PARALLEL COMPUTERS

Dr Noor Mahamamd Sk

Hprcse

Introduction

- *Parallel computing is a number of "compute elements" (cores)* solve a problem in a cooperative way.
- All modern supercomputer architectures depend heavily on parallelism, and the number of CPUs in large-scale supercomputers increases steadily.
- A common measure for supercomputer "speed"

Taxonomy of Parallel Computing Paradigms

- A widely used taxonomy for describing the amount of concurrent control and data streams present in a parallel architecture was proposed by Flynn
- **SIMD *Single Instruction, Multiple Data.***
- A single instruction stream, either on a single processor (core) or on multiple compute elements, provides parallelism by operating on multiple data streams concurrently.
- Examples: Vector processors
- The SIMD capabilities of modern superscalar microprocessors, and Graphics Processing Units (GPUs).

Taxonomy of Parallel Computing Paradigms

- **MIMD *Multiple Instruction, Multiple Data.***
- *Multiple instruction streams on multiple* processors (cores) operate on different data items concurrently.
- The shared memory and distributed-memory parallel computers are typical examples for the MIMD paradigm.

Shared-memory computers

- A *shared-memory parallel computer is a system in which a number of CPUs* work on a common shared physical address space.
- Although transparent to the programmer as far as functionality is concerned
- There are two varieties of shared memory systems that have very different performance characteristics in terms of main memory access:
- *Uniform Memory Access (UMA)*
- *Cache-coherent Nonuniform Memory Access (ccNUMA)*

Uniform Memory Access (UMA)

- Systems exhibit a “flat” memory model:
- Latency and bandwidth are the same for all processors and all memory locations.
- This is also called *symmetric multiprocessing (SMP).*
- *At the time of writing,* single multicore processor chips are “UMA machines.”
- However, “cluster on a chip” designs that assign separate memory controllers to different groups of cores on a die are already beginning to appear.

Non Uniform Memory Access (NUMA)

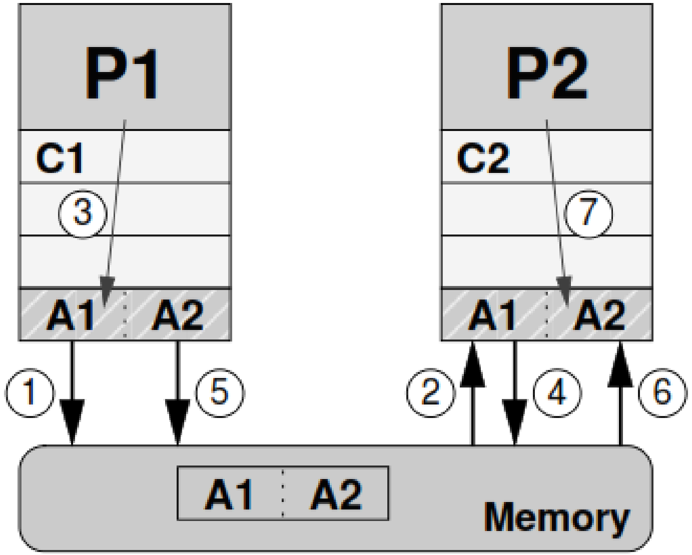
- Memory is *physically distributed but logically shared.*
- *The physical layout of such* systems is quite similar to the distributed-memory case, but network logic makes the aggregated memory of the whole system appear as one single address space.
- Due to the distributed nature, memory access performance varies depending on which CPU accesses which parts of memory (“local” vs. “remote” access).

Cache Coherence

- Cache coherence mechanisms are required in all cache-based multiprocessor systems, whether they are of the UMA or the ccNUMA kind.
- This is because copies of the same cache line could potentially reside in several CPU caches.
- If, e.g., one of those gets modified and evicted to memory, the other caches’ contents reflect outdated data.
- Cache coherence protocols ensure a consistent view of memory under all circumstances.

MESI

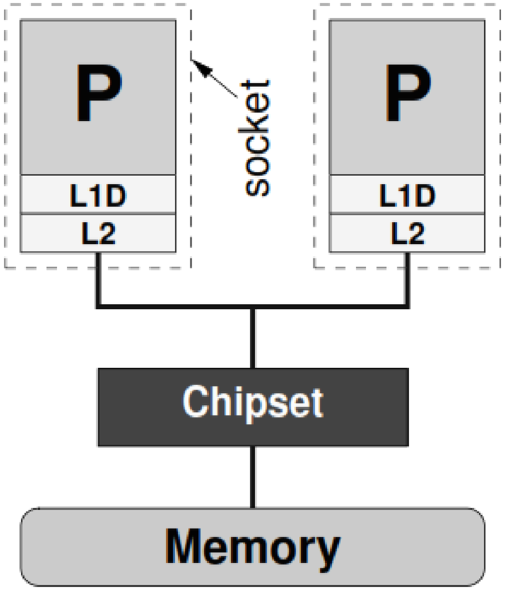
- Two processors P1, P2 modify the two parts A1, A2 of the same cache line in caches C1 and C2.
- The MESI coherence protocol ensures consistency between cache and memory.
- 1. C1 requests exclusive CL ownership
- 2. set CL in C2 to state I
- 3. CL has state E in C1 → modify A1 in C1
- and set to state M
- 4. C2 requests exclusive CL ownership
- 5. evict CL from C1 and set to state I
- 6. load CL to C2 and set to state E
- 7. modify A2 in C2 and set to state M in C2



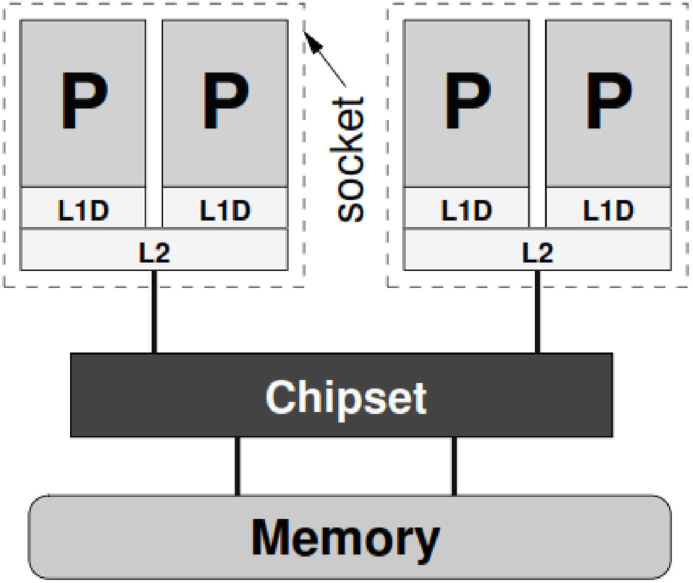
MESI

- **M *modified:***
- *The cache line has been modified in this cache, and it resides in no* other cache than this one.
- Only upon eviction will memory reflect the most current state.
- **E *exclusive:***
- *The cache line has been read from memory but not (yet) modified.*
- However, it resides in no other cache.
- **S *shared:***
- *The cache line has been read from memory but not (yet) modified.*
- *There* may be other copies in other caches of the machine.
- **I *invalid:***
- *The cache line does not reflect any sensible data.*
- *Under normal circumstances* this happens if the cache line was in the shared state and another processor has requested exclusive ownership.

