

CSCE 5160 Parallel Processing

Homework #1: 2.2, 2.3, 2.12 2.13
Due Sept 10, 2009

**Remote classes using WebEx
if necessary**

Grader: Shu Chen
shuchen@my.unt.edu
Office Hours: Thursday 8-12

Review

- Memory latency and Bandwidth
- Cache memories
- Multithreading and prefetching
- PRAM models
 - EREW, CREW, ERCW, CRCW models
 - Concurrent write: Common, arbitrary, priority, sum (reduction)
- Complexity analysis
- Interconnection networks

Interconnection Networks

Processor - Memory and Processor-Processor networks

We may have networks connecting processors directly to memory units (shared memory)

Or

We may have networks connecting processors to processors (message passing)

Completely connected: every memory (or processor) is connected to every processing element (Figure 2.14, page 39)

Star Connected – Figure 2.14 page 39

Cross-bar networks. We need $p*b$ switching elements and each element will have $p+m$ wires. See page 35, Figure 2.8

Interconnection Networks

Bus-based networks: not scalable in performance, but cost-effective.

The bus contention can be reduced by using caches with each processor (page 34, Figure 2.7)

Here we need only $p+m$ switches and each switch need only one connection (or wire)

Multi-stage networks (page 36-38)

Multistage networks are constructed using $a \times b$ switch boxes with a inputs and b outputs. Most commonly we have 2×2 switch boxes (figure 2.11, page 37). Here we have either straight (pass-through) connected or cross-connected.

At each stage of the multi-stage network, we will have $\log_2(n)$ switches.

“perfect shuffle” – see figure 2.10 page 37

$$j = 2i \text{ if } 0 \leq i \leq p/2-1$$

$$j = 2i+1-p \text{ if } p/2 \leq i \leq p-1$$

Omega (Figure 2.12 page 38) is most popular among the multi-stage networks.

Interconnection Networks

Omega Network.

Routing a message from a processor s to a memory module t can be easily achieved. Consider s and t as binary numbers. The message arrives at the first switch.

If the most significant bits of s and t are the same, the switch at first stage is connected for pass-through;

If they differ, the switch is connected for cross-connection.

We use the next significant bit to determine the connection of the switch in second stage, and so on.

An example of communication path(s) shown in figure 2.13

Only one path can succeed –known as **blocking networks**

When there is no conflict for a “switch” multiple paths are possible

How many stages and how many switches?

$\log_2(p=b)$ stages and $\log_2(p)*p/2$ switches with p processors and $p=b$ memory banks.

Interconnection Networks

Rings – page 40, Figure 2.15

Mesh connected – 2-D, 3D etc (page 40)

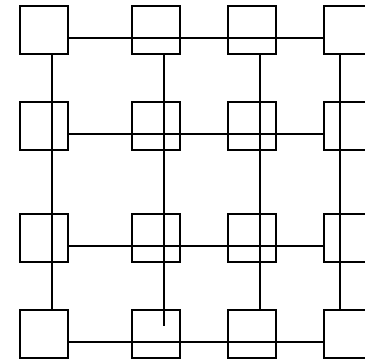
Torres network or wrap-around connected mesh networks

Routing: X-Y routing

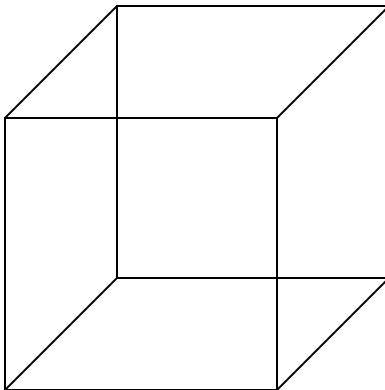
Go along X until you reach the column

Then go along Y dimension

(and then along Z etc.)

