Lecture 2

Parallel & Distributed Architecture

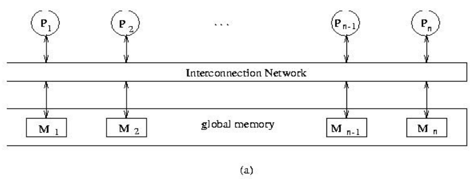
Parallel Computer Architectures

1. Memory organization
   - Memory hierarchy
   - Shared versus distributed memory
2. Processor/node architecture
   - Flynn’s taxonomy
3. Interconnection network
   - Dynamic versus static networks

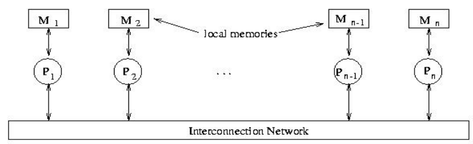
1. Classification based on Memory

1. Based on memory organization:
   SM=Shared-Memory versus DM=Distributed Memory
2. Based on access time:
   - UMA =Uniform Memory Access (time);
   - NUMA=Non-uniform Memory Access (time);
   - SMP = Symmetric Multi-Processors;

Shared memory



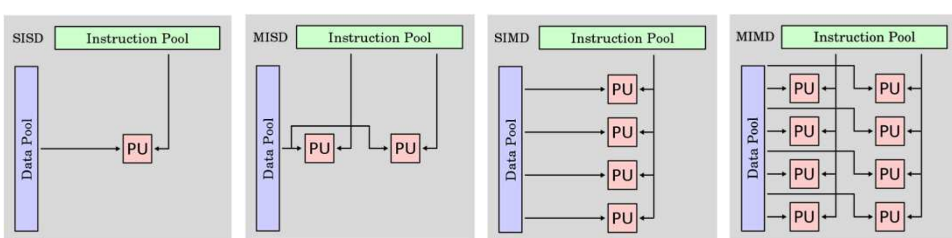
Distributed memory



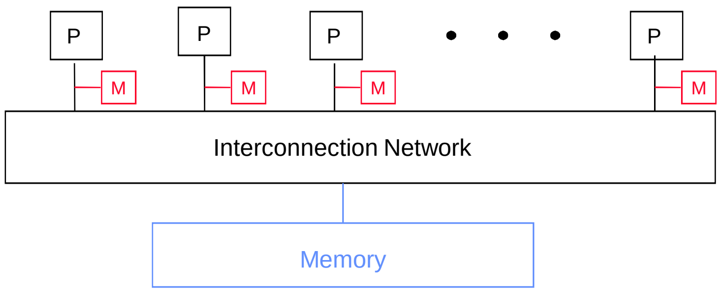
2. Classification based on data and control flows

Flynn's taxonomy:

1. Single Instruction Single Data (SISD) – exp. classical Von Neumann machine (sequential computer).
2. Multiple Instruction Single Data (MISD) – exp. none
3. Single Instruction Multiple Data (SIMD) – exp. GPU
4. Multiple Instruction Multiple Data (MIMD) – exp. cluster of computers

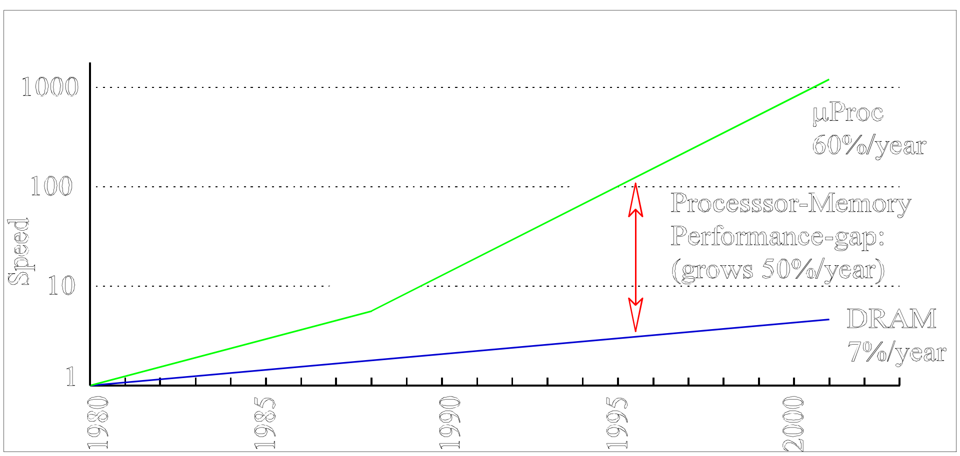


A generic parallel architecture



P – Processor; M - Mmeory

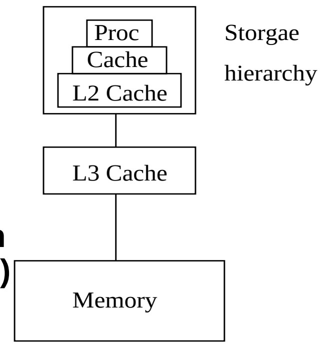
Inside one CPU:
Growing gap betw. processor and memory