UMA



- A UMA system with two single core CPUs that share a common front side bus (FSB).



- A UMA system in which the FSBs of two dual-core chips are connected separately to the chipset.

UMA

- The simplest implementation of a UMA system is a dual-core processor, in which two CPUs on one chip share a single path to memory.
- It is very common in high performance computing to use more than one chip in a compute node, be they single core or multicore.
- Two (single-core) processors, each in its own socket, communicate and access memory over a common bus, the so-called *front side bus (FSB).*
- *All arbitration* protocols required to make this work are already built into the CPUs.
- The chipset (often termed “northbridge”) is responsible for driving the memory modules and connects to other parts of the node like I/O subsystems.
- This kind of design is outdated and is not used any more in modern systems.

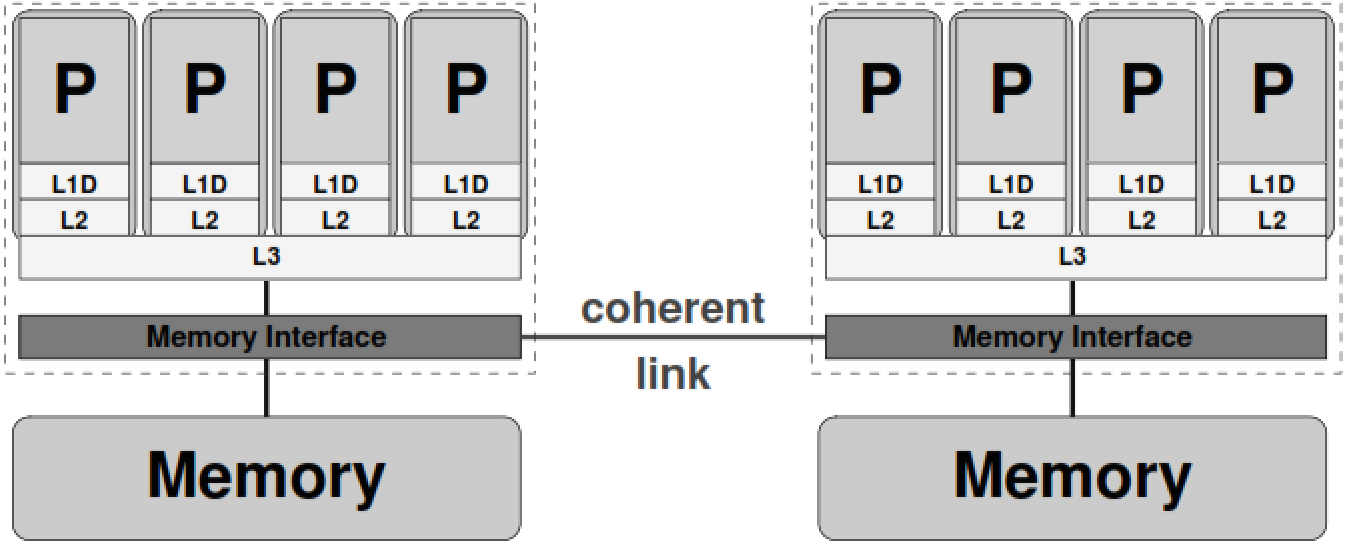
UMA

- Two dual-core chips connect to the chipset, each with its own FSB.
- The chipset plays an important role in enforcing cache coherence and also mediates the connection to memory.
- In principle, a system like this could be designed so that the bandwidth from chipset to memory matches the aggregated bandwidth of the front side buses.
- Each chip features a separate L1 on each core and a dual-core L2 group.
- The arrangement of cores, caches, and sockets make the system inherently *anisotropic, i.e., the "distance" between one core and another varies depending on* whether they are on the same socket or not.
- With large many-core processors comprising multilevel cache groups, the situation gets more complex still.

Problems of UMA

- The general problem of UMA systems is that bandwidth bottlenecks are bound to occur when the number of sockets (or FSBs) is larger than a certain limit.
- In very simple designs a common *memory bus is used that can* only transfer data to one CPU at a time.
- In order to maintain scalability of memory bandwidth with CPU number, nonblocking *crossbar switches can be built that establish point-to-point connections between* sockets and memory modules
- Due to the very large aggregated bandwidths those become very expensive for a larger number of sockets.
- At the time of writing, the largest UMA systems with scalable bandwidth (the NEC SX-9 vector nodes) have sixteen sockets.
- This problem can only be solved by giving up the UMA principle.

ccNUMA



- A ccNUMA system with two locality domains (one per socket) and eight cores.

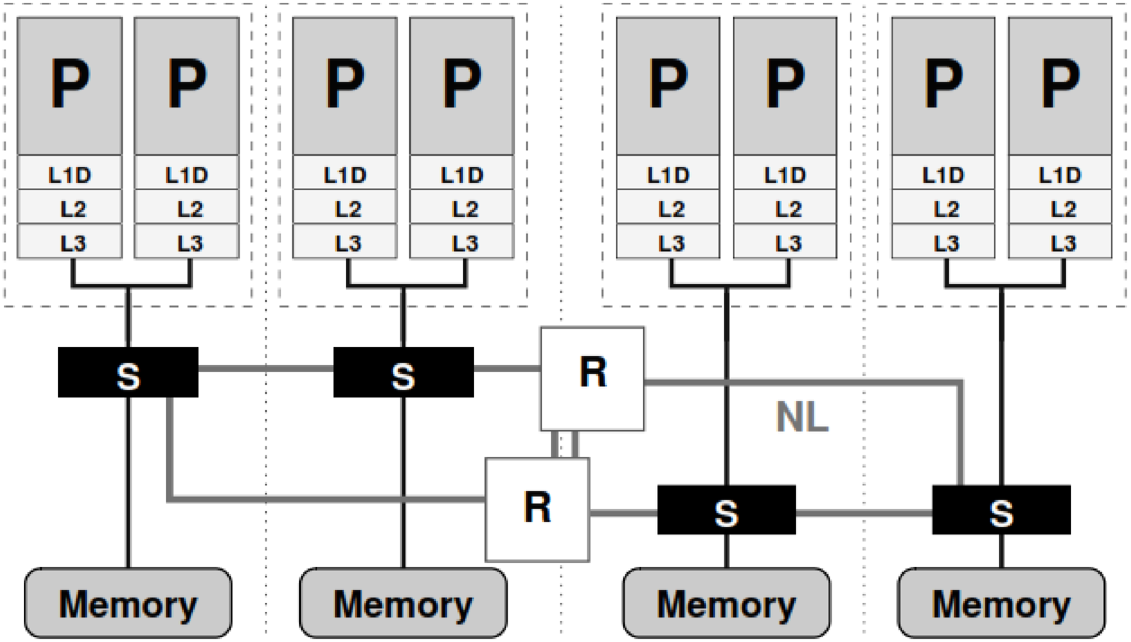
ccNUMA

- A *locality domain (LD) is a set of processor cores together with* locally connected memory.
- This memory can be accessed in the most efficient way, i.e., without resorting to a network of any kind.
- Multiple LDs are linked via a *coherent* interconnect, which allows transparent access from any processor to any other processor's memory.
- In this sense, a locality domain can be seen as a UMA "building block."
- The whole system is still of the shared-memory kind, and runs a single OS instance.
- Although the ccNUMA principle provides scalable bandwidth for very large processor counts, it is also found in inexpensive small two- or four-socket nodes frequently used for HPC clustering.

