SFWRENG 4F03
Parallel Computing

Winter 2020

06 Parallel Software

Dr Asghar Bokhari

Faculty of Engineering, McMaster University

February 6, 2020

Types of Parallelism

- Parallelism in Hardware
  - Uniprocessor
    - Pipelining
    - Multiple issue processor - Superscaler
  - Multiprocessors
    - SIMD - Vector Processors, GPUs
    - Shared memory multiprocessors
    - Distributed memory multiprocessors
    - Computer clusters
- Parallelism in Software
  - Very little commodity software making use of parallel hardware
  - Software engineers must learn how to write applications that exploit the available hardware.

Terminology

- Threads
  - Shared memory systems start a process and fork several threads.
  - The problem is divided among a number of threads.
- Processes
  - The problem is divided among processes in distributed memory sytems
- When the discussion applies equally well to both types of systems process/thread may be used

SPMD

- We start discussing software for MIMD systems
- Instead of creating different programs to be run on each processor, it is possible to write a single program
- SPMD (single program multiple data)
- A single executable behaves as if it were multiple different programs.

Example:

```
if( I am thread/process 0)
     do task1
else
     do task 2
```

- Task parallelism

SPMD

- SPMD programs can implement data parallelism also

Example:

```
if( I am thread/process 0)
     Operate on first half of the array
else
    Operate on the second half
```

Parallel Program Design

- How to design parallel applications?
- Have a serial program and want to parallelize it
- No silver bullet
- A problem may have several parallel solutions
- Different strategies and tools
- Goal - to be able choose among the many available methods to design efficient parallel applications

PCAM

- Four stage methodology - PCAM (Ian Foster)[1]
  1: Partition
    - Divide the problem into smaller tasks
    - Fine grained decomposition (sand is easier to pour than bricks)
    - Focus on the oppourtunities for parallelism ignoring the number of processors on target computer system
    - First decompose data associated with problem to approximately equal sized chunks
    - Associate each operation with the data on which it operates

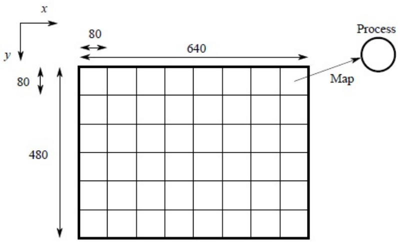
[1] Designing and Building Parallel Programs, by Ian Foster
http://www-unix.mcs.anl.gov/∼itf/dbpp/

PCAM

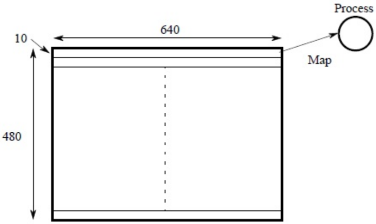
- Four stage methodology - PCAM (Ian Foster)[2]
  1: Partition Continued
     - This results in a set of tasks with some data and a set of operations on it
     - Representing each box as a task (ref next slide)- Fine-grained decomposition: large number of small tasks
     - Reoresenting a row as a task - Coarse-grained decomposition: small number of large tasks

[2] Designing and Building Parallel Programs, by Ian Foster http://www-unix.mcs.anl.gov/∼itf/dbpp/

Partitioning



(a) Square region for each process



(b) Row region for each process

PCAM

2: Communication

- An operation may require data from other tasks
- Data must be transferred between tasks so that computation may proceed
- Need to have communication strategies - point-to-point or broadcast

3: Aggregation (or Agglomeration)

- Re-evaluate the task and communication structures defined in above two stages
- Combine the tasks into larger tasks to improve performance or to reduce costs

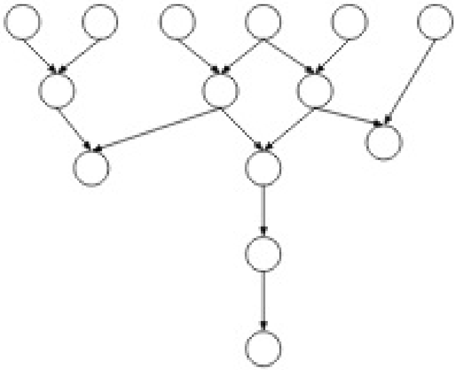
4: Mapping

- Assign tasks to the available processors with a goal to maximize process utilization and minimize communication costs

Task Dependency Graph

- Task: programmer-defined unit of computation
- A node (circle or elipse) represents a task in the graph
- Edge represents control dependence
- Start Node: node with no incoming edge
- Finish Node: node with no outgoing edge

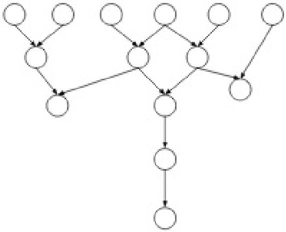
Dependency Graph



Degrees of Concurrency

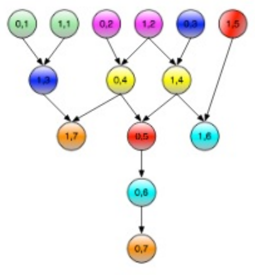
- **Maximum degree of concurrency:** the maximum number of tasks that can be executed in parallel at any stage of execution
- **Average degree of concurrency:** the average number of tasks that can be executed in parallel
- The average degree of concurrency is a more useful measure
- **Critical path:** the longest directed path between any pair of start and finish nodes
- **Critical path length:** sum of the weights of the nodes on a critical path
- Average degree of concurrency$=\frac{total\ \ amount\ \ of\ \ work}{critical\ \ path\ \ length}$
- Weight: units of time required to complete a task
- Speedup= sequential execution time/parallel execution time

Dependency Graph



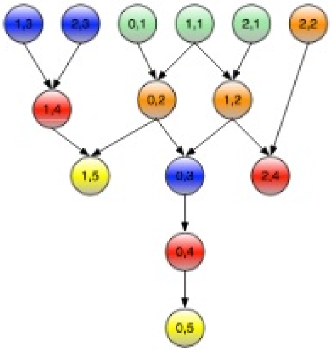
- maximum degree of concurrency is 6
- critical path length is 5
- total amount of work is 14 (assuming each task takes one unit of time)
- average degree of concurrency is 14/5=2.8
- max achieveable speedup = 14/5 = 2.8

Assignment to 2 processes



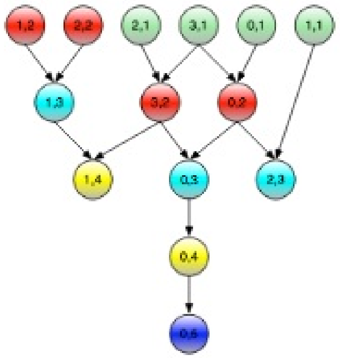
- First number is the process number
- Number of tasks per process = $14/2$ =7 (use ceiling if fraction)
- critical path length is 5
- speedup = $14/7 = 2$
- average degree of concurrency is $14/5$=2.8

Assignment to 3 processes



- First number is the process number
- critical path length is 5
- speedup $= 14/5 = 2.8$

Assignment to 4 processes



- First number is the process number
- critical path length is 5
- speedup $= 14/5 = 2.8$ What is the speedup for 8 processors??

Example

-

Title

-
-
- Template!