Models of Computation for Massive Data

Jeff M. Phillips

August 22, 2011

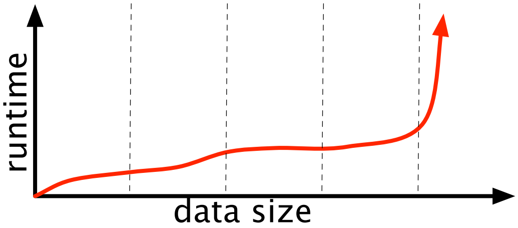
Outline

Sequential:

- External Memory / (I/O)-Efficient
- Streaming

Parallel:

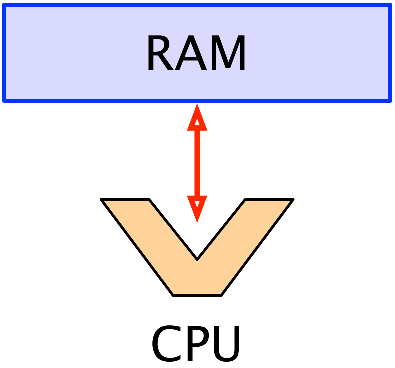
- PRAM and BSP
- MapReduce
- GP-GPU
- Distributed Computing



RAM Model

RAM model (Von Neumann Architecture):

- CPU and Memory
- CPU Operations $(+, -, *, \ldots)$ constant time
- READ, WRITE take constant time.

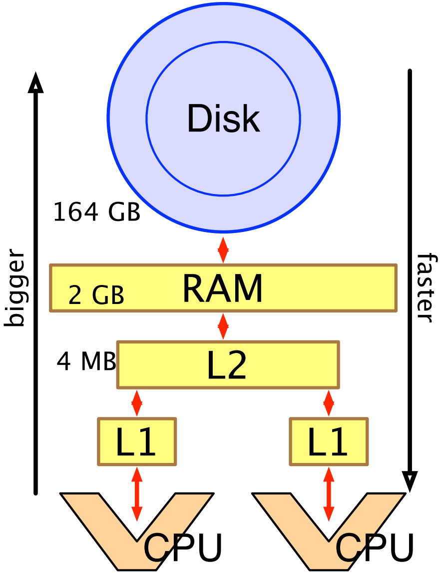


Today's Reality

What your computer actually looks like:

- 3+ layers of memory hierarchy.
- Small number of CPUs.

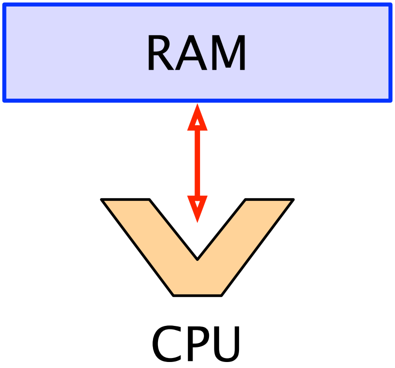
Many variations!



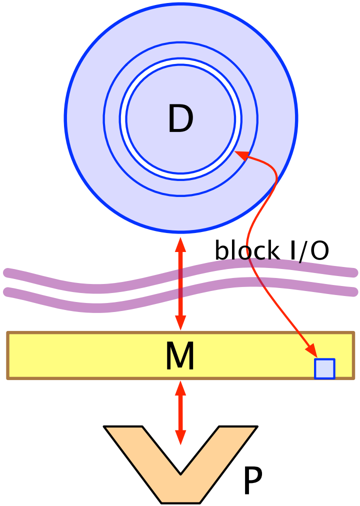
RAM Model

RAM model (Von Neumann Architecture):

- CPU and Memory
- CPU Operations $(+, -, *, \ldots)$ constant time
- READ, WRITE take constant time.

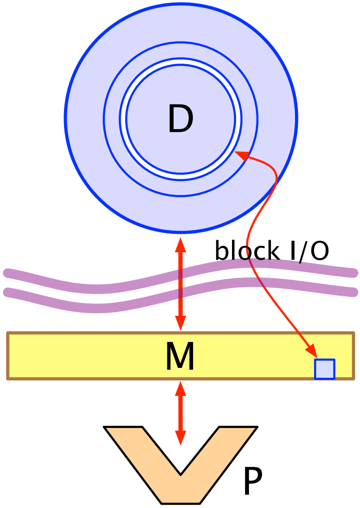


External Memory Model



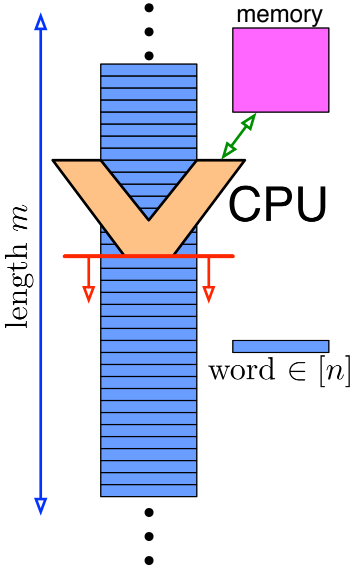
- N = size of problem instance
- B = size of disk block
- M = number of items that fits in Memory
- T = number of items in output
- I/O = block move between Memory and Disk

