Advanced Algorithms **Fall 2015**

Lecture 5: Parallel Algorithms, Nov. 10,17, 2015

Faculty: K.R. Chowdhary *: Professor of CS*

5.1 Introduction

Most of the present algorithms are sequential.i.e., they specify sequence of steps in which each of the step consist of a single operation. However, the architectures in the market are going more and more towards the Parallel, from the start of multi-core systems. In contrast to a sequential algorithm, a Parallel algorithm will perform multiple operations in a single step. Consider the problem of computing sum of a sequence, A of n numbers. The standard sequential algorithm computes sum by making a single pass through the sequence, keeping a running sum of numbers seen so far. It is not difficult to device an algorithm to perform many operations in parallel. For example, each number of A having an index i paired with it, it is possible to pair $A[0]$ with $A[1]$, $A[2]$ with $A[3]$, and so on; the result is a new sequence $\lceil \frac{n}{2} \rceil$ numbers, whose sum is identical to earlier and will take half the steps. If this pairing is repeated, it would take only $\lceil \log_2 n \rceil$ steps for complete sum. Note that in $[\frac{n}{2}]$ steps there will be $[\frac{n}{2}]$ sums, next similarly, $[\frac{n}{4}]$, and so on.

The parallelism can result to improved performance. For example, in parallel architecture, operations in a single parallel algorithm can be executed simultaneously on different processors. The parallel algorithm can be executed even in a single processor system, using multiple functional units, for example, pipeline and array processors. A parallel algorithm will run at least as fast, as there are parallel processing units are existing in a computer. A good parallel algorithm is expected to run in both the parallel and sequential architecture.

5.2 RAM Model

A sequential algorithm is formulated using an abstract model of computer called RAM (random access model), where a single processor is connected with a random access memory system. All the arithmetic and logical operations require one time step.

The random access machine (RAM) models a one-accumulator computer in which instructions are not permitted to modify themselves. The input tape is sequence of squares, and holds an integer each. Write instruction prints an integer in each square y_i. The memory consist of locations $r_0, r_1, \ldots$, each of which can hold an integer of arbitrary size. The instruction set of RAM machine is: add, sub, mul, div, load, store, read, write, j, jnz, jz, hlt. In general, a RAM program defines a mapping from input tape to output tape.

5.3 Multiprocessor models to model Parallel Computation

Modeling a parallel Computation is more difficult because the parallel computers' organization have too much variations than sequential, and there is no consensus on a right model, neither there is sufficient understanding

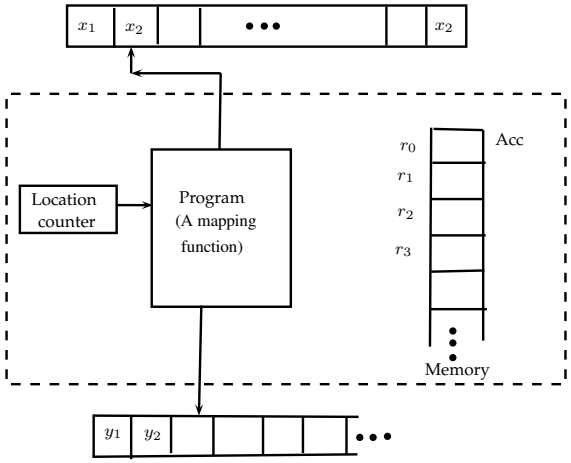

Figure 5.1: RAM model.

about the relationship between these models. The parallel models are broken into 1) multiprocessor models and 2) work-depth models.

A multiprocessor model is a generalization of RAM model, and has more than one processor. These models are classified into three basic types: 1) local memory machines, 2) modular memory machines, 3) parallel random access machines (PRAM) (see fig. 5.2).

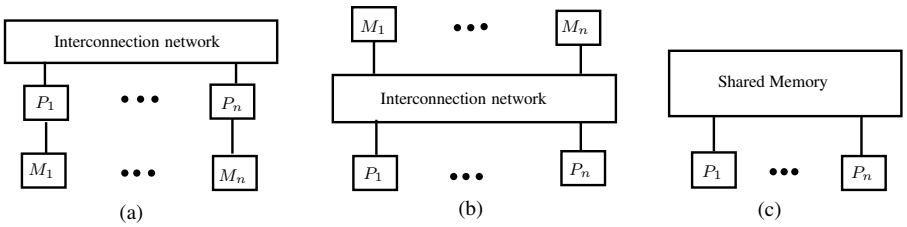

Figure 5.2: Multiprocessor machine models: a) Local memory, b) Modular memory, c) PRAM.

