CSE 599c
Scientific Data Management

Magdalena Balazinska and Bill Howe

Spring 2010

Lecture 4 – Data Intensive Analytics

References

- **Parallel Database Systems: The Future of High Performance Database Systems.** Dave DeWitt and Jim Gray. Com. of the ACM. 1992. Also in Red Book 4th Ed. Sec. 1 and 2.
- **MapReduce: Simplified Data Processing on Large Clusters.** Jeffrey Dean and Sanjay Ghemawat. OSDI 2004. Sec. 1 - 4.
- **Pig Latin: A Not-So-Foreign Language for Data Processing.** C. Olston, B. Reed, U. Srivastava, R. Kumar and A. Tomkins. SIGMOD 2008. Introduction.
- **CloudBurst: highly sensitive read mapping with MapReduce**. Michael C. Schatz. Bioinformatics 2009 25(11):1363-1369
- **Skew-Resistant Parallel Processing of Feature-Extracting Scientific User-Defined Functions.** YongChul Kwon, Magdalena Balazinska, Bill Howe, and Jerome Rolia. To appear in SOCC 2010.

The Problem

- As we have seen over the past few lectures, science is becoming data intensive
- Not all scientists but increasingly many scientists can no longer perform their analysis locally on their desktop using Excel
- They need more powerful data analysis tools
- Many scientists even need to use clusters of machines

State of the Art

- Surprisingly few scientists use databases today
  - Although some do use dbs and even data warehousing systems
- Even fewer use parallel databases or MapReduce
- Why is that? See discussion from past lecture
- So what do they use for their data intensive analytics?
  - MPI
  - OpenMP
  - Specialized libraries such as ScaLAPACK (matrix ops and more)

Message Passing Interface (MPI)

- ”A message-passing API, together with protocol and semantic specifications for how its features must behave in any implementation.” [Wikipedia]
  - Designed for shared-nothing clusters
  - Very popular for HPC programming
- Example functionality:
  - Message send and receive
  - Broadcast, scatter/gather, synchronization barrier
- Limitations
  - Low-level; Need to know and think in terms of parallel processing
  - Libraries on top of MPI exist but are domain-specific and limited

OpenMP

- Alternative: forget clusters get shared-memory machine
- "The OpenMP Application Program Interface (API) supports multi-platform shared-memory parallel programming in C/C++ and Fortran" http://openmp.org/wp/
- OpenMP allows one to give directives to the compiler as to how to parallelize the code
- Limitation: shared-memory machines can be expensive

Our Discussion Today

- Can parallel databases and MapReduce help?
- What are the challenges behind applying these technologies?

- Outline for the rest of the lecture
  - Quick review of parallel dbs and MapReduce
  - CloudBurst paper
    - Concrete example of applying MapReduce
  - SkewReduce
    - Example of extending MapReduce motivated by science requirements

Parallel DBMSs

- Goal
  - Improve performance by executing multiple operations in parallel
- Key benefit
  - Cheaper to scale than relying on a single increasingly more powerful processor
- Key challenge
  - Ensure overhead and contention do not kill performance

Performance Metrics for Parallel DBMSs

- Speedup
  - More processors ➔ higher speed
  - Individual queries should run faster
  - Should do more transactions per second (TPS)

- Scaleup
  - More processors ➔ can process more data
  - Batch scaleup
    - Same query on larger input data should take the same time
  - Transaction scaleup
    - N-times as many TPS on N-times larger database
    - But each transaction typically remains small

Challenges to Linear Speedup and Scaleup

- Startup cost
  - Cost of starting an operation on many processors
- Interference
  - Contention for resources between processors
- Skew
  - Slowest step becomes the bottleneck

Architectures for Parallel Databases

- Shared memory
- Shared disk
- Shared nothing

Shared Nothing

- Most scalable architecture
  - Minimizes interference by minimizing resource sharing
  - Can use commodity hardware

- Also most difficult to program and manage

- Processor = server = node
- P = number of nodes

We will focus on shared nothing

Taxonomy for Parallel Query Evaluation

- Inter-query parallelism
  - Each query runs on one processor
- Inter-operator parallelism
  - A query runs on multiple processors
  - An operator runs on one processor
- Intra-operator parallelism
  - An operator runs on multiple processors

We study only intra-operator parallelism: most scalable

Horizontal Data Partitioning

- Relation R split into P chunks R_0, …, R_{P-1}, stored at the P nodes
- Round robin: tuple t_i to chunk (i mod P)
- Hash based partitioning on attribute A:
  - Tuple t to chunk h(t.A) mod P
- Range based partitioning on attribute A:
  - Tuple t to chunk i if $v_{i-1} < t.A < v_i$

Parallel Selection

Compute $\sigma_{A=v}(R)$, or $\sigma_{v1<A<v2}(R)$

- On a conventional database: cost = B(R)
- Q: What is the cost on a parallel database with P processors ?
  - Round robin
  - Hash partitioned
  - Range partitioned

Parallel Selection

- Q: What is the cost on a parallel database with P processors ?

- A: B(R) / P in all cases

- However, different processors do the work:
  - Round robin: all servers do the work
  - Hash: one server for $\sigma_{A=v}(R)$, all for $\sigma_{v1<A<v2}(R)$
  - Range: one server only

Data Partitioning Revisited

What are the pros and cons ?

- Round robin
  - Good load balance but always needs to read all the data
- Hash based partitioning
  - Good load balance but works only for equality predicates and full scans
- Range based partitioning
  - Works well for range predicates but can suffer from data skew

Parallel Group By

- Compute $\gamma_{A,\ sum(B)}(R)$
- Step 1: server i partitions chunk R_i using a hash function h(t.A) mod P: R_{i0}, R_{i1}, …, $R_{i,P-1}$
- Step 2: server i sends partition R_{ij} to serve j
- Step 3: server j computes $\gamma_{A,\ sum(B)}$ on R_{0j}, R_{1j}, …, $R_{P-1,j}$

Parallel Join

- Step 1
  - For all servers in [0,k], server i partitions chunk R_i using a hash function h(t.A) mod P: R_{i0}, R_{i1}, …, $R_{i,P-1}$
  - For all servers in [k+1,P], server j partitions chunk S_j using a hash function h(t.A) mod P: S_{j0}, S_{j1}, …, $R_{j,P-1}$

- Step 2:
  - Server i sends partition R_{iu} to server u
  - Server j sends partition S_{ju} to server u

- Steps 3: Server u computes the join of R_{iu} with S_{ju}

Parallel Dataflow Implementation

- Use relational operators unchanged
- Add special split and merge operators
  - Handle data routing, buffering, and flow control
- Example: exchange operator
  - Inserted between consecutive operators in the query plan
  - Can act as either a producer or consumer
  - Producer pulls data from operator and sends to n consumers
    - Producer acts as driver for operators below it in query plan
  - Consumer buffers input data from n producers and makes it available to operator through getNext interface

