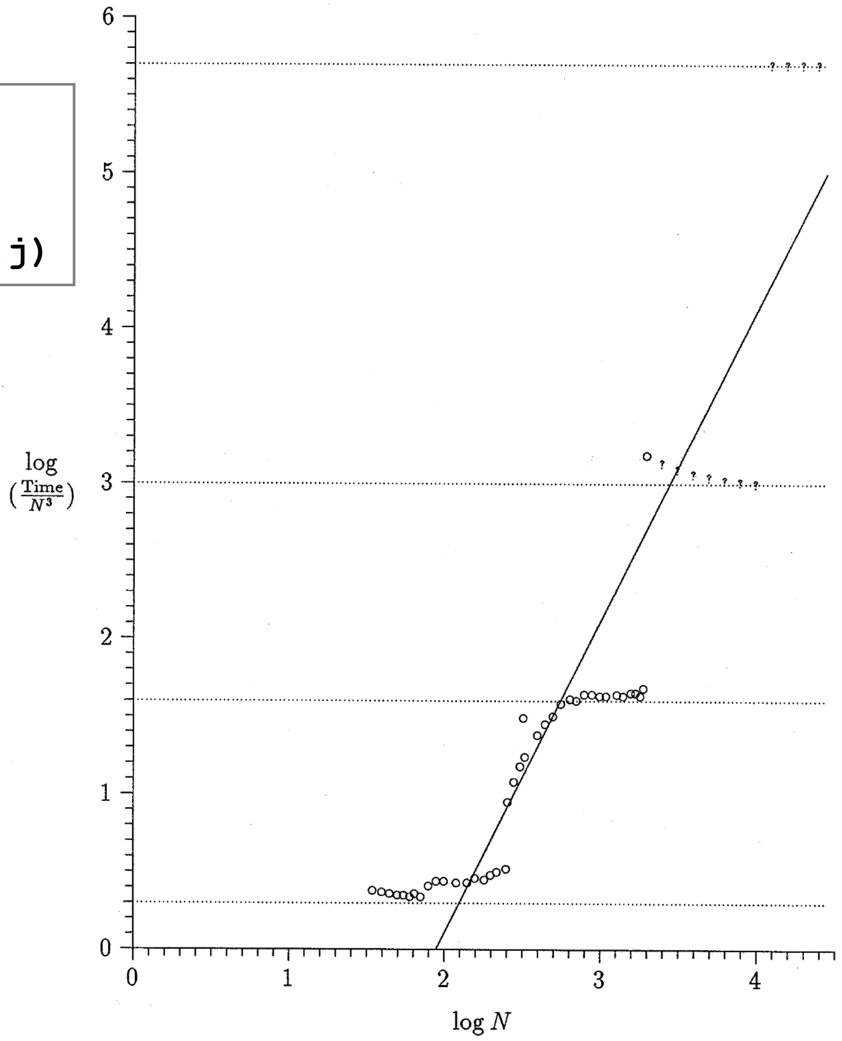
	name	defn	flops	refs	I
BLAS 1	scale	$y = ax$	n	$2n$	0.5
	triad	$y = ax + y$	$2n$	$3n$	0.67
	dot product	$x \bullet y$	$2n$	$2n$	1
BLAS 2	Matrix-vector	$y = y + Ax$	$2n^2+n$	n^2+3n	~ 2
	rank-1 update	$A = A + xy^T$	$2n^2$	$2n^2+2n$	~ 1
BLAS 3	Matrix product	$C = C + AB$	$2n^3$	$4n^2$	$n/2$

Effect of the memory hierarchy on execution time

- $C^{NxN} = A^{NxN} \bullet B^{NxN}$ naïve implementation

```
do i = 1,N
   do j = 1,N
      do k = 1,N
         C(i,j) = C(i,j) + A(i,k)*B(k,j)
```

