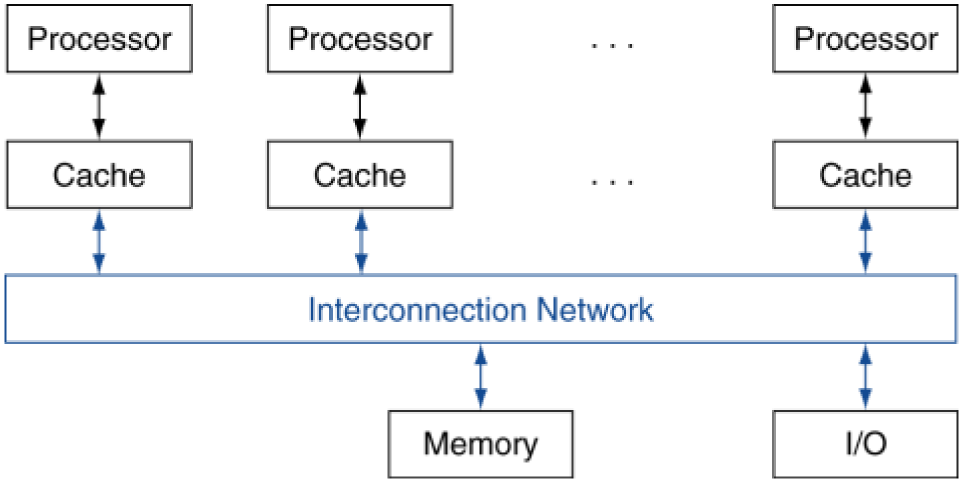
Message-Passing Multiprocessors

Computer Organization

TÉCNICO LISBOA

Review: Shared Memory Multiprocessors (SMP)

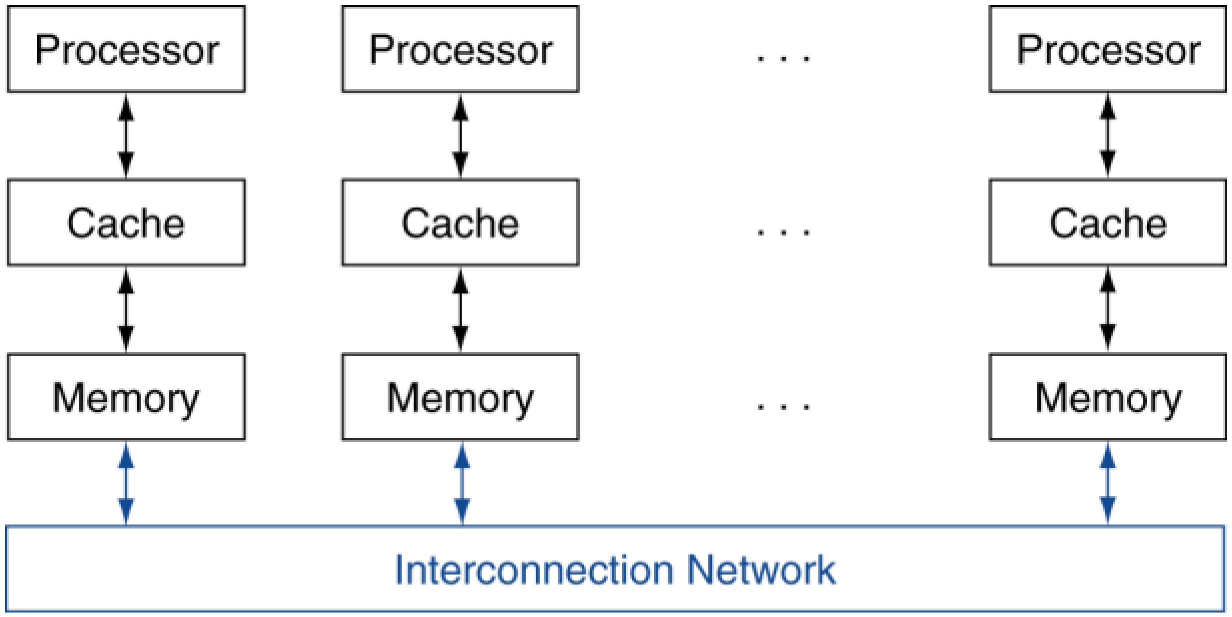
- Single address space shared by all processors
- Processors coordinate/communicate through shared variables in memory (via loads and stores)
  - Use of shared data must be coordinated via synchronization primitives (locks) that allow access to data to only one processor at a time



- They come in two styles
  - Uniform memory access (UMA) multiprocessors
  - Nonuniform memory access (NUMA) multiprocessors

Message Passing Multiprocessors (MPP)

- Each processor has its own private address space
- Processors share data by *explicitly* sending and receiving information (message passing)
- Coordination is built into message passing primitives (message send and message receive)



Loosely Coupled Clusters

- Network of independent computers
  - Each has private memory and OS
  - Connected using I/O system
    - E.g., Ethernet/switch, Internet
- Suitable for applications with independent tasks
  - Web servers, databases, simulations, …
- High availability, scalable, affordable
- Problems
  - Administration cost (prefer virtual machines)
  - Low interconnect bandwidth
    - c.f. processor/memory bandwidth on an SMP

Grid Computing

- Separate computers interconnected by long-haul networks
  - E.g., Internet connections
  - Work units farmed out, results sent back

- Can make use of idle time on PCs
  - E.g., SETI@home, World Community Grid

Sum Reduction (Again)

- Sum 100,000 elements on 100 processors
- First distribute 1000 numbers to each
  - Then do partial sums

```
sum = 0;
for (i = 0; i < 1000; i = i + 1)
   sum = sum + A[i];
```

- Reduction
  - Half the processors send, other half receive and add
  - The quarter send, quarter receive and add, …

