Introduction to Parallel Algorithms and Architectures

RAM (Random Access Machine):

1. Memory
   a) Memory with M locations, where M is (large) finite number.
   b) Each memory location is capable of storing a piece of data.
   c) Each memory location has a unique location.
2. Processor
   a) A single processor operates under control of a sequential algorithm.
   b) The processor can load/store data from/to memory and can perform basic arithmetic and logical operations.
3. Memory Access Unit
   a) Creates a path from Processor to Memory
   b) Establishes a direct connection between memory and processor

Each step of an algorithm consists of:
1. Read phase - processor reads data from memory into register
2. Compute phase - processor performs basic operations in memory
3. Write phase - processor writes contents of register into memory

Time: discuss the time that it takes for each of these 3 phases - special attention to the time to access arbitrary location in memory. (Note: each register must be of size log M to accommodate distinct memory locations.)
1. So, compute takes O(log log M) time.
2. Memory access for individual access requires O(log M) time.
3. To process k memory accesses requires O(k + log M) time due to pipelining.
4. Since these terms don't typically effect the running time analysis and comparison, we say a step takes O(1) time, which is termed *uniform analysis*.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

PRAM (Parallel Random Access Machine):

1. Processors
   a) There are n identical processors (PEs), $P_1, P_2, \ldots, P_n$, each of which is identical to the RAM processor. Assume that n is (large) finite.
2. Memory
   a) Common/Global memory with M locations, $M \geq N$
3. Memory Access Unit
   a) similar to MAU of RAM, but allows any PE to get to any memory location

Two processors that want to communicate can use the shared memory as a bulletin board. Show example.

Each step of an algorithm consists of:
1. Read phase - up to n PEs may simultaneously perform one read from Memory to its local memory (i.e., a register)
2. Compute phase - every processor is entitled to perform a (small) fixed number of logical or arithmetic operations on the contents of its local memory (registers)
3. Write phase - up to n PEs may simultaneously write a value that is its local memory (i.e., a register) to the global/common memory.

Note: “up to” means that there may be reasons why some PEs don’t want to perform the operation (they might be masked out or the read/write may be conditional).

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Memory Access:

1. Exclusive Read (ER)
2. Concurrent Read (CR)
3. Exclusive Write (EW)
4. Concurrent Write(CW)
   a) Priority CW - only PE with highest priority succeeds
   b) Common CW - all PEs writing to the same location must write the same value
   c) Arbitrary CW - one PE, chosen arbitrarily, succeeds
   d) Combining CW
      i) Arithmetic functions - SUM, PRODUCT
      ii) Logical functions - AND, XOR
      iii) Selection/Semigroup - MAX, MIN

Common Models of PRAM:

1. CREW
2. EREW
3. CRCW
4. ERCW

Time:

1. Compute - each processor is same as a RAM and the instruction set is identical, so the time is the same: O(log log M)
2. Read/Write - if Memory Access Unit is implemented as combinational circuit, then the access time is O(log M), though pipelining can again improve things.

For similar reasons in terms of comparison of running times, choose each step of a PRAM to take unit time, i.e., O(1) time.

Note: 1. The PRAM is not a physically realizable machine.
2. It is a powerful model for studying the logical structure of parallel computation without worrying about communication.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Examples:

1. Minimum - bottom-up tree-like computation
2. Boadcast - top-down tree-like computation (ER) or CR
3. Search
4. Parallel Prefix: Given $x_1, x_2, \ldots, x_n$ and a binary associative operator $\otimes$, compute $x_1, x_1 \otimes x_2, x_1 \otimes x_2 \otimes x_3, \ldots$ where the k^{th} item $x_1 \otimes x_2 \otimes \ldots \otimes x_k$ is called the k^{th} prefix.

    For **Arrays** *only (not linked lists)* do:

    i) Running Sum

    ii) Running Minimum

Distributed Memory vs. Shared Memory: General Discussion

- Shared Memory
    - in general, shared memory machines don't scale well
    - unit-time access cannot be preserved
- Distributed Memory
    - scales better
    - typically involves message passing

Distributed Address Space vs. Shared Address Space: General Discussion

- permits shared memory programming and concepts
- can involve physically shared memory or physically distributed memory

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

Interconnection Networks:

1. Measures
   a) Degree of the network - i.e., maximum degree of any PE in the network.
   b) Communication Diameter - maximum of the minimum distance between any pair of PEs.
   c) Bisection Width - minimum number of wires that have to be removed (severed) in order to disconnect the network into 2 “equal” size subnetworks.
   d) I/O bandwidth - usually not critical as typically assumed that data already resides in machine.
   e) Time to perform basic operations (min, sum)

2. Processor Organizations

   a) PRAM (discuss measures)

   b) Mesh: - communication diameter
      - bisection width
      - minimum
      - sorting and/or lower bound on sorting

      i) Linear Array
         a) *class demonstration* of sorting
         b) introduction of a *rotation* operation
         c) show parallel prefix
      ii) Ring
      iii) 2-D Array
         a) do min two ways
         b) do sorting lower bound
         c) do parallel prefix

   c) Tree
      i) Communication diameter O(log n)
      ii) Bisection Width O(1)
      iii) show min
      iv) lower bound on sorting is $\Omega(n)$

