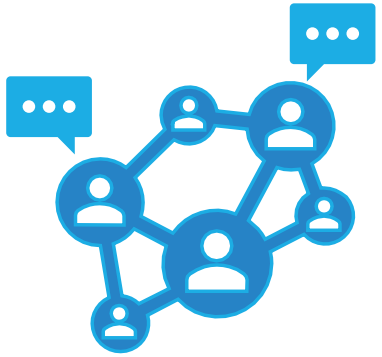
CS3210

Parallel Computing

Changes from Monday in Green



Tut 4

Mon (4pm)
Tues (2pm)

Admin
Updates

- Assignment 3 released - due 16 Nov Oct, 11am
  - Topic: Distributed Task Scheduling with MPI
  - Weightage: 14%
  - Please start early - there will be heavy contention for the machines near the deadline
- Lab machines scheduled for daily reboot at 9.30am
  - Should clear most long-running processes; drop me a message if too many accumulate through the day
- End-tutorial Quiz 4 later

Q1
Message-Passing Programming

- Consider the following incomplete MPI program

```
int rank, p, size = 8;
int left, right;
char send_buffer1[8], recv_buffer1[8];
char send_buffer2[8], recv_buffer2[8];
...
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
left = (rank - 1 + p) % p;
right = (rank + 1) % p;
...
MPI_Send(send_buffer1, size, MPI_CHAR, left, ...);
MPI_Recv(recv_buffer1, size, MPI_CHAR, right, ...);

MPI_Send(send_buffer2, size, MPI_CHAR, right, ...);
MPI_Recv(recv_buffer2, size, MPI_CHAR, left, ...);
```

Q1(a)
Message-Passing Programming

- The processors are arranged in a logical ring
  - Each processor should exchange its hostname with its left and right neighbour
  - Goal: Assign a unique name to each MPI process and fill in the missing sections of the program such that it prints its name along with those of its neighbours

- Answer
  - Retrieve hostname: **`gethostname(send_buff, size);`**
  - Complete body of **`MPI_Send`** and **`MPI_Recv`**

Q1(b)
Message-Passing Programming

- In the given program, **`MPI_Send`** and **`MPI_Recv`** calls are arranged in a way that may result in deadlock
  - Describe a scenario in which this can occur

- Answer
  - Deadlock can occur in either pairs of **`MPI_Send`** and **`MPI_Recv`** operations
  - If the MPI runtime uses no system buffers, the `MPI_Send` calls block until the receiving process calls `MPI_Recv`
  - As all processes wait for `MPI_Recv` in the ring, a deadlock occurs

Q1(c)
Message-Passing Programming

- Modify the program to prevent deadlock by re-arranging the order of `MPI_Send` and `MPI_Recv` operations

- Answer
  - Remove cyclic dependencies by inverting order of operations for some processes
  - Each process examines its own rank and performs one of the following depending on parity:
  - Even-ranked processes: `MPI_Send` first then `MPI_Recv`
  - Odd-ranked processes: `MPI_Recv` first then `MPI_Send`

Q1(d)
Message-Passing Programming

- Modify the program to prevent deadlock by using `MPI_Sendrecv`

- Answer
  - Replace each pair of **`MPI_Send`** and **`MPI_Recv`** operations with a **`MPI_Sendrecv`** operation (this is blocking)
  -
```
int MPI_Sendrecv(
  const void *sendbuf, int sendcount,
  MPI_Datatype sendtype, int dest, int sendtag,
  void *recvbuf, int recvcount,
  MPI_Datatype recvtype, int source, int recvtag,
  MPI_Comm comm, MPI_Status *status)
```

Q1(e)
Message-Passing Programming

- Modify the program to prevent deadlock by using `MPI_Isend` and `MPI_IRecv`

- Answer
  - `MPI_Isend` and `MPI_Irecv` are both non-blocking, so there is no need to re-order the operations
  - However, before reading the hostnames of the neighbours, the operation must be checked for completion with **`MPI_Test`**, **`MPI_Wait`** or **`MPI_Waitany`**

Q2(a)
Pipelining Pattern in MPI

- Suppose you need to compute prefix sum for multiple arrays for image convolution operations in MPI
  - Not allowed to use **`MPI_Scan`**

- Explain how you would design an MPI program to compute prefix sums in a pipeline programming model
  - Data distribution: each process of rank i receives i-th element of each array
  - Each process of rank $\boldsymbol{i}$ receives prefix sum from previous process of rank $(\boldsymbol{i-1})$ and adds the $\boldsymbol{i}$-th value, then sends the array of prefix sums to the process of rank $(\boldsymbol{i+1})$