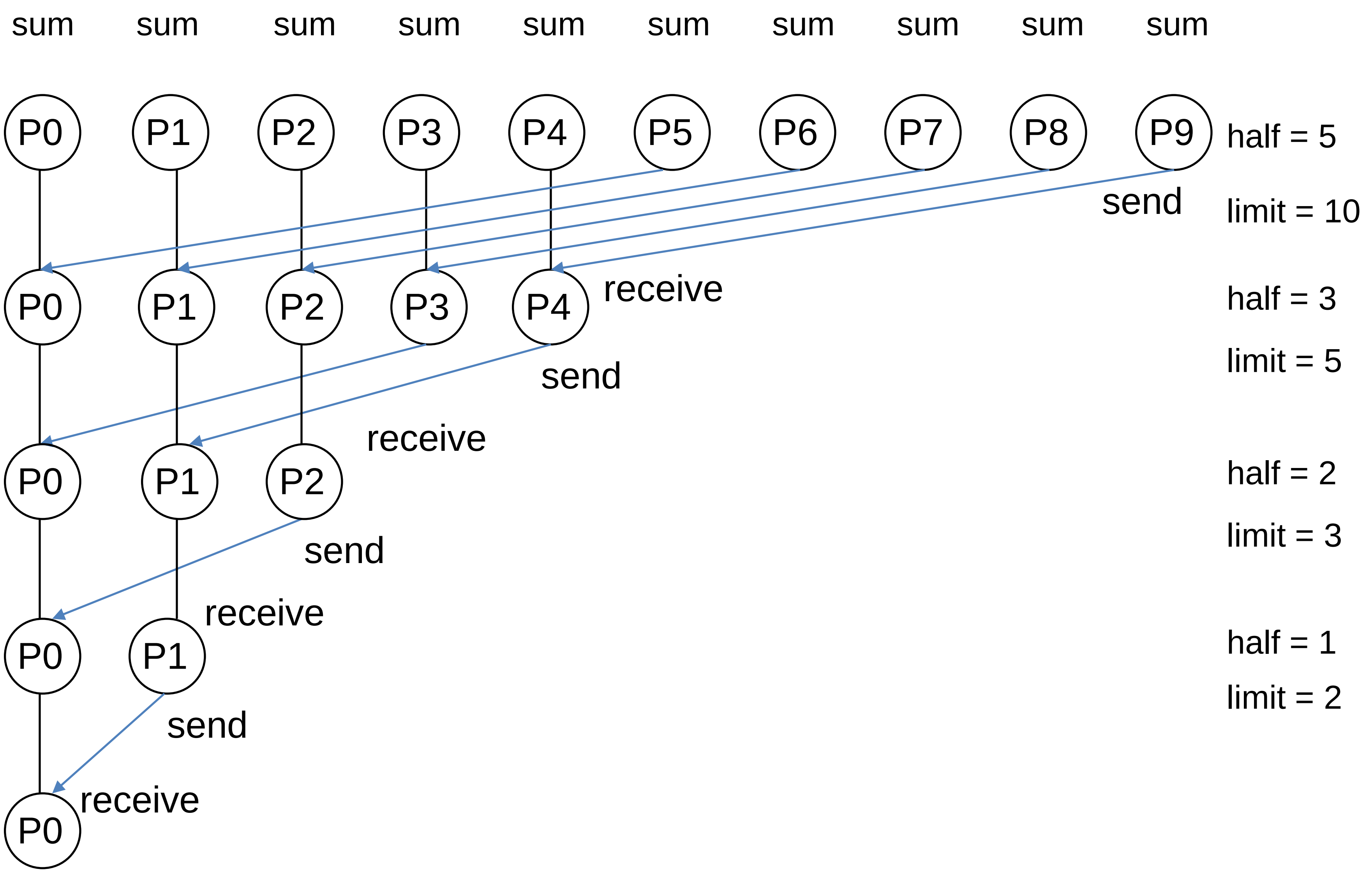
Sum Reduction (Again)

- Given send() and receive() operations

```
limit = 100; half = 100;  // 100 processors
repeat
  half = (half+1)/2;        // send vs. receive
                            // dividing line
  if (Pn >= half && Pn < limit)
    send(Pn - half, sum);
  if (Pn < (limit/2))
    sum = sum + receive();
  limit = half;       // upper limit of senders
until (half == 1);  // exit with final sum
```

  – Send/receive also provide synchronization

An Example with 10 Processors



Pros and Cons of Message Passing

- Message sending and receiving is much slower than addition, for example
- But message passing multiprocessors are much easier for hardware designers to design
  - Don't have to worry about cache coherency for example
- The advantage for programmers is that communication is explicit, so there are fewer "performance surprises" than with the implicit communication in cache-coherent SMPs.
  - Message passing standard MPI (www.mpi-forum.org)
- However, it's harder to port a sequential program to a message passing multiprocessor since every communication must be identified in advance.
  - With cache-coherent shared memory the hardware figures out what data needs to be communicated

MPI – Message Passing Interface

MPI is the de facto standard for scientific programming on distributed memory parallel computers.

- Library of routines that enable message passing applications
- Interface specification, not a concrete implementation

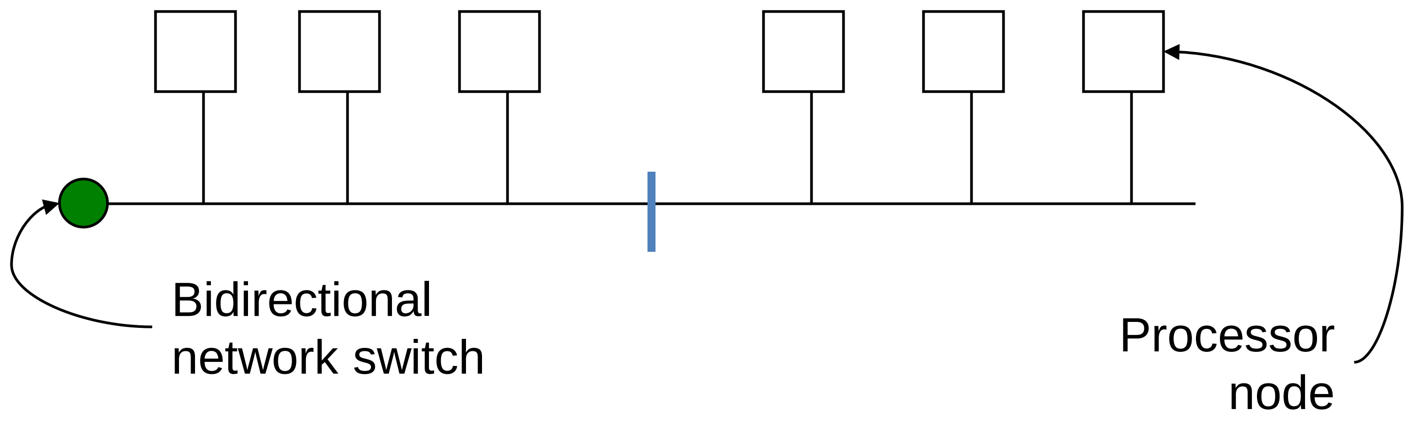
MPI basically manages a set of tasks and maintains communication channels between them.

