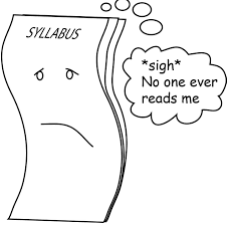


PARALLEL AND DISTRIBUTED ALGORITHMS
BY
DEBDEEP MUKHOPADHYAY
AND
ABHISHEK SOMANI

http://cse.iitkgp.ac.in/~debdeep/courses_iitkgp/PAlgo/index.htm

SYLLABUS



- The Idea of Parallelism: A Parallelised version of the Sieve of Eratosthenes
- PRAM Model of Parallel Computation
- Pointer Jumping and Divide & Conquer: Useful Techniques for Parallelization
- PRAM Algorithms: Parallel Reduction, Prefix Sums, List Ranking, Preorder Tree Traversal, Merging Two Sorted Lists, Graph Coloring
- Reducing the Number of Processors and Brent's Theorem
- Dichotomy of Parallel Computing Platforms
- Cost of Communication
- Programmer's view of modern multi-core processors
- The role of compilers and writing efficient serial programs

2

SYLLABUS (CONTD.)

Parallel Complexity: The P-Complete Class

Mapping and Scheduling

Elementary Parallel Algorithms

Sorting

Parallel Programming Languages: Shared Memory Parallel Programming using OpenMP

Writing efficient OpenMP programs

Dictionary Operations: Parallel Search

Graph Algorithms

Matrix Multiplication

Industrial Strength programming 1:

Programming for performance; Dense Matrix-matrix multiplication through various stages: data access optimization, loop interchange, blocking and tiling

Analyze BLAS (Basic Linear Algebra System) and ATLAS (Automatically Tuned Linear Algebra System) code

3

SYLLABUS (CONTD.)

Distributed Algorithms: models and complexity measures.

Safety, liveness, termination, logical time and event ordering

Global state and snapshot algorithms

Mutual exclusion and Clock Synchronization

Distributed Graph algorithms

Distributed Memory Parallel Programming: Cover MPI programming basics with simple programs and most useful directives; Demonstrate Parallel Monte Carlo

Industrial strength programming 2:

Scalable programming for capacity

Distributed sorting of massive arrays

Distributed Breadth-First Search of huge graphs and finding Connected Components

4

TEXT BOOKS AND RESOURCES

Michael J Quinn, Parallel Computing, TMH

**Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar,
Introduction to Parallel Computing, Pearson**

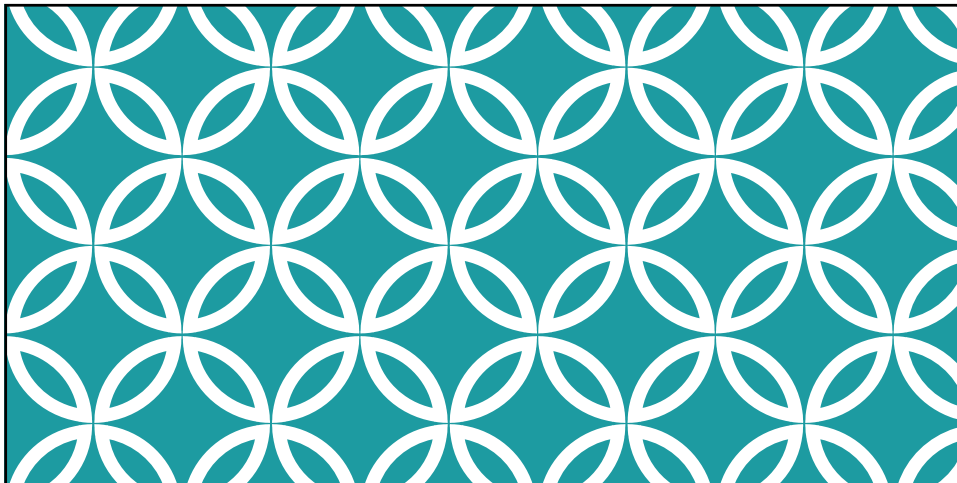
Joseph Jaja, An Introduction to Parallel Algorithms, Addison Wesley

Mukesh Singhal and Niranjana G. Shivaratri, Advanced Concepts in
Operating Systems, TMH

Course site:

http://cse.iitkgp.ac.in/~debdeep/courses_iitkgp/PAlgo/index.htm

5

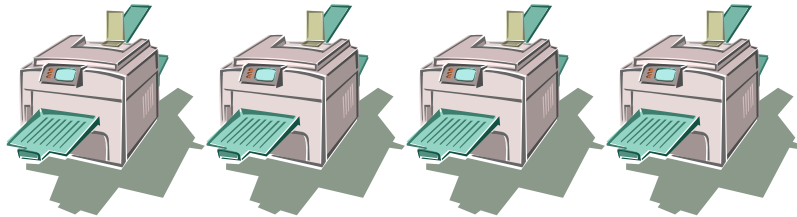


THE IDEA OF PARALLELISM

6

PARALLELISM AND HIGH SPEED COMPUTING

Many problems can be solved by massive parallelism.



p steps on 1 printer, 1 step on p printers

p = speed-up factor (best case)

Given a sequential algorithm, how can we parallelize it?