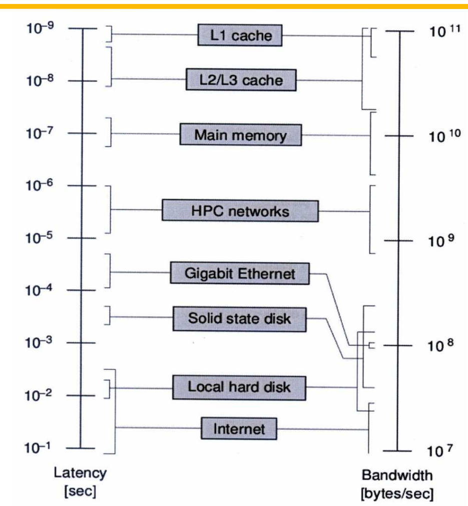
Memory hierarchy: Caches

In order reduce the big delay of directly accessing the main or remote memory, it requires:

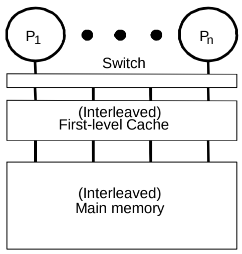
- Optimizing the data movement, maximize reuse of data already in fastest memory;
- Minimizing data movement between ‘remote’ memories (communication)

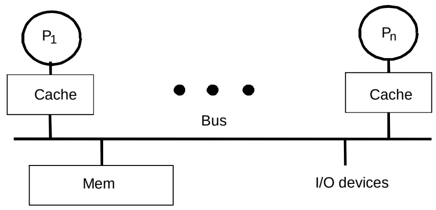


Across processors and CPUs:
Speed differences in memory and networks



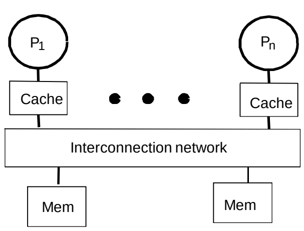
Extensions of Memory System (1)

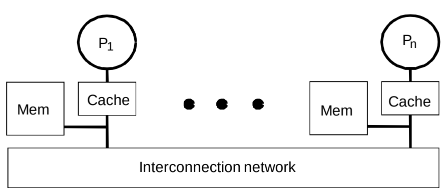




(a) Shared Cache **(b) Bus-based shared memory (SMP)**

Extensions of Memory System (2)

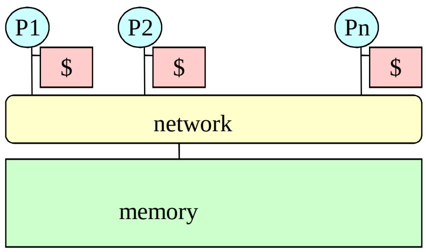




(c) Dance hall (UMA) **(d) Distributed-memory (NUMA)**

Machine Model 1

- A shared memory machine
- Processors all connected to a large shared memory
- "Local" memory is not (usually) part of the hardware
  - Symmetric Mutliprocessors (SMP), e.g. SGI Origin
- Speed: much quicker to cache than main memory



Bus-based Shared Memory Multiprocessors

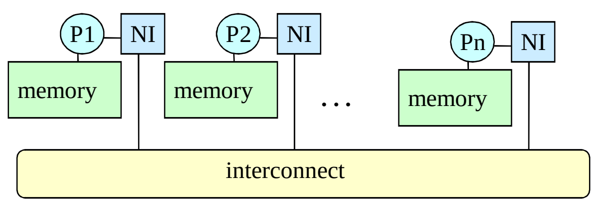
- Symmetric Multiprocessors (SMPs)
  - Symmetric access to all of main memory from any processor
- Dominate the server market
  - Building blocks for larger systems; today multi-core laptops are common
- Attractive as throughput servers and for parallel programs
  - Fine-grain resource sharing
  - Uniform access via loads/stores
  - Automatic data movement and coherent replication in caches
  - Useful for operating system too