ccNUMA

- In this particular example two locality domains, i.e., quad-core chips with separate caches and a common interface to local memory, are linked using a high-speed connection.
- *HyperTransport (HT)* and *QuickPath (QPI) are the current technologies favored by AMD and Intel, respectively,* but other solutions do exist.
- Apart from the minor peculiarity that the sockets can drive memory directly, making separate interface chips obsolete, the inter socket link can mediate direct, cache-coherent memory accesses.
- From the programmer’s point of view this mechanism is transparent: All the required protocols are handled by hardware.

ccNUMA



- A ccNUMA system (SGI Altix) with four locality domains, each comprising one socket with two cores.
- The LDs are connected via a routed NUMALink (NL) network using routers (R).

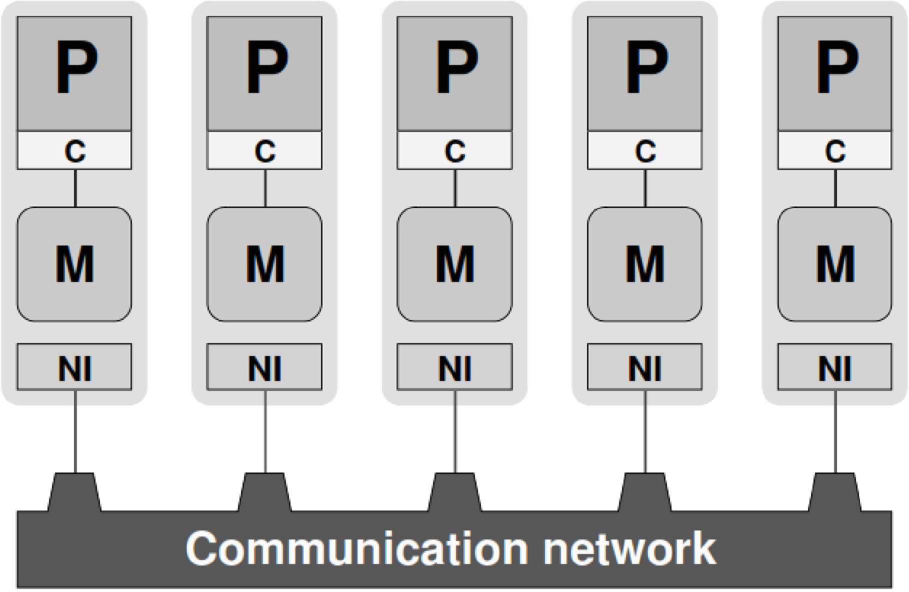
bandwidth and latency

- Network connections must have bandwidth and latency characteristics that are at least the same order of magnitude as for local memory.
- ***Locality Problem****:*
- It occurs even if there is only one serial program running on a ccNUMA machine.
- The second problem is potential *contention if two processors from different locality domains access memory in* the same locality domain, fighting for memory bandwidth.
- Even if the network is nonblocking and its performance matches the bandwidth and latency of local access, contention can occur.
- Both problems can be solved by carefully observing the data access patterns of an application and restricting data access of each processor to its own locality domain.

ccNUMA

- In inexpensive ccNUMA systems I/O interfaces are often connected to a single LD.
- Although I/O transfers are usually slow compared to memory bandwidth
- There are high-speed network interconnects that feature multi-GB bandwidths between compute nodes.

Distributed-Memory Computers



- Simplified programmer's view, or "programming model," of a distributed-memory parallel computer: Separate processes run on processors (P), communicating via interfaces (NI) over some network.
- No process can access another process' memory (M) directly, although processors may reside in shared memory.

Distributed-Memory Computers

- Each processor P is connected to exclusive local memory, i.e., no other CPU has direct access to it.
- Nowadays there are actually no distributed-memory systems any more that implement such a layout.
- *For price/performance reasons all parallel* machines today, first and foremost the popular PC clusters, consist of a number of shared-memory “compute nodes” with two or more CPUs
- The “distributed-memory programmer’s” view does not reflect that.
- It is even possible to use distributed-memory programming on pure shared memory machines.

Distributed-Memory Computers

- Each node comprises at least one network interface (NI) that mediates the connection to a *communication network.*
- *A serial process runs on each CPU that can* communicate with other processes on other CPUs by means of the network.
- It is easy to envision how several processors could work together on a common problem in a shared-memory parallel computer
- But as there is no remote memory access on distributed-memory machines
- The problem has to be solved cooperatively by sending messages back and forth between processes.