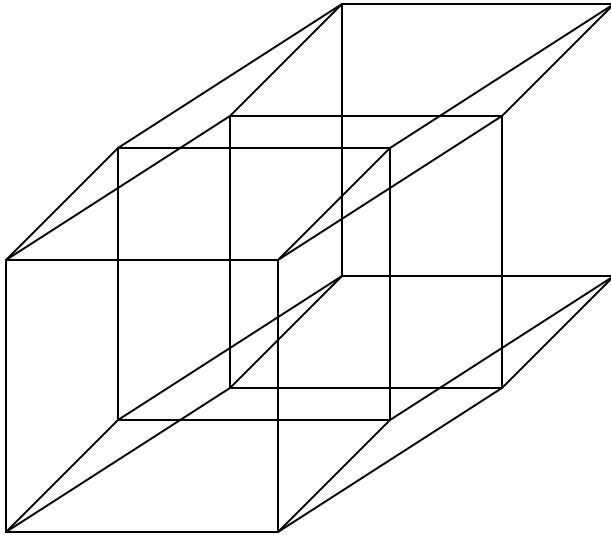


Hypercubes -- usually described as dimensions d for a network with 2^d nodes



3-D cube $p = 2^3 = 8$

Interconnection Networks



4-Dimensional

If processors are numbers in binary, the number of bits needed to number all processors will equal d (the number of dimensions).

To send a message from processor s to processor t , we compare each bit of s with that of t . If they are different, we move one edge in that dimension.

Interconnection Networks

Properties of Hypercube --

Note that each node must be connected to d other nodes. Each node has d neighbors.

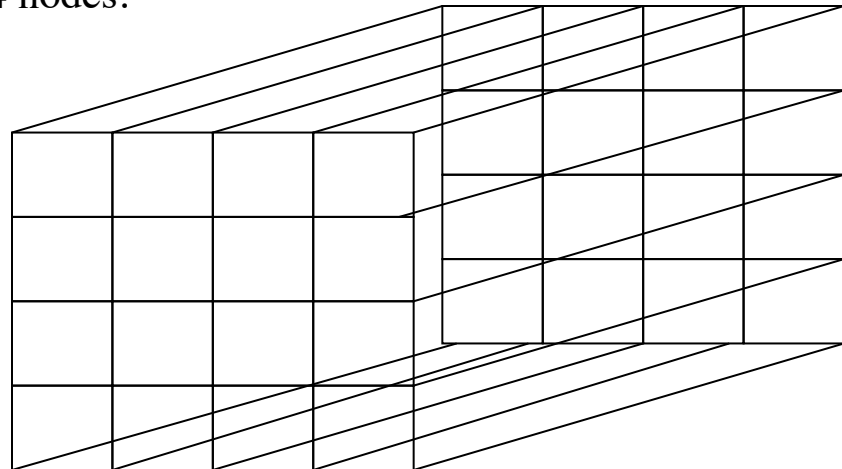
Hamming distance -- number bits in which two numbers differ.

The Hypercubes we have seen are sometimes called binary hypercubes or 2-ary hypercubes. There are only two values in each dimensions (0, 1).

In general we can have k-ary hyper cubes. Each dimension will have k values (0, 1, 2, ..., $k-1$)
Consider 4-ary 3-d cube. We will have $k^3 = 64$ nodes.

Nodes will be numbered

000	100	200	300
001	101	201	301
002	102	202	302
003	103	203	303
010	110	210	310
011	111	211	311
012	112	212	312
013	113	213	313
----	----	----	----



Interconnection Networks

Routing in k-ary Hypercube.

In each dimension, we will rely on a linear-array to communicate among the processing nodes.

How many connections are needed?

Each node is still connected to only d neighbors.

with $4^3 = 64$ nodes, in k -ary we need only 3 connections from each node

with $2^6 = 64$ nodes in binary Hypercube we need 6 connections from each node

Of course, binary hypercube requires only d -steps to communicate between any pair of nodes.

k -ary hypercube requires $(k-1)*d$ steps

Other Interconnection Networks

Trees -- there is exactly one path between any pair of nodes.

Normally, the processing nodes are the leaves and we have switching elements at other nodes -- see figure 2.18 on page 42

How to route a message? Go up the tree until you find a common parent, then move down the tree.

What is the problem?

Switches at higher levels of the tree (closer to the root) will be congested.

Fat trees -- provide greater bandwidth as you go up the tree.

See figure 2.19 on page 42

Properties of Interconnection Networks

How to compare the various networks?

Diameter: The maximum distance between any two processors (in terms of the number of links to traverse).

Connectivity (or arc-connectivity): The minimum number of links that must be removed to divide the network into two disconnected networks.

We can also define node-connectivity in a similar manner -- the minimum number of nodes (and associated links) to disconnect the network into two disjoint networks.

Bisection width: The minimum number of links that must be removed to disconnect the network into two-equal sized sub-networks.