In local memory system, each processor accesses its own memory, but can access others memory by sending requests through network. Like in RAM, local operations, including memory access can unit time. But, for others memory access depends on the capability of the network, and pattern of the memory access by other processors.

In modular machine, any processor can access any memory, by sending request through the network. However, the arrangement is such that time of memory access is uniform for all the memory modules. As in PRAM single step can access number of memory locations, simultaneously.

The purpose of (b) model is that an algorithm designed for this can be modified to be run on any other model.

5.4 PRAM model and its variations

The purpose of the theoretical models for parallel computation is to give frameworks by which we can describe and analyze algorithms. These ideal models are used to obtain performance bounds and complexity estimates. The parallel random access machine (PRAM) model has been used extensively. The PRAM model was introduced by Fortune and Wyllie in 1978 for modeling idealized parallel computers in which communication cost and synchronization overhead are negligible.

A PRAM consists of a control unit, a global memory shared by n processors, each of which has a unique index as follows: P_1, P_2, ..., P_n. In addition to the global memory via which the processors can communicate, each processor has its own private memory. The n processors operate on a synchronized *read*, *compute*, and *write cycle*. During a computational step, an active processor may *read* a data value from a memory location, *perform* a single operation, and finally *write* back the result into a memory location. Active processors must execute the same instruction, generally, on different data. Hence, this model is sometimes called the shared memory, single instruction, multiple data (SM SIMD) machine.

Algorithms are assumed to run without interference as long as only one memory access is permitted at a time. We say that PRAM guarantees atomic access to data located in shared memory. An operation is considered to be atomic if it is completed in its entirety or it is not performed at all (all or nothing).

There are different modes for read and write operations in a PRAM. These different modes are summarized as follows:

1. Exclusive read (ER): Only one processor can read from any memory location at a time.
2. Exclusive write (EW): Only one processor can write to any memory location at a time.
3. Concurrent read (CR): Multiple processors can read from the same memory location simultaneously

5.5 Network Topology

A network is a collection of switches, connected by communication channels. A processor and memory have more than one ports connected to these switches. The connection may be in bus, mesh, hypercube, two stage multi-stage network or fat-tree topology.

The simplest network topology is a bus. This network can be used in both local-memory machine models and modular memory machine models. In either case, all processors and memory modules are typically connected to a single bus. In each step, at most one piece of data can be written onto the bus. This data might be a request from a processor to read or write a memory value, or it might be the response from the processor or memory module that holds the value. In practice, the advantage of using a bus is that it is simple to build and, because all processors and memory modules can observe the traffic on the bus, it is relatively easy to develop protocols that allow processors to cache memory values locally. The disadvantage of using a bus is that the processors have to take turns accessing the bus. Hence, as more processors are added to a bus, the average time to perform a memory access grows proportionately.

In two-dimensional rectangular mesh topology, each switch has distinct label (x, y), where $0 \leq x \leq X - 1$ and $0 \leq y \leq Y - 1$, where X, Y are lengths of the sides of the rectangle. The number of switches in a mesh is thus $X.Y$. Every switch, except those on boundary, communicate to four neighbors. This network is good for local memory machines.

A hypercube is a network with 2^n switches in which each switch has a distinct n-bit label. Two switches are connected by a communication channel in a hypercube if and only if the labels of the switches differ in

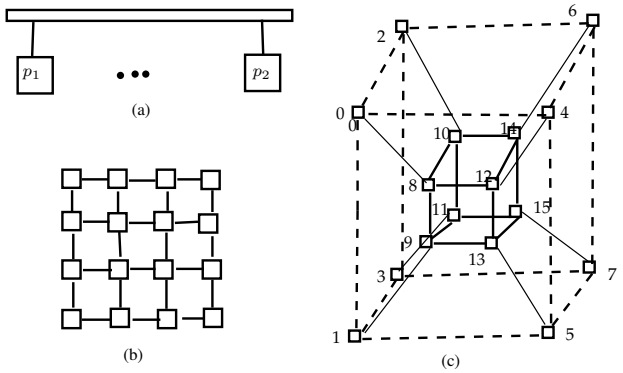

Figure 5.3: Interconnection Networks: (a) Bus, (b) Mesh, (c) hypercube.

precisely one bit position.

