

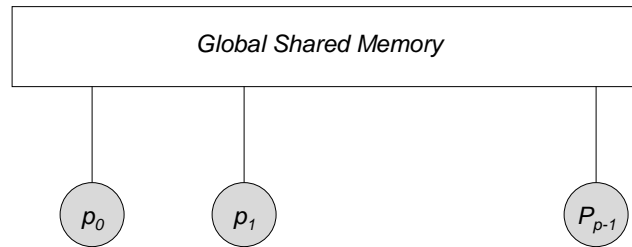
Parallel models

9/3/07

PRAM submodels

- Memory is shared amongst all processors and can be accessed in unit time
- Concurrent access restrictions resolve cases where several processors attempt to access the same memory location
- The corresponding submodels are:
 - Exclusive read and exclusive write (EREW)
 - Concurrent read and exclusive write (CREW)
 - Exclusive read and concurrent write (ERCW)
 - Concurrent read and concurrent write (CRCW)

Parallel Random Access Machine



Concurrent writes

- The result of a concurrent write is not always clear. Some results are:
 - Common: all values must be the same
 - Arbitrary: one processor succeeds
 - Priority: processors are assigned priorities
 - Combination: the value stored is a function of all of the values written (i.e., sum)

Discussion

- Consider the example of computing a logical OR of p bits using a p -processor machine using CRCW
- This can be computed in constant time using any CRCW submodel:
 - One processor writes a 0
 - A 1 is written by all processors whose bit is 1
- The CREW model, however, requires at least $\log p$ time. Thus, CRCW can be more powerful if memory concurrent access is feasible

Advantages

- The PRAM model is simple and allows maximum parallelism because of limited theoretical overhead
 - Memory access is bounded by some constant and therefore does not affect our analysis
- For this reason, these algorithms are useful for:
 - Deriving a starting point for computation parallelism
 - Proving lower bounds as it can not run faster with communication overhead
 - Show some problems are difficult to parallelize (similar to P vs. NP problems)

Limitations

- Memory is inherently sequential: it can accept only one address at a time and deliver its contents
- One mechanism to minimize this consequence is to construct *memory modules* where each can be accessed simultaneously
- Truly parallel access requires all p processors should be connected to all m memory modules
 - $\Theta(pm)$ connections total

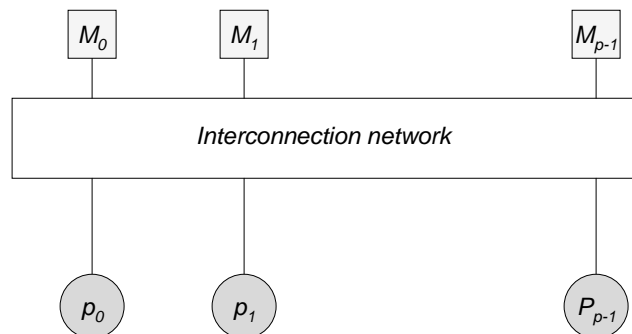
Realistic models

- Maximizing concurrent memory access leads to improved performance, but we want to also minimize overall cost.
- If we have p processors:
 - Having fewer than p memory modules results in potential serialization (pigeonhole principle)
 - Having more than p memory modules increases the number of connections
 - p modules is the best compromise

Network shared memory

- p processors and p memory modules connected through an interconnection network
- Every processor can access any memory module in this framework

Parallel shared memory model



Design criteria

- Since each processor will require one connection to avoid serialization, the connection complexity should be $\Omega(p)$
- Connecting every processor to every memory module requires $\Theta(p^2)$ connections