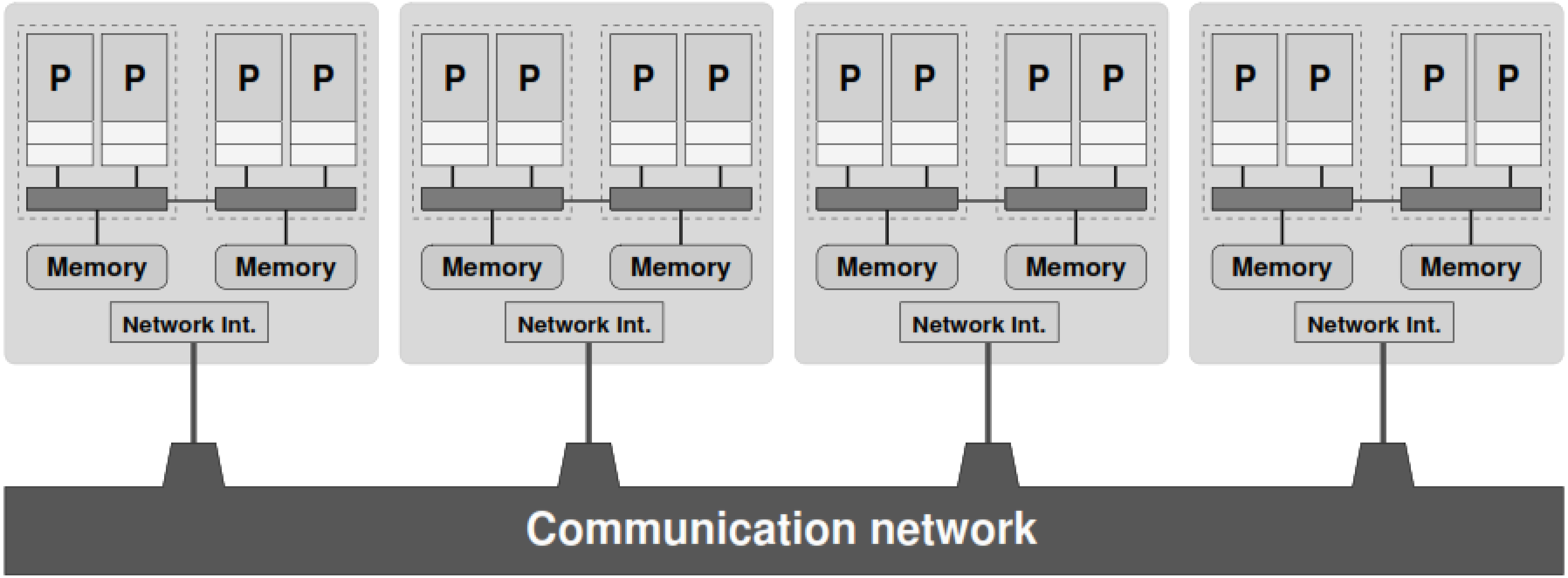
NORMA

- The distributed-memory architecture outlined here is also named *No Remote Memory Access (NORMA).*
- *Some vendors provide libraries and sometimes hardware* support for limited remote memory access functionality even on distributed-memory machines.
- Since such features are strongly vendor-specific, and there is no widely accepted standard available

Hierarchical (hybrid) systems

- As already mentioned, large-scale parallel computers are neither of the purely shared-memory nor of the purely distributed-memory type
- But a mixture of both, i.e., there are shared-memory building blocks connected via a fast network.
- This makes the overall system design even more anisotropic than with multicore processors and cNUMA nodes, because the network adds another level of communication characteristics
- The concept has clear advantages in terms of price vs. performance;
- It is cheaper to build a shared-memory node with two sockets instead of two nodes with one socket each, as much of the infrastructure can be shared.
- Moreover, with more cores or sockets sharing a single network connection, the cost for networking is reduced.

Hierarchical (hybrid) systems



- Typical hybrid system with shared-memory nodes (ccNUMA type).
- Two-socket building blocks represent the price vs. performance “sweet spot” and are thus found in many commodity clusters.

Hybrid Systems

- Two-socket building blocks are currently the “sweet spot” for inexpensive *commodity clusters,*
- *i.e., systems built from standard components that were not specifically* designed for high performance computing.
- Depending on which applications are run on the system, this compromise may lead to performance limitations due to the reduced available network bandwidth per core.
- Moreover, it is *per se unclear how* the complex hierarchy of cores, cache groups, sockets and nodes can be utilized efficiently.
- The only general consensus is that the optimal programming model is highly application- and system-dependent.

Networks

- The communication overhead can have significant impact on application performance.
- The characteristics of the network that connects the “execution units,” “processors,” “compute nodes,” or whatever play a dominant role here.
- A large variety of network technologies and topologies are available on the market, some proprietary and some open.

Basic performance characteristics of Networks

- The simplest and cheapest solution is Gigabit Ethernet
- Which will suffice for many throughput applications but is far too slow for parallel programs with any need for fast communication.
- *InfiniBand is* the dominating distributed-memory interconnect in commodity clusters.

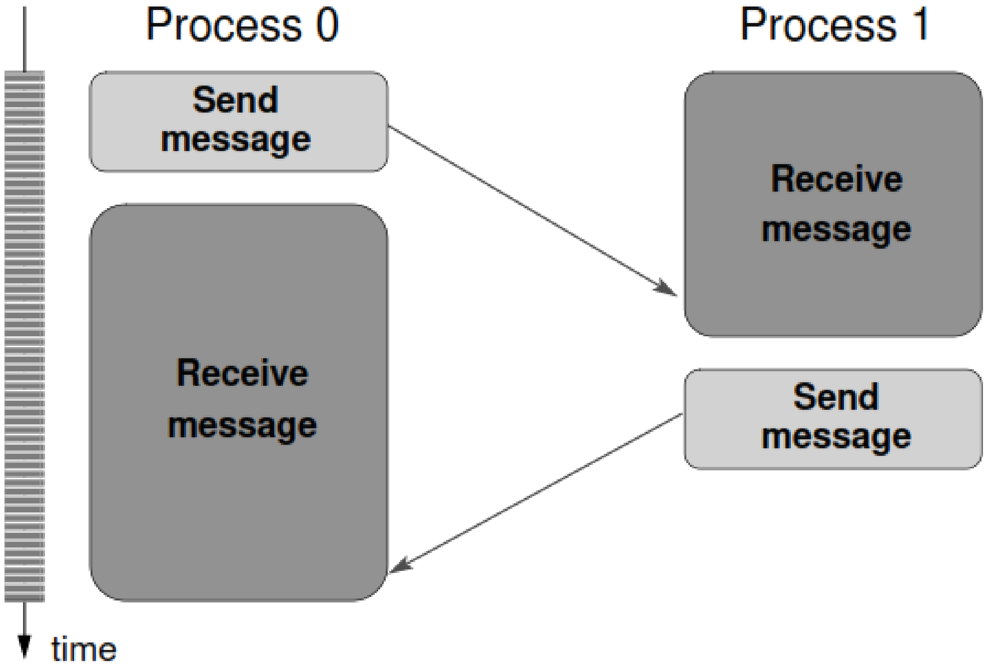
Latency

- All data transmission protocols have some overhead in the form of administrative data like message headers, etc.
- Some protocols (like, e.g., TCP/IP as used over Ethernet) define minimum message sizes, so even if the application sends a single byte, a small “frame” of *N > 1 bytes is transmitted.*
- Initiating a message transfer is a complicated process that involves multiple software layers, depending on the complexity of the protocol.
- Each software layer adds to latency.
- Standard PC hardware as frequently used in clusters is not optimized towards low-latency I/O

high-performance networks

- High-performance networks try to improve latency by reducing the influence of all of the above.
- Lightweight protocols, optimized drivers, and communication devices directly attached to processor buses are all employed by vendors to provide low latency.