Q2(b)
Pipelining Pattern in MPI

- How many operations are required for each pipeline stage? What are they? What assumptions are needed?
  - Assumption: array size equal to number of processes
  - Each process (stage): 1 receive, 1 add, 1 send for each array

- What is the maximum achievable speedup for arrays of size N when the pipeline is full?
  - Recall Tutorial 1/Lecture 5
  - Maximum speedup = number of pipeline stages = N

Q2(c)
Pipelining Pattern in MPI

- How would the responses to the previous questions change if we use specifically `MPI_Scan`?
  - We can use multiple `MPI_Scan` operations, one to compute prefix sum of each array
  - However, speedup achieved will much lower (as each `MPI_Scan` is a blocking collective operation)

- Food for thought: how can we mitigate this?

Q3
Interconnect Topologies

- Bob is trying to set up a 64-node NUMA network, but has not decided which direct interconnection network to use
  - Main consideration: cost, especially those of network links

Item	Cost
Each *uni-direction* link (base cost with base data rate of 1Mbps)	SGD100
Enhanced data rate (for every additional 1Mbps)	SGD50

- Nodes in network can process (send & receive) messages at 2 Mbps

Q3
Interconnect Topologies

- Summary table of characteristics of different topologies

network G with n nodes	degree $g(G)$	diameter $\delta(G)$	edge-connectivity $ec(G)$	bisection bandwidth $B(G)$
complete graph	$n-1$	1	$n-1$	$\left(\frac{n}{2}\right)^2$
linear array	2	$n-1$	1	1
ring	2	$\lfloor \frac{n}{2} \rfloor$	2	2
d-dimensional mesh $(n = r^d)$	$2d$	$d(\sqrt[d]{n}-1)$	d	$n^{\frac{d-1}{d}}$
d-dimensional torus $(n = r^d)$	$2d$	$d\left\lfloor \frac{\sqrt[d]{n}}{2} \right\rfloor$	$2d$	$2n^{\frac{d-1}{d}}$
k-dimensional hyper-cube $(n = 2^k)$	$\log n$	$\log n$	$\log n$	$\frac{n}{2}$
k-dimensional CCC-network $(n = k2^k$ for $k \geq 3)$	3	$2k-1+\lfloor k/2 \rfloor$	3	$\frac{n}{2k}$
complete binary tree $(n = 2^k - 1)$	3	$2\log\frac{n+1}{2}$	1	1
k-ary d-cube $(n = k^d)$	$2d$	$d\lfloor \frac{k}{2} \rfloor$	$2d$	$2k^{d-1}$

Q3
Interconnect Topologies

- What topology metric can we use to determine
  - The number of links needed

- Degree $g(v)$: number of direct neighbour nodes of a node v
  - Usefulness: small node degree reduces node hardware overhead (number of network links)
  - $\boldsymbol{N_{links} = n \times g(G)}$
  - Here, N_{links} is the number of uni-directional links

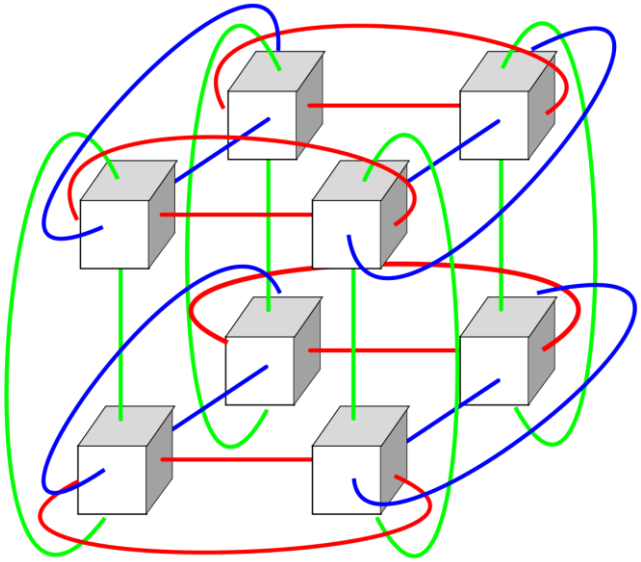
Q3
Interconnect Topologies

- What topology metric can we use to determine
  - The data rate needed

- Bisection width $B(G)$: number of edges that must be removed to divide network into two equal halves
  - Bisection bandwidth: total bandwidth available between two bisected partitions
  - Usefulness: measures capacity of network when transmitting messages simultaneously
  - Data rate $= \frac{n}{2} \times \frac{\text{Message rate}}{B(G)}$