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

d) Pyramid
   i) tapering array of meshes
   ii) combines advantages of mesh and tree
   iii) each PE connected to
      a) 4 mesh neighbors
      b) 4 children
      c) 1 parent
   iv) *n* base PEs $\Rightarrow$ (4/3)*n* - 1/3 PEs
   v) apex is the bottleneck
   vi) Communication Diameter: $\Theta(\log n)$
   vii) Bisection Width: $\Theta(n^{1/2})$
   viii) Min: $\Theta(\log n)$
   ix) Sorting: $\Omega(n^{1/2})$

e) Mesh-of-Trees
   i) mesh with a tree of PEs over every row *and* every column
   ii) MOT of base size *n* has
      a) *n* PEs in the base mesh
      b) $n^{1/2}$ column trees, each with $n^{1/2}-1$ non-base PEs
      c) $n^{1/2}$ row trees, each with $n^{1/2}-1$ non-base Pes
      d) $3n-2n^{1/2}$ total PEs
      e) All PEs are identical except for neighboring connections
         (1) Base PE: 4 MCC neighbors; parent in row tree; parent in column tree
         (2) Interior tree PE: parent in tree; 2 children in tree
         (3) Tree root PE: 2 children
   iii) Communication diameter: $\Theta(\log n)$
   iv) Bisection Width: $\Theta(n^{1/2})$
   v) Minimum: $\Theta(\log n)$
   vi) Sorting: $\Omega(n^{1/2})$
   vii) Bottlenecks for moving data not as bad as pyramid
   viii) Examples:
      a) Cross-product of $n^{1/2}$ pieces of data on MOT of base size *n*
      b) Sorting $n^{1/2}$ pieces of data on MOT of base size *n*.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

f) Hypercube: An r-dimensional hypercube has $N = 2^r$ nodes and $r2^{r-1}$ edges. Each node corresponds to an r-bit string, where 2 nodes share an edge iff their r-bit strings differ in exactly 1 position.
   i) Each node is connected to $\log_2 N$ other nodes.
   ii) This is **not** a fixed degree network.
   iii) Show how to build an r-cube recursively from two (r-1)-cubes.
   iv) An edge is a k-dimensional edge iff it connects nodes differing in the k^{th} bit position. So, the notion of edges of dimension k is well defined.
   v) Communication diameter: $\log_2 N$
      a) Note: includes multiple paths between nodes
   vi) Bisection Width: $N/2$
   vii) **Therefore:** The hypercube has low communication diameter and high bisection width. This is very desirable!
   viii) The hypercube is both node and edge symmetric in that by relabeling nodes, we can map any node to any other node and preserve communication links.
   ix) Discuss lower bounds on sorting based on
      a) Bisection Width
      b) Communication Diameter

SIMD vs. MIMD: General Discussion (Flynn's Taxonomy, 1966)

Granularity: Fine-Grained vs. Coarse-Grained

General Performance Measures:

1. **Throughput**: The number of results produced per unit time (typically, wall-clock)
2. **Running time**: $T_{par}(n)$ represents the length of time from the beginning of the algorithm until the last processor terminates.
3. **Cost**: $C(n,p) = p(n) \times T_{par}(n,p)$ is an upper bound on the total number of elementary steps executed by this algorithm with $p(n)$ processors on input of size n.
4. **Work** is the total number of operations performed (not counting NOPs)
5. **Speedup**: $S(n,p) = T_{seq}(n) / T_{par}(n,p)$, is the ratio between the time taken for the *most efficient* sequential algorithm to perform a task compared to the

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.

time needed for the *most efficient* parallel algorithm to perform the same task on an input of size *n* with *p* processors.

a) Linear Speedup: $S_p = p$

b) Superlinear Speedup: $S_p > p$. Discuss:

   i) not possible since a single PE can always emulate the parallel machine

   ii) but, choose algorithm before problem instance

   iii) emulation can have problems due to cache management

   iv) parallel algorithm can get lucky

6. **Efficiency**: $E(n,p) = T_{seq}(n) / C(n,p) = S(n,p) / p(n)$, measures how well utilized the processors are. Measures the "cost-effectiveness" of the computation. Typically, the best efficiency is at most 1.

7. Discuss relationship between measures and discuss goals of algorithm development

8. Amdahl's Law: $S_p \leq 1/[f + (1-f)/p]$, where f is the fraction of operations that must be performed sequentially and *p* is the number of processors.

   a) That is, a small number of sequential operations can significantly limit the speedup on a parallel computer.

   b) E.g., if 10% of the operations must be performed sequentially, then $S_p \leq 10$ **regardless of how many processors are used**.

   c) Discuss the falacy of the argument in terms of increased problem size.

9. **Scalable:**

   a) An algorithm is scalable if the level of parallelism increases at least linearly with the problem size.

   b) An architecture is scalable if it continues to yield the same performance per processor as the number of PEs increases.

   c) Scalability is important in that it allows users to solve larger problems in the same amount of time by purchasing a larger machine.

Copyright © 1996-1997 by Russ Miller. All Rights Reserved.