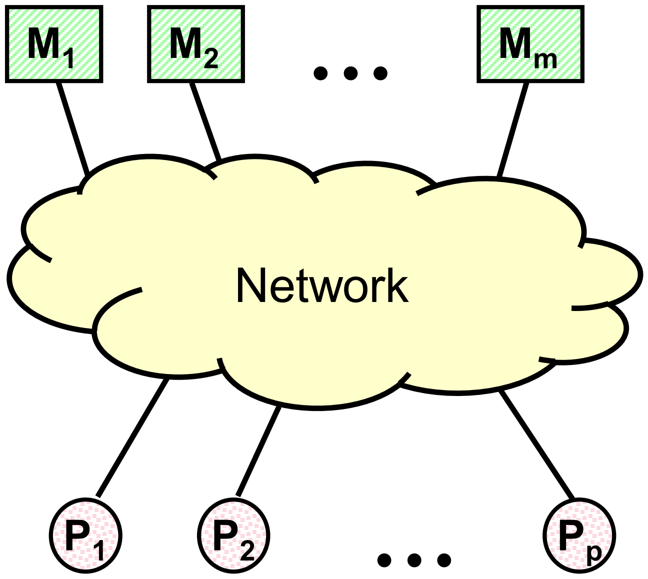
- Machine
  - simple L1 cache
    - block size = 16 words
    - capacity = 512 blocks
    - fully associative
  - main memory
    - 4K pages
- Layout of A,B,C in memory
  - Fortran: column-major order
- RAM model suggests $O(N^3)$ run time
  - actual time follows $O(N^5)$ growth!



Performance of naive $N \times N$ matrix multiply on an IBM RS6000/320 uniprocessor. Time in clock cycles per multiply-add (note $\log_{10}$ scales). Source: Alpern *et al.*, “The Uniform Memory Hierarchy Model of Computation", *Algorithmica*, 1994

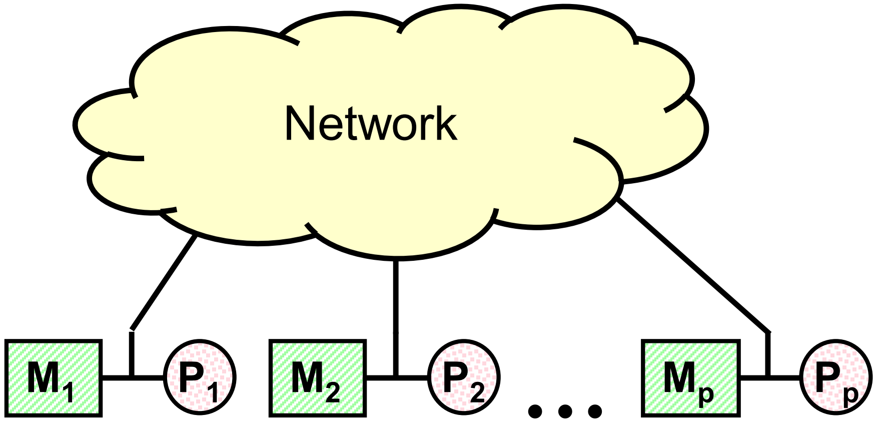
Shared memory taxonomy

- Uniform Memory Access (UMA)
  - Processors and memory separated by network
  - All memory references cross network
  - Only practical for machines with full latency hiding
    - Parallel vector processors, multi-threaded processors
    - Expensive, rarely available in practice