External Memory Model



- N = size of problem instance
- B = size of disk block
- M = number of items that fits in Memory
- T = number of items in output
- I/O = block move between Memory and Disk

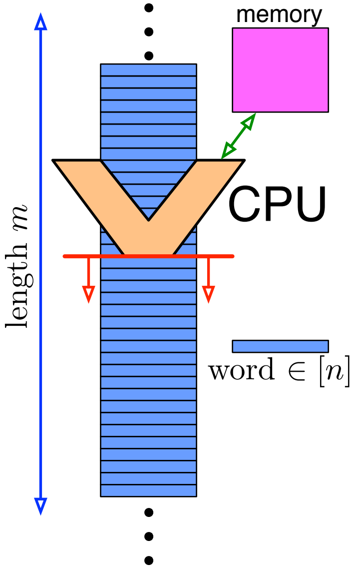
Streaming Model



CPU makes "one pass" on data

- Ordered set $A = \langle a_1, a_2, \ldots, a_m \rangle$
- Each $a_i \in [n]$, size $\log n$
- Compute $f(A)$ or maintain $f(A_i)$ for $A_i = \langle a_1, a_2, \ldots, a_i \rangle$.
- Space restricted to $S = O(\text{poly}(\log m, \log n))$.
- Updates $O(\text{poly}(S))$ for each a_i.

Streaming Model



CPU makes "one pass" on data

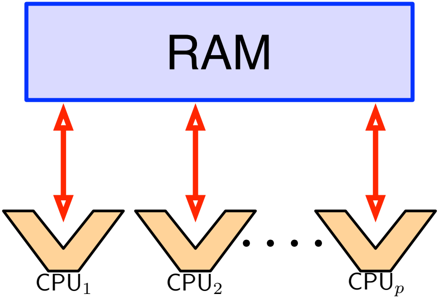
- Ordered set $A = \langle a_1, a_2, \ldots, a_m \rangle$
- Each $a_i \in [n]$, size $\log n$
- Compute $f(A)$ or maintain $f(A_i)$ for $A_i = \langle a_1, a_2, \ldots, a_i \rangle$.
- Space restricted to $S = O(\text{poly}(\log m, \log n))$.
- Updates $O(\text{poly}(S))$ for each a_i.

PRAM

Many (p) processors. Access shared memory:

- EREW : Exclusive Read Exclusive Write
- CREW : Concurrent Read Exclusive Write
- CRCW : Concurrent Read Concurrent Write

Simple model, but has shortcomings...
...such as Synchronization.

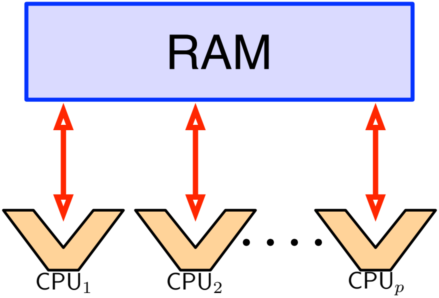


PRAM

Many (p) processors. Access shared memory:

- EREW : Exclusive Read Exclusive Write
- CREW : Concurrent Read Exclusive Write
- CRCW : Concurrent Read Concurrent Write

Simple model, but has shortcomings...
...such as Synchronization.

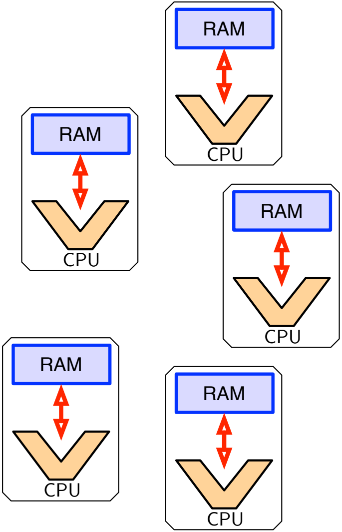


Bulk Synchronous Parallel

Each Processor has its own Memory
Parallelism Procedes in Rounds:

1. Compute: Each processor computes on its own Data: w_i.
2. Synchronize: Each processor sends messages to others: $s_i = m \times g \times h$.
3. Barrier: All processors wait until others done.