Map Reduce

- Google: paper published 2004
- Open source variant: Hadoop

- Map-reduce = high-level programming model and implementation for large-scale parallel data processing

- Competing alternatives include:
  - Dryad from Microsoft
  - Clustera from Wisconsin

Data Model

- Files !
- A file = a bag of (key, value) pairs
- A map-reduce program:
  - Input: a bag of (input key, value) pairs
  - Output: a bag of (output key, value) pairs

Step 1: the MAP Phase

- User provides the MAP-function:
  - Input: one (input key, value)
  - Ouput: a bag of (intermediate key, value) pairs

- System applies map function in parallel to all (input key, value) pairs in the input file

Step 2: the REDUCE Phase

- User provides the REDUCE function:
  - Input: intermediate key, and bag of values
  - Output: bag of output values

- System groups all pairs with the same intermediate key, and passes the bag of values to the REDUCE function

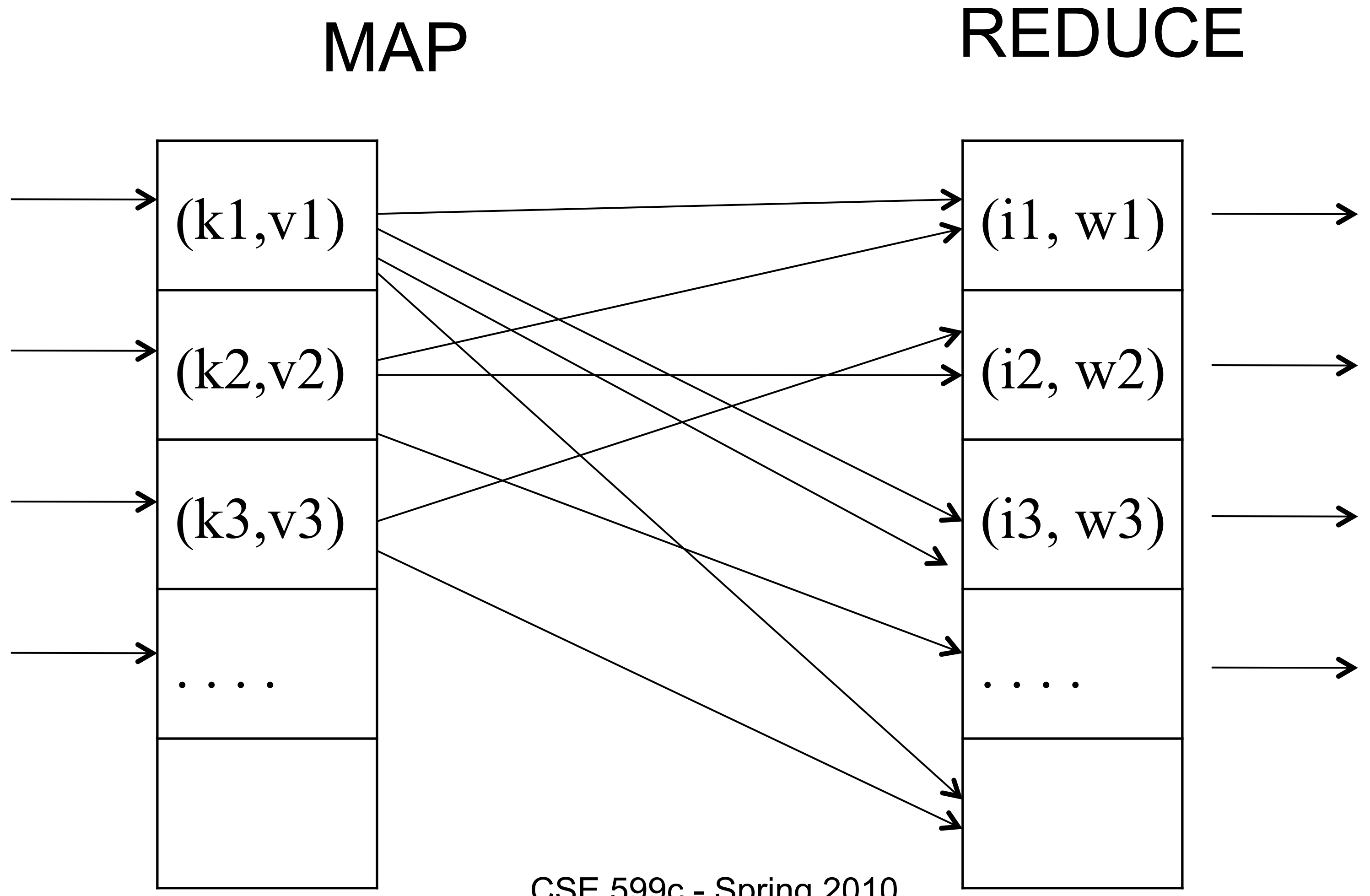
Example

- Counting the number of occurrences of each word in a large collection of documents

```
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1"):
```

```
reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));
```

MapReduce Execution



Map = GROUP BY, Reduce = Aggregate

R(documentKey, word)

```
SELECT word, sum(1)
FROM R
GROUP BY word
```

Example 2: MR word length count

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.

When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.