Developed by the MPI Forum, a voluntary organization representing industry, government labs and academia.

Network Performance Metrics

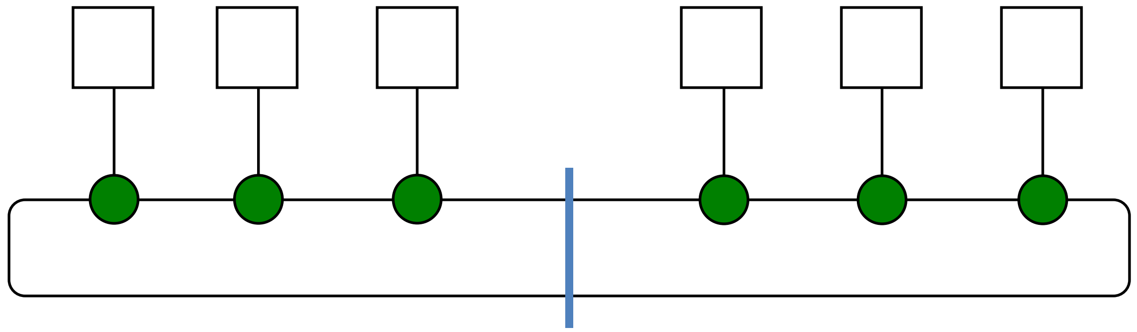
- Network cost
  - number of switches
  - number of (bidirectional) links on a switch to connect to the network (plus one link to connect to the processor)
  - width in bits per link, length of link wires (on chip)
- Network bandwidth (NB) – represents the best case
  - bandwidth of each link **x** number of links
- Bisection bandwidth (BB) – closer to the worst case
  - divide the machine in two parts, each with half the nodes and sum the bandwidth of the links that cross the dividing line
- Other interconnection network (IN) performance issues
  - Latency on an unloaded network to send and receive messages
  - Throughput: maximum # of messages transmitted per unit time
  - # routing hops worst case, congestion control and delay, fault tolerance, power efficiency

Bus IN



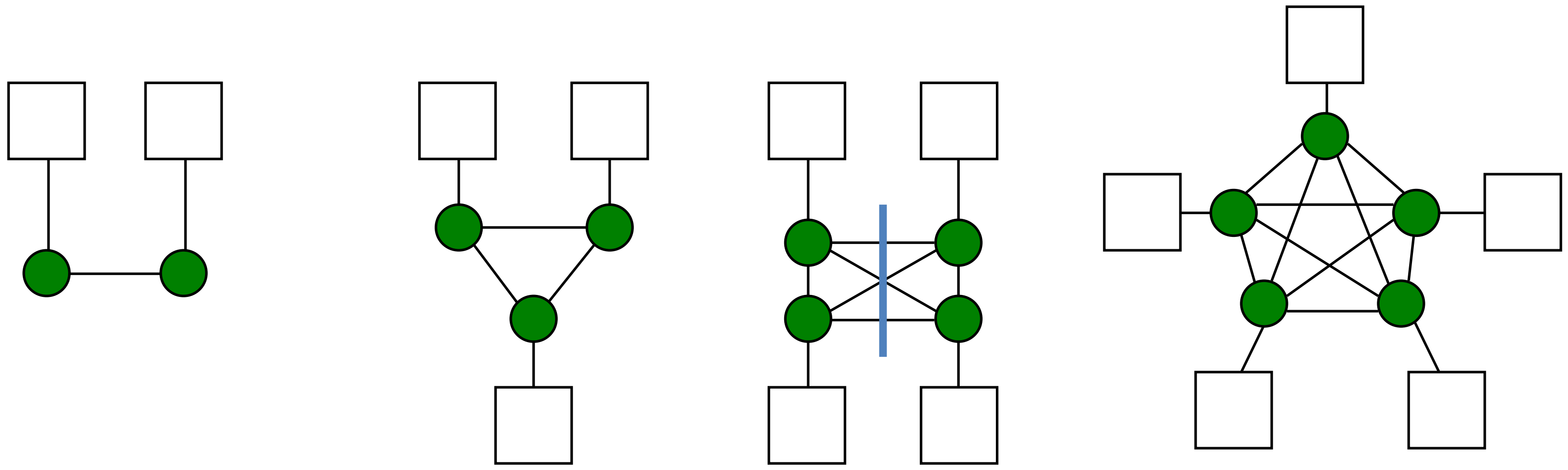
- N processors, 1 switch (●), 1 link (the bus)
- Only 1 simultaneous transfer at a time
  - NB = link (bus) bandwidth **x** 1
  - BB = link (bus) bandwidth **x** 1