Runtime: $\max w_i + \max s_i$



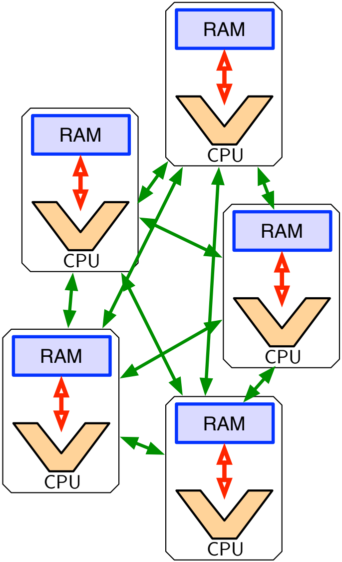
Pro: Captures Parallelism *and* Synchronization
Con: Ignores Locality.

Bulk Synchronous Parallel

Each Processor has its own Memory
Parallelism Procedes in Rounds:

1. Compute: Each processor computes on its own Data: w_i.
2. Synchronize: Each processor sends messages to others: $s_i = m \times g \times h$.
3. Barrier: All processors wait until others done.

Runtime: $\max w_i + \max s_i$



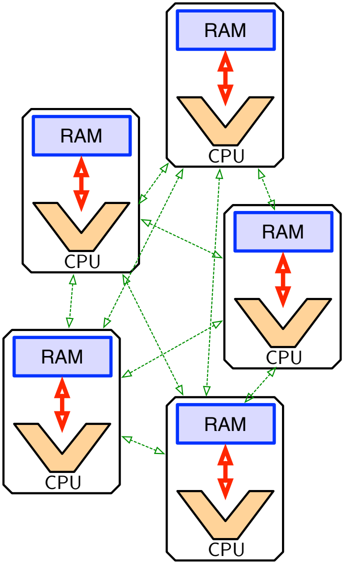
Pro: Captures Parallelism *and* Synchronization
Con: Ignores Locality.

Bulk Synchronous Parallel

Each Processor has its own Memory
Parallelism Procedes in Rounds:

1. Compute: Each processor computes on its own Data: w_i.
2. Synchronize: Each processor sends messages to others: $s_i = m \times g \times h$.
3. Barrier: All processors wait until others done.

Runtime: $\max w_i + \max s_i$



Pro: Captures Parallelism *and* Synchronization
Con: Ignores Locality.

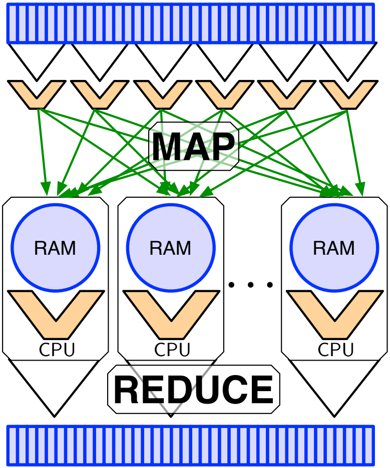
MapReduce

Each Processor has full hard drive, data items $< \text{KEY}, \text{VALUE} >$.
Parallelism Procedes in Rounds:

- Map: assigns items to processor by KEY.
- Reduce: processes all items using VALUE. Usually combines many items with same KEY.

Repeat M+R a constant number of times, often only one round.

- Optional post-processing step.



Pro: Robust (duplication) and simple. Can harness Locality
Con: Somewhat restrictive model

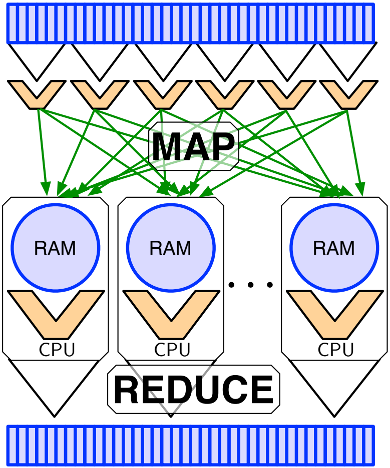
MapReduce

Each Processor has full hard drive, data items $< \text{KEY}, \text{VALUE} >$.
Parallelism Procedes in Rounds:

- Map: assigns items to processor by KEY.
- Reduce: processes all items using VALUE. Usually combines many items with same KEY.

Repeat M+R a constant number of times, often only one round.

- Optional post-processing step.

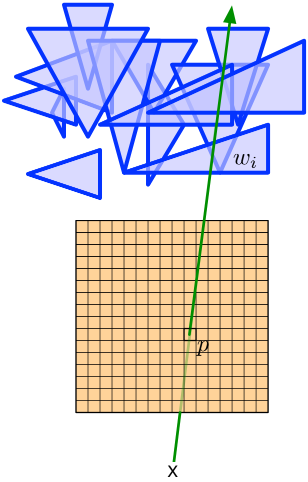


Pro: Robust (duplication) and simple. Can harness Locality
Con: Somewhat restrictive model

General Purpose GPU

Massive parallelism on your desktop.
Uses **G**raphics **P**rocessing **U**nit.
Designed for efficient video rasterizing.
Each *processor* corresponds to pixel p

- depth buffer:
  $D(p) = \min_i ||x - w_i||$
- color buffer: $C(p) = \sum_i \alpha_i \chi_i$
- ...