We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

Example 2: MR word length count

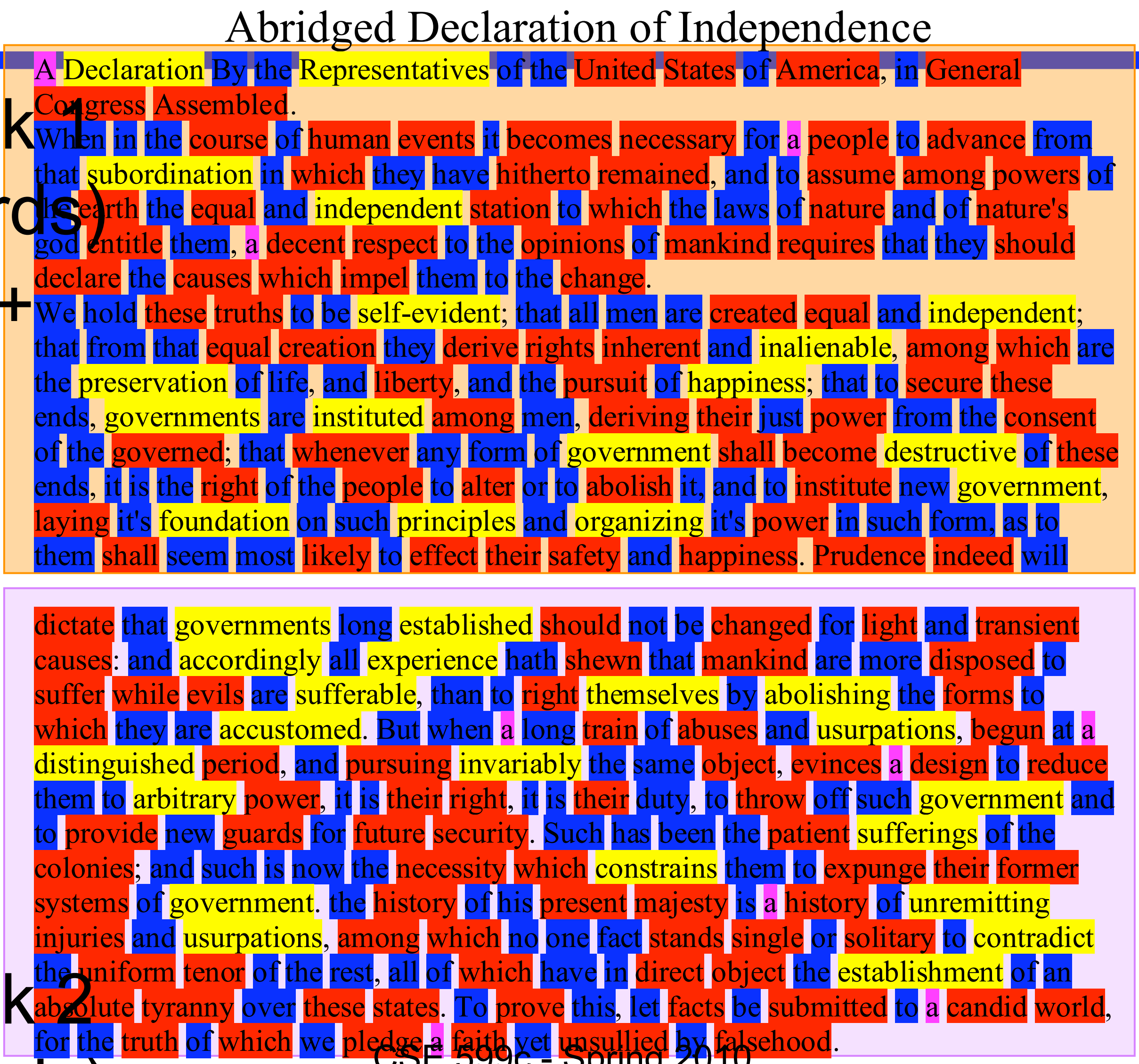
Map Task 1
(204 words)

Yellow: 10+

Red: 5..9

Blue: 2..4

Pink: = 1



Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.

When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.

We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will

dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

(key, value)

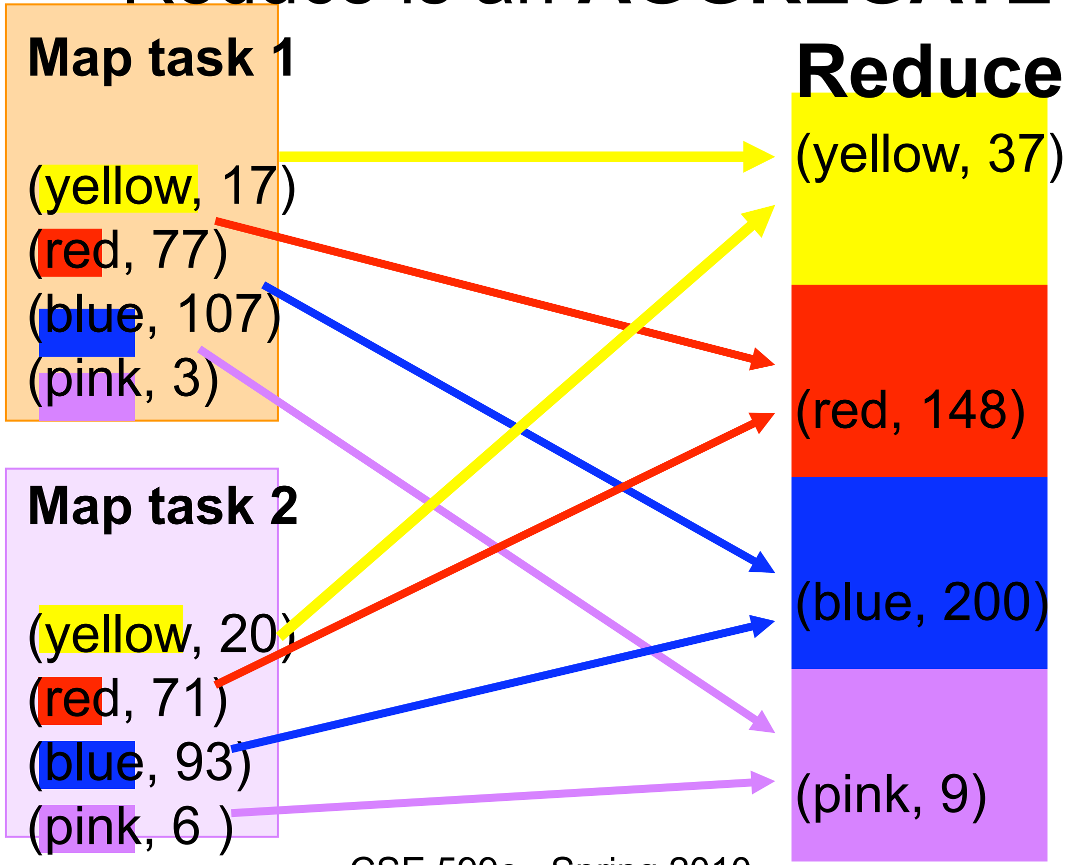
(yellow, 17)
(red, 77)
(blue, 107)
(pink, 3)

(yellow, 20)
(red, 71)
(blue, 93)
(pink, 6)

Map Task 2
(190 words)