Q3(a)
Interconnect Topologies

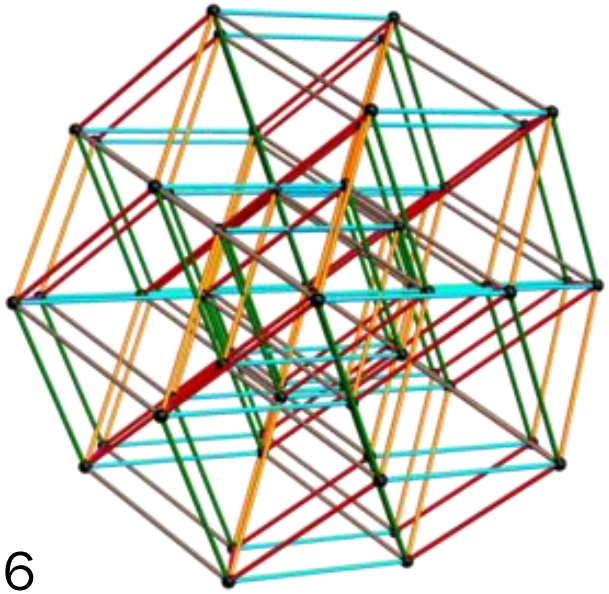
- Estimate the cost if the network is a 3D torus
  - Torus: cyclic in all dimensions



- Answer
  - 64 nodes, each with out-degree 6
  - $N_{links} = n \times g(G) = 64 \times 6 = 384$ uni-direction links
  - Data rate $= \frac{n}{2} \times \frac{\text{Message rate}}{B(G)} = \frac{64}{2} \times \frac{2\ Mbps/link}{32\ links} = 2\ Mbps$
  - Total cost $C = 384 \times (100 + 50) = \$57600$

Q3(b)
Interconnect Topologies

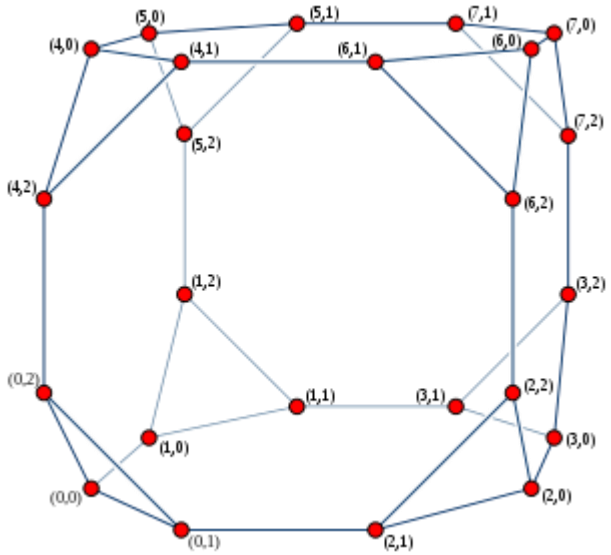
- Estimate the cost if the network is a hypercube
  - A k-dimensional hypercube has 2^k nodes
  - Each node has $k = \log_2 n$ links



- Answer
  - 64 nodes ⇒ $6D$ hypercube, out-deg 6
  - $N_{links} = n \times g(G) = 64 \times 6 = 384$ uni-direction links
  - Data rate $= \frac{n}{2} \times \frac{\text{Message rate}}{B(G)} = \frac{64}{2} \times \frac{2\ Mbps/link}{32\ links} = 2\ Mbps$
  - Total cost $C = 384 \times (100 + 50) = \$57600$

