

CS 312: Algorithm Analysis



Lecture #38: Parallel Algorithms

Announcements

- Help Session on Proj. #7 today at 6:30
- Proj. #3 Graded and on BlackBoard
 - Please check your early bonus count and your late day budget on BlackBoard
- Complete the Course Evaluation for “1 HW”
 - In sec. 002 & 003, this will mean that you will receive extra credit in the amount of the average value of a homework asst.

Objectives

Lecture #37: Linear Programming: Simplex Method

- Quickly Review Slack Form
- Introduce Pivot Algorithm
- Walk through Example
- Understand Simplex Method

Lecture #38: Parallel Algorithms

- Explain “computational model”
- Compare P-RAM with Fixed Network model of parallel computation
- Try some examples
- Compute speedup and efficiency

Definitions

- **Sequential Algorithm**
 - Any algorithm designed for a single-processor machine
- **Parallel Algorithm**
 - Any algorithm designed for a multiprocessor machine

Simple Example

- I have 5 loads of clothes to wash
- It takes 25 minutes to wash one load
- How long will it take me?

Computational Models

- For performance evaluation...
- A computational model describes how much it costs to perform atomic, or basic, operations.
- For today, we'll talk only about time cost.

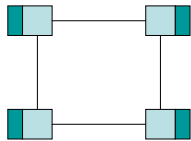
Computational Models

- **Sequential**
 - RAM: random access machine
 - Read, write, add, subtract, compare etc. all done in one time step.
- **Parallel**
 - Given n machines, perform n adds, subtracts, compares etc in one step.
 - Communication is more complicated because it requires a shared resource
 - Shared resource implies a policy for sharing.

Two Communication Models

- **Fixed Network**
- **Parallel Random Access Machine (P-RAM)**
- **Hybrids exist, but we won't cover them.**
 - Fixed network of P-RAMs. Etc.

Fixed Network

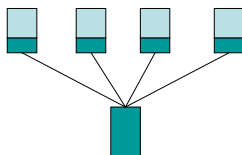


- Processors are connected by communication links.
- Two processors can communicate in one time step (if there is an edge between them).
- E.g., Hypercube

Fixed Network Policies

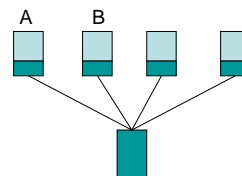
- Links simplex or duplex?
- Completely connected?
- Forwarding, bypassing, dropping?

Parallel RAM (P-RAM)



- All processors share access to a common memory
- All processors have some local memory as well
- Two processors can communicate in two time steps.
- Nearest real implementation: SIMD (e.g., AltiVec)

P-RAM Policies



What if A and B want to read the same location at the same time?
What if A and B want to write the same location at the same time?

P-RAM Policies

- Concurrent reads
 - A and B can read at the same time
- Concurrent writes
 - A and B can write at the same time
- CREW, EREW, ~~ERCW~~, CRCW
- Read/write bypassing
- Dropping reads/writes
- Fences – for synchronization

Another Example

- 10 numbers to be added
- Two people (A and B)
- How do we parallelize this?
 - On a fixed network
 - On a P-RAM
- How much time is saved?

Another Example

- 10 numbers to be added
- Two people (A and B)
- How do we parallelize this?
 - On a P-RAM $5(\text{read}) + 4(\text{add}) + 1(\text{comm.}) + 1(\text{add}) = 11$
 - On a Fixed Network $5(\text{read}) + 4(\text{add}) + 1(\text{write}) + 1(\text{read}) + 1(\text{add}) = 12$
- How much time is saved? 1

In-class Demo

- 7 volunteers (everyone!) to add 14 numbers
- CREW P-RAM $p = 7$
 - Chalkboard is the shared memory
 - Send writes to the queue. (I'll be the queue)
 - Read at anytime
- Fixed Network
 - Completely connected simplex network.

In-class Demo

Sequential: Read 14, Add 13, Write 1 // 28

CREW P-RAM p processors = 7

Step 1. (7p)	Step 2. (4p)	Step 3. (2p)	Step 4. (1p)
Read: 2	Read: 1	Read: 1	Read: 1
Add: 1	Add: 1	Add: 1	Add: 1
Write: 7	Write: 4	Write: 2	Write: 1

Fixed Network

Step 1. (7p)	Step 2. (4p)	Step 3. (2p)	Step 4. (1p)
Read: 2	Add: 1	Add: 1	Add: 1
Add: 1	Comm: 1	Comm: 1	Write: 1
Comm: 1			

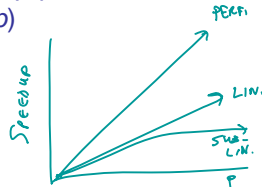
Parallel Sum (p. 379)

```

function parsum (T, n)
  for i ← lg n - 1 down to 0 do
    for 2i ≤ j ≤ 2i+1 - 1 in parallel do
      T[j] ← T[2j] + T[2j+1]
  return T[1]
    
```

Speedup

- π : Given problem for which the best-known sequential algorithm has a run time of $S(n)$
- n : problem size
- Parallel algorithm on a p -processor machine in time $T(n, p)$
- $Speedup(p) = \frac{S(n)}{T(n, p)}$



Efficiency

The efficiency of a parallel algorithm is:

$$E = \frac{S(n)}{pT(n, p)} = \frac{SPEEDUP(p)}{p}$$

Where...

- $S(n)$ = bound on the sequential algorithm, and
- $T(n, p)$ = bound on a parallel algorithm with p processes
- n = instance size
- p = number of processors

What is the speedup for 100 numbers problem?

- Sequential time = 200 units of time
- Person A can add 50 numbers in 49 units of time (+50 reads, +1 write)
- At the same time, Person B can add the other 50 numbers
- In one more unit of time, the two sums can be added.
- $Speedup = \frac{200}{100} = 2$

Heap Sort

- Optimal run time: $\Theta(n \log n)$ - SEQUENTIAL
- Let A be an n -processor parallel algorithm that sorts n keys in $\Theta(\log n)$
- Let B be an $2n$ -processor algorithm that also sorts n keys in $\Theta(\log n)$ time
- Speedup of A is $\frac{\Theta(n \log n)}{\Theta(\log n)} = \Theta(n)$
- Speedup of B is also $\frac{\Theta(n \log n)}{\Theta(\log n)} = \Theta(n)$

Superlinear speedup

- Is it possible?

Superlinear speedup

- Is it possible?
- Only when speedup is defined with runtimes
 1. p processors have more aggregate memory
 2. the cache-hit frequency might be better for the parallel machine w/more aggregate cache than one processor

In practice

Assignment

HW #29 – see the blog

1. Problem 11.4 (requires that you read sec. 11.2)
2. Suppose you have a parallel algorithm that runs in $O(T(n,p))$ time for $T(n,p) = \log n$ for a problem of size n using p processors. Suppose that the best known sequential algorithm solves problems of size n in $O(S(n))$ time for $S(n) = n \log n$. What is the speedup for this algorithm and what is the efficiency? Did this algorithm achieve linear speedup?