Multicomputers and Interconnection Networks

Lecture for CPSC 5155
Edward Bosworth, Ph.D.
Computer Science Department
Columbus State University
The Origin of Multicomputing

• The basic multicomputing organization dates from early on.
• The difference is that, before 1945, all computers were human; a “computer” was defined to be “a person who computes”. An office dedicated to computing employed dozens of human computers who would cooperate on solution of one large problem.
• They used mechanical desk calculators to solve numeric equations, and paper as a medium of communication between the computers. Kathleen McNulty, an Irish immigrant, was one of the more famous computers. As she later described it:

  “You do a multiplication and when the answer appeared, you had to write it down and reenter it. ... To hand compute one trajectory took 30 to 40 hours.”
Fitting the Problem to Multicomputing

• This example, from the time of U.S. participation in the Second World War illustrates the important features of multicomputing.

1. The problem was large, but could be broken into a large number of independent sub-problems, each of which was rather small and manageable.

2. Each sub-problem could be assigned to one computer, with little requirement for communication between independent computers. The computation intensity was high and the communication intensity was low.
Task Granularity

• In any time-sharing system, each process is allocated a time quantum, about 100 milliseconds. After that time, it is blocked and another process is scheduled.
• Let $R$ be the length of the run–time quantum.
• Let $C$ be the amount of time during that run–time quantum that the process spends in communication.
• The applicable ratio is $(R/C)$, which is defined only for $0 < C \leq R$.
• In course–grain parallelism, $R/C$ is fairly high so that computation is efficient.
• In fine–grain parallelism, $R/C$ is low and little work gets done due to the excessive overhead of communication and coordination among processors.
An Early Multicomputer

- Each computer is working on a mechanical adding machine.
- There is intensive computation, but not very much communication of data.
Sharing Data

• **Multiprocessors** are computing systems in which all programs share a single address space. This may be achieved by use of a single memory or a collection of memory modules that are closely connected and addressable as a single unit.

• **Multicomputers** are computing systems in which a collection of processors, each with its private memory, communicate via some dedicated network. Programs communicate by use of specific message and receive message primitives. Examples: Clusters and **MPP (Massively Parallel Processors)**.
Message Passing

- Each processor has private physical address space
- Hardware sends/receives messages between processors
Massively Parallel Processors

- We shall see that MPP systems finally evolved due to a number of factors, at least one of which only became operative in the late 1990’s.
  1. The availability of small and inexpensive commodity microprocessor units (AMD, Pentium, etc.) that could be efficiently packaged into a small unit.
  2. The discovery that many very important problems were quite amenable to parallel implementation.
  3. The discovery that many of these important problems had structures of such regularity that sequential code could be automatically translated for parallel execution with little loss in efficiency.
Interconnection Topologies

• Any computer system (multiprocessor or multicomputer) that uses multiple processors must provide a data channel by which these processors can communicate.

• One way of characterizing these interconnection networks is by the topology of each. The term refers to the physical arrangement of communicating nodes (processors, memories, etc.) and the links that connect these nodes.
A Single–Bus Interconnection

Consider now a number of communicating nodes connected by a single bus. We have two equivalent ways to represent such an architecture.

The top view emphasizes the function of the NIC (Network Interface Cards).
Two Ring Topologies

- Here are two equivalent unidirectional ring topologies. In each one, the packets can circulate in one direction only.
- This topology is simpler than a bidirectional ring, but it is less fault–tolerant.
Two Bidirectional Ring Topologies
Bidirectional Rings are Single–Fault Tolerant
Mathematical Graph Theory and Interconnection Networks

• A mathematical graph $G = (V, E)$ is a set of vertices (nodes) connected by a set of edges.
• Graph theory provides many powerful tools for analysis of interconnection networks based only on the topology (structure).
• The vertex set, $V$, represents the nodes that communicate in the network. The edge set, $E$, represents the communication links.
Formal Graph Theory

• **Definition:** A graph $G$ is a finite non-empty set of vertices, denoted $V(G)$, together with a (possibly empty) finite set $E \subseteq V(G) \times V(G)$.

• In the formal theory, we identify each vertex by an integer.

• If $|V(G)| = N$, we say that $V(G) = \{1, 2, 3, \ldots, N\}$ and $E(G) \subseteq \{(J, K) \mid 1 \leq J \leq N \text{ and } 1 \leq K \leq N\}$

• A simple graph is one without loops; no edges of the form $(K, K)$; from a vertex back to itself.
Directed and Undirected Graphs

• A **directed graph** is a simple graph in which the edge \((J, K)\) is not the same as the edge \((K, J)\); \((J, K) \neq (K, J)\). Think “one way streets”.

• An **undirected graph** is a simple graph in which the edge \((J, K)\) is defined to be the same as the edge \((K, J)\); \((J, K) = (K, J)\).

• Most networks are represented by undirected graphs.
More on Directed and Undirected Graphs

- **Directed graphs** correspond to edge sets that contain ordered pairs.
- **Undirected graphs** correspond to edge sets that contain unordered pairs.
The Edge Sets

- Each graph has $V(G) = \{1, 2, 3, 4\}$

Undirected: $E(G) = \{(1,4), (2,3), (2,4), (3,4)\}$

Directed: $E(G) = \{(1,4), (2,4), (3,2), (4,1), (4,3)\}$
Graphs: Walks, Paths, and Cycles

- Let $u$ and $v$ be vertices in a graph $G$, with $u$ and $v$ not necessarily distinct.
- A $u$-$v$ walk of $G$ is a finite, alternating sequence of vertices and edges starting with $u$ and ending with $v$.
- The number $s$, the number of edges in the sequence, is called the length of the walk.
- A $u$-$v$ path is a $u$-$v$ walk in which no vertex is repeated.
- A cycle is a $u$-$v$ walk in which all vertices are distinct with the sole exception that $u = v$. 
Examples: Paths and Cycles