Shared Memory Multiprocessors

- Normal uniprocessor mechanisms to access data through reads and writes
- Key is extension of memory hierarchy to support multiple processors (e.g., crossbar network is expensive for large number, therefore often virtual shared memory system is implemented)

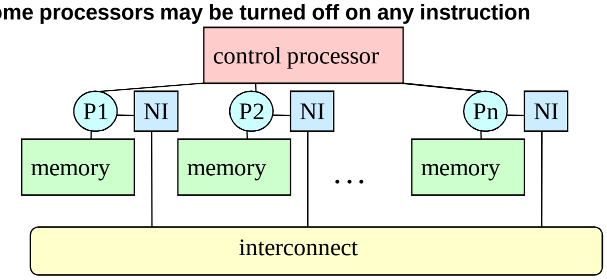
Machine Model 2

- **A distributed memory machine**
  - **Cluster of computers, IBM Blue Gene,Tianhe, etc.**
  - **Processors all connected to own memory (and caches)**
  - **cannot directly access another processor's memory**
- **Each "node" has a network interface (NI)**
  - **all communication and synchronization done through this**



Machine Model 3

- A SIMD (Single Instruction Multiple Data) machine
- A large number of small processors
- A single "control processor" issues each instruction
  - each processor executes the same instruction
  - some processors may be turned off on any instruction



Earlier example machine Connection Machine (CM). Programming model is

- implemented by mapping n-fold parallelism to p processors
- mostly done in the compilers (HPF = High Performance Fortran)

Machine Model 4, Cluster of SMPs

- Since small shared memory machines (SMP's) are the fastest commodity machine, why not build a larger machine by connecting many of them with a network?
- Shared memory within one SMP
- Message passing outside
- ASCI Red (Intel), Blue Gene (IBM), ...
- Programming model?
  - Treat machine as "flat", always use message passing, even within SMP (simple, but ignore important part of memory hierarchy)
  - Expose two layers: shared memory and message passing (higher performance, e.g., mixed MPI&OpenMP, but ugly to program)

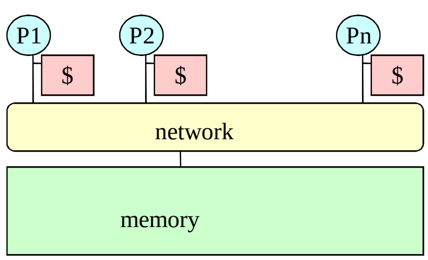
Shared Memory Systems

- Shared cache
- Shared memory
- Cache coherence problem

- Emphasis is not on network, but on memory hierarchy.

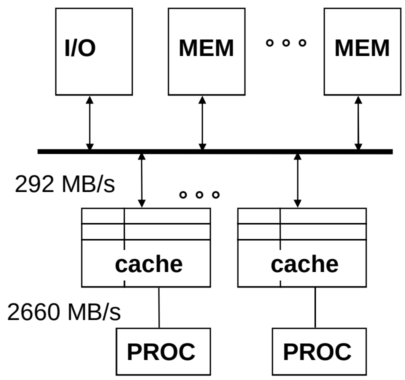
Basic Shared Memory Architecture

- Processors all connected to a large shared memory
- Local caches for each processor
- speed: much quicker to cache than main memory



° Simplest to program, but hard to build with many processors

Limits of using Bus as Network



Assume a 1000 MB/s bus

500 MIPS processor w/o cache

=> 2000 MB/s instr BW per processor

=> 660 MB/s data BW if 33% load-store

(assuming 4 bytes per instruction/word)

Suppose 98% instr. hit rate and 95% data hit rate (assume 16 bytes block=cache line)