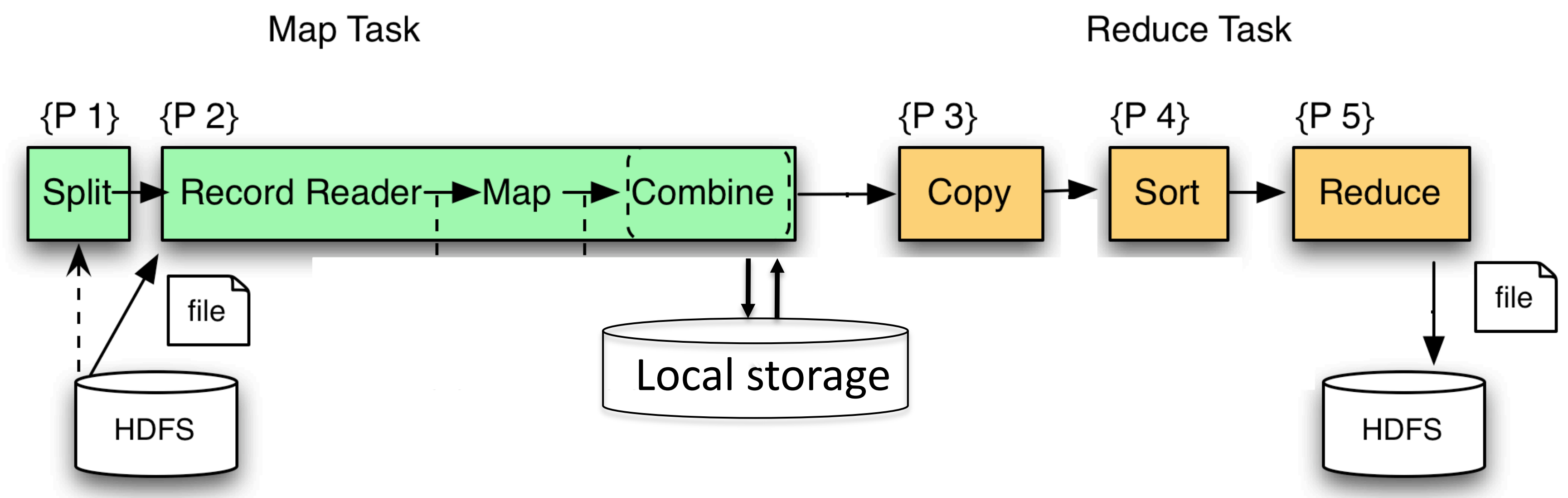
Example 2: MR word length count

Map is a **GROUP BY** operation
Reduce is an **AGGREGATE** operation



MR Phases

- Each Map and Reduce task has multiple phases:



Implementation

- There is one master node
- Master partitions input file into M *splits*, by key
- Master assigns workers (=servers) to the M map tasks, keeps track of their progress
- Workers write their output to local disk, partition into R regions
- Master assigns workers to the R reduce tasks
- Reduce workers read regions from the map workers' local disks

Interesting Implementation Details

- Worker failure:
  - Master pings workers periodically,
  - If down then reassigns the task to another worker

- Choice of M and R:
  - Larger is better for load balancing
  - Limitation: master needs O(M×R) memory

Interesting Implementation Details

- Backup tasks:
  - “Straggler” = a machine that takes unusually long time to complete one of the last tasks. Eg:
    - Bad disk forces frequent correctable errors (30MB/s → 1MB/s)
    - The cluster scheduler has scheduled other tasks on that machine
  - Stragglers are a main reason for slowdown
  - Solution: pre-emptive backup execution of the last few remaining in-progress tasks

Map-Reduce Summary

- Hides scheduling and parallelization details
- However, very limited queries
  - Difficult to write more complex tasks
  - Need multiple map-reduce operations
- Solution: more general query languages:
  - PIG Latin (Y!): its own language, freely available
  - Scope (MS): SQL ! But proprietary…
  - DryadLINQ (MS): LINQ ! But also proprietary…
  - Clustera (other UW) : SQL ! Not publicly available

MapReduce (MR) tools

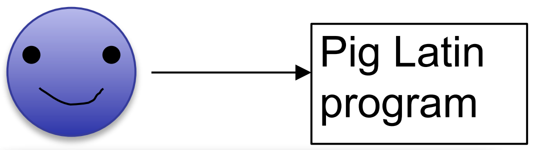
MR implementation: hadoop

One MR query language: ***Pig Latin***

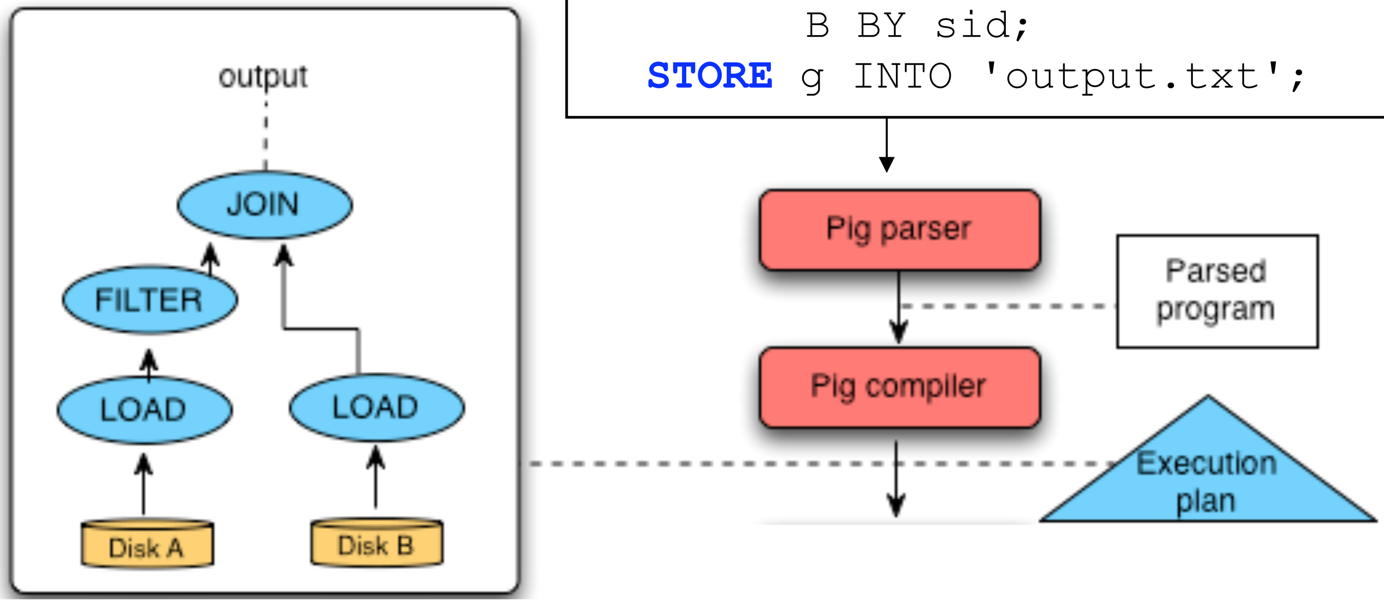
Query engine: pig

Graphics taken from: hadoop.apache.org and research.yahoo.com/node/90

Background: Pig system



```
A = LOAD 'file1' AS (sid,pid,mass,px:double);
B = LOAD 'file2' AS (sid,pid,mass,px:double);
C = FILTER A BY px < 1.0;
D = JOIN C BY sid,
         B BY sid;
  STORE g INTO 'output.txt';
```



Applying MapReduce to Science

- What are the potential benefits of applying parallel dbs or MapReduce to science?
- What are the potential challenges?