7

PARALLEL ALGORITHMS

The fastest computers in the world are built of numerous conventional microprocessors.

The emergence of these high performance, massively parallel computers demand the development of new algorithms to take advantage of this technology.

Objectives:

- Design
- Analyze
- Implement

Parallel algorithms on such computers with numerous processors

8

APPLICATIONS

Scientists often use high performance computing to validate their theory.

- “Data!data!data!” he cried impatiently. “I can’t make bricks without clay.”
— Arthur Conan Doyle, *The Adventure of the Copper Beeches*

Many scientific problems are so complex that solving them requires extremely powerful machines.

Some complex problems where parallel computing is useful:

- Computer Aided Design
- Cryptanalysis
- Quantum Chemistry, statistical mechanics, relativistic physics
- Cosmology and astrophysics
- Weather and environment modeling
- Biology and pharmacology
- Material design

9

MOTIVATIONS OF PARALLELISM

Computational Power Argument – from Transistors to FLOPS

In 1965, Gordon Moore said

“The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue if not to increase. Over the long term, the rate of increase is a bit more uncertain although there is no reason to believe it will not remain nearly constant for at least 10 years.”

From 1975, he revised the doubling period to 18 months, what came to be known as Moore’s law.

With more devices in chip, the pressing issue is of achieving increasing OPS (operation per second):

- A logical recourse is to rely on parallelism.

10

MOTIVATIONS OF PARALLELISM

The memory/disk speed argument

Overall speed of computation is determined not just by speed of processor, but also by the ability of memory system to feed data to it.

- While clock rates of high speed processors have increased by 40% in last 10 years, DRAM access times have increased by only 10%.
- Coupled with increase in instructions executed per clock cycles this gap presents a huge performance bottleneck.

Typically bridged by using a hierarchy of memory (Cache Memory)

Performance of memory system is determined by the fraction of total memory requests that can be satisfied from cache.

Parallel Platforms yield better memory system performance:

1. Larger aggregate caches
2. Higher aggregate bandwidths to the memory system
(both typically linear with the number of processors).

Further the principles of parallel algorithms themselves lend to cache friendly serial algorithms!

11

MOTIVATIONS OF PARALLELISM

The Data Communication Argument

The internet is often envisioned as one large heterogenous parallel/distributed computing environment.

Some most impressive applications:

- SETI (Search for Extra Terrestrial Intelligence) utilizes the power of large number of home computers to analyze electromagnetic signals from outer space.
- Factoring large integers
- Bitcoin mining (tries to find the collision of a cryptographic hash function)

Even having a powerful centralized processor will not work because of the low bandwidth network. It is infeasible to collect data at a node.

12

ADVENT OF PARALLEL MACHINES

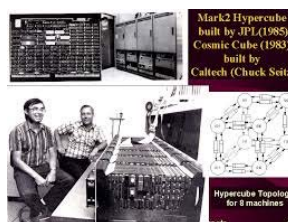
Daniel Slotnick at University of Illinois designed two early parallel computers.

- Solomon, constructed by Westinghouse Electric Company (early 60's)
- ILLIAC IV, assembled at Buroughs Corporation (early 70's)



During 70's at CMU, C.mmp and Cm* were constructed.

During 80's at Caltech, the Cosmic cube was built (the ancestor of multicomputers)



13

MICROPROCESSORS VS SUPERCOMPUTERS: PARALLEL MACHINES

Parallelism is exploited on a variety of high performance computers, in particular *massively parallel computers (MPPs)* and *clusters*.

MPPs, clusters, and high-performance vector computers are termed *supercomputers*

(vector processors have instructions that operate on a one dimension array)

Example: Cray Y/MP and NEC SX-3

Supercomputers were augmented with several architectural advances by 70's like: bit parallel memory, bit parallel arithmetic, cache memory, interleaved memory, instruction lookahead, multiple functional units, pipelining:

- However microprocessors had a long way to go!
- Huge development due to advances in their architectures, coupled with reduced instruction cycle times have lead to the convergence in relative performances of them and supercomputers.

Lead to the development of commercially viable parallel computers consisting of 10s, hundreds, or even 1000s of microprocessors.