=> 160 MB/s instr. BW per processor

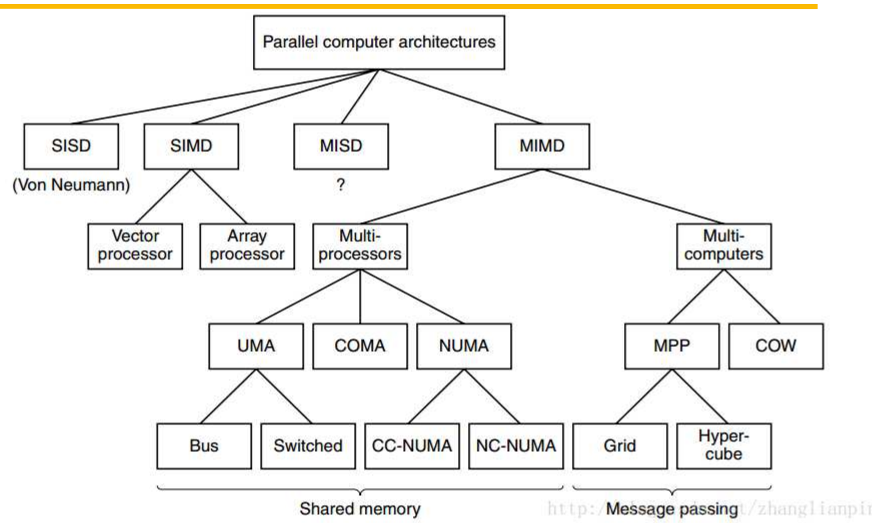
=> 132 MB/s data BW per processor

=> 292 MB/s combined BW

∴ **4 processors will saturate the bus!**

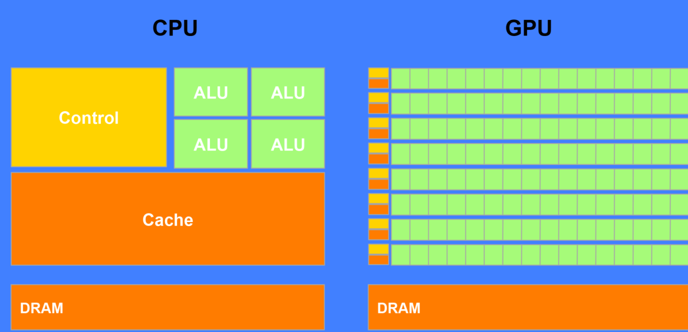
→ Bus only useful in small systems

Summary: Parallel Computer Architectures



Accelerators (for floating point processing)

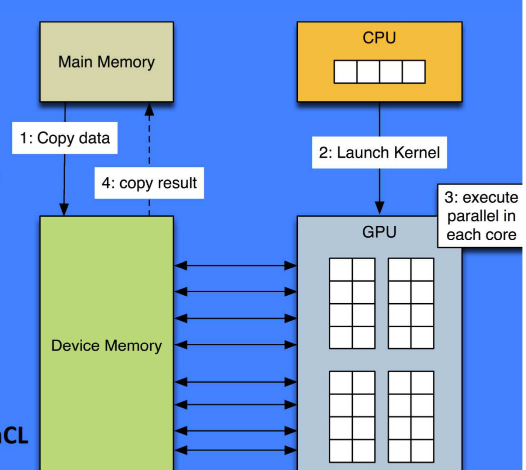
A Graphics Processing Unit is not a CPU!



And there is no GPU without a CPU…

Graphics Processing Unit

- ✓Originally dedicated to specific operations required by video cards for accelerating graphics, GPU have become flexible **general-purpose** computational engines (GPGPU):
  - ✓Optimized for **high memory throughput**
  - ✓Very suitable to **data-parallel processing**
- ✓Three vendors: Chipzilla (*a.k.a.* Intel), ADI (*i.e.,* AMD), NVIDIA.
- ✓Traditionally GPU programming has been tricky but **CUDA** and **OpenCL** made it affordable.



Parallel Computer Architectures

1. Memory organization
   - Memory hierarchy
   - Shared versus distributed memory
2. Processor/node architecture
   - Flynn's taxonomy
3. Interconnection network
   - Dynamic versus static networks