Q3(c)
Interconnect Topologies

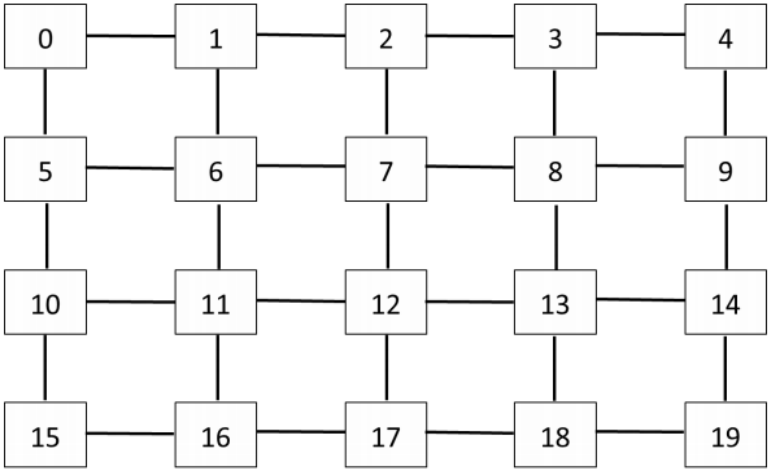
- Estimate the cost if the network is a Cycle-Connected Cube
  - A k-dimensional CCC has $k2^k$ nodes
  - Each node has exactly 3 links



- Answer
  - 64 nodes ⇒ $4D$ CCC
  - $N_{links} = n \times v(G) = 64 \times 3 = 192$ uni-direction links
  - Data rate $= \frac{n}{2} \times \frac{\text{Message rate}}{B(G)} = \frac{64}{2} \times \frac{2\ Mbps/link}{8\ links} = 8\ Mbps$
  - Total cost $C = 192 \times (100 + 7 \times 50) = \$86400$

Q4
YX Routing

- Consider this mesh network with YX message routing
  - YX routing: route vertically then horizontally
  - Each network link can transmit 1 byte of data/clock



- Two messages are sent on the mesh at the same time
  - Message 1 from node 0 to 14
  - Message 2 from node 11 to 13
  - Both: 2 packets of data, each packet 4 bytes in size

Q4
YX Routing

- Claim: "*it looks like we're going to need a more complicated routing scheme to avoid contention*"
  - Do you agree or disagree?

- Answer
  - No contention exists in this scenario
  - First packet from Message 1 takes $3 \times 4 = 12$ cycles to arrive at node 11 (3 hops)
  - But, after 8 cycles Message 2 has already been completely transmitted over this link $(12 \rightarrow 13)$, so the link is free
  - Note: packets are not sent byte by byte

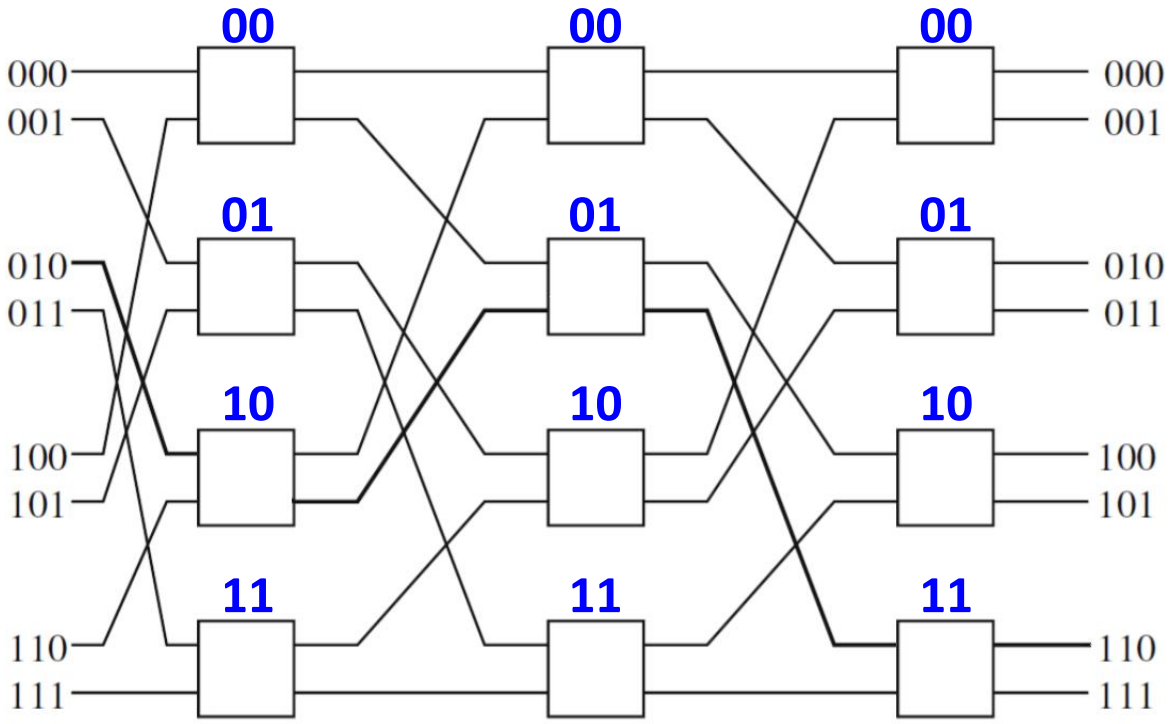
Q5(a)
Indirect Interconnects

- Assume an Omega network is used to connect 8 processing networks with 8 memory units
  - How many stages are required? How many switches are required at each stage?