- Example: Intel's Paragon XP/S, MasPar's MP-2, Thinking machines CM-5

14

PARALLEL PROCESSING TERMINOLOGY: WHAT IS PARALLELISM?

Parallelism refers to the simultaneous occurrence of events on a computer.

An event typically means one of the following:

- An arithmetical operation
- A logical operation
- Accessing memory
- Performing input or output (I/O)

15

TYPES OF PARALLELISM

Parallelism can be examined at several levels.

- **Job level:** several independent jobs simultaneously run on the same computer system.
- **Program level:** several tasks are performed simultaneously to solve a single common problem.
- **Instruction level:** the processing of an instruction such as adding two numbers can be divided into subinstruction. If several similar instructions are to be performed their subinstructions may be overlapped using a technique called pipelining
- **Bit level:** when the bits in a word are handled one after the other this is called a bit-serial operation. If the bits are acted on in parallel the operation is bit-parallel.

16

JOB LEVEL PARALLELISM

This is parallelism between different independent jobs or phases of a jobs on a computer

Example: Consider a computer with 4 processors. It runs jobs that are classified as small (S), medium (M) and large (L). Small jobs require 1 processor, medium jobs 2 processors, and large jobs 4 processor. Each job on the computer takes one unit of time. Initially there is a queue of jobs: S M L S S M L L S M M

Show how these jobs would be scheduled to run if the queue is treated in order. Also give a better schedule.

17

SCHEDULING EXAMPLE

| Time | Jobs running | Utilisation |
|------|--------------|-------------|
| 1 | S, M | 75% |
| 2 | L | 100% |
| 3 | S, S, M | 100% |
| 4 | L | 100% |
| 5 | L | 100% |
| 6 | S, M | 75% |
| 7 | M | 50% |

Average utilisation is 83.3%

Time to complete all jobs is 7 time units.

18

A BETTER SCHEDULE

A better schedule would allow jobs to be taken out of order to give higher utilisation.

S M L S S M L L S M M

Allow jobs to “float” to the front to the queue to maintain high utilisation.

| Time | Jobs running | Utilisation |
|------|--------------|-------------|
| 1 | S, M, S | 100% |
| 2 | L | 100% |
| 3 | S, M, S | 100% |
| 4 | L | 100% |
| 5 | L | 100% |
| 6 | M, M | 100% |

19

NOTES ON SCHEDULING EXAMPLE

In the last example:

- Average utilisation is 100%.
- Time to complete all jobs is 6 time units.

Actual situation is more complex as jobs may run for differing lengths of time.

Real job scheduler must balance high utilisation with fairness (otherwise large jobs may never run).

20

PARALLELISM BETWEEN JOB PHASES

Parallelism also arises when different independent jobs running on a machine have several phases, e.g., computation, writing to a graphics buffer, I/O to disk or tape, and system calls.

Suppose a job is executing and needs to perform I/O before it can progress further. I/O is usually expensive compared with computation, so the job currently running is suspended, and another is started. The original job resumes after the I/O operation has completed.

This requires special hardware: I/O channels or special I/O processor.

The *operating system* controls how different jobs are scheduled and share resources.

21

PROGRAM LEVEL PARALLELISM

This is parallelism between different parts of the same job.

Example

A robot has been programmed to look for electrical sockets when it runs low on power. When it finds one it goes over to it and plugs itself in to recharge. Three subsystems are involved in this - the vision, manipulation, and motion subsystems. Each subsystem is controlled by a different processor, and they act in parallel as the robot does different things.

22

ROBOT EXAMPLE

| Task | Vision | Manipulation | Motion |
|------------------------------------|--------|--------------|--------|
| 1. Looking for electrical socket | × | | × |
| 2. Going to electrical socket | × | | × |
| 3. Plugging into electrical socket | × | × | |

23

NOTES ON ROBOT EXAMPLE

The subsystems are fairly independent, with the vision subsystem guiding the others.

There may also be a central “brain” processor.

This is an example of *task parallelism* in which different tasks are performed concurrently to achieve a common goal.

24

DOMAIN DECOMPOSITION

A common form of program-level parallelism arises from the division of the data to be programmed into subsets.

This division is called *domain decomposition*.

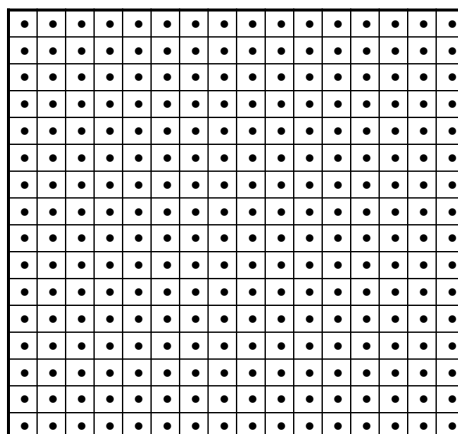
Parallelism that arises through domain decomposition is called *data parallelism*.