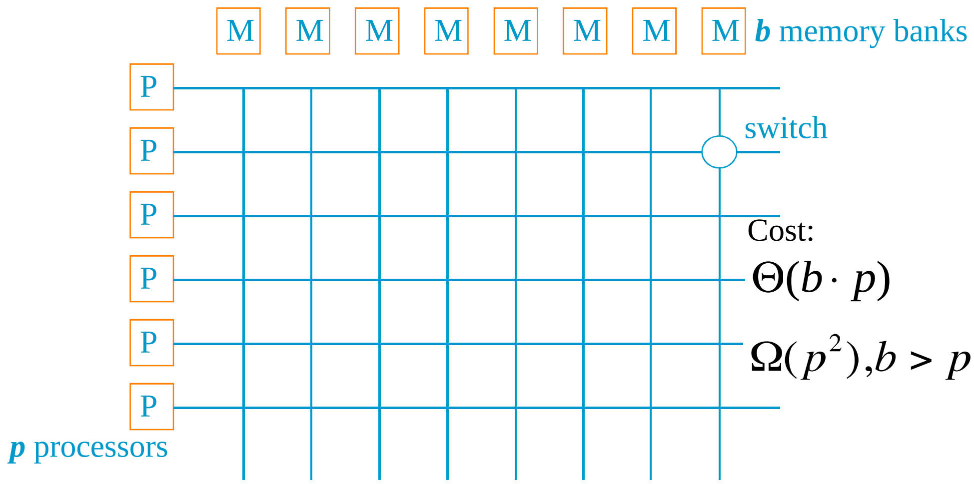
3. Interconnection Networks

Besides the **node architecture** and **memory organization**, **interconnection network** is another important component which characterizes a parallel computer.

Types of Networks:

1. Dynamic
2. Static

Cross-bar networks (dynamic)

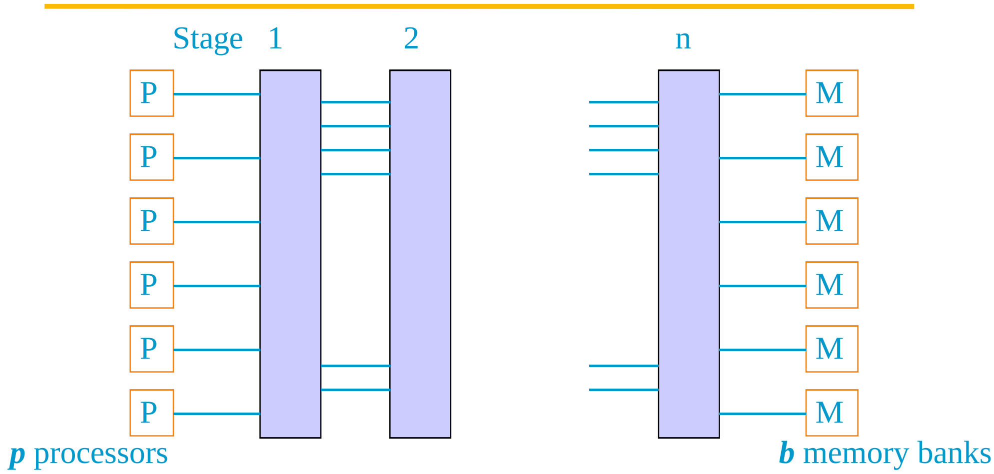


Cost:

$\Theta(b \cdot p)$

$\Omega(p^2), b > p$

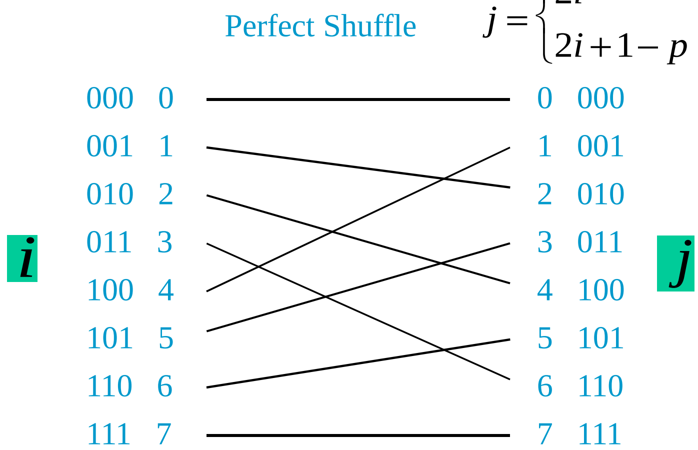
Multi-stage networks (dynamic)



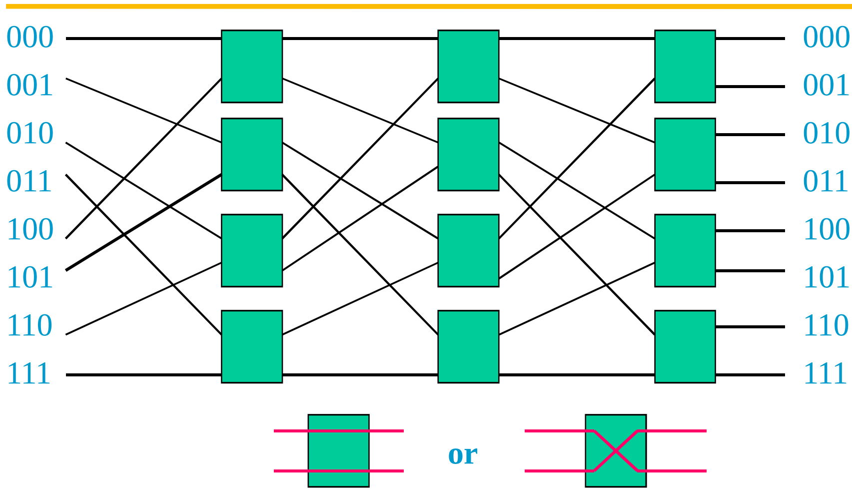
$O(p \cdot \log_2(p))$, $b=p$, $n=\log_2(p)$