PingPong

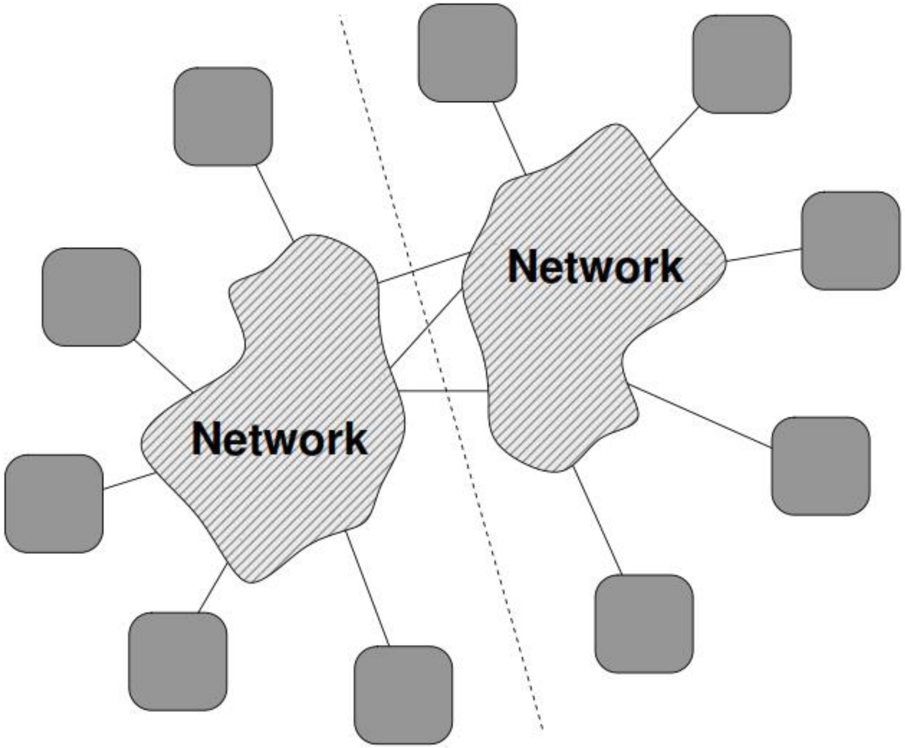


- Timeline for a “PingPong” data exchange between two processes.
- PingPong reports the time it takes for a message of length *N bytes to travel from process* 0 to process 1 and back.

Bisection bandwidth

- Note that the simple PingPong algorithm described above cannot pinpoint “global” saturation effects:
- If the network fabric is not completely nonblocking and all nodes transmit or receive data at the same time,
- Aggregated bandwidth, i.e., the sum over all effective bandwidths for all point-to-point connections, is lower than the theoretical limit.
- This can severely throttle the performance of applications on large CPU numbers as well as overall throughput of the machine.
- One helpful metric to quantify the maximum aggregated communication capacity across the whole network is its *bisection bandwidth* B_b
- It is the sum of the bandwidths of the minimal number of connections cut when splitting the system into two equal-sized parts.
- In hybrid/hierarchical systems, a more meaningful metric is actually the available bandwidth per core, i.e., bisection bandwidth divided by the overall number of compute cores.
- It is one additional adverse effect of the multicore transition that bisection bandwidth per core goes down.

Bisection Bandwidth



- The bisection bandwidth B_b is the sum of the bandwidths of the minimal number of connections cut (three in this example) when dividing the system into two equal parts.

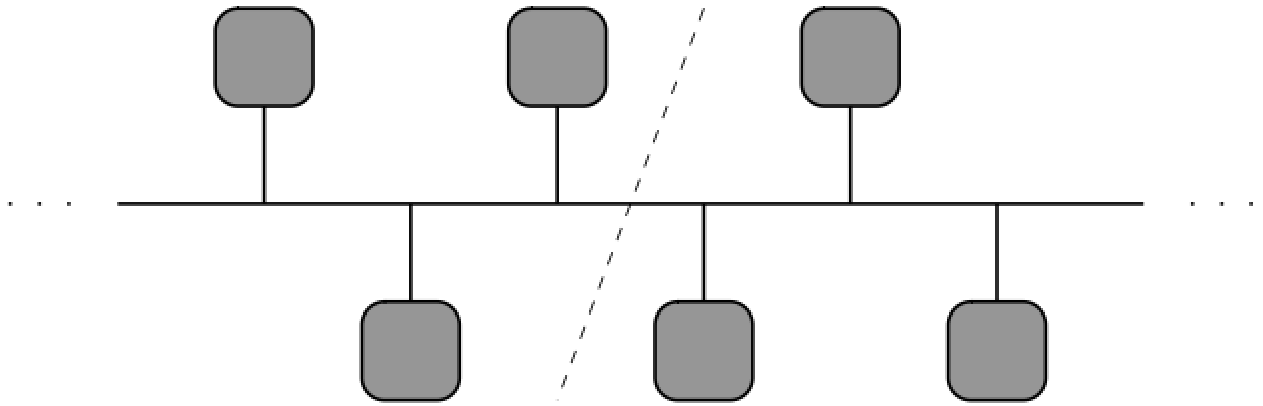
Buses

- A bus is a shared medium that can be used by exactly one communicating device at a time.
- Some appropriate hardware mechanism must be present that detects collisions (i.e., attempts by two or more devices to transmit concurrently).
- Buses are very common in computer systems.
- They are easy to implement, feature lowest latency at small utilization, and ready-made hardware components are available that take care of the necessary protocols.
- A typical example is the PCI bus, which is used in many commodity systems to connect I/O components.
- In some current multicore designs, a bus connects separate CPU chips in a common package with main memory.

Buses

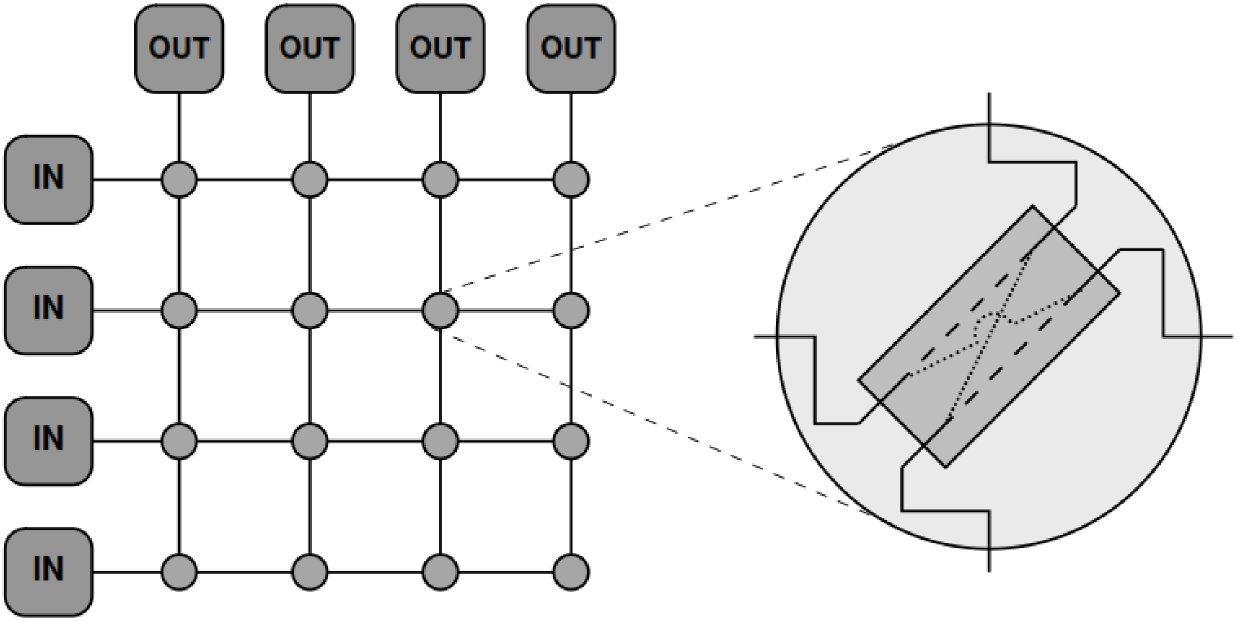
- The most important drawback of a bus is that it is *blocking.*
- *All devices share* a constant bandwidth, which means that the more devices are connected, the lower the average available bandwidth per device.
- Moreover, it is technically involved to design fast buses for large systems as capacitive and inductive loads limit transmission speeds.
- And finally, buses are susceptible to failures because a local problem can easily influence all devices.
- In high performance computing the use of buses for high-speed communication is usually limited to the processor or socket level, or to diagnostic networks.