The data subsets are assigned to different computational processes. This is called *data distribution*.

Processes may be assigned to hardware processors by the program or by the runtime system. There may be more than one process on each processor.

25

DATA PARALLELISM

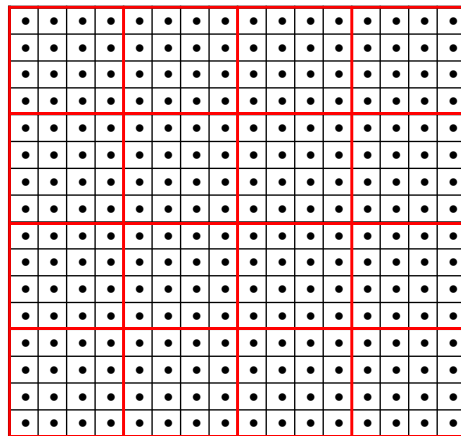


Consider an image digitised as a square array of pixels which we want to process by replacing each pixel value by the average of its neighbours.

The *domain* of the problem is the two-dimensional pixel array.

26

DOMAIN DECOMPOSITION



Suppose we decompose the problem into 16 subdomains

We then distribute the data by assigning each subdomain to a process.

The pixel array is a *regular* domain because the geometry is simple.

This is a homogeneous problem because each pixel requires the same amount of computation (almost - which pixels are different?).

27

PARALLEL PROCESSING

Information processing that emphasizes the concurrent manipulation of data elements belonging to one or more process solving a single problem.

A parallel computer is a multiple processor computer capable of parallel processing.

The throughput of the device is the number of results it produces per unit time.

- Throughput can be improved by increasing the speed of processing on data.
- Also, by increasing the number of operations that are performed at a time.

Pipelining and data parallelism are two ways of doing so.

28

PIPELINING AND DATA PARALLELISM

A pipelined computation is divided into number of steps, called segments or stages.

- Each segment works on full speed on a part of the computation.

Data parallelism is the use of multiple functional units to apply the same operation simultaneously to elements of a data set.

Speed up is the ratio between the time needed for the **most efficient sequential algorithm** to perform a computation and the time needed to perform the same computation on a machine incorporating pipelining and/or parallelism.

29

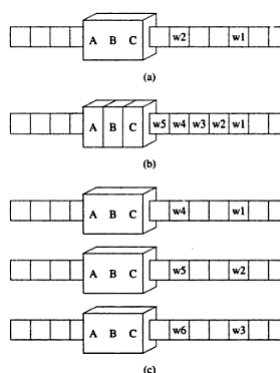
CONTRASTING THE IDEAS

Each assembly has three steps: A, B, and C, each taking 1 unit of time.

A sequential widget assembly machine makes a widget in 3 units of time (Fig a)

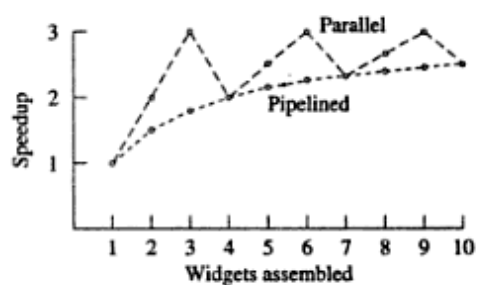
A pipeline with 3 segments, produces the first widget in 3 units of time (Fig b)

A 3-way data parallel widget assembly machine produces 3 widgets every 3 units of time (Fig c)



30

COMPARE THE SPEED-UPS



31

CONTROL PARALLELISM

Pipeline is actually a special case of a more general class of parallel algorithms, called control-path parallelism.

Data Parallelism: Same operation is performed on a data set.

Control Parallelism: Different operations are performed on different data elements concurrently.

Consider an example: The task of maintenance of an estate's landscape as quickly as possible>

- Mowing the lawn, edging the lawn, checking the sprinklers, weeding the flower beds.
- Except checking the sprinklers, other jobs would be quick if there are multiple workers.
- Increasing the lawn mowing speed by creating a team and assigning each member a portion of the lawn is an example of data parallelism.
- We can also perform the other tasks concurrently. This is an example of control parallelism.
- There is a precedence relationship, since all other tasks must be completed before the sprinklers are tested.

32

SCALABILITY

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

An architecture is scalable if it continues to yield the same performance per processor, albeit used on a larger problem size, as the number of processors increase.

This allows a user to solve larger problems in the same amount of time by using a parallel computer with more processors.

Data parallel algorithms are more scalable than control parallel algorithms, which is usually a constant, independent of the problem size.

- We shall study more such algorithms which are amenable to data parallelism!