A channel width is the number of bits (or bytes) that can simultaneously travel along a link (number of wires). The rate at which data can be transmitted on a link along with the channel width define the channel **bandwidth**.

Using the channel bandwidth to define **bisection bandwidth**.

Properties of Interconnection Networks

Communication performance

Latency (set up time, overhead in routing
saving messages at intermediate nodes, etc)

Bandwidth

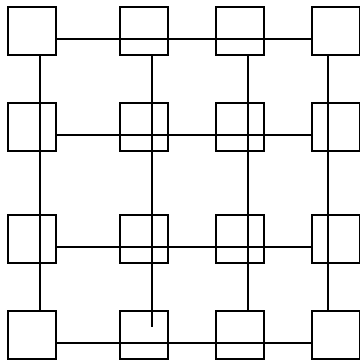
Cost. Typically in terms of switches, links or wires

Using these various parameters, we can find out how long it takes a message to reach its destination. We can also define the maximum capacity of a network (or reliability) using connectivity measures.

See Table 2.1 on page 44 that summarizes some of these properties for different networks

Impact of Interconnection Networks

Let us consider an example to see the impact of an interconnection on the performance of a parallel algorithm. We will examine 4x4 Matrix multiply using a 4*4 mesh of **16 processors**.



0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

How do we distribute the data?

Allocate each array element to a different processor?

$$c[i,j] = c[i,j] + a[i,k]*b[k,j]$$

[i,j] is the processor number

$c[i,j]$, one $a[i,k]$ and one $b[k,j]$ are local to the processors -- other processors must receive $a[i,k]$ and $b[k,j]$ from their neighbors.

worst-case time = 6 steps = $O(\sqrt{(n*n)}) = O(n)$

we will derive more accurate equations for “communication performance”

Impact of Interconnection Networks

Note our n is 4 since we are dealing with 4×4 matrix

How many times is the k loop repeated?

```
for (i=0; i< n; i++;)
    for (j=0; j<n; j++
        {c[i,j] =0; for (k=0; k<n; k++) {c[i,j] = c[i,j]+a[i,k]*b[k,j];}}
```

The i and j loops are repeated only once, k is repeated n times

So, the total algorithm complexity = $n * O(n) = O(n^2)$

With n^2 processors we should expect a speed up of n^2 ,
or the algorithm should have a complexity of $O(n)$.

Abstract Parallel Processing Model

Consider an EREW PRAM with p processors and m memory locations.

We can emulate this model on a p -processor message-passing parallel computer in which each processor has m/p memory locations.

Let t be the run time of an algorithm on a p -processor EREW PRAM model.

Find an upper bound on the run time of the algorithm for the following architectures.

- a) a p -processor ring
- b) a p -processor mesh
- c) a p -processor hypercube

Some assumptions. Since we are starting with a PRAM model (shared memory model), we will assume that the run-time of t consists of only memory accesses.

Since we started with EREW model, we will simply assume that all memory accesses are read (since both reads and writes are sequential).

Abstract Parallel Processing Model

Also, since we are interested in an upper bound when the algorithm is implemented on a message-passing architecture, we will assume that each memory access by a processor is to a memory location at a remote node that is at the *farthest* distance from requesting processor.

Thus, each of the t memory accesses will have to travel the maximum distance possible.

Of course this is very poor distribution of the data. Ideally, we should minimize the communication between two processors by distribution the data needed by a processor into its local memory.

a). When we are dealing with a ring, the maximum distance any message must travel is $p/2$ (actually the I should use the ceiling operator for $p/2$ but since I do not have the correct font, I will not show that).

So, the upper bound on the algorithm is $t*(p/2)$.or in terms of complexity, $O(t*p)$

Abstract Parallel Processing Model

b). When dealing with a 2-D mesh, the maximum distance a message must travel is $2 * (p^{1/2})$, giving an upper bound for our algorithm as $O(t * (p^{1/2}))$.

c). For a hypercube, the maximum distance a message must travel is $\log_2(p)$, giving us an upper bound on our algorithm of $t * \log_2(p)$, or a complexity of $O[t * \log_2(p)]$

How to implement concurrent write?