Ring IN



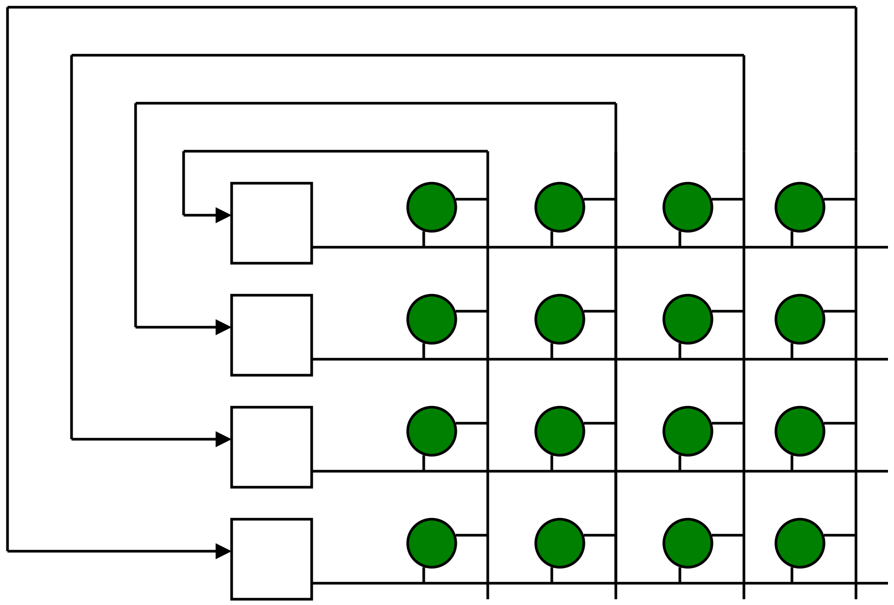
- N processors, N switches, 2 links/switch, N links
- N simultaneous transfers
  - NB = link bandwidth **x** N
  - BB = link bandwidth **x** 2
- If a link is as fast as a bus, the ring is only twice as fast as a bus in the worst case, but is N times faster in the best case

Fully Connected IN



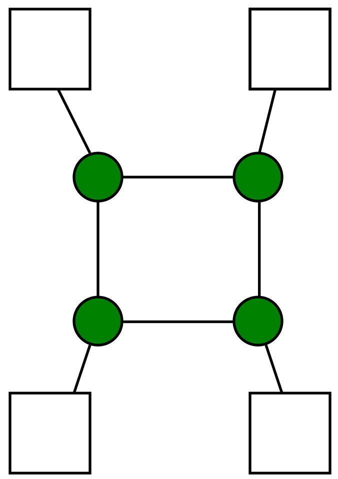
- N processors, N switches, N-1 links/switch, (N**x**(N-1))/2 links
- N simultaneous transfers
  - NB = link bandwidth **x** (N (N-1)) / 2
  - BB = link bandwidth **x** $(N/2)^2$

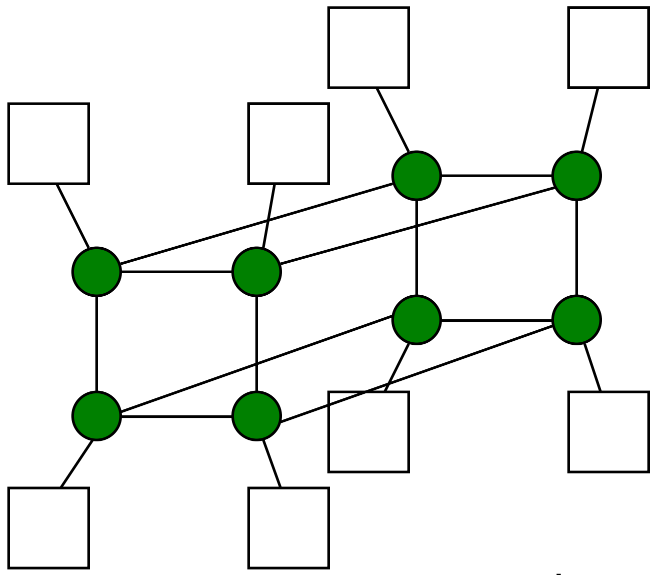
Crossbar (Xbar) Connected IN



- N processors, N^2 switches (unidirectional), 2 links/switch, N^2 links
- N simultaneous transfers
  - NB = link bandwidth **x** N
  - BB = link bandwidth **x** N/2

