Parallel Computing In A Nutshell

Alexandre David
B2-206

Overview

Hardware

Models
- PRAM
- hyperthreading
- cache coherence
- interconnection networks
- bandwidth/latency
- superscalar processors
- pipelines

Practice
- Caches.

Software/Algorithms

Models
- communication costs
- mapping techniques
- task dependency
- load balancing
- communication algorithms
- optimality, scalability

Practice
- message passing (MPI)
- pthreads
- applications (matrix, graphs, sorting, search).

Cache

Keywords: Cache, cache coherence, false sharing.

- What it is.
- Cache coherence protocol.
  - Why we need one – to keep multiple copies of the same data consistent.
- False sharing – performance issue.

Examples

- Matrix multiplication (matrix*matrix, matrix*vector).
- Prefix computation.
- LU decomposition.
- Gaussian elimination.

Keyword: PRAM

PRAM Model

- Global, synchronous, parallel RAM.
- Classes: EREW, CREW, ERCW, CRCW.
  - Most powerful: CRCW.
  - Least powerful: EREW.
  - Simulation of CRCW on EREW & keep optimality.
- Issue: How to resolve contention?

Design Principle

Construct optimal algorithms to run **as fast as possible**.

=

Construct optimal algorithms using **as many processors as possible**!

Keywords: Brent scheduling principle, degree of concurrency, modeling.

Brent's Scheduling Principle

Theorem

If a parallel computation consists of
k **phases**
taking time $t_1, t_2, \ldots, t_k$
using $a_1, a_2, \ldots, a_k$ processors
in phases $1, 2, \ldots, k$
then the computation can be done in time **$O(a/p+t)$** using **p processors** where
$t = \text{sum}(t_i)$, $a = \text{sum}(a_i t_i)$.

Optimality

Keywords: Optimality, sequential/parallel execution time, complexity.

- Complexity: What to evaluate?
- Compare to optimal sequential algorithm (running time W).
- Parallel execution time (T_P) vs. total work (p^*T_P).
- Optimal iff $p^*T_P = \Theta(W)$.

Fundamental concept.

Interconnection Networks

- Bus, cross-bar, multistage (omega networks.
- Issues: Performance vs. cost.

Keywords: Network topologies, meshes, hypercubes.

Topologies

- Linear array, rings.
- 2-D, 3-D… meshes.
- Hypercubes.
- Tree based.
- Issue: How to evaluate them?
  - Cost vs. performance.
  - Define metrics.

Linear Arrays and Meshes

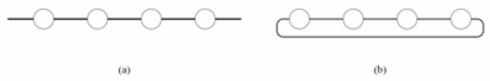


Figure 2.15 Linear arrays: (a) with no wraparound links; (b) with wraparound link.

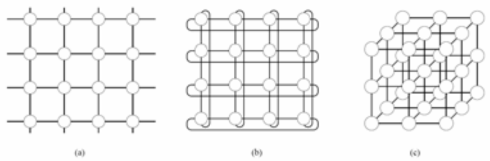


Figure 2.16 Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

Hypercubes

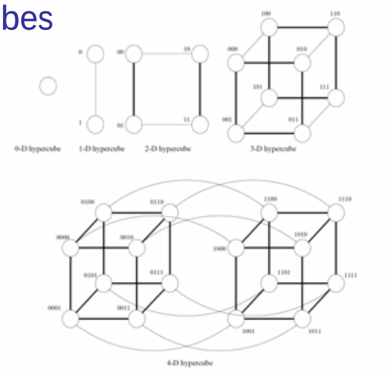


Figure 2.17 Construction of hypercubes from hypercubes of lower dimension.

Hypercubes

- Very good but very expensive.
- Naming scheme property – neighbors IDs differ by only one bit.
- E-cube routing protocol.

Metrics

- Diameter: Maximum distance $p_a \leftrightarrow p_b$.
- Connectivity: Measure of multiplicity of paths.
- Bisection width: Minimum number of links to cut in order to partition the network in 2 equal halves.
- Bisection bandwidth: Minimum volume of communication allowed between 2 halves.
- Cost: Number of communication links, i.e., wires.

Keywords: Latency, bandwidth.

Communication Costs

- Model latency & bandwidth.
- Total time for communication =
  - Startup time (t_s), *once per message*
  - + Per-hop time (t_h), *between directly connected nodes*
  - + Per-word transfer time (t_w), *1/bandwidth*.
  - Generic: $T=t_s+mt_w$.

Keyword: congestion.