• Here are some examples of paths and cycles.
Undirected Graphs

- Consider a graph $G$ with vertex set $V(G)$, and let $J \in V(G)$ be an arbitrary vertex.
- The **open neighborhood**, denoted $N(J)$, of vertex $J$ is that set of vertices adjacent to it.
- More formally, $N(J) = \{ K \mid (J, K) \in E(G) \}$
- The **degree** of a vertex is the number of vertices adjacent to it; $d(J) = |N(J)|$.
- The **maximum degree** of the graph, denoted $\Delta(G)$, is the largest vertex degree.
- The **minimum degree** of the graph, denoted $\delta(G)$, is the smallest vertex degree.
Paths and Cycles

- The graph $P_N$ is a path on $N$ vertices.

- The cycle $C_N$ is a cycle comprising $N$ distinct vertices.
Complete Graphs

• The complete graph $K_N$ is a graph on $N$ vertices with each vertex adjacent to all others.

• A network based on $K_N$ has the maximum connectivity for $N$ vertices, but also the maximum edge count: $N \cdot (N - 1)/2$
Connected Graphs and Components

- A graph $G$ is said to be **connected** if there exists a path between every pair of distinct vertices in the graph, otherwise it is **disconnected**.

- A **connected component**, or simply a **component**, of a graph $G$ is a maximal connected subgraph of $G$. 
Cut Nodes and Vertex Connectivity

• A **cut node** is a node the removal of which will increase the number of components in a graph. If a cut node is removed from a connected graph, the graph becomes disconnected and has two connected components that are isolated from each other.

• The **vertex connectivity** of a graph, denoted $\kappa(G)$, is defined to be the smallest number of nodes, the removal of which will disconnect the graph. If $\kappa(G) = 1$, there is a cut node.
A Cut Node

• Removal of node 5 will break the network into two independent sub-nets that cannot communicate with each other.

Most robust network designs avoid this single point of failure.
The Cut Node Removed

• Here is a simple redesign of the previous network. There are no cut nodes; it requires the removal of 2 nodes to cut the network.

• Removal of node 7 allows the rest of the network to function.
Distances in Graphs

• Let $U$ and $V$ be two distinct nodes in a graph. The distance between $U$ and $V$, denoted as $\text{dist}(U, V)$ is defined as follows.

• If there is a path between $U$ and $V$, the distance is the number of edges on the shortest path between the two vertices.

• If there is no path between $U$ and $V$, the distance is not defined.
The Graph Diameter

• The diameter of a graph is the largest distance between any two vertices in the graph.
• For a communication network, the diameter is the minimum number of links required for the two most distant nodes to communicate.
• While each network can be characterized by its diameter, it is not clear that this measure is always of great importance.
Network Topologies: Fully Interconnected and Rings

• Depicted as a graph, the **fully interconnected topology** with N nodes is the complete graph on N vertices. It is the most robust, requiring the removal of \((N - 1)\) links to isolate a node. It is also the most costly, requiring \(N \cdot \frac{(N - 1)}{2}\) communication links.

• Depicted as a graph, the **ring topology** with N nodes is a cycle on N vertices. As mentioned early in this lecture, a bidirectional ring can function with the loss of either one node or one connection link.
Hypercubes

- An **N–dimensional hypercube** corresponds to an **N–regular graph** with $2^N$ vertices.
  As indicated in the name, each node communicates directly with N adjacent nodes. Here are depictions of the first four hypercubes.
The Crossbar Switch

- Here is a typical depiction of a crossbar switch, an 8–by–8 crossbar with 64 switches.
The Omega Network: A Staged Network

- This staged network is built from simple switches with very few operations. Each switch in the network has 2 inputs, 2 outputs, and 4 states.
The Omega Network

• This figure shows an 8–by–8 Omega switching network. It has 12 switches.
Connecting Computers

a. Crossbar

b. Omega network
Modified Grids and Tori

• We now introduce a connection topology based on either a grid or a torus.
• Here are the 1D and 2D variants of the grid and torus. The torus “wraps around”.

[Diagrams of linear array, circular array, mesh, and torus]
The Continuum Model

• Many significant problems can be approached using the *continuum model* of algorithms.

• In this model, the problem to be solved is set in 3D space, which is divided into adjacent cubic volumes of equal size.

• The algorithm is executed in two repeated steps.
  1. The algorithm is applied to each volume separately.
  2. The flow between adjacent volumes is then used to update the values in each volume.
Some Interesting Problems

2. Prediction of hurricane paths. This is weather modeling on a large scale.
3. Air flow around wings and other aerodynamic structures.
A Match to 2D Continuum
A Grid and a Torus

In a grid, there are processors on the boundaries that have fewer than expected neighbors. In a torus, each processor has the same number of neighbors. The eastern neighbor of the easternmost processor is at the extreme west.
A 3D Design from Cray

- Here, each processor is adjacent to six other processors.
- The directions are often called North and South, East and West, Up and Down.
- The torus “wraps around”.

![Diagram of 3D Cray design]

- Compute Node
- Service Node
- Dual PCI-Express Links
- 4-32 GB Memory
- 25.6 GB/s Direct Connect Memory
- 6.4 GB/s Direct Connect Memory
- Cray SeaStar 2+ 3-D Interconnect
A 3D Torus (3 x 4 x 4) with 3 “Wrap Arounds” Shown
Toroidal vs. Other Interconnection Schemes

• In a standard UMA configuration, each CPU can communicate directly with any other CPU through the main memory bus.
• This is a more general connection scheme.
• In the toroidal approach, each CPU can communicate with only a small number (say 4 to 6) of other CPUs. This is faster.