Arbitrary or priority

need a lock or some similar synchronization mechanism

We will come back to this later

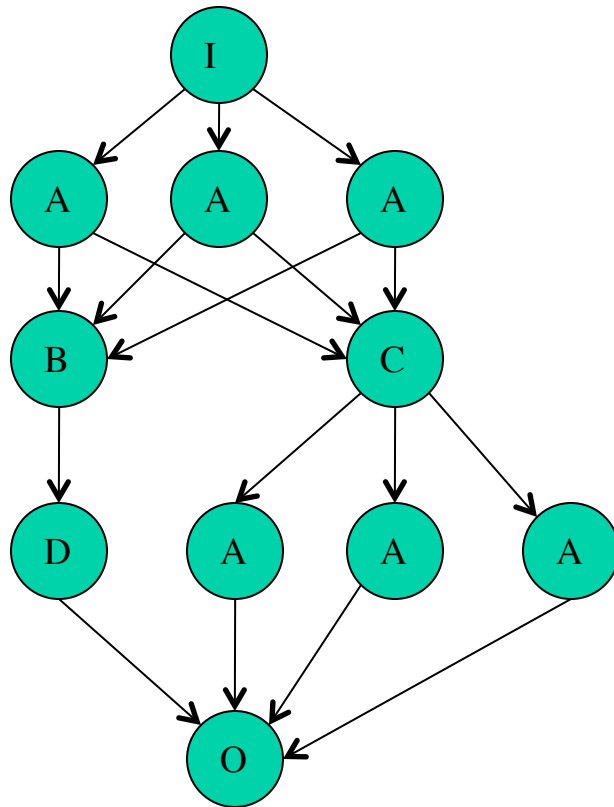
Do local caches with processors cause complications?

Cache coherency – pages 45-53

we will come back to this at a later time as we understand the need for
locks and other synchronization methods

Identifying Functional and Data Parallelism

Consider the figure– identify data and task parallelisms



Data parallelism

All A's after I and after C

Task parallelism

B and C can be executed in parallel

D and A's can be executed in parallel

Task parallelism can be identified using
“leveling” algorithms

Identifying Functional and Data Parallelism

Consider another example

Add 1000 4-digit numbers. Each number is written on an index card

You have 1000 students at your service to complete this problem

Students are seated in 25 rows of tables and each table seats 40 students (columns)

Each student can pass up to 4 cards to his/her neighbors – 2D mesh

a). What is the fastest way of distributing the cards

Give 20 cards to each student at the two ends of each row

They distribute 19 to their neighbor in the same row, and so on

Takes 20 steps

Any other ideas?

Identifying Functional and Data Parallelism

b). What is the fastest way of accumulating subtotals from students?

Once cards are distributed, you can send your card (and accumulated sum) to your neighbors

We can accumulate sums along rows and then across columns

Or we can accumulate sums from data received from all 4 neighbors

c). Draw a graph to show the communication paths and also estimated time needed

Mapping Algorithm to Network

If we assume that an algorithm can be defined as a graph representing communication requirements among the various sub-computations, each computation on a different processing node, then we can see how well this graph can be mapped onto a network
-- the network itself is represented as a graph.

In general we are considering mapping a graph $G(V,E)$ -- algorithm
onto another graph $G'(V',E')$ -- communication

It is possible that one link of E may be mapped to more than one link in E' (called **dilation factor**-- leads to delays due to more links that must be traversed);

or several links in E may be mapped to a single link of E' (called **congestion factor** and leads to delays due to congestion).

Likewise several nodes of V may be mapped to a single node in V' (called **expansion**, and this implies that one processor is executing several sub-computations).

Mapping Algorithm to Network

Our goal is to take any algorithm which assumes one network of interconnection and map the algorithm to another physical interconnections

Let us see how we can map Hypercubes to a linear array.

Linear array nodes will be numbered 0, 1, 2, ... n-1

Hypercube nodes will be numbered as binary numbers

But one property of linear array is that two consecutive numbers are connected (are neighbors).

If we use straight binary coding, this will not happen.

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Let us look at nodes 1 and 2 (001 and 010) --

According to our hypercube routing -- Hamming distance
there is a distance of 2 between these nodes -- they are not neighbors.

Likewise between 3 (011) and 4 (100) -- in the hypercube
they are not neighbors -- but they are in the linear array.

Mapping Algorithm to Network

So what do we do?

We can use **Gray codes** for this purpose -- actually called reflected Gray codes

0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

How about mapping Hypercubes to Meshes -- say 2-D mesh?

Each dimension in the mesh will be treated as a linear array and we use reflected Gray code mapping.

Mapping Hypercubes to Trees?

We will digress from the textbook to learn how to use MPI and OpenMP