Lecture 18

Parallel Computing and Big Data

Course: Algorithms for Big Data

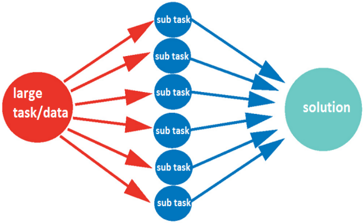
Instructor: Hossein Jowhari

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Spring 2021

Parallel Processing: Basic Idea

Speeding up via breaking large tasks into (independent) sub-tasks and executing them in parallel.

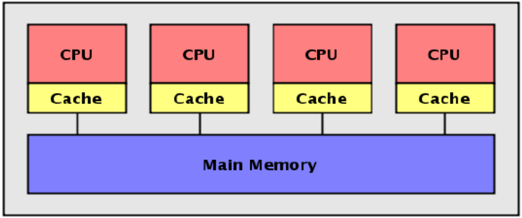


$$\text{performance} = \frac{\text{stime}}{\text{ptime}}$$

- Sequential time (stime):
  Time to finish the job in sequential manner
- Parallel time (ptime, wall-clock time):
  Time to finish the job using parallel processes

Models for parallel computing

- PRAM (shared memory)



- Multi-threading

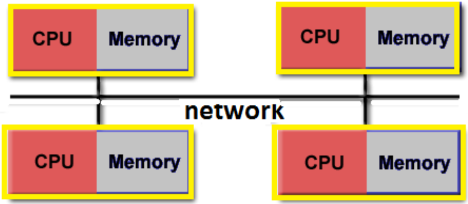
  Java, Python, ...

- OpenMP

  C,C++, Fortran

- CUDA (GPU)

- Message Passing (local memory)



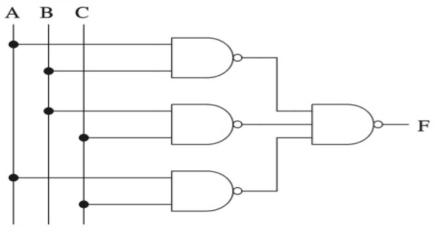
- MPI (C++)
- MapReduce, Hadoop
- Spark

Theoretical Models for Parallel Computing

- Circuit Complexity
- PRAM (Parallel Random Access Model)
- MPC (Massively Parallel Computing)
- ...

Circuit Complexity

aka Non-uniform complexity



How many AND, OR, NOT gates are needed to compute a boolean function $f : \{0,1\}^n \to \{0,1\}$?

total number of circuits : time complexity

depth of the circuit: parallel time complexity

[Theorem] For any boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, there is a circuit with depth 4 and 2^{n+1} gates.

Related Complexity Classes:

- AC: Functions solvable with (unbounded fan-in) circuits with $O(\log^{O(1)} n)$ depth and $n^{O(1)}$ number of gates.

  unbounded fan-in gate: a gate with multiple inputs

- AC_0: Functions solvable with (unbounded fan-in) circuits with $O(1)$ depth and $n^{O(1)}$ number of gates.
- NC: Functions solvable with (bounded fan-in) circuits with $O(\log^{O(1)} n)$ depth and $n^{O(1)}$ number of gates.
- NC_k: Functions solvable with (bounded fan-in) circuits with $O(\log^{O(k)} n)$ depth and $n^{O(1)}$ number of gates.

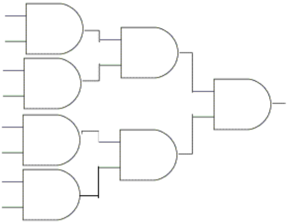
Circuit Complexity

Examples:

- AND of n bits: $f(x_1, \dots, x_n) = x_1 \wedge \dots \wedge x_n$

  AC: (depth$=$ 2, number of gates $=$ 1)

  NC: (depth$= \log n$, number of gates $= n-1$)



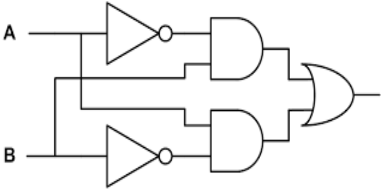
Circuit Complexity

Examples:

- XOR of n bits: $f(x_1, \ldots, x_n) = x_1 + \ldots + x_n \mod 2$

  AC: (depth$= O(\frac{\log n}{\log \log n})$, number of gates $= n^{O(1)}$)

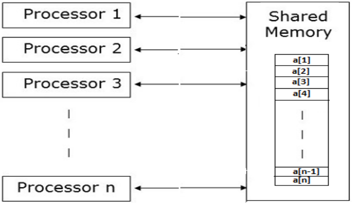
  NC: (depth$= O(\log n)$, number of gates $= O(n)$)



A xor B = A'B + AB'

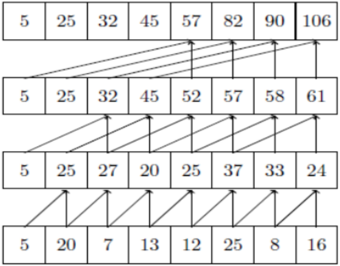
Planar Perfect Matching $\in NC$ Perfect Matching $\in NC$?