Topology Embedding

- Embed one topology in another (hypercube on mesh…).
- Embedding metrics: Dilation, Expansion, **Congestion**.
- Possible to map denser into sparser:
  - Map (expensive) logical topology into (cheaper) physical hardware!
  - Mesh with links faster by *sqrt(p)/2* than hypercube links has same performance!

Keywords: Concurrency, task dependency graph, degree of concurrency.

Decomposition

- Decomposition into concurrent tasks.
- Task dependency graph.
- Granularity: number & size of tasks.
- Maximal degree of concurrency.
- Critical path.
- Task interaction graph (dependency = sub-graph).

Keyword: Load balancing.

Process Mapping

- Mapping = Assignment of task to processes.
- Decomposition techniques:
  - Recursive (divide & conquer).
  - Data decomposition (1-D, 2-D block data partitioning).
  - Exploratory decomposition (search).
  - Block-cyclic decompositions.
  - Graph partitioning.

Dynamic Mapping

- Centralized scheme (Master-slave).
- Distributed scheme (exchange).
- Goals: Minimize contention & hot spots, maximize data locality.

Communication Primitives

- Algorithms & cost analysis for:
  - Broadcast one-to-all, all-to-all.
  - Reduce all-to-one, all-to-all.
  - Scatter & gather.
  - Circular shifts.
- For different topologies (arrays, meshes, hypercubes).

Keywords: Broadcast communication primitives.

Analytical Modeling – Performance Metrics

- What is scalability? How to measure it?
- Measure overhead: $T_O = pT_P - T_S$.
- Speedup: $S=T_S/T_P$ – recall: w.r.t. T_S **best** sequential algorithm running time.
- Efficiency (normalized speedup) $E=S/p$.
- Cost optimal if E is a constant.

Keywords: Speedup, efficiency.

Cost Optimality

- 2 techniques to be cost optimal (not always work, cheating in a sense):
  - Re-schedule the tasks on fewer processes (Brent's scheduling principle).
  - Decrease granularity: Assign n/p elements per process.
- 2 techniques, one effect: Decrease p, increase work/process.

Keyword: Scalability.

Scalability

- Fix problem size, increase processors: Decrease efficiency.
- Fix processors, increase problem size: Increase efficiency.
- Related to communication overhead.
- Scalability measures the ability to increase speedup in function of *p*.
  - Fix E: How to increase *W* in function of *p*? $W=KT_0(W,p)$. Isoefficiency function.

MPI

- Point to point communication.
- Group communication – communicator.
- Topology support.

6 Golden Functions

MPI_Init	Initializes MPI.
MPI_Finalize	Terminates MPI.
MPI_Comm_size	Determines the number of processes.
MPI_Comm_rank	Determines the label of the calling process.
MPI_Send	Sends a message.
MPI_Recv	Receives a message.

Focus: Communication.

MPI

- Key principles:
  - partitioned address space,
  - explicit parallelization.
- Blocking vs. non blocking communication.
- Deadlocks (circular send).
- Collective communication primitives.
- Barriers.

Keyword: Deadlock.

\+ More scalable, costs explicit.
− Load balancing, mapping.

Focus: Synchronization.

Multithreading – Pthreads

- Latency hiding but need more bandwidth.
- Implicit scheduling & load balancing.
- Thread creation, termination, cancellation, mutex-locks, condition variables.
- Critical sections, race conditions.
- How to implement monitors.

Keywords: Race, critical sections.

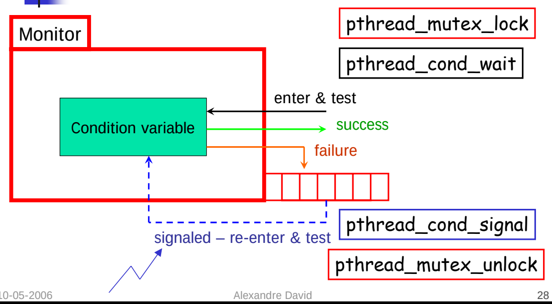
\+ Implicit, easier.
− Less scalable.

Keyword: Monitors.

Pthread Constructs

- Monitors with condition variables.
- Read-write locks.
- Barriers.

Conditional Wait in Pthread



Applications

- Dense matrix algorithms.
  - 1-D, 2-D, 3-D block mapping for matrix multiplication.
  - Gaussian elimination.
- Sorting.
  - Bitonic sort, odd-even sort, quicksort.

Keywords: Acceleration & deceleration anomalies.

Applications

- Graphs.
  - Adjacency matrix vs. adjacency lists.
  - Minimum spanning tree.
  - Single-source & all-pairs shortest path.
- Discrete optimization problems formulated as search.
  - Super-linear anomalies.
  - Work splitting, load balancing (Round-robin, random polling).
  - Termination detection.