CloudBurst Overview

- DNA sequencing machines produce lots of data
- After sequencing DNA, researchers often map the reads to a reference genome to find the locations where each read occurs
  - Output: for each read, one or more alignments
    - Alignment is valid if nb errors (mismatch, insert, delete) < threshold
  - Algorithm: seed-and-extend

CloudBurst Map Phase

- Input: Two datasets split into chunks
  - Reads from sequencing machine and reference genome
  - Schema: (id, SeqInfo), where SeqInfo is (sequence, start_offset)
- Configuration parameters
  - Max nb of mismatches k
  - Minimum length of reads m
  - Seed size: s = m/(k+1)
- Output
  - For each input sequence, emit a set of (seed, MerInfo)
  - Where seed is a sequence of length s
  - MerInfo is tuple (id, position, isRef, isRC, left_flank, right_flank)

CloudBurst Reduce Phase

- Input: (seed, sequence of MerInfo)
  - Recall that MerInfo comes either from reads or from reference
  - So there are two sets of MerInfo: R and Q
- Goal:
  - Extend the exact alignment seeds into longer inexact alignments
  - Need to process the Cartesian product RxQ
  - For each alignment found, check and eliminate duplicates
- Output:
  - File with every alignment of every read with at most k mismatches

Results

- See Figures 3 through 5

Benefits and Challenges

- Does it work well?
- Is this useful?
- Any problems with this approach?
- Is this a significant contribution?

SkewReduce

- Goal: Help scientists express their analysis in a way that leads to an efficient execution in a share-nothing cluster

- Focus: Feature-extracting analysis functions
  - Input: points in a multi-dimensional space (or a multi-d array)
  - Processing: identify and extract features (e.g., clusters)
  - Outputs:
    - The list of features
    - The input data annotated with these features

Example Applications

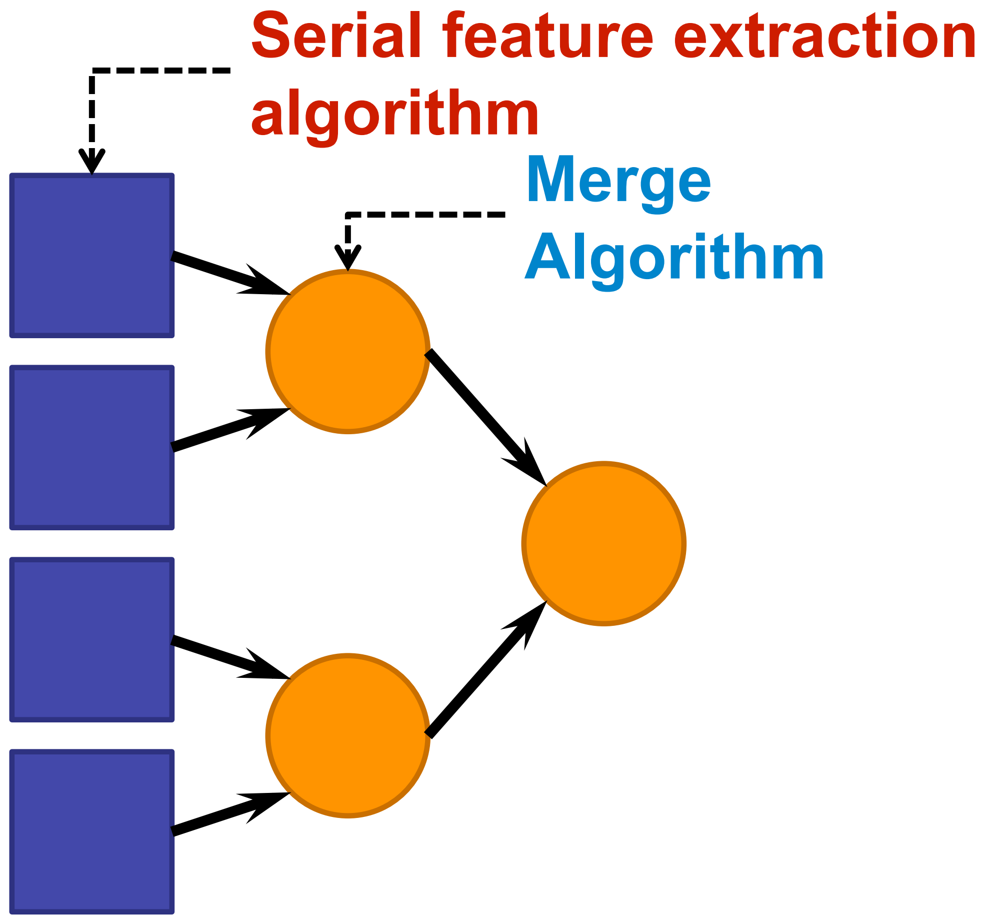
- Data clustering
  - Find galaxies in a 3D snapshot of a simulated universe
  - Find families of organisms in a 6D space of measured properties
- Image processing
  - Find flocks of birds, hurricanes, stars, etc. in 2D images