Abstract Theoretical Models: PRAM



- Exclusive read exclusive write (EREW)
- Concurrent read exclusive write (CREW)
- Exclusive read concurrent write (ERCW)
- Concurrent read concurrent write (CRCW)

CREW PRAM for Prefix Sum



$$\forall k \in [n], \quad S_k = \sum_{i=1}^{k} a[i]$$

n processors

$\log n$ parallel time

PRAM: Merging sorted arrays

A	10	15	22	80	91	98						
B	5	8	11	15	70	90						
Output	5	8	10	11	15	15	22	70	80	90	91	98

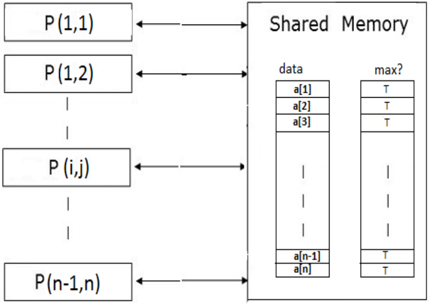
n processors: $P_1, \ldots, P_n$

P_i binary searches array B to find how many numbers in B are greater than $A[i]$. Requires $\log n$ comparisons.

Having this information, the processor P_i puts $A[i]$ in its right position in the output array.

$\log n$ parallel time

PRAM (CRCW): find max



$\binom{n}{2}$ processors. n distinct numbers.

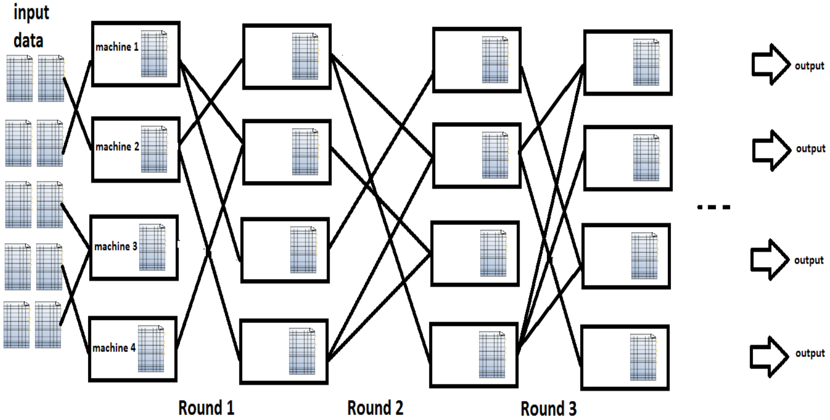
For $i \in [n]$, processor P_i writes $m[i] = \mathrm{T}$.

Processor P_{ij} compares $a[i]$ and $a[j]$. If $a[i] < a[j]$ writes $m[i] = \mathrm{F}$

$O(1)$ parallel time

Abstract Theoretical Models: MPC

Massively Parallel Computing



- N input data size, M machines
- S memory size per machine

At first, input data is distributed among machines. Each machine gets a part of the input. We assume $S = o(N)$. Typically $S = O(N^{\epsilon})$ for some $\epsilon < 1$.

If $S \geq N$, we could give all data to a single machine which can solve the entire problem locally.

Computation is done in rounds.

In each round, the machines do local computations on their share of the input. Then in a synchronous manner they communicate with each other.

Each machine transmits at most $O(S)$ words in each round!

With these assumptions, the real bottleneck is the number of rounds.

MPC for graph problems

- Big graph $G = (V, E)$ with m edges on n vertices
- Input size $N = O(m)$
- Each machine gets a subset of the edges (repetitions?)
- We assume number of machines $M = O(\frac{m}{S})$
- Strongly super-linear memory $S = O(n^{1+\epsilon})$ when $\epsilon \in (0, 1]$
- Near linear memory $S = \tilde{O}(n)$
- Strongly sub-linear memory $S = O(n^{\epsilon})$ when $\epsilon < 1$

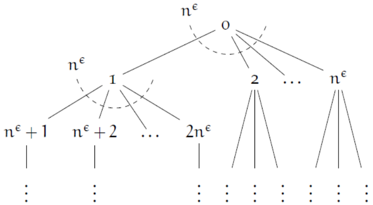
Broadcasting trees

Every machines sends at most S words in each round.

Suppose machine number 0 wants to broadcast n words to all M machines. How many rounds does it take?

Suppose $S = n^{1+\epsilon}$ (super-linear case). If $M \geq n^{\epsilon}$ broadcasting cannot be done in one round.

using a broadcast tree, this can be done in $O(\frac{1}{\epsilon})$ rounds.



In the first round, machine 0 sends n words to machines $1, \ldots, n^{\epsilon}$. (second level of the tree)

In the second round, each machine in the second level sends n words to n^{ϵ} new machines (third level)

After $O(\frac{1}{\epsilon})$ rounds, all $M \leq n^2$ machines have received the words.