Perfect Shuffle

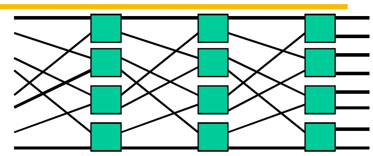
$$j = \begin{cases} 2i & 0 \le i \le p/2 - 1 \\ 2i + 1 - p & p/2 \le i \le p - 1 \end{cases}$$



Omega Network (dynamic)



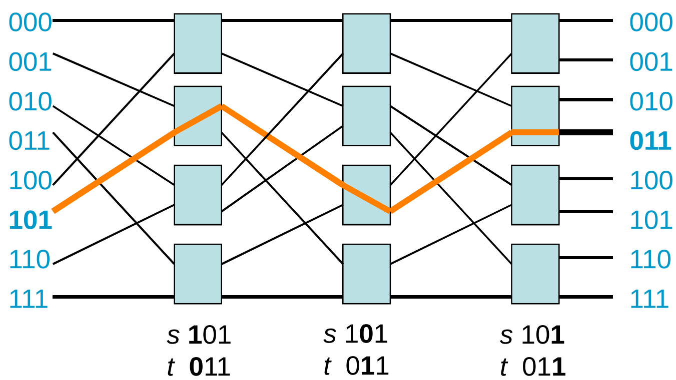
Complexity Omega Network

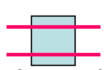
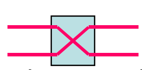


- **Processors:** p
- **Stages:** $\log p$
- **Switches per Stage:** $p/2$
- **Total number of switches:** $\frac{p}{2}\log p$

Routing in Omega Network

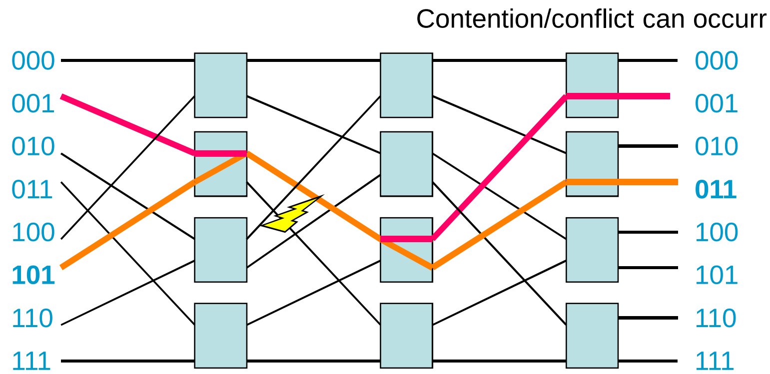
$s = \alpha_1\alpha_2...\alpha_k$
to
$t = \beta_1\beta_2...\beta_k$



Switching rules:  or 

$\boldsymbol{\beta}_k$ = 0: the message is forwarded over the upper link of the switch;
$\boldsymbol{\beta}_k$ = 1: the message is forwarded over the lower link of the switch.

Collision in Omega Network



Exactly one path for every pair: $s = \alpha_1\alpha_2 \ldots \alpha_k$ to $t = \beta_1\beta_2 \ldots \beta_k$

In total $(n/2)\log(n)$ swtiches $\rightarrow$ $2^{(n/2)\log(n)} = n^{n/2}$ different switchings compared to $n!$ permutations (for n input to n output), only $n^{n/2}$ of the $n!$ possible permutations can be performed without conflict.