Bus



- A bus network (shared medium).
- Only one device can use the bus at any time, and bisection bandwidth is independent of the number of nodes.

2D Crossbar Network



- A flat, fully nonblocking two-dimensional crossbar network.
- Each circle represents a possible connection between two devices from the “IN” and “OUT” groups, respectively, and is implemented as a 2×2 switching element.
- The whole circuit can act as a four-port nonblocking switch.

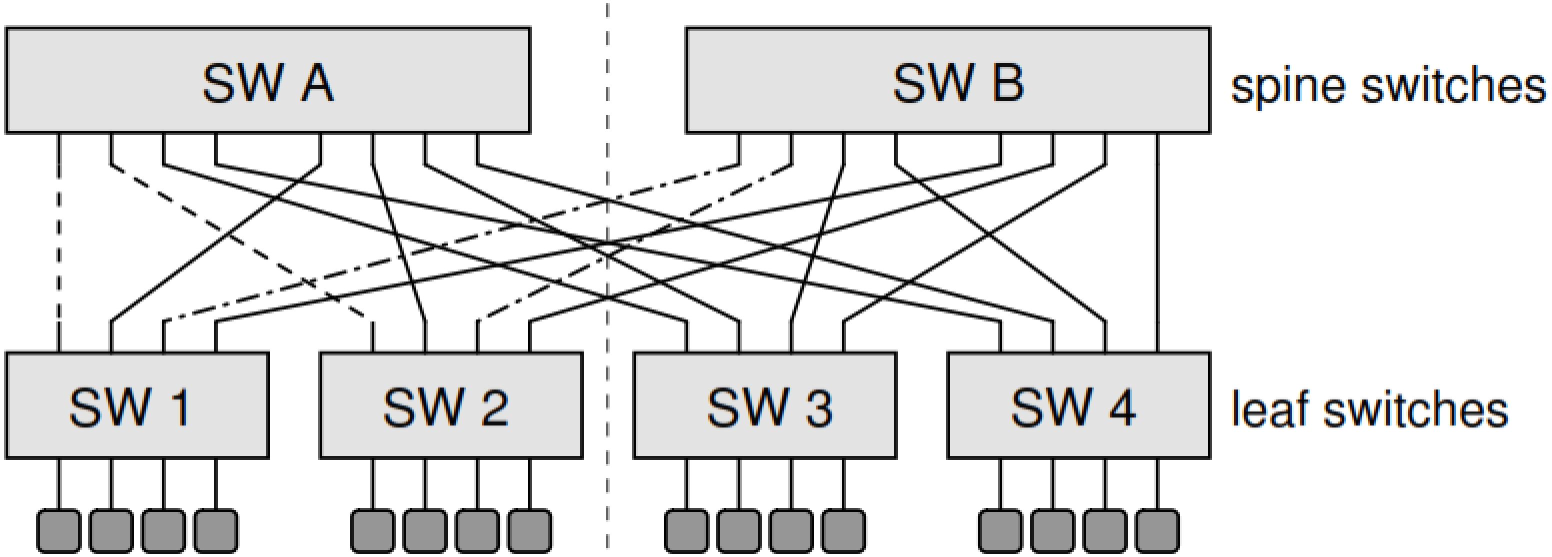
Switched and fat-tree networks

- A switched network subdivides all communicating devices into groups.
- The devices in one group are all connected to a central network entity called a *switch in* a star-like manner.
- Switches are then connected with each other or using additional switch layers.
- In such a network, the distance between two communicating devices varies according to how many “hops” a message has to sustain before it reaches its destination.
- Therefore, a multiswitch hierarchy is necessarily heterogeneous with respect to latency.
- The maximum number of hops required to connect two arbitrary devices is called the *diameter of the network.*

Switched and fat-tree networks

- A single switch can either support a fully *nonblocking operation,*
    - *Which means* that all pairs of ports can use their full bandwidth concurrently, or
    - It can have partly or completely — a bus-like design where bandwidth is limited.
- One possible implementation of a fully nonblocking switch is a *crossbar.*
- *Such* building blocks can be combined and cascaded to form a *fat tree switch hierarchy*
- In this case the bisection bandwidth per compute element is less than half the leaf switch bandwidth per port, and contention will occur even if static routing is itself not a problem.
- Note that the network infrastructure must be capable of (dynamically or statically) balancing the traffic from all the leaves over the thinly populated higher-level connections.
- If this is not possible, some node-to-node connections may be faster than others even if the network is lightly loaded.
- On the other hand, maximum latency between two arbitrary compute elements usually depends on the number of switch hierarchy layers only.

