

Hypercubes

9/5/07

Static Interconnection Networks

- All of the examples we discussed last class were static networks, *i.e.*, links connect processors directly
- Today, we will discuss more examples of these configurations and begin to look at dynamic interconnection networks

Arrays vs. rings

- By increasing the number of links of an array, we previously reduced the diameter
- An array is a simple one dimensional mesh. Increasing dimensionality will further improve diameter.
- How can we construct effective meshes for parallel algorithms?

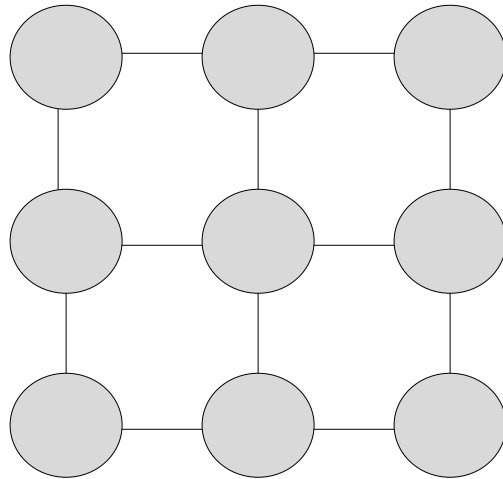
Links per node of a mesh

- There are 2 links per dimension for each node of the mesh
- Therefore, the total number of links per node is $2d$
- We can conclude that meshes are scalable

Basics

- Suppose we have a d dimensional mesh of p processors with k processors along each dimension
 - $p = k^d$
- To travel between the farthest two points, $k - 1$ links along each dimension must be traversed
 - Diameter = $d(k - 1) = \Theta(dp^{1/d})$
- Note meshes with the same number of processors along each dimension minimize the diameter

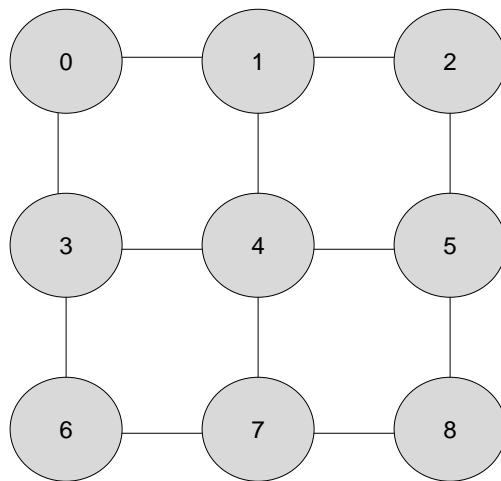
3X3 mesh



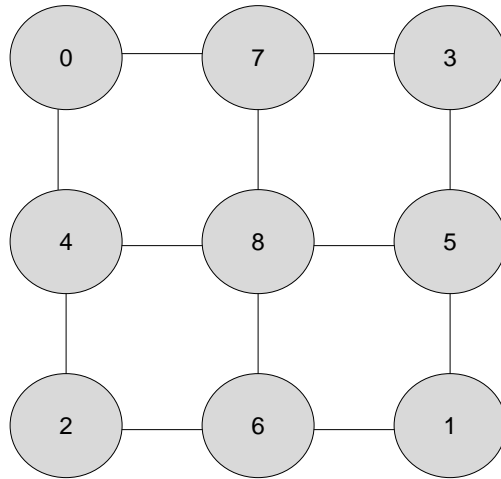
Bisection width

- A mesh can be broken into two by splitting it along one of the dimensions
- To do so, we must cut k^{d-1} links
- Bisection width is estimated to be:
 - $\Theta(p^{1-1/d})$

Labeling the processors of a mesh



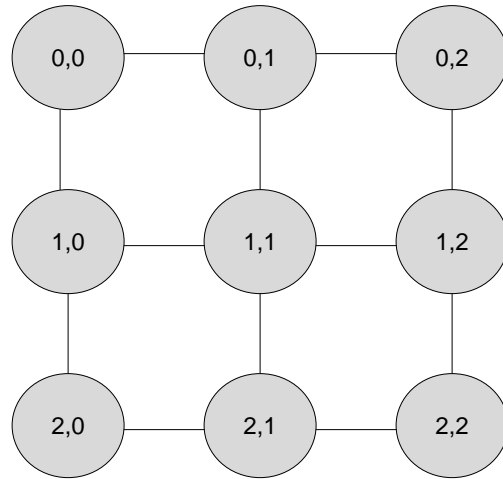
Labeling the processors of a mesh



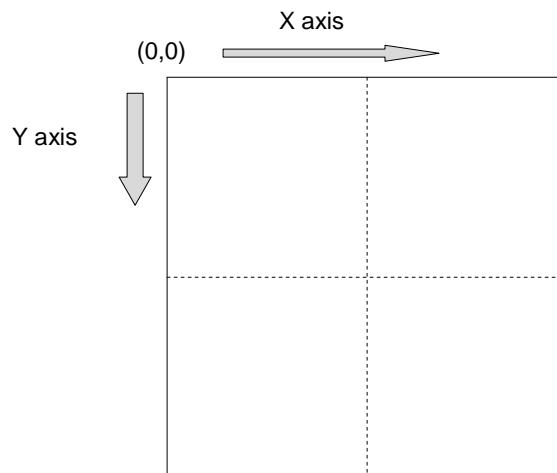
Practical considerations

- In a parallel algorithm, a given processor often needs to know its neighbors based solely on its identifier/location
- We achieve this result by labeling a processor by its row and column
 - This framework can be extended beyond two dimensions, using special care for the corners