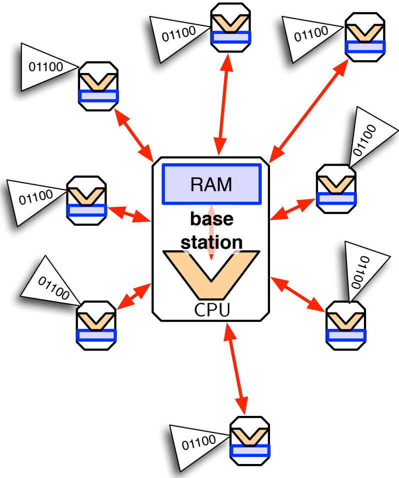


Pro: Fine grain, massive parallelism. Cheap.
Con: Somewhat restrictive model. Small memory.

Distributed Computing

Many small slow processors with data.
Communication very expensive.

- Report to *base station*
- Merge tree
- Unorganized (peer-to-peer)

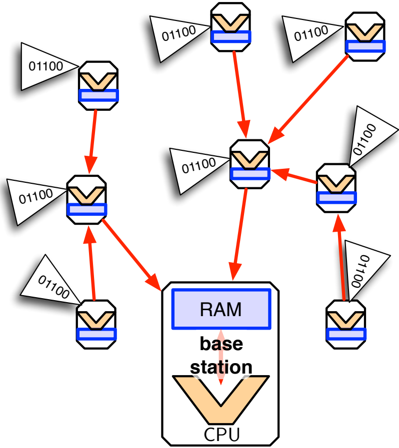


Data collection or Distribution

Distributed Computing

Many small slow processors with data.
Communication very expensive.

- Report to *base station*
- Merge tree
- Unorganized (peer-to-peer)

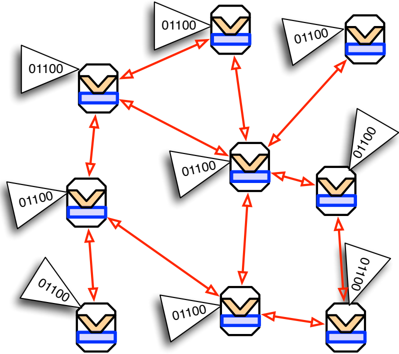


Data collection or Distribution

Distributed Computing

Many small slow processors with data.
Communication very expensive.

- Report to *base station*
- Merge tree
- Unorganized (peer-to-peer)

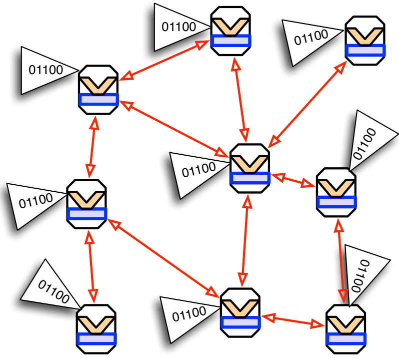


Data collection or Distribution

Distributed Computing

Many small slow processors with data.
Communication very expensive.

- Report to *base station*
- Merge tree
- Unorganized (peer-to-peer)



Data collection or Distribution

Advanced Algorithms: Approximate, Randomized

Themes

What are course goals?

- How to analyze algorithms in each model
- Taste of how to use each model
- When to use each model

Themes

What are course goals?

- How to analyze algorithms in each model
- Taste of how to use each model
- When to use each model

Work Plan:

- 2-3 weeks each model.
  - Background and Model.
  - Example algorithms analysis in each model.
  - Small assignment (total $= 1/2$ grade).
- Course Project ($1/2$ grade).
  - Compare single problem in multiple models
  - Solve challenging problem in one model
  - Analyze challenging problem in one model
- Access to Amazon's EC2. More in about 1 month.

Class Survey

Q1: Algorithms Background

A What is the highest algorithms class you have taken?

B What was the hardest topic?

C Have you seen a randomized algorithm? (which one?)

D Have you seen an approximation algorithm? (which one?)

Q2: Programming Background

A Have you used C or C++?

B Have you used Matlab?

C What other languages have you coded in?

Q3: Class interest

A Are you registered?

B How certain are you to stay in the class? (choose one)

(a) Definitely staying in!
(b) Probably staying in.
(c) Deciding between this and another class.
(d) Just shopping around...

Data Group

Data Group Meeting

Thursdays @ 12-1pm in Graphics Annex

`http://datagroup.cs.utah.edu/dbgroup.php`