full-bandwidth fat-tree network

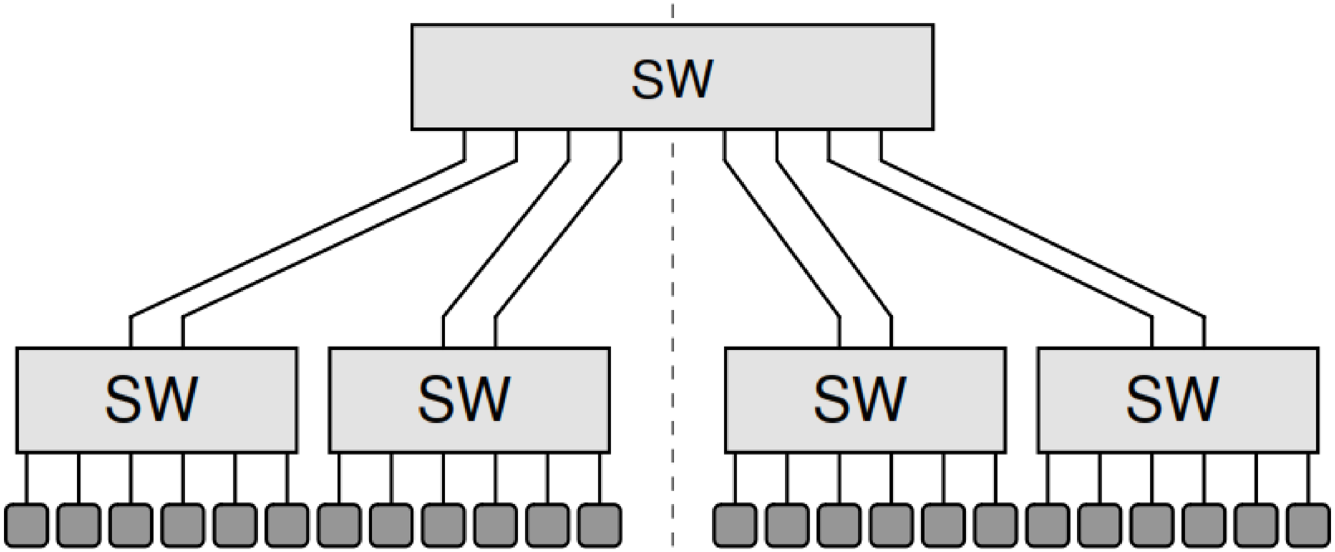


- A fully nonblocking full-bandwidth fat-tree network with two switch layers.
- The switches connected to the actual compute elements are called *leaf switches, whereas the upper* layers form the *spines of the hierarchy.*

Static and Adaptive Routing

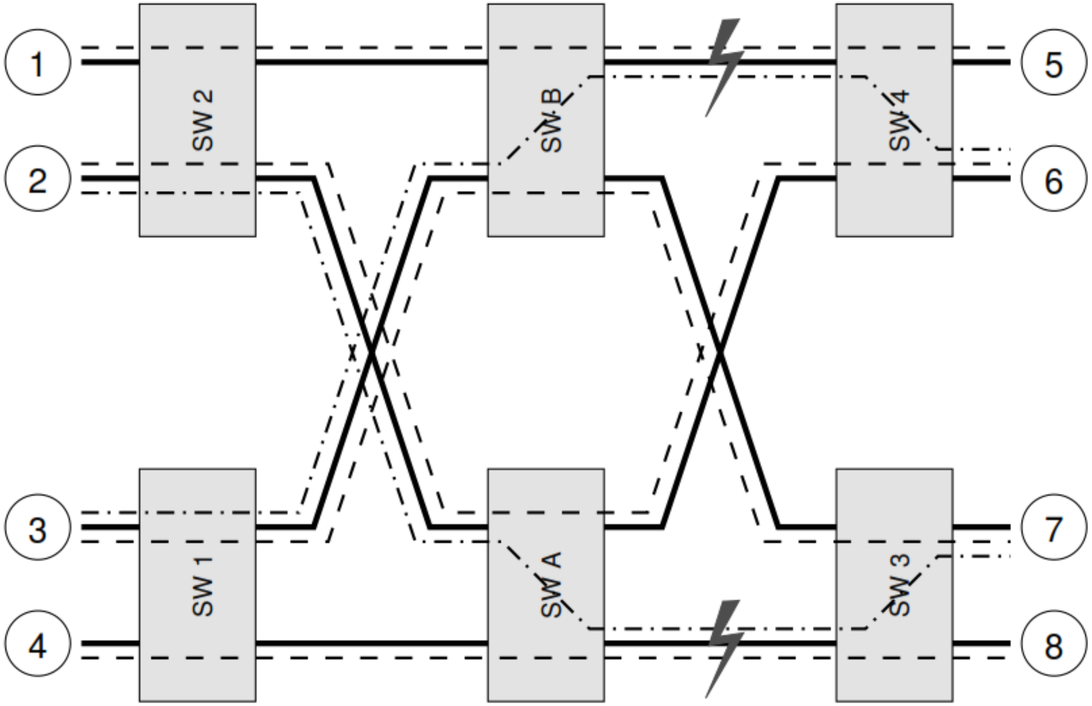
- If *static routing is used, i.e., if connections* between compute elements are “hardwired” in the sense that there is one and only one chosen data path (sequence of switches traversed) between any two
- One can easily encounter situations where the utilization of spine switch ports is unbalanced, leading to collisions when the load is high.
- Many commodity switch products today use static routing tables.
- In contrast, *adaptive routing selects data paths depending on the network load and thus avoids* collisions.
- Only adaptive routing bears the potential of making full use of the available bisection bandwidth for all communication patterns.

Flat Free Network



- A fat-tree network with a bottleneck due to “1:3 oversubscription” of communication links to the spine.
- By using a single spine switch, the bisection bandwidth is cut in half as compared to the layout in last Figure because only four nonblocking pairs of connections are possible.
- Bisection bandwidth per compute element is even lower.

Fat Tree Switch Hierarchy



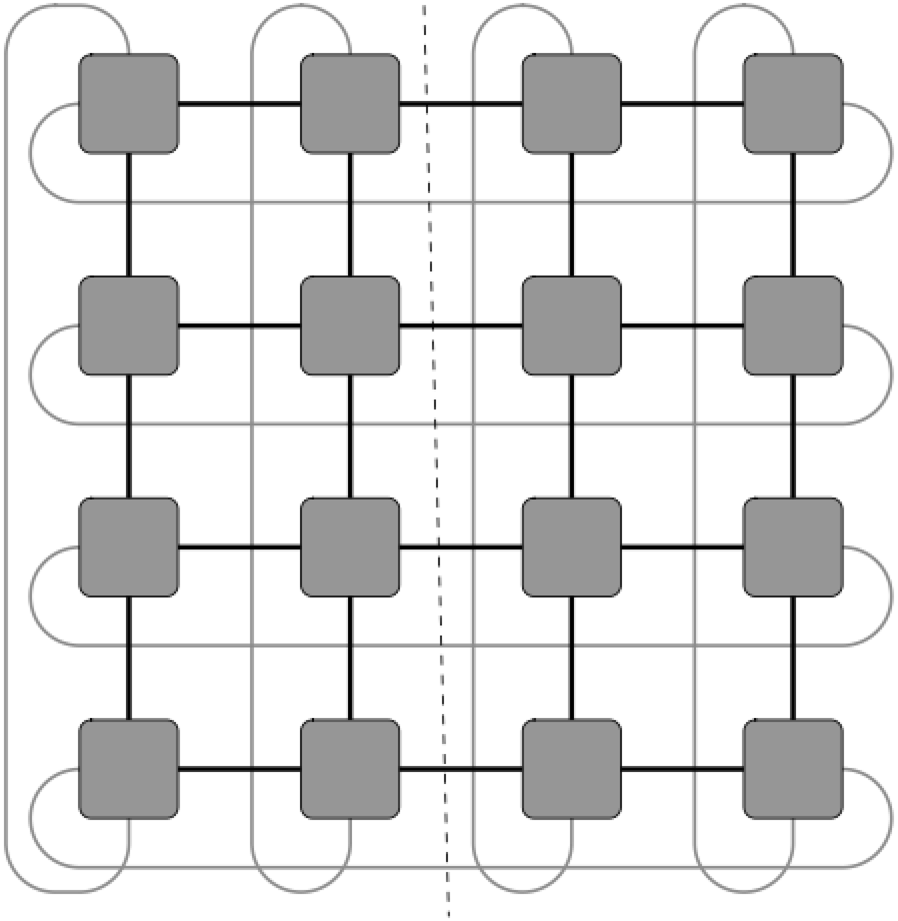
- Even in a fully nonblocking fat-tree switch hierarchy (network cabling shown as solid lines), not all possible combinations of *N/2 point-to-point connections allow collisionfree* operation under static routing.
- When, starting from the collision-free connection pattern shown with dashed lines, the connections 2⟷6 and 3⟷7 are changed to 2⟷7 and 3⟷6, respectively (dotted-dashed lines), collisions occur, e.g., on the highlighted links *if connections* 1⟷5 and 4⟷8 are not re-routed at the same time.