Dynamic networks (switching network)

- ❖ **Flexible in realizing communication of diff. pairs of processors/memories**
- ❖ **Simple bus type network cheap but not scalable (in performance) to large number of processors**
- ❖ **Crossbars are very fast but too expensive to scale to large systems**
- ❖ **Trade-off between cost and performance**

TU Delft
Delft University of Technology

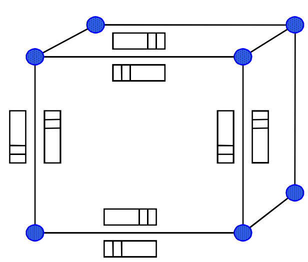
Static Networks

- Topologies
- Cost models

- Emphasis on nodal-connectivity

Historical Perspective

- Early machines were:
  - Collection of microprocessors
  - bi-directional queues between neighbors
- Messages were forwarded by processors on path
- Strong emphasis on topology in algorithms



Network Analogy

- To have a large number of transfers occurring at once, you need a large number of distinct wires
- Networks are like streets
  - link = street
  - switch = intersection
  - distances (hops) = number of blocks traveled
  - routing algorithm = travel plans
- Properties
  - latency: how long to get somewhere in the network
  - bandwidth: how much data can be moved per unit time
    - » limited by the number of wires
    - » and the rate at which each wire can accept data

Components of a Network

Networks are characterized by

- Topology - how things are connected
  - two types of nodes: hosts and switches
- Routing algorithm - paths used
  - e.g., all east-west then all north-south (avoids deadlock)
- Switching strategy
  - circuit switching: full path reserved for entire message
    - » like the telephone
  - packet switching: message broken into separately-routed packets
    - » like the post office
- Flow control - what if there is congestion
  - if two or more messages attempt to use the same channel
  - may stall, move to buffers, reroute, discard, etc.

Properties of a Network

- **Diameter is the maximal length of shortest paths between any two nodes in the graph. (another metric: average distance)**
- **The bandwidth of a link is: w * 1/t**
  - **w is the number of wires**
  - **t is the time per bit**
- **Effective bandwidth lower due to packet overhead**

Trailer	Error code	Data payload	Routing and control header

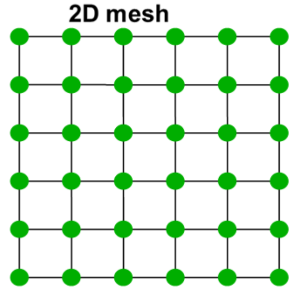
- **Bisection bandwidth**
  - **sum of the minimum number of channels which, if removed, will separate the network into two equal parts**

    (A network is partitioned if some nodes cannot reach others.)

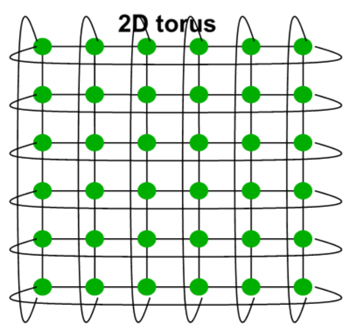
Meshes and Tori

- **Diameter:** $2\sqrt{n}$ **(in 2D)**
- **Bisection bandwidth:** $\sqrt{n}$

2D mesh



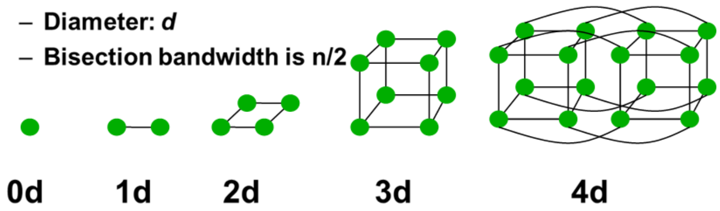
2D torus



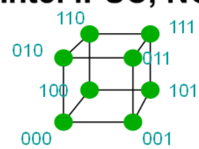
- Often used as network in machines
- Generalizes to higher dimensions (Cray T3D used 3D Torus)
- Natural for algorithms with 2D, 3D arrays

Hypercubes

- Number of nodes n = 2^d for dimension *d*
  - Diameter: *d*
  - Bisection bandwidth is n/2

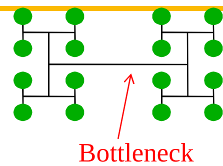


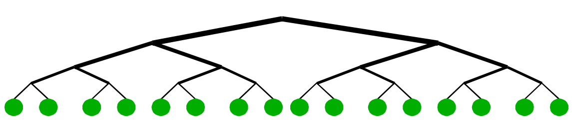
- Popular in early machines (Intel iPSC, NCUBE)
  - Lots of clever algorithms
- Greycode addressing
  - each node connected to d others with 1 bit different

