Week 4 – Collective communication, topologies

- Main topics this week
  - Collective communication, topologies
- Reading
  - Pacheco, Chapters 4 and 5

Lecture 6

Collective Communication and topologies
Network topology slides adopted from:
Networks: Topologies - HPC Advisory Council
www.hpcadvisorycouncil.com/events/2011/.../pdf/.../10_HPCAC.pdf

The project

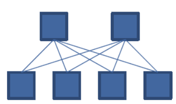
Milestone	Due Date
Project Matchmaking	Wednesday, Feb.15
Project Proposal	Wednesday, Feb.24
CSCI 5576 Paper Presentations	March 12-16
Annotated Bibliography	March 23
Project Checkpoint	Monday, Apr. 4
Final Presentations	April 30 – May 4
Final Paper	May 9

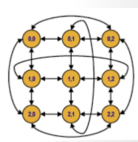
World leading large scale machines

- National Supercomputing Centre in Shenzhen
  - Fat-tree, 5.2K nodes, 120K cores, NVIDIA GPUs, China (Petaflop)
- Tokyo Institute of Technology
  - Fat-tree, 4K nodes, NVIDIA GPUs, Japan (Petaflop)
- Commissariat a l'Energie Atomique (CEA)
  - Fat-tree, 4K nodes, 140K cores, France (Petaflop)
- Los Alamos National Lab – Roadrunner
  - Fat-tree, 4K nodes, 130K cores, USA (Petaflop)
- NASA
  - Hypercube, 9.2K nodes, 82K cores – NASA, USA
- Jülich JuRoPa
  - Fat-tree, 3K nodes, 30K cores, Germany
- Sandia National Labs – Red Sky
  - 3D-Torus, 5.4K nodes, 43K cores – Sandia "Red Sky", USA

Network topologies

- Fat-tree (CLOS), Mesh, 3D-Torus topologies
- CLOS (fat-tree)
  - Can be fully non-blocking (1:1) or blocking (x:1)
  - Typically enables best performance
    - Non blocking bandwidth, lowest network latency
- Mesh or 3D Torus
  - Blocking network, cost-effective for systems at scale
  - Great performance solutions for applications with locality
  - Support for dedicate sub-networks
  - Simple expansion for future growth

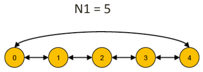




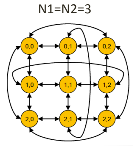
d-Dimensional Torus Topology

- Formal definition
  - T=(V,E) is said to be d-dimensional torus of size $N_1 x N_2 x \dots x N_d$ if:
    - $V=\{(v_1,v_2,\dots,v_d) : 0 \le v_i \le N_i-1\}$
    - $E=\{(u \rightarrow v)$: exists j s.t. 1) for each $i \neq j$, $v_i=u_i$ AND 2) $v_j=(u_j \pm 1) \bmod N_j\}$
- Examples

N1 = 5

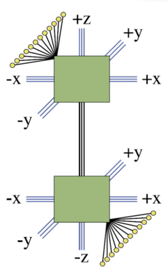


N1=N2=3

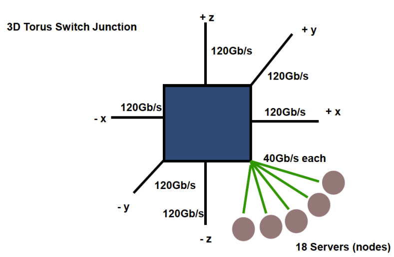


3D-Torus System with Infiniband – Key Items

- Multiple server nodes per cube junction
- Smaller 3D cube size the better
  - Lowest latency between remote nodes
- Minimizing throughput contention
- Ability to connect storage
- Support for separate networks
  - Dedicated network (links) for specific applications/usage
  - Example: links dedicated for collectives or specific jobs

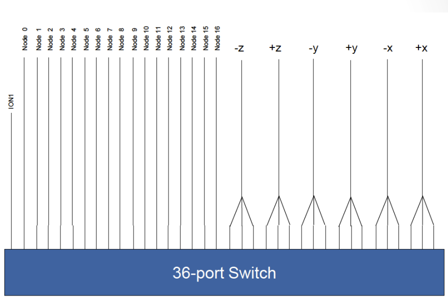


Example 3D Torus

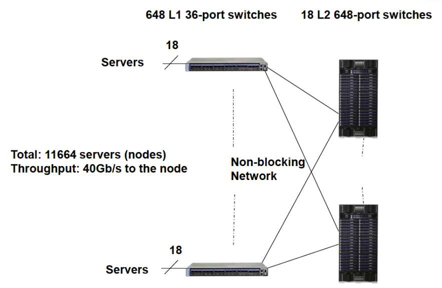


3D Torus size: 8x8x8 (512 36-port switches)
Total number of servers: 9216

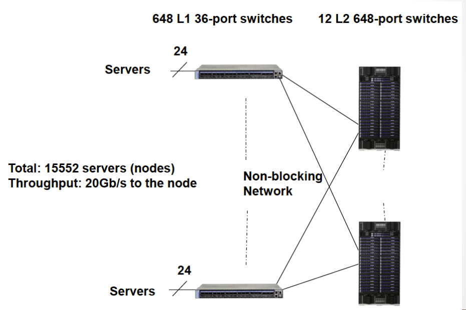
3D Torus Connections Example



Example: Non-blocking, fat tree, Infiniband



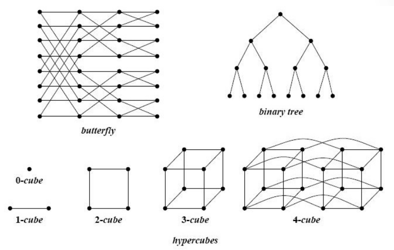
Example: 2:1 Oversubscription



Determining the network topology

Interconnection Network (API)	**Topology Query Tool(s)**
Myrinet (MX)	`fm_db2wirelist`
InfiniBand (OFED)	`ibdiagnet & ibnet-discover`
SeaStar (Cray XT)	`xtprocadmin & xtdb2proc`
BlueGene/P (DCMF)	`DCMF API`