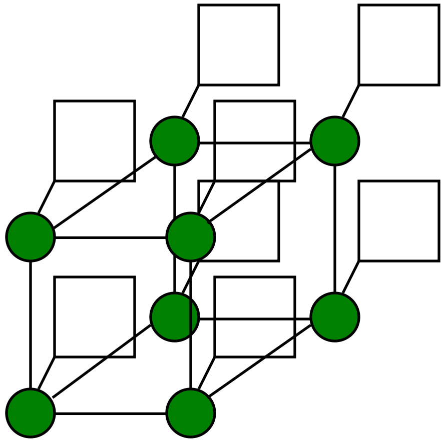
Hypercube (Binary N-cube) Connected IN

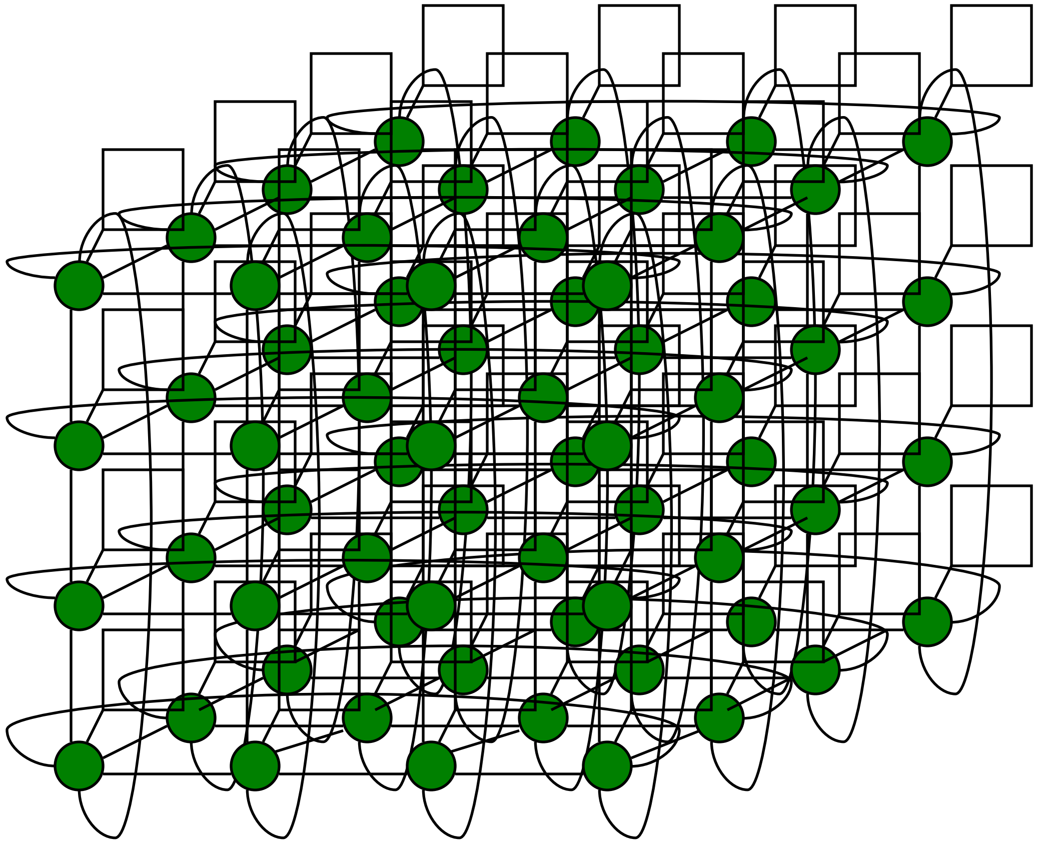

2-cube


3-cube

- N processors, N switches, logN links/switch, (N logN)/2 links
- N simultaneous transfers
  - NB = link bandwidth **x** (N logN)/2
  - BB = link bandwidth **x** N/2

2D and 3D Mesh/Torus Connected IN





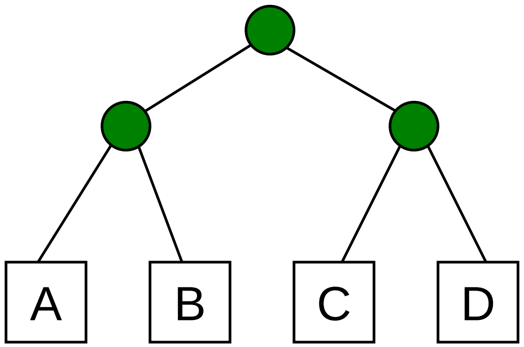
- N processors, N switches, 2, 3, 4 (2D torus) or 6 (3D torus) links/switch, 4 N/2 links or 6 N/2 links
- N simultaneous transfers
  - NB = link bandwidth **x** 4N or link bandwidth **x** 6N
  - BB = link bandwidth **x** $2N^{1/2}$ or link bandwidth **x** $2N^{2/3}$

IN Comparison

- For a 64 processor system

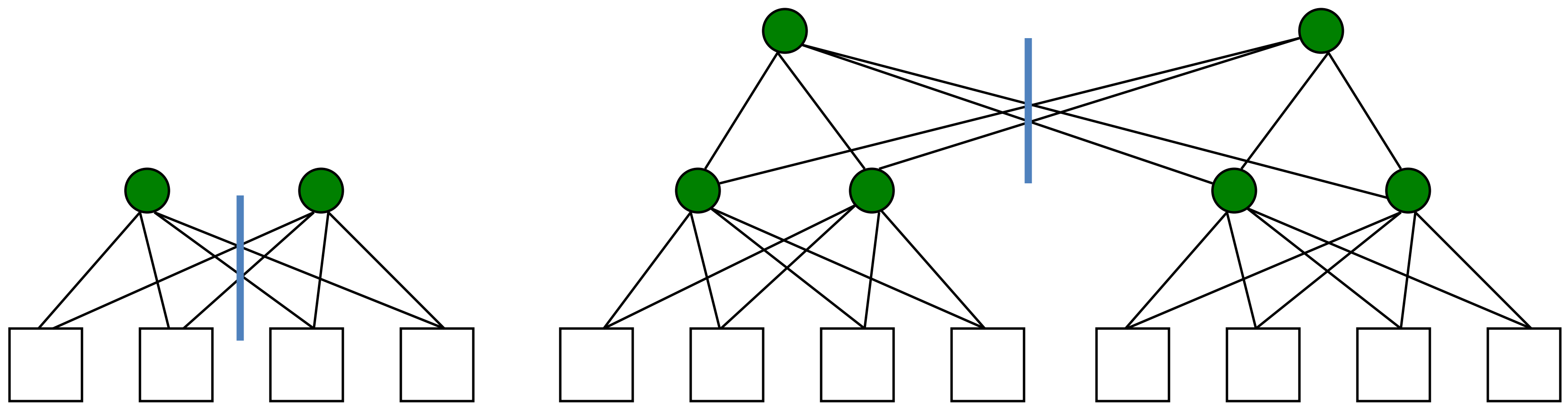
	Bus	Ring	2D Torus	6-cube	Fully connected
Network bandwidth	1	64	256	192	2016
Bisection bandwidth	1	2	16	32	1024
Total # of switches	1	64	64	64	64
Links per switch	1	2+1	4+1	6+1	63+1
Total # of links (bidi)	1	64+64	128+64	192+64	2016+64

"Fat" Trees



- Any time A wants to send to C, it ties up the upper links, so that B can't send to D.
  - The bisection bandwidth on a tree is horrible - 1 link, at all times
- The solution is to 'thicken' the upper links.
  - Have more links as you work towards the root of the tree increases the bisection bandwidth
- Rather than design a bunch of N-port switches, use pairs of switches