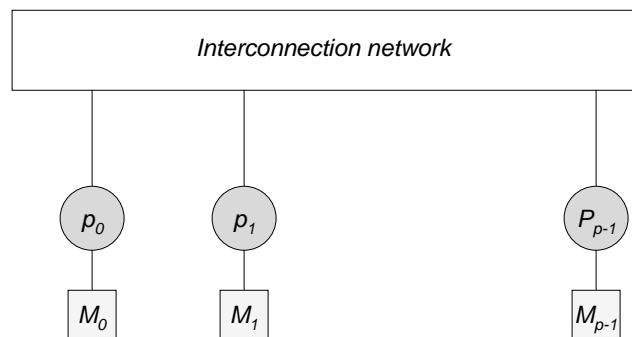
Scalability

- An ideal system that has a quadratic number of connections is not practical
- An architecture is said to be scalable if it is possible to build large configurations
- We will focus on architectures that are scalable and can accommodate large numbers of processors as needed
 - Complexity should be close to $O(p)$

Networked distributed memory

- Suppose we connect each memory module M_i to a processor P_i
- Note that this does not affect the overall connection complexity because only p connections are added (network $\Omega(p)$)
- Practical implementations ensure that remote access of M_i does not affect P_i

Networked Distributed Memory



Advantages

- Local memory can be accessed as quickly as done in sequential computation
- As such, we can structure our computation to utilize local memory as much as possible
- This also helps design practical PRAM implementations (e.g., ccNUMA)

Conclusions

- Networked distributed memory is superior to networked shared memory
- Shared memory, in theory, is superior to networked distributed memory but not feasible in practice
- Most scalable parallel computers have distributed memory configurations

Algorithm design

- Note that any distributed memory algorithm can also run on a shared memory system
- Many shared memory algorithms, however, do not minimize remote access and as a result suffer on distributed systems
- We will focus on distributed memory algorithms in this course

Distributed vs. shared (cont.)

- In the discussion in the parallel computing community, programming paradigms are often confused with parallel architectures.
- Both architectures allow the opposite programming paradigm, although shared memory programming is simpler
- Commercial scalable systems that offer non-uniform memory access, or NUMA systems, are available but minimizing remote accesses minimizes overall runtime

Our framework

- Throughout the course we will describe algorithms that interact by explicit communication
- This will allow us to better estimate the cost of using the interconnection network to transfer information, or the communication complexity

Network parameters

- Diameter
 - Maximum distance between any pair of processors on the network
 - Smaller the diameter, better the network
- Bisection width
 - Minimum number of links to cut to separate the network into two independent halves
 - Helps us identify “bottlenecks” in the network

Importance of diameter

- Maximum distance between two processors can be used to define the lower bound for communication
- Best: fully connected network = 1
- Worst: array network = $p - 1$

Importance of bisection width

- Tests the ability of the network to deal with traffic
- Specifically, if we need to transfer n data items during any one time step, the communication requires at least $\Omega(n/BW)$

Example

- Suppose we would like to sort n unique numbers in increasing order
- Each processor has n/p numbers
- If the input is such that the order is in decreasing order, all n items must be shifted across the bisection width
 - This implies at least $\Omega(n/BW)$ communication time

One more: links per node

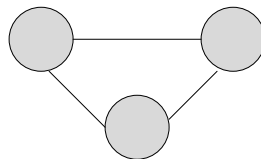
- Another important metric is how many links each node has to the network
- If the number of links is not a linear function of p , it may be inefficient when added to a larger network

Static connection networks

- Processors are connected directly
- To visualize, consider a graph where the nodes are the processors and the edges are communication links
- The best case scenario is a fully connected graph

Fully-connected network

- Diameter = 1
- Number of nodes per link = $p - 1$
- Highly non-scalable
- Bisection width = $\Theta(p^2)$ (see text)



Bisection width

- In general, it is easier to estimate the number of links to cut than the actual bisection width of a network
- This is because the estimated bisection width should be at least as large as the actual bisection width
- This is often sufficient for lower bounds

Array/ring network

- Number of links per node = 2
- Allows arbitrarily large networks to be constructed with minimal cost
- Diameter can be reduced by a factor of two by converting an array into a ring network
- Bandwidth is improved by a factor of two

Multidimensional mesh/torus

- An array is a one-dimensional mesh
- We will only discuss meshes with the same number of processors along each dimension because they minimize the diameter
- Connecting the first column of processors to the last column results in a torus