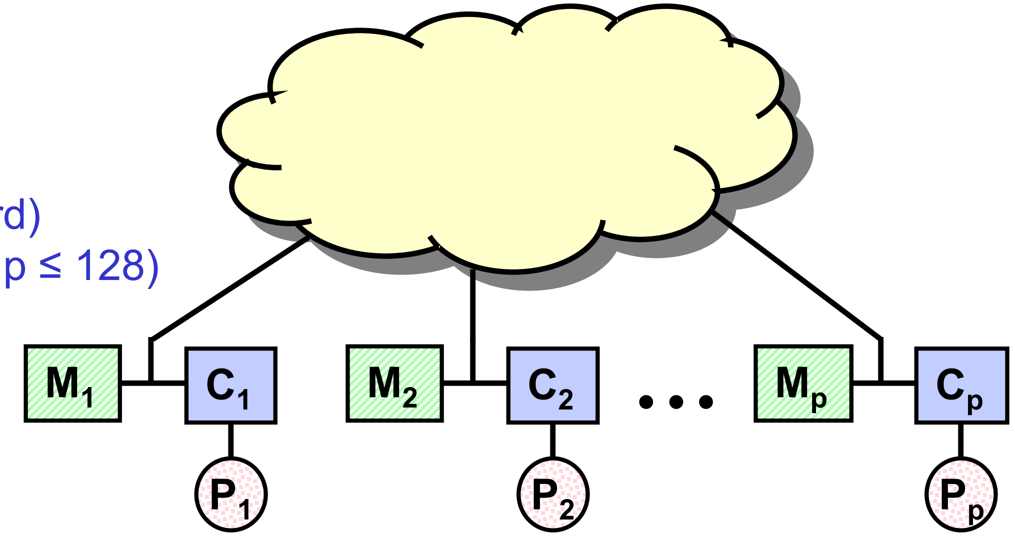
Shared memory taxonomy

- Non-Uniform Memory Access (NUMA)
  - Memory is partitioned across processors
  - References are local or non-local
    - Local references
      - low latency
    - Non-local references
      - high latency
    - non-local : local latency
      - large
  - Examples
    - BBN TC2000 (1989)
  - Poor performance unless extreme care is taken in data placement



Combining (N)UMA with cache memories

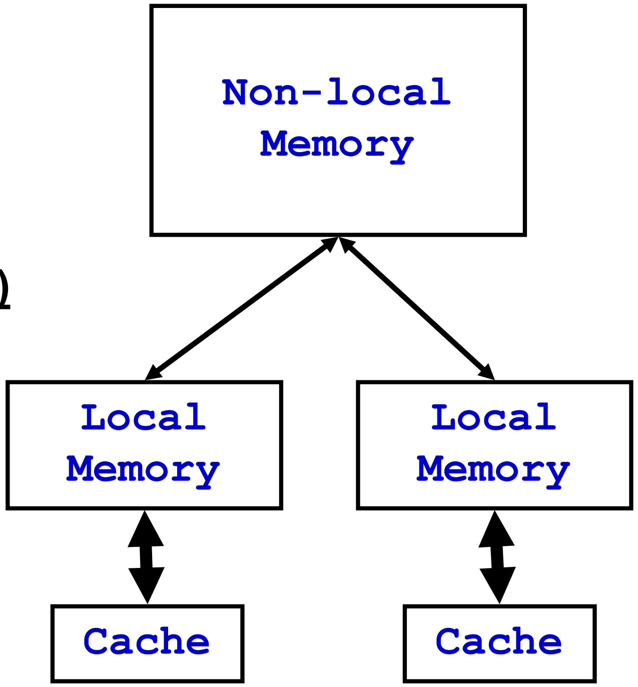
- Processor-local caches
  - Cache all memory references
  - Must reflect changes in value due to other processors in system
  - Cache-misses
    - Usual: compulsory, capacity, and conflict misses
    - New: *coherence* misses
- Cache-coherent UMA examples
  - Conventional PC-based SMP systems
    - Network is a shared bus
    - Limited scaling ($p \leq 4$)
    - mostly extinct
  - Server-class machines
    - Dual or Quad socket (single card)
    - Intel Xeon or AMD EPYC ($20 \leq p \leq 128$)
    - prevalent
- Cache-coherent NUMA examples
  - scales to larger processor count
    - SGI UltraViolet ($p \sim 1024$)
    - rare