Fat Tree IN



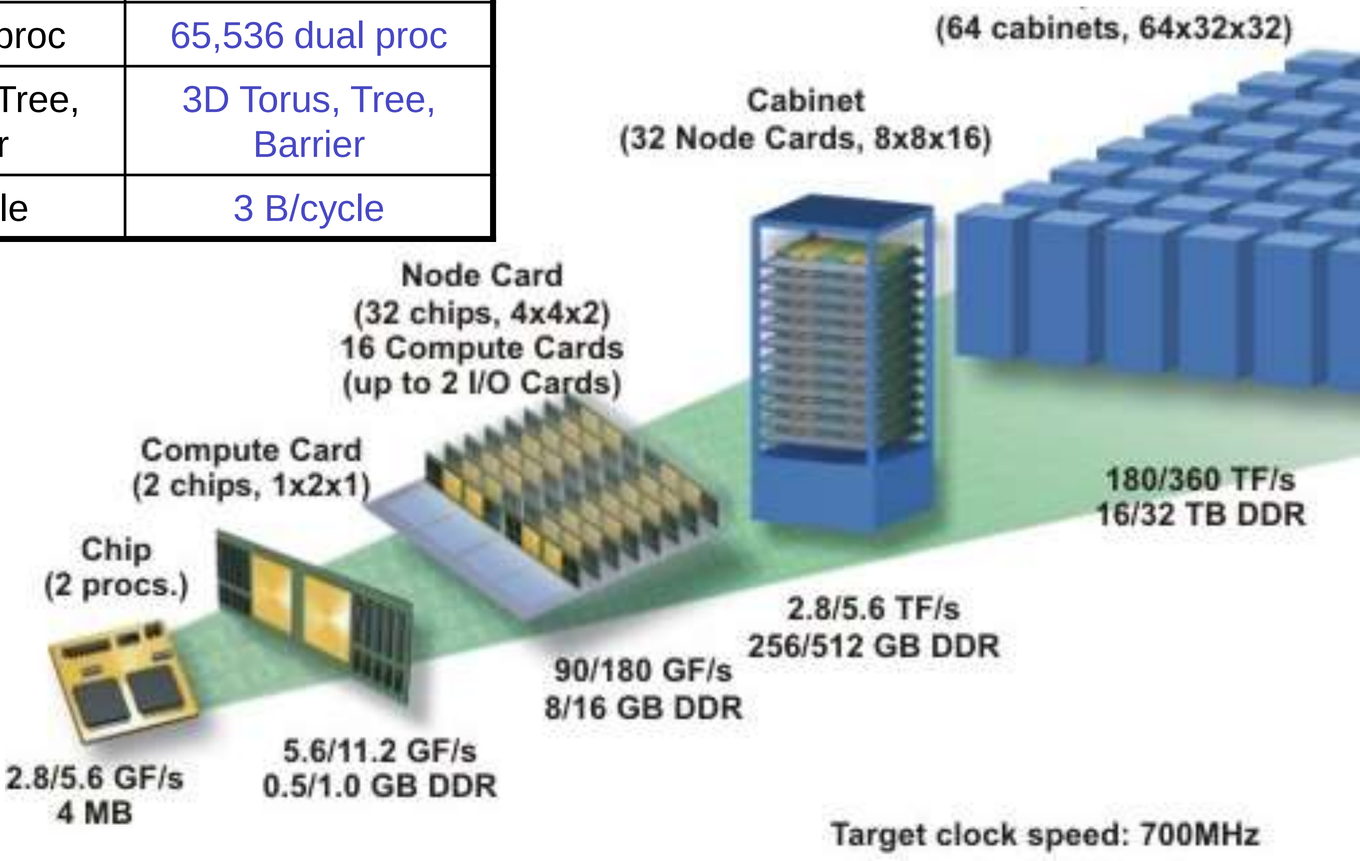
- N processors, log(N-1) **x** logN switches, 2 up + 4 down = 6 links/switch, N **x** logN links
- N simultaneous transfers
  - NB = link bandwidth **x** N logN
  - BB = link bandwidth **x** 4

Network Connected Multiprocessors

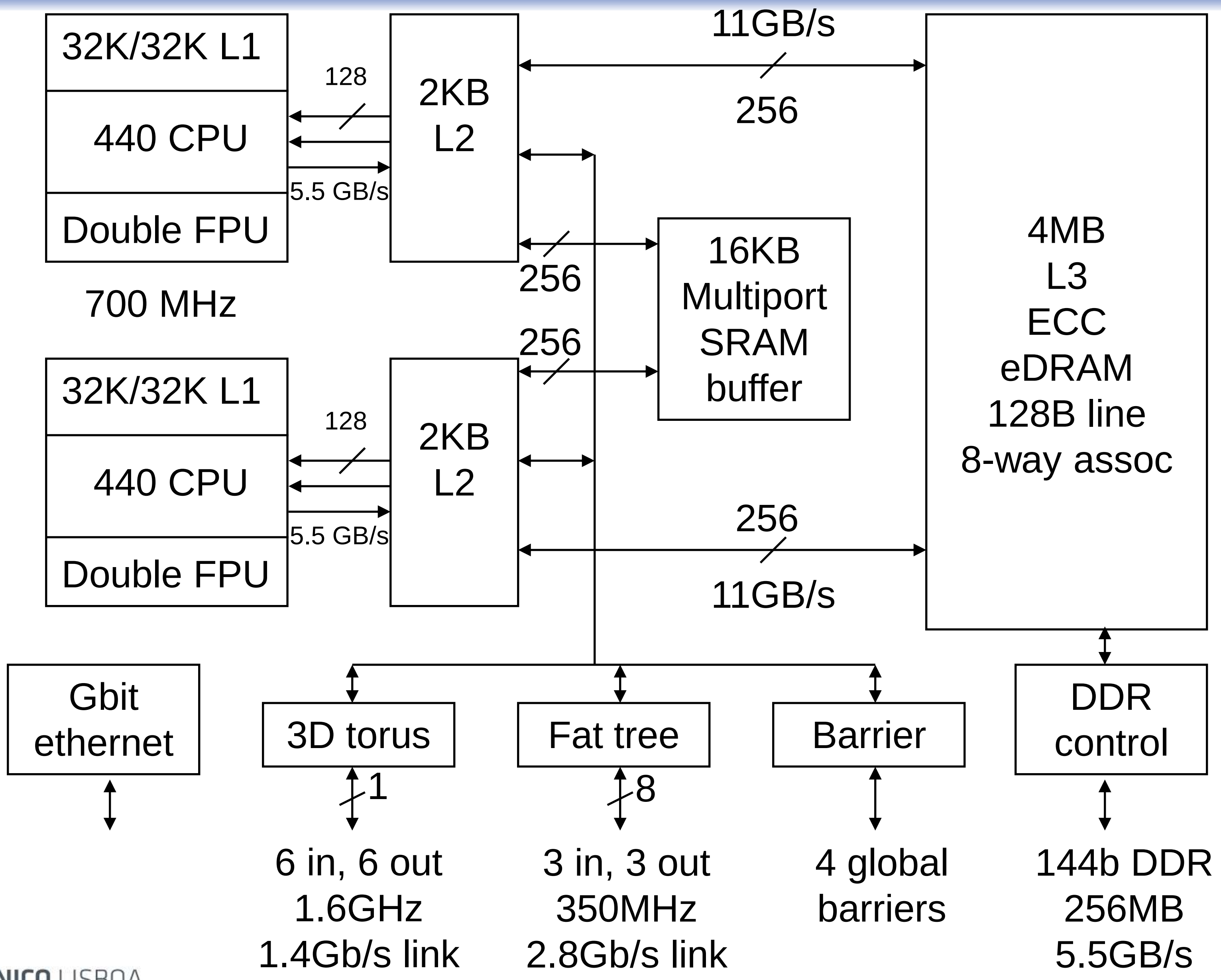
	Proc	Proc Speed	# Proc	IN Topology	BW/link (MB/sec)
SGI Origin	R16000		128	fat tree	800
Cray 3TE	Alpha 21164	300MHz	2,048	3D torus	600
Intel ASCI Red	Intel	333MHz	9,632	mesh	800
IBM ASCI White	Power3	375MHz	8,192	multistag e Omega	500
NEC ES	SX-5	500MHz	640 x 8	640-xbar	16,000
NASA Columbia	Intel Itanium2	1.5GHz	512 x 20	fat tree, Infiniband	
IBM BG/L	Power PC 440	0.7GHz	65,536 x 2	3D torus, fat tree, barrier	

