Parallel Computer Architecture

Parallel Computer Architecture = Processor + Processor + Processor + ……. + Network

Processor performance doubles every 18 months

Network performance grows, but not as fast

Time is often lost in communication

Other engineering considerations are:

- machine connectivity
- machine synchronisation etc.

Performance modeling

Accurate model with detailed simulation of individual hardware components is analytically intractable

Abstract machine models have been proposed to model the performance of any parallel algorithm and predict its speedup and cost

Parallel Algorithms

New issues to be considered -

- Speed up - the factor by which it is faster than an equivalent optimal sequential algorithm
- Cost - time multiplied by the number of processors used
- Scalability - Can it handle a significant increase in the size of the input without an increase in the number of processors?

Speed up / performance of a parallel algorithms depends on -

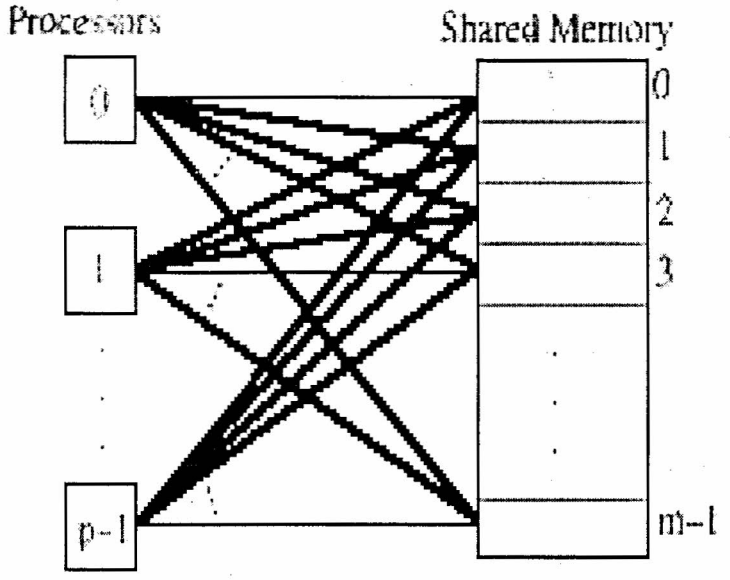
- The underlined parallel architecture
  - SMP or NUMA or computational cluster
- The programming model
  - Shared memory or message passing
- The actual implementation details
  - data locality, granularity of the subtasks etc.

Performance modeling

Abstract machine models help us to analyse and predict program performance without worrying about the details of physical machines

The PRAM model

Parallel Random Access Machine [Fortune and Wyllie, 1978]



Processor i can do the following in 3 phases of one cycle

1. Fetch an operand from address s_i in shared memory
2. Perform computations on data held in local registers
3. Store a value into address d_i in shared memory

The PRAM Model

The model contains

- unlimited number of processors (1 time unit / op)
- global shared memory (1 time unit / word)

Characteristics

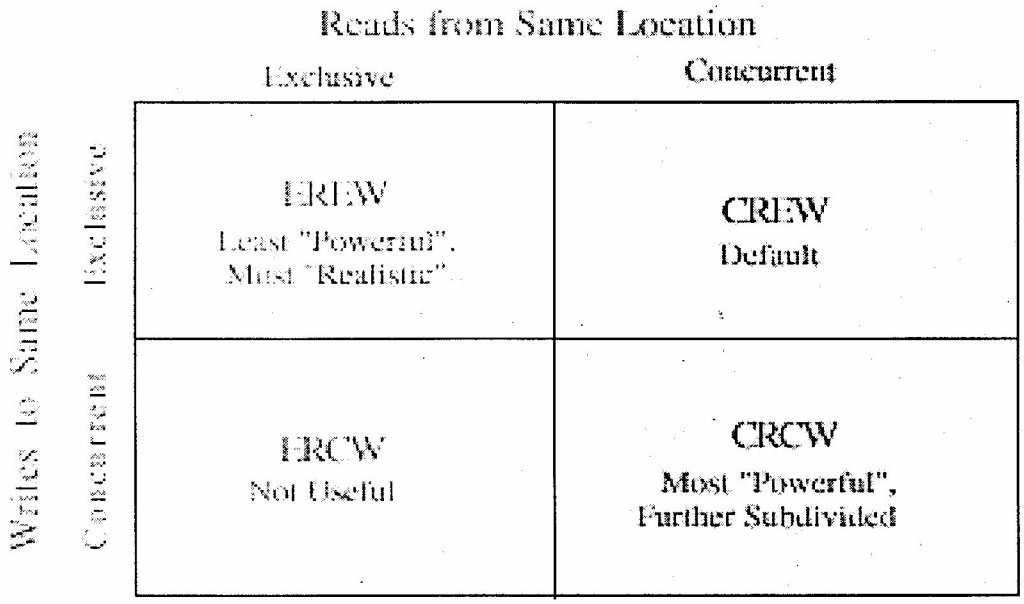
- Each processor is equivalent to a unit cost RAM
- All processors have access to the global shared memory
- Machines execute the same code synchronously, but work on different data