Mesh networks

- Fat-tree switch hierarchies have the disadvantage of limited scalability in very large systems, mostly in terms of price vs. performance.
- The cost of active components and the vast amount of cabling are prohibitive and often force compromises like the reduction of bisection bandwidth per compute element.
- In order to overcome those drawbacks and still arrive at a controllable scaling of bisection bandwidth, large MPP machines like the IBM Blue Gene or the Cray XT use *mesh networks, usually in the form of multidimensional (hyper-)cubes.*
- *Each* compute element is located at a Cartesian grid intersection.
- Usually the connections are wrapped around the boundaries of the hypercube to form a torus topology.
- There are no direct connections between elements that are not next neighbors.
- The task of routing data through the system is usually accomplished by special ASICs (application specific integrated circuits), which take care of all network traffic, bypassing the CPU whenever possible.
- The network diameter is the sum of the system's sizes in all three Cartesian directions.

A two-dimensional (square) torus network

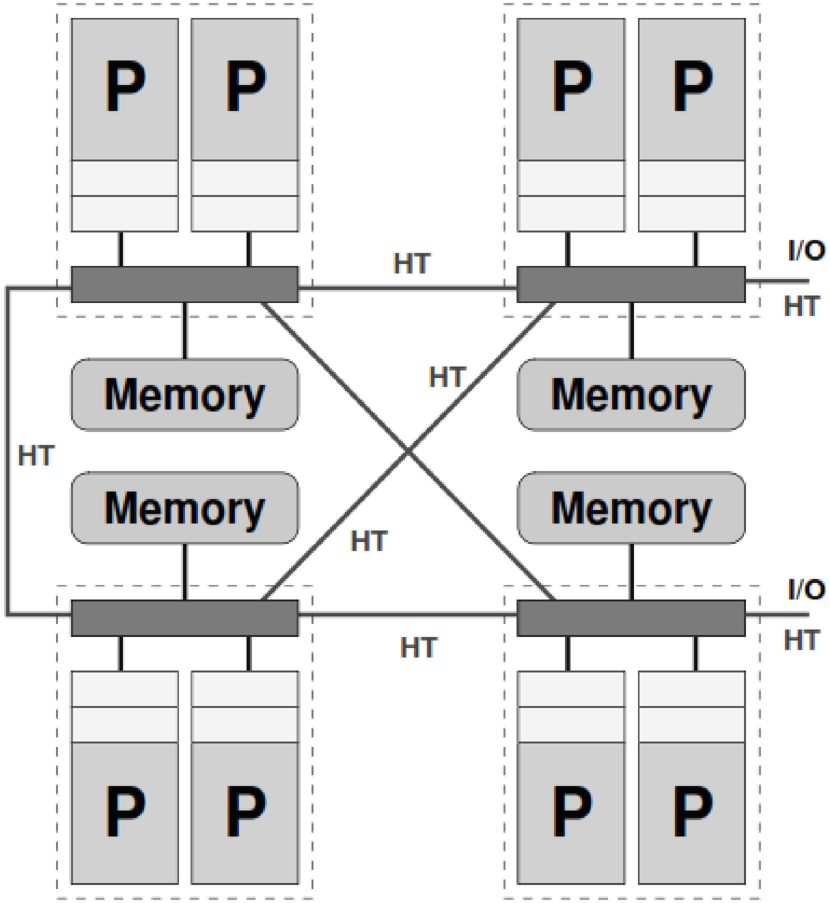
- Bisection bandwidth scales like Sqrt(*N) in this case.*



- Maximum latency scales like $N^{1/d}$.
- Although these properties appear unfavorable at first sight, the torus topology is an acceptable and extremely cost-effective compromise for the large class of applications that are dominated by nearest-neighbor communication.
- If the maximum bandwidth per link is substantially larger than what a single compute element can "feed" into the network (its *injection bandwidth), there is enough headroom to support* more demanding communication patterns as well.
- Another advantage of a cubic mesh is that the amount of cabling is limited, and most cables can be kept short.
- As with fat-tree networks, there is some heterogeneity in bandwidth and latency behavior, but if compute elements that work in parallel to solve a problem are located close together, these characteristics are well predictable.
- Moreover, there is no "arbitrary" system size at which bisection bandwidth per node suddenly has to drop due to cost and manageability concerns.

Four-socket ccNUMA system

- Figure A four-socket ccNUMA system with a HyperTransport-based mesh network.
- Each socket has only three HT links, so the network has to be heterogeneous in order to accommodate I/O connections and still utilize all provided HT ports.
- On smaller scales, simple mesh networks are used in shared-memory systems for ccNUMA-capable connections between locality domains.
- Figure shows an example of a four-socket server with HyperTransport interconnect.
- This node actually implements a heterogeneous topology (in terms of intersocket latency) because two HT connections are used for I/O connectivity:
- Any communication between the two locality domains on the right incurs an additional hop via one of the other domains.



Hybrids

- If a network is built as a combination of at least two of the topologies described above, it is called *hybrid.*
- *In a sense, a cluster of shared-memory nodes* implements a hybrid network even if the internode network itself is not hybrid.
- This is because intranode connections tend to be buses (in multicore chips) or simple meshes (for ccNUMA-capable fabrics like HyperTransport or QuickPath).
- On the large scale, using a cubic topology for node groups of limited size and a nonblocking fat tree further up reduces the bisection bandwidth problems of pure cubic meshes.

Reference

- **Georg Hager and Gerhard Wellein**, Introduction to High Performance Computing for Scientists and Engineers, CRC Press, 2011.