Basic Approach

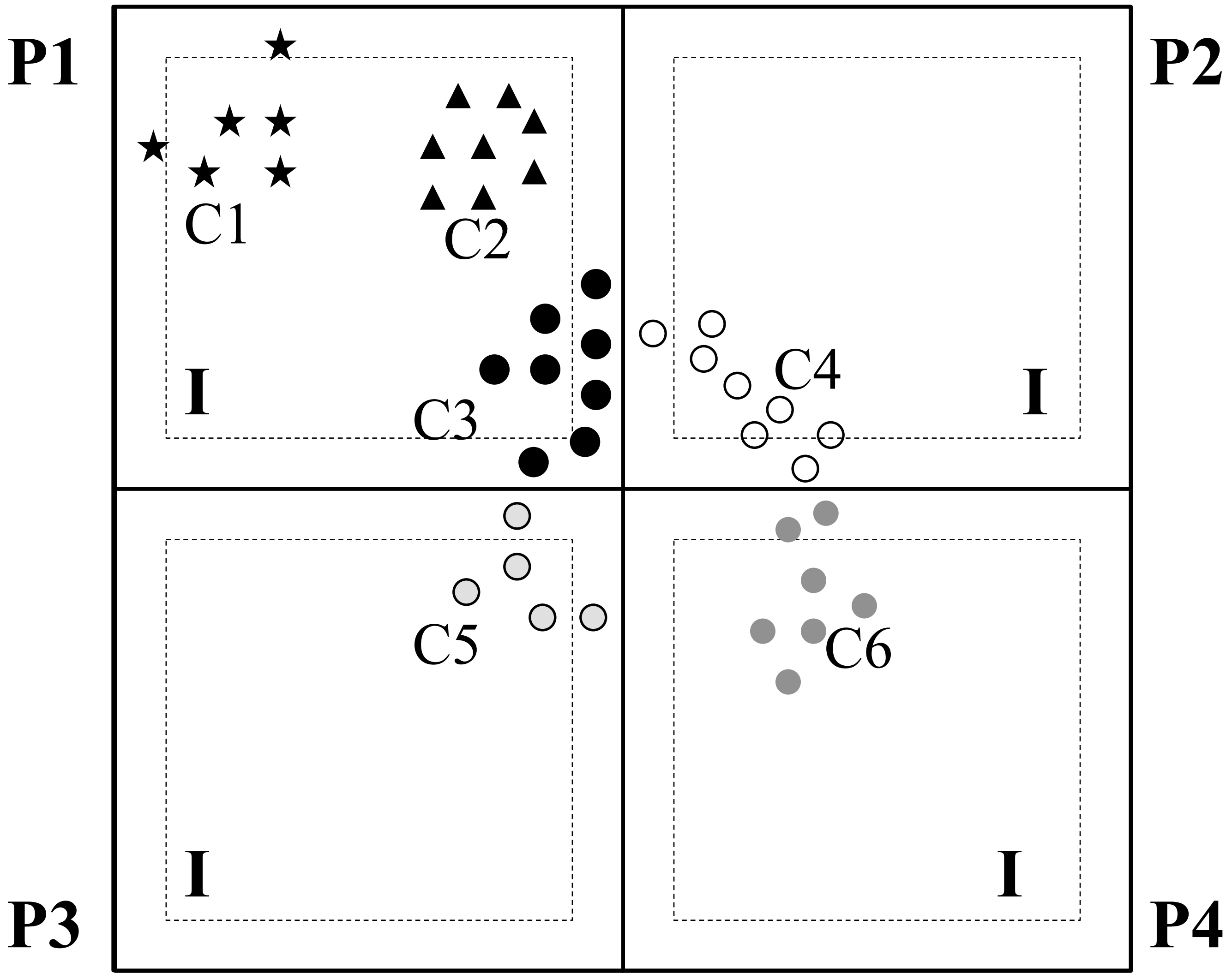
Can parallelize feature extracting functions as follows

- Step 1: Split input into N equal-sized hypercubes
- Step 2: Extract features in each partition separately
  - And annotate the input data with these features
- Step 3: Reconcile features that span partition boundaries
- Step 4: Re-label the input data with the final feature IDs