IBM BlueGene

	512-node proto	BlueGene/L
Peak Perf	1.0 / 2.0 TFlops/s	180 / 360 TFlops/s
Memory Size	128 GByte	16 / 32 TByte
Foot Print	9 sq feet	2500 sq feet
Total Power	9 KW	1.5 MW
# Processors	512 dual proc	65,536 dual proc
Networks	3D Torus, Tree, Barrier	3D Torus, Tree, Barrier
Torus BW	3 B/cycle	3 B/cycle



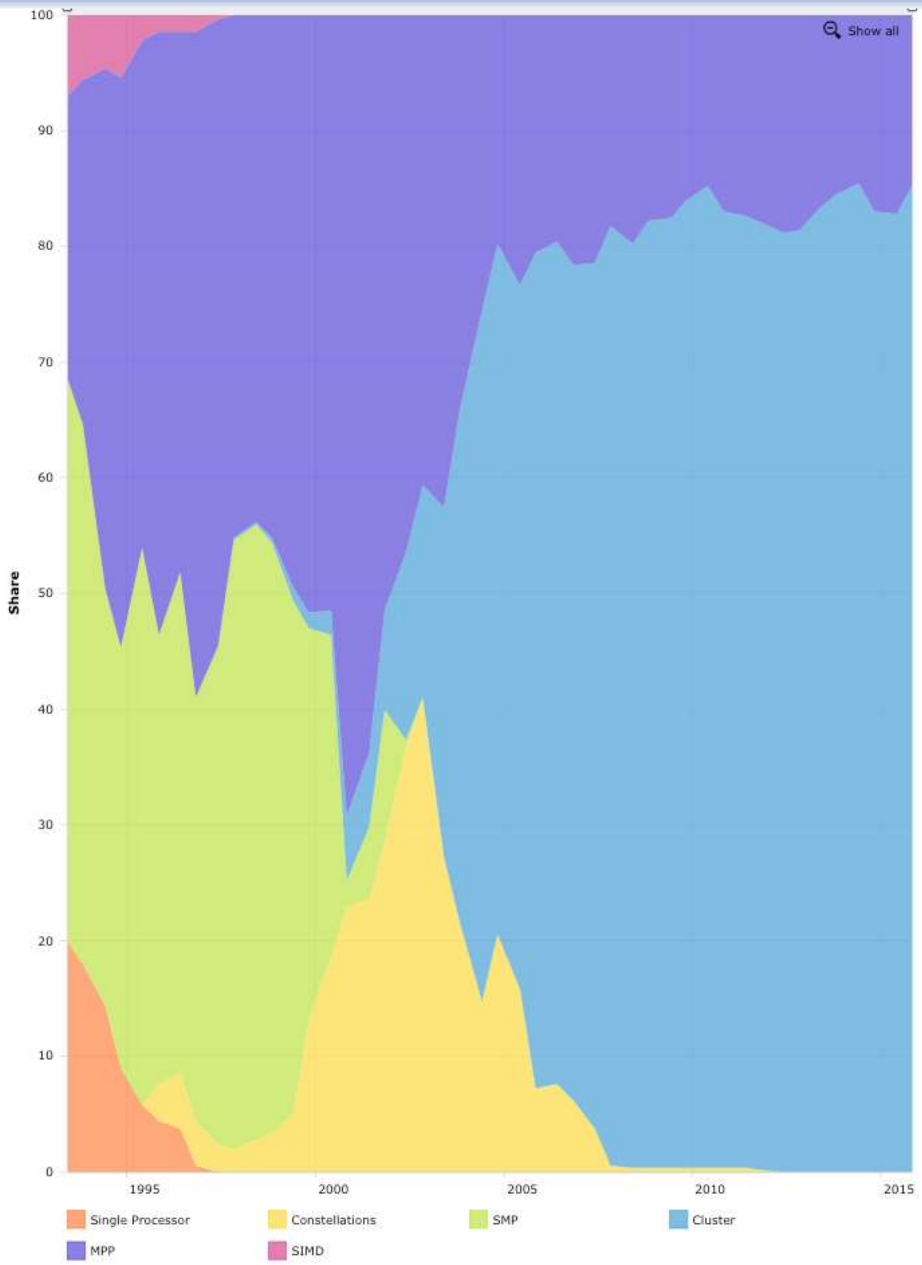
A BlueGene/L Chip



Multiprocessor Benchmarks

	Scaling?	**Reprogram?**	**Description**
Linpack	Weak	Yes	Dense matrix linear algebra
SPECrate	Weak	No	Independent job parallelism
SPLASH 2	Strong	No	Independent job parallelism (both kernels and applications, many from high-performance computing)
NAS Parallel	Weak	Yes (c or Fortran)	Five kernels, mostly from computational fluid dynamics
PARSEC	Weak	No	Multithreaded programs that use Pthreads and OpenMP. Nine applications and 3 kernels – 8 with data parallelism, 3 with pipelined parallelism, one unstructured
Berkeley Design Patterns	Strong or Weak	Yes	13 design patterns implemented by frameworks or kernels