Many parallel algorithms exists for mesh and hypercube networks, they are fine tuned, but have disadvantages that they may not perform well other networks. Hence, in order to work on a new machine, an algorithm need to be developed from scratch. The algorithm that routes the information through network should exploit the network topology.

A multistage network is used to connect one set of switches called the input switches to another set called the output switches through a sequence of stages of switches. Such networks were originally designed for telephone networks. The stages of a multistage network are numbered 1 through L, where L is the depth of the network. The switches on stage 1 are the input switches, and those on stage L are the output switches. In most multistage networks, it is possible to send a message from any input switch to any output switch along a path that traverses the stages of the network in order from 1 to L. Multistage networks are frequently used in modular memory computers; typically processors are attached to input switches, and memory modules are attached to output switches. A processor accesses a word of memory by injecting a memory access request message into the network. This message then travels through the network to the appropriate memory module. If the request is to read a word of memory, then the memory module sends the data back through the network to the requesting processor. There are many different multistage network topologies. Figure 5.4, for example, shows a depth-2 network that connects 4 processors to 16 memory modules. Each switch in this network has two channels at the bottom and four channels at the top. The ratio of processors to memory modules in this example is chosen to reflect the fact that, in practice, a processor is capable of generating memory access requests faster than a memory module is capable of servicing them.

A complete binary tree of height $k, k \geq 0$ an integer, has $n = 2^{k+1} - 1$ nodes. The root node is at level 0 and the 2^k leaves are at level k. Each node at level $1, \ldots, k-1$ has two children and one parent, the root node does not have a parent node, and the leaves at level k do not have children nodes. Notice that the degree of the network is 3 and that the communication diameter is $2k = 2\lfloor \log_2 n \rfloor$. One severe disadvantage of a tree is that when extensive communication occurs, all messages traveling from one side of the tree to the other must pass through the root, causing a bottleneck. This is because the bisection bandwidth is only 1. Fat trees (fig. 5.4), alleviate this problem by increasing the bandwidth of the communication links near the root. This increase can come from changing the nature of the links, or, more easily, by using parallel communication links. Other generalizations of binary trees include complete t-ary trees of height k, where each node at level $0, \ldots, k-1$ has t children. There are $(t^{k+1} - 1)/(t1)$ nodes, the maximum degree is $t + 1$, and the diameter is $2k = 2\lfloor \log_t n \rfloor$.

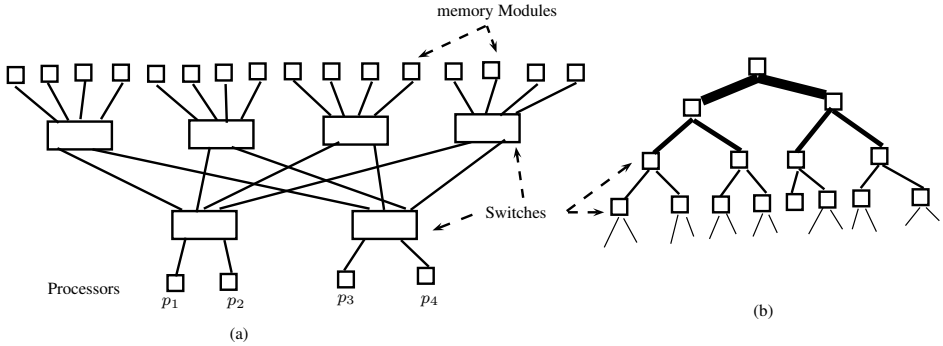

Figure 5.4: (a)2-Level multi-stage network (b) Fate-tree network topologies.

References

[1] Thomas H. Corman, et al., "Introduction to Algorithms, 3rd ed." *PHI*, 2009.

[2] Mikhail J. Atallah and Marina Blanton, "Algorithms and Theory of Computation handbook, Second Edition, Special Topics and Techniques", CRC Press, Taylor and Francis Group - A Chapman and Hall Book, 2010.

[3] Allen B. Tucker, Jr., "The computer Science and Engineering Handbook," *CRC Press*, 1997.