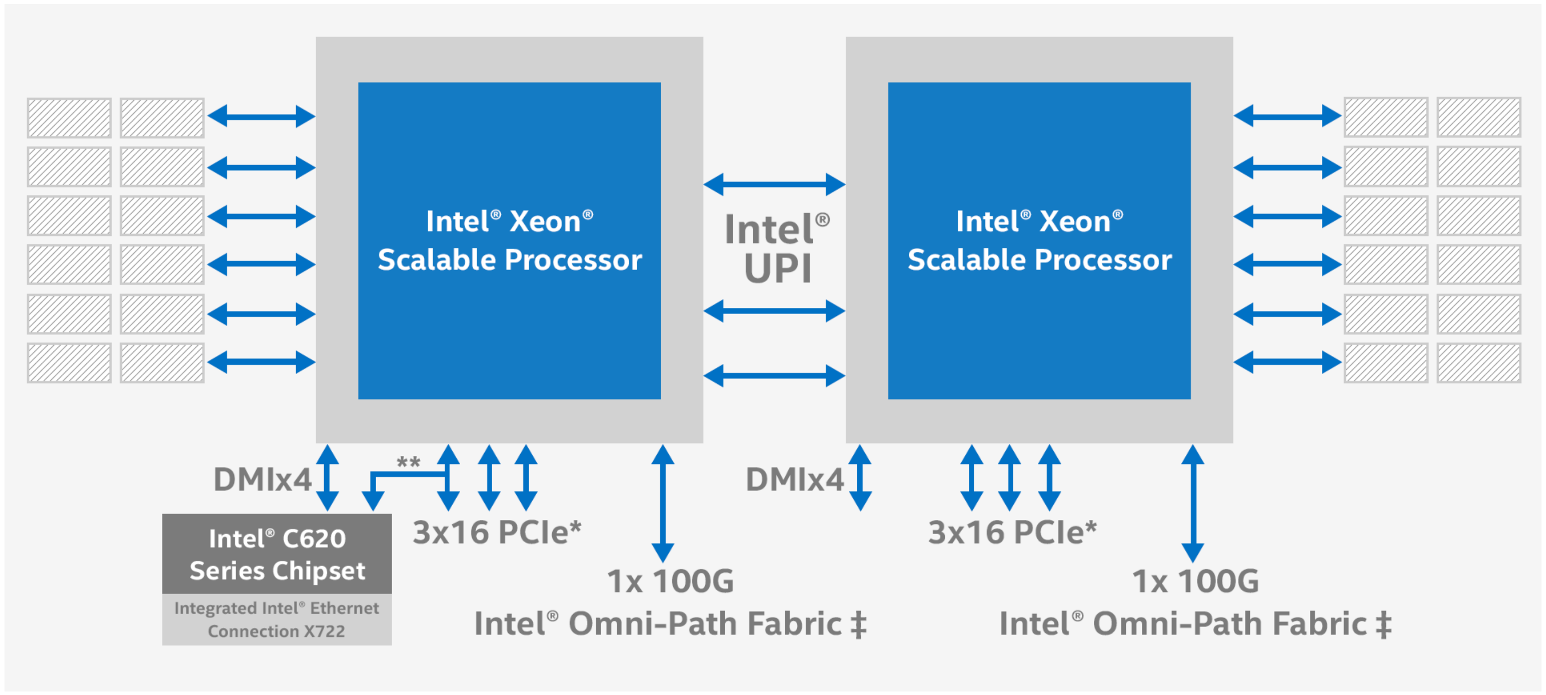
Incorporating *shared* memory in the hierarchy

- Non-local shared memory
  - can be viewed as additional level in processor-memory hierarchy
- Shared-memory parallel programming
  - extension of memory hierarchy techniques
  - goal:
    - concurrent transfer through parallel levels

Storage component	Latency (cycles)	Transfer Rate (words [8B] / cycle)
Disk	**1,000,000**	**0.001**
Non-local memory	**180 - 500**	**0.1 - 0.01**
Local memory	**60**	**0.1**
Cache	**2**	**1**
Registers	**0**	**3**



Modern shared-memory server: Intel Xeon series



Intel® Xeon® Scalable Processor
Intel® UPI
Intel® Xeon® Scalable Processor
DMIx4
**
3x16 PCIe*
DMIx4
3x16 PCIe*
Intel® C620 Series Chipset
Integrated Intel® Ethernet Connection X722
1x 100G Intel® Omni-Path Fabric ‡
1x 100G Intel® Omni-Path Fabric ‡

AMD Infinity

- Speed of light inconveniently slow!
  - miniaturize size of memory and processors
- Single card server
  - 7 nm process technology
  - 64 – 256 cores total,
  - 4 TB memory