Supercomputer Style Migration (Top500)



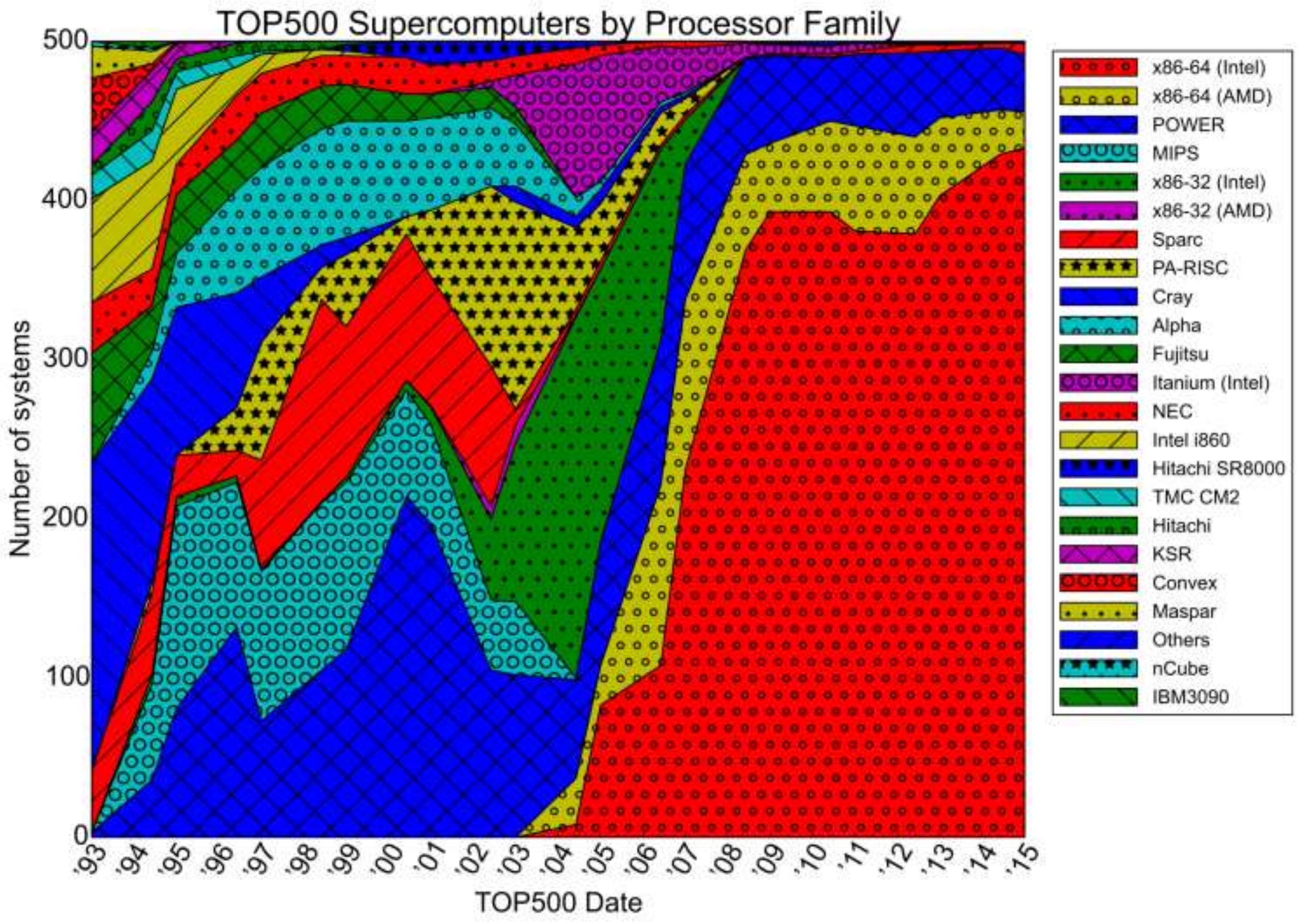
http://www.top500.org/
(November 2015)

Cluster – whole computers interconnected using their I/O bus

Constellation – a cluster that uses an SMP multiprocessor as the building block

- Uniprocessors and SIMDs disappeared while Clusters and Constellations grew from 3% to 80%.
- Now its 100% Clusters and MPPs.

Processor families in TOP500



Fallacies

- Amdahl’s Law doesn’t apply to parallel computers
  - Since we can achieve linear speedup
  - But only on applications with weak scaling

- Peak performance tracks observed performance
  - Marketers like this approach!
  - Need to be aware of bottlenecks

Pitfalls

- Not developing the software to take account of a multiprocessor architecture
  - Example: using a single lock for a shared composite resource
    - Serializes accesses, even if they could be done in parallel
    - Use finer-granularity locking

Concluding Remarks

- Goal: higher performance by using multiple processors
- Difficulties
  - Developing parallel software
  - Devising appropriate architectures
- Many reasons for optimism
  - Changing software and application environment
  - Chip-level multiprocessors with lower latency, higher bandwidth interconnect
- An ongoing challenge for computer architects!

Message-Passing Multiprocessors

Computer Organization

Many slides adapted from:
Computer Organization and Design,
Patterson & Hennessy
4th Edition, © 2011, MK
and from Prof. Mary Jane Irwin, PSU

TÉCNICO LISBOA