Row-column identification



Neighbor identification



Choose your system

- Given a choice of the following 16 processor systems, which would you choose and why?
 - 4X4 two-dimensional mesh
 - 2X2X4 three-dimensional mesh
 - 2X2X2X2 four-dimensional mesh

Optimal meshes

- The diameter of a mesh is $\Theta(dp^{1/d})$ and the bisection width is $\Theta(p^{1-1/d})$
- From above, note as we increase the dimensionality d , the diameter decreases and the bisection width increases.
- Why not form a 1X1X1 system?

Generalizing this concept

- We would like to build a p processor system, and $p = 2^d$
- Based on the previous discussion, we maximize d by creating a $2 \times 2 \times \dots \times 2$ system.
 - In other words, $d = \log_2 p$

Conclusions

- Given a mesh, the diameter is $\Theta(dp^{1/d})$ and the bisection width is $\Theta(p^{1-1/d})$
- $d = \log p$
- Therefore, diameter is $\Theta(\log p)$ and the bisection width is $\Theta(p)$

Conclusions

- Advantages:
 - Best diameter and bisection width we have seen so far
 - Practically, these are very useful bounds
- Disadvantages:
 - Number of links per node is not constant
 - $p = 2^d$, $2d$ links per node = $\Theta(\log p)$

Hypercubes

- We call these structures hypercubes because they have the same length or number of processors along every dimension
- Note that higher order structures are possible with modest numbers of processors

Example cubes

Labeling processors in a hypercube

- Because a hypercube is a mesh, we will use the same strategy to label processors
- Fix a coordinate system of the appropriate dimension at one corner of the hypercube, imagine all links to be unit length, and label based on these coordinates.

Bit labeling

- Because the number of processors along any given dimension is 2, we can represent a single coordinate by a single bit: 0 or 1
- Each processor, therefore, has an identifier of d bits
 - For example, (0,1,1) in a 8-processor hypercube

Finding neighbors

- If a processor has a 0 in a given dimension, it does not have a left neighbor
 - We can tell if the coordinates are negative
- Likewise, a processor with a 1 in a given dimension does not have a right neighbor
- The unique neighbor can be found by fixing all dimensions except one and flipping the bit from 0 to 1, or visa versa

Generalization

- Flipping any of the d bits returns a neighbor of a given processor
 - Each flip corresponds to a link in the network
- Because at most d bits can be flipped, a processor has d neighbors
- This also gives us a quick algorithm to check if any two processors are neighbors
 - Hamming distance of 1

Conclusions

- A hypercube is a symmetric structure
- We can pick any one of the d dimensions and split the hypercube into two hypercubes of the next smaller dimension
- For example, separate based on the most significant bit

Communication paths

- First, find the number of bits that two messages differ
- Route the message to a neighbor, and continue until it arrives
- If the processor ids differ by k bits, there are $k!$ possible shortest paths possible
 - High connectivity

Other facts

- What is the diameter of a hypercube and why?
- We can remove $p/2$ links to split a hypercube into two and that is the actual bisection width

Butterfly networks

- Suppose we would like to have a constant number of links per node but still have the connectivity advantages of a hypercube
- Lets replace each hypercube processor with $\Theta(\log p)$ processors
 - This allows us to accommodate the $\log p$ links of a hypercube

Overview

- We will use $p(\log p + 1)$ processors and arrange them into $\log p + 1$ columns
 - Each column will have p processors
- Each processor can be identified using its row identifier
 - This is also called a hypercube id as it represents one hypercube processor

Dynamic interconnection networks

- Suppose we develop a switch with two inputs and two outputs, and this switch is capable of connecting any input to any output
- A general switch of this type is called a *crossbar*, and we are discussing a 2X2 crossbar
- Any network with switches that route messages from one processor to another is called a dynamic interconnection network

Dynamic butterfly

- We will replace processors in a static butterfly network with switches
- Because of properties of the crossbar, we require only $p/2$ switches per column and $\log p$ columns

Our overall goal

- We would like distinct pairs to communicate simultaneously
- Here, we consider cases where each processor sends a single message and no two destinations are the same
- We can represent these as permutations
 - For example, (1 2 3) or (3 2 1)

An ideal network

- We want our network to support such permutations and we call such a configuration a permutation network
- All multistage interconnection networks do not achieve this, but any permutation can be broken into 3 sets of communications that can be achieved in parallel.
- In practice, this means there is a constant overhead for any permutation communication step.