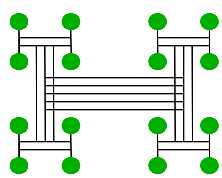


Trees

- Diameter: log(n)
- Bisection bandwidth: 1
- Easy layout as planar graph
- Many tree algorithms (summation)
- Fat trees avoid bisection bandwidth problem
  - more (or wider) links near top
  - example, Thinking Machines CM-5



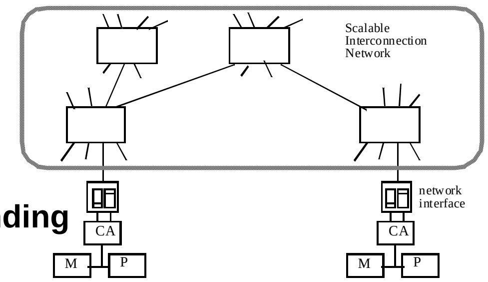




Scalable, High Perf. Interconnection Network

- At the core of parallel computer architecture
- Requirements and trade-offs at many levels
  - Elegant mathematical structure
  - Deep relationships to algorithm structure
  - Managing many traffic flows
  - Electrical / Optical link properties
- Little consensus
  - interactions across levels
  - Performance metrics?
  - Cost metrics?
  - Workload?

=> need a holistic understanding



Example speccification: Summit supercomputer

Processors:
4,356 nodes, each with two 22-core Power9 CPUs, and six NVIDIA Tesla V100 GPUs. Total 2,414,592 cores

A Fat-tree network topology
InfiniBand uses a switched fabric topology, as opposed to early shared medium Ethernet.
Implemented using Mellanox 100-Gb/s EDR InfiniBand ConnectX-5 adapters and Switch-IB2 switches
Messages
InfiniBand transmits data in packets of up to 4 KB that are taken together to form a message.

Example 2 specification: Fukagu Supercomputer

• Configuration: 158,976 Fujitsu A64FX 48C 2.2GHz; total 7299072 cores

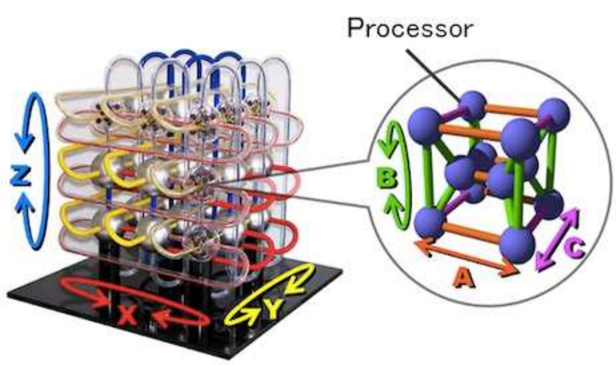
• CPU (node) features: 48 cores, 2 SIMD operations/core; memory 32 GB/node with mem. bandwidth 1TB/s.

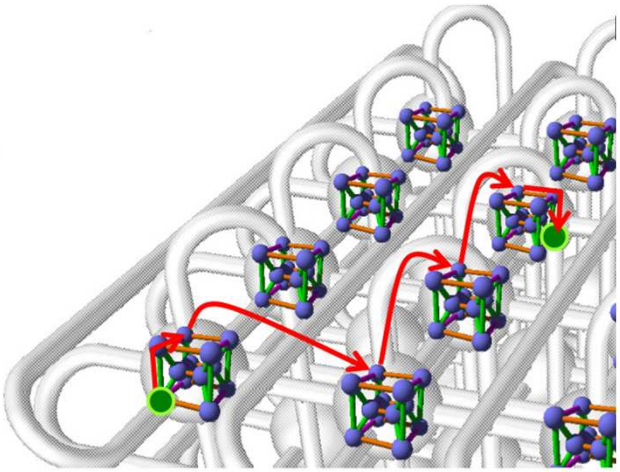
• Interconnection Network: a 6-dimensional mesh/torus network TOFU, each node with 10 inter-node connections (logically a 3-D torus network)

TOFU interconnect: 6-D mesh/torus

❖ Six coordinate axes: X, Y, Z, A, B, C
- X, Y, Z: size varies according to system configuration
- A, B, C: fixed 2×3×2 (building block)

❖ TOFU: Torus fusion X×Y×Z ×(2×3×2)
- X×Y×Z forms a 3-D torus of building blocks.





TOFU node design: Each pair of adjacent ABC mesh/torus is interconnected via twelve links.

Quiz 1

Submission deadline: 14 September, 15:45 PM

Questions on brightspace in Assignments/Quiz1

Upload your answers to Assigments/Quiz1.