- Answer
  - This is a 8×8 Omega network
  - Number of stages $= log_2(n) = log_2(8) = 3$
  - Number of switches per stage $= \frac{n}{2} = \frac{8}{2} = 4$

Q5(a)
Indirect Interconnects

- Sketch this 8×8 Omega network



8×8 Omega Network using 2x2 switches

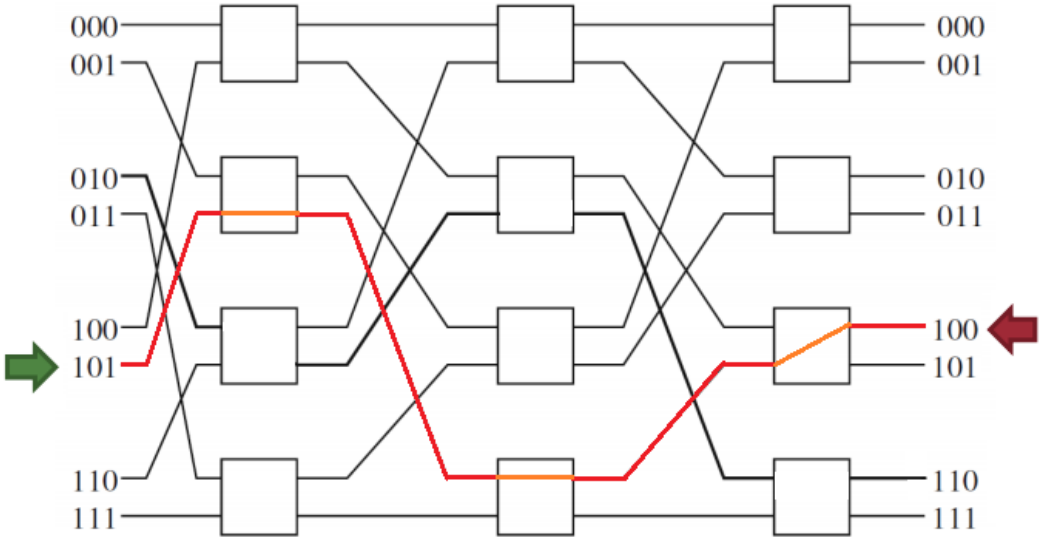
Q5(b)
Indirect Interconnects

- Assume the 8×8 Omega network previously is used to send a packet from source 101 to destination 100
  - Explain how this packet is routed at each stage using XOR-tag routing

- Answer
  - First XOR the source and destination: $\mathbf{101} \oplus \mathbf{100} = \mathbf{001}$
  - Then examine the i-th bit, which determines whether this packet takes the straight or crossover edge at i-th stage
  - Thus, $\mathbf{001} \Rightarrow$ straight, straight, crossover

Q5(b)
Indirect Interconnects

- Answer
  - First XOR the source and destination: $101 \oplus 100 = 001$
  - Thus, $001 \Rightarrow$ straight, straight, crossover



8×8 Omega Network using 2x2 switches

Admin
Tutorial Quiz

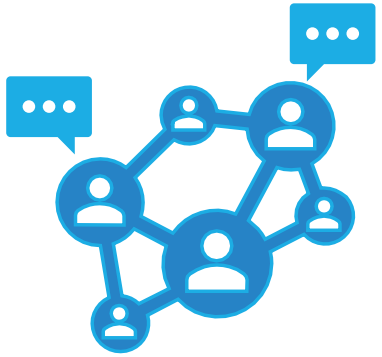
- Go to Luminus > Quiz
  - 1% of your grade
  - 10 minutes for 5 MCQ questions
  - Don’t be stressed - full credit for ≥ 50% correct (you should be able to do this after the tutorial, well, hopefully)

- 15 minutes now to attempt it

CS3210

Parallel Computing

Thank you! Any questions?



Tut 4

Mon (4pm)
Tues (2pm)