Topologies from algorithm design



Collective communication

- Broadcast
  - one to all: data from root to all others
- Gather
  - all to one: all send data to root (result is vector)
- Scatter
  - one to all: reverse of gather
- Reduce
  - all to one: combine results on all at root via specified operation

MPI operations

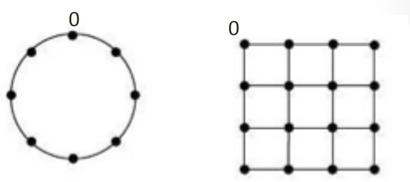
- MPI_BCAST
- MPI_GATHER
- MPI_SCATTER
- MPI_REDUCE

arguments similar to those of MPI_SEND and MPI_RECV

How to do gather without collective routine

- The simple way of where all send to 0 (root) at once
- More efficiently,
  take advantage of network topology
  - Can reduce number of steps
  - Does reduce contention for wires

Contention



Two simple and important topologies show
topologies are graphs: nodes and edges

Consider a ring of p processors

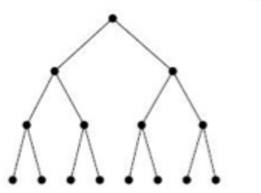
- Gather requires p/2 communication steps
- Cost per step = $T = t_s + k\ t_c$
  t_s = startup or latency
  k = bytes in message (constant size here)
  t_c = communication cost
  $1/t_c$ = bandwidth
- Total cost = (p/2)*T

What's a more efficient topology?

Do we immediately see how to do a gather on a mesh?

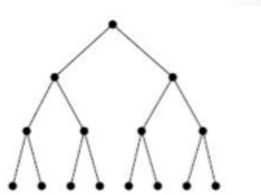
Tree

- Gather requires ____ steps
- Cost per step =
- Total cost = ____



Tree

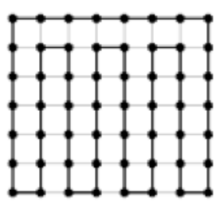
- Gather requires $\log_2 p$ steps
- Cost per step = $T = t_s + k\, t_c$
- Total cost = $\log_2 p * T$



But we don't have a tree!

- Graph embeddings allow us to find one graph in another
  (one topology in another)
  = virtual circuit
- Designing efficient parallel programs requires us to understand graph embeddings
  - What are efficient communication schemes?
  - What schemes avoid contention?
  - What schemes permit nearest neighbor communication?

One example



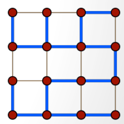
ring in 2-D mesh

Graph embedding

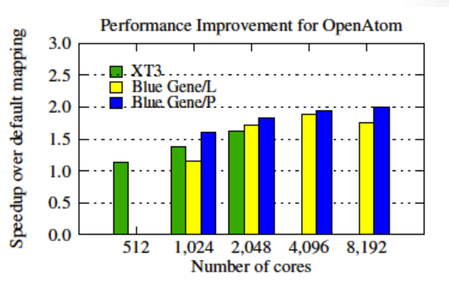
- Graph embedding gives ~~us~~ flexibility in how we design algorithm: use convenient communication graph V_s and embed it into network graph V_t
- Concerns
  - load : maximum number of nodes in V_s mapped to same node in V_t (CPU use)
  - congestion : maximum number of edges in E_s mapped to paths containing same edge in E_t (bandwidth, contention)
  - dilation : maximum distance between any two nodes $(\Omega(u), \Omega(v))$ € V_t such that (u, v) € E_s (nearest neighbors)

How to embed

- Finding optimal embedding NP-complete, in general, so heuristics used to determine good embedding
- Special graphs
  - Tree : connected graph containing no cycles
  - Spanning tree : subgraph that includes all nodes of given graph and is also a tree
  - Spanning tree algorithm prevents redundant communication

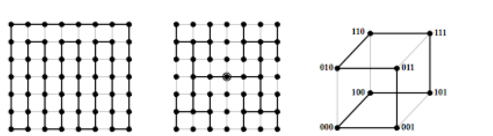


Example of speedup by mapping



Examples: graph embeddings

- For some important cases, good or optimal embeddings are known:

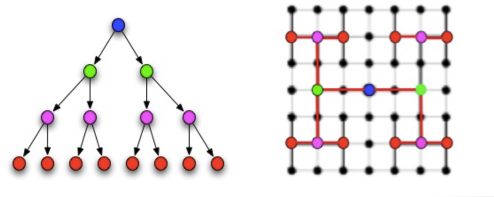


ring in 2-D mesh
dilation 1

binary tree in 2-D mesh
dilation $\lceil (k-1)/2 \rceil$

ring in hypercube
dilation 1

So focus on embedding tree into mesh (torus)



binary tree (not spanning tree) into 2-D mesh
dilation = ceil((k-1)/2)

Other operations similarly based on trees

- Reduce has the same communication scheme as gather but with operations on values
- Broadcast and scatter go from root to leaves
- Other ops like MPI_ALLGATHER: all to all, communication goes in one direction then other
- Next weeks' lab assignment: butterfly allreduce
  - Topology is butterfly
  - Communication goes in both directions