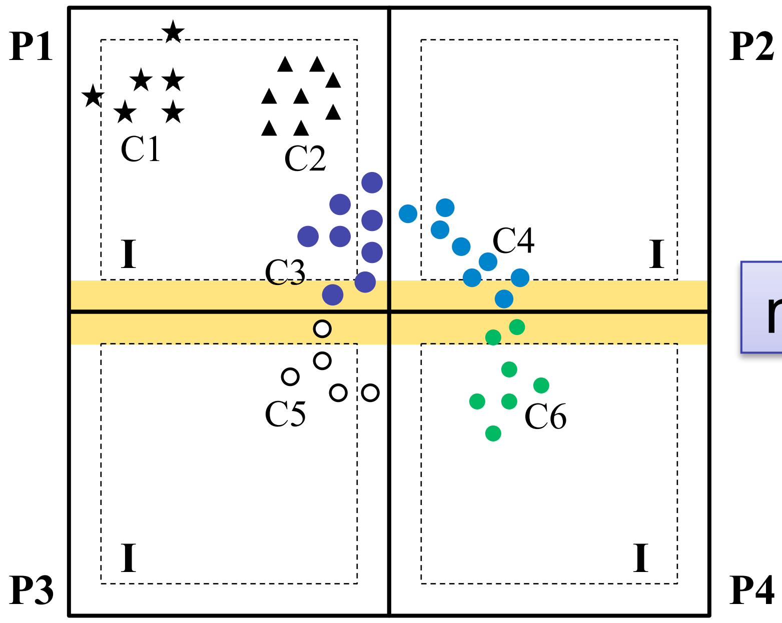
Basic Approach



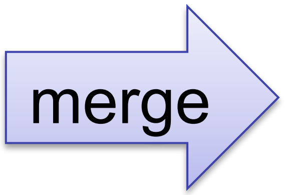
Example: Friends of Friends

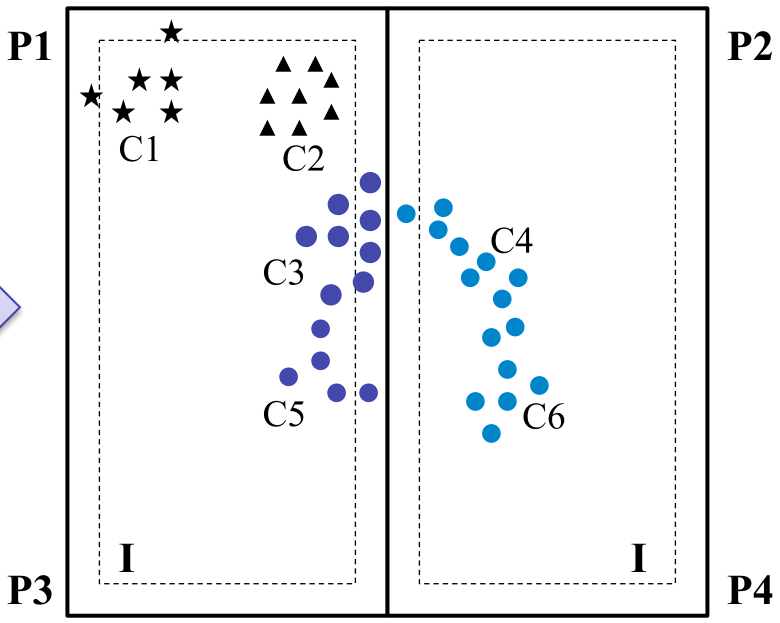


Example: Friends of Friends



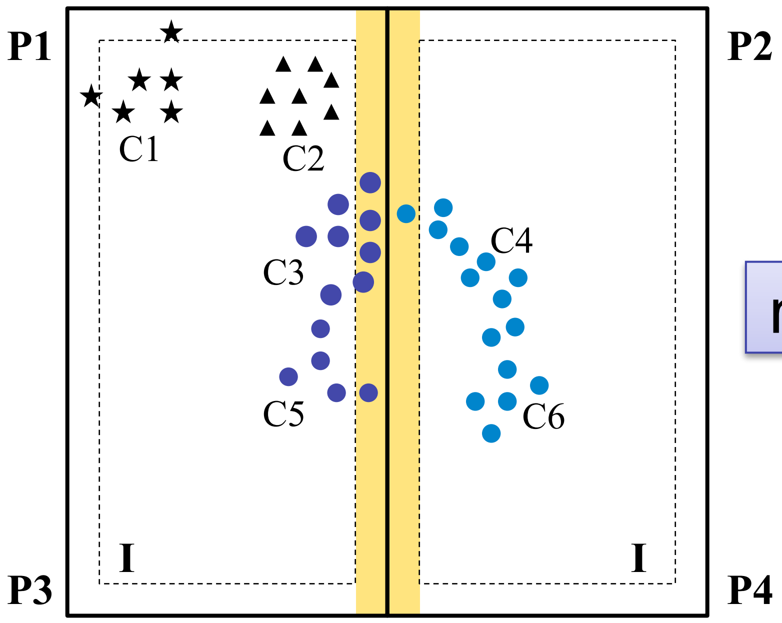
Merge P1, P3
Merge P2, P4





C5 → C3
C6 → C4

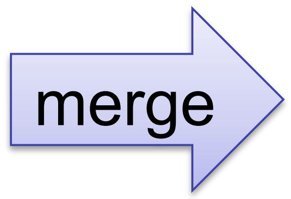
Example: Friends of Friends

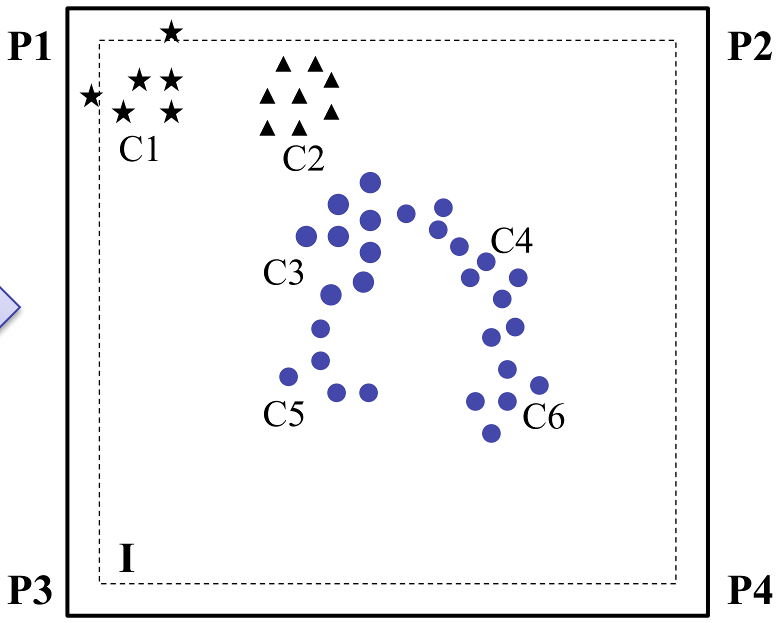


Merge P1-P3, P2-P4

C5 → C3

C6 → C4





C4 → C3

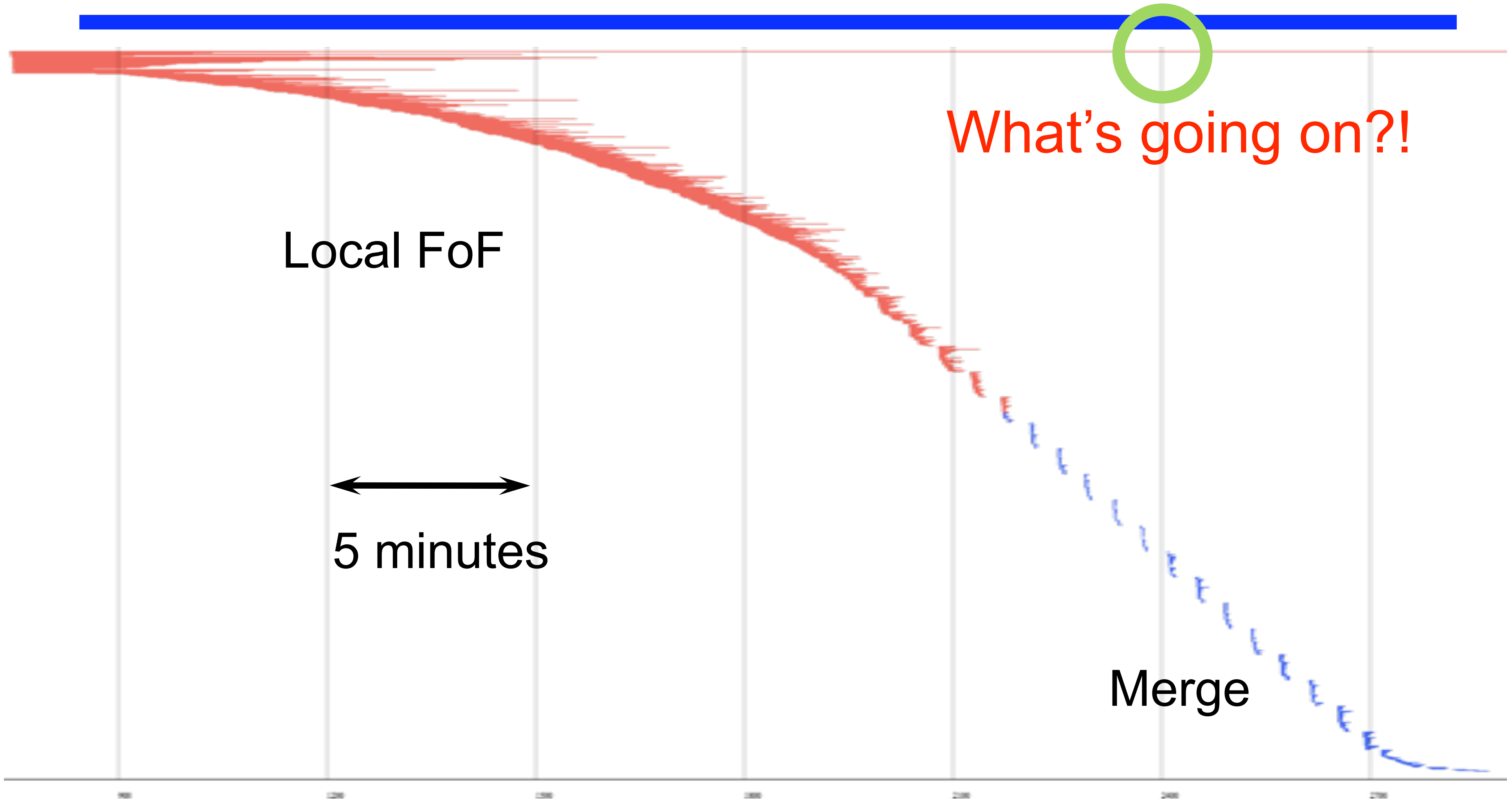
C5 → C3

C6 → C3

Challenge

- A naïve implementation can lead to an extremely inefficient execution and can even fail!
- Why? Because of computation skew
  - Some partitions take much longer than others
- For one astronomy dataset with 43 GB: 20 hours!
- Optimized implementation: 70 minutes!
  - But took a few weeks to develop, which is not scalable!