Advantages

- Natural generalisation of RAM
- One can concentrate on the inherent parallelism in the problem and ignore engineering details

Submodels of PRAM

- Based on read-write conventions
  - Do we allow more than one processor to read from the same memory locations?
  - Do we allow more than one processor to write onto the same memory locations?



CRCW PRAM variants

- Based on write conflicts resolution rules
  - Common CRCW PRAM - all processors writing to the same memory location write the same value
  - Arbitrary CRCW PRAM - any of the processors writing to the same memory location may succeed
  - Priority CRCW PRAM - processors are numbered and the processor with the minimum id succeeds
- Other variants are Random, Max/min etc.

All the above variants do not differ much in computational power

The PRAM Model

Disadvantages

- Not accurate, because ignores many details
- Ignores costs
- Can not model many real-life problems
  - In real-life problems full synchrony is hard to achieve
  - Sometimes asynchrony is best
- A performance model should consider
  - scalability
  - portability
  - predictability

of parallel algorithms. PRAM ignores all these issues

Problem:
Compute the maximum of $n = 2^m$ numbers

for K = m-1 step -1 to 0 do
    for all j, 2^k <= j <= 2^{k+1} -1 in parallel do
        A(j) =: max{A(2j), A(2j+1)}

Efficiency of PRAM Algorithm

Time $T(n)$, where n is the size of the problem

Number of processors $p(n)$

Work $w(n) = p(n).T(n)$

Any PRAM algorithm that does work $w(n)$ can be converted into a sequential algorithm that runs in $w(n)$ time

A parallel algorithm is work optimal iff

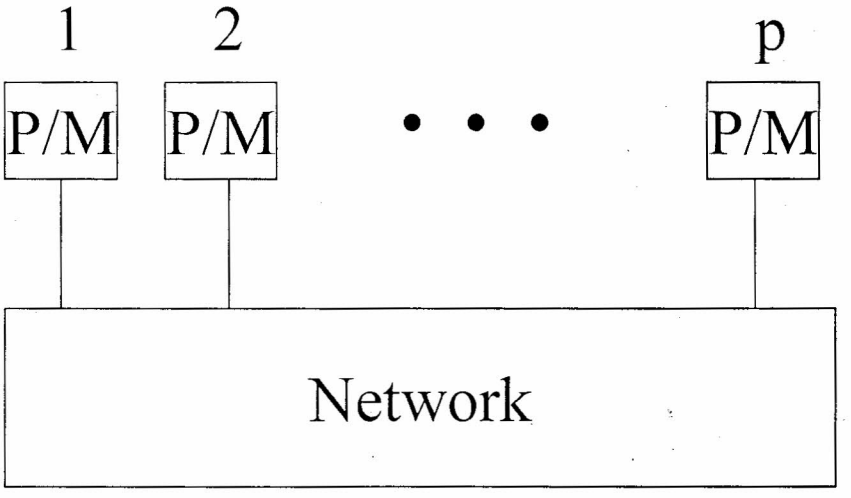
$$w(n) = T_{seq}(n)$$

where $T_{seq}(n)$ is the best sequential time

The BSP Model

Bulk Synchronous Parallel Computer [Valiant, 1990]

A realistic general-purpose parallel model



Contains

- p processors with local memory, 1 time unit per operation
- communication network
  - g time units per word delivered
- synchronisation mechanism
  - l time units per barrier synchronisation

More scalable, portable and predictable than PRAM

The BSP Model

Processors execute algorithms in a series of supersteps

During a superstep a processor processes local data and initiate any requests to remote memory locations

At the end of a superstep period all the processors must synchronize using the barrier mechanism

During the barrier any unresolved remote memory accesses are completed and then, and only then, the processors can continue onto the next superstep