Example: Unbalanced Computation

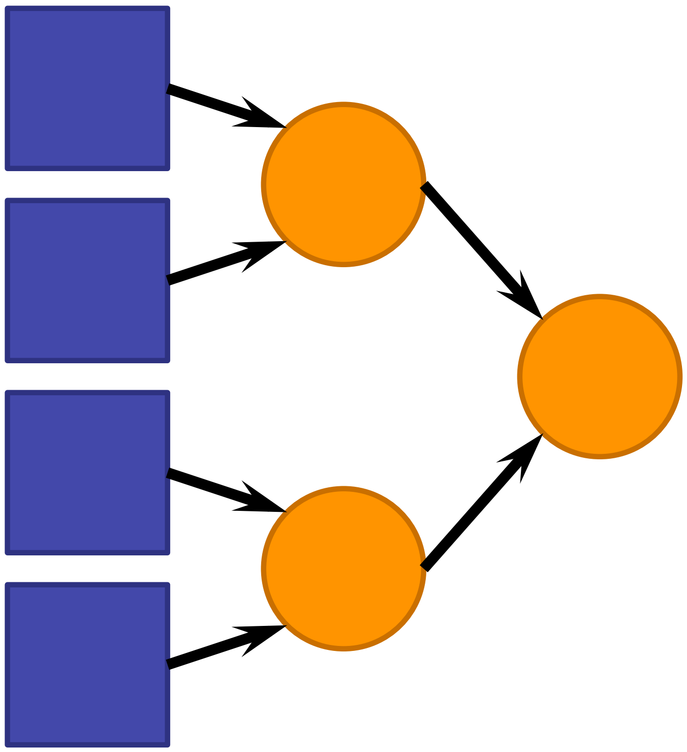


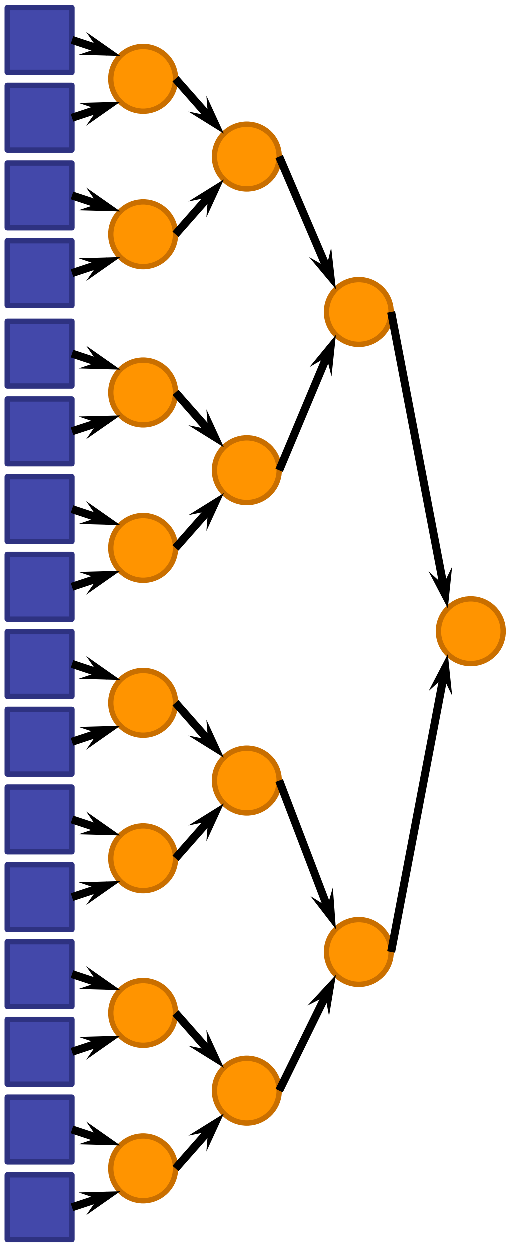
- The top red line runs for 1.5 hours

Unbalanced Computation: Skew

- A few partitions are slow
  - All other partitions are waiting for the slow ones
- MapReduce: Speculative execution
  - Effective for machine skew
  - What if faster machine can’t help?
- Spatial analysis can exhibit significant skew
  - Density varies wildly: Stars, Galaxies, Cities, …

How About Micro Partitions?

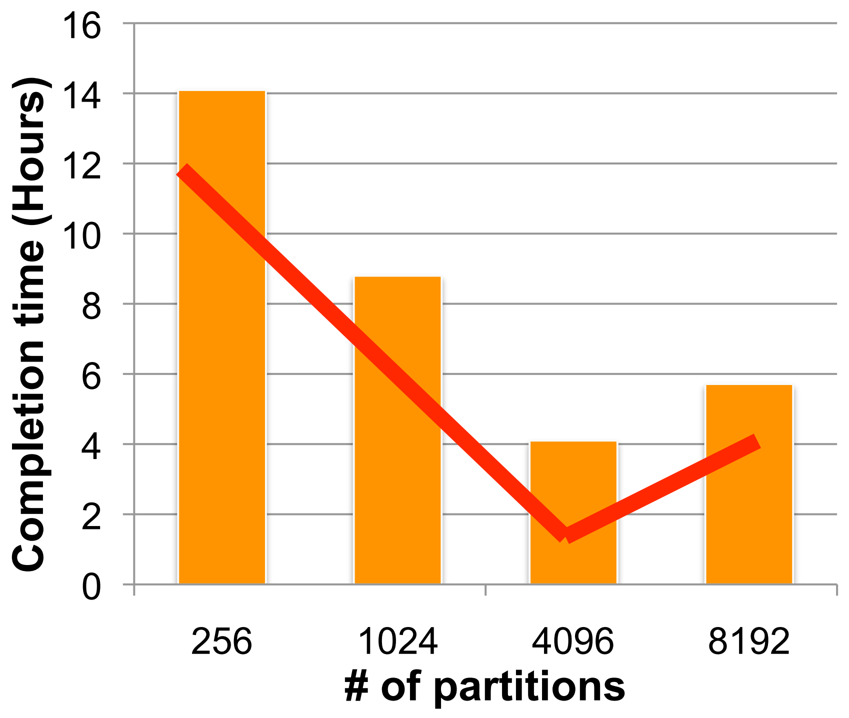




How about having micro partitions?

- Super fine grain partitions
  - Less data = Less skew
  - But framework overhead!

- Finding sweet spot is time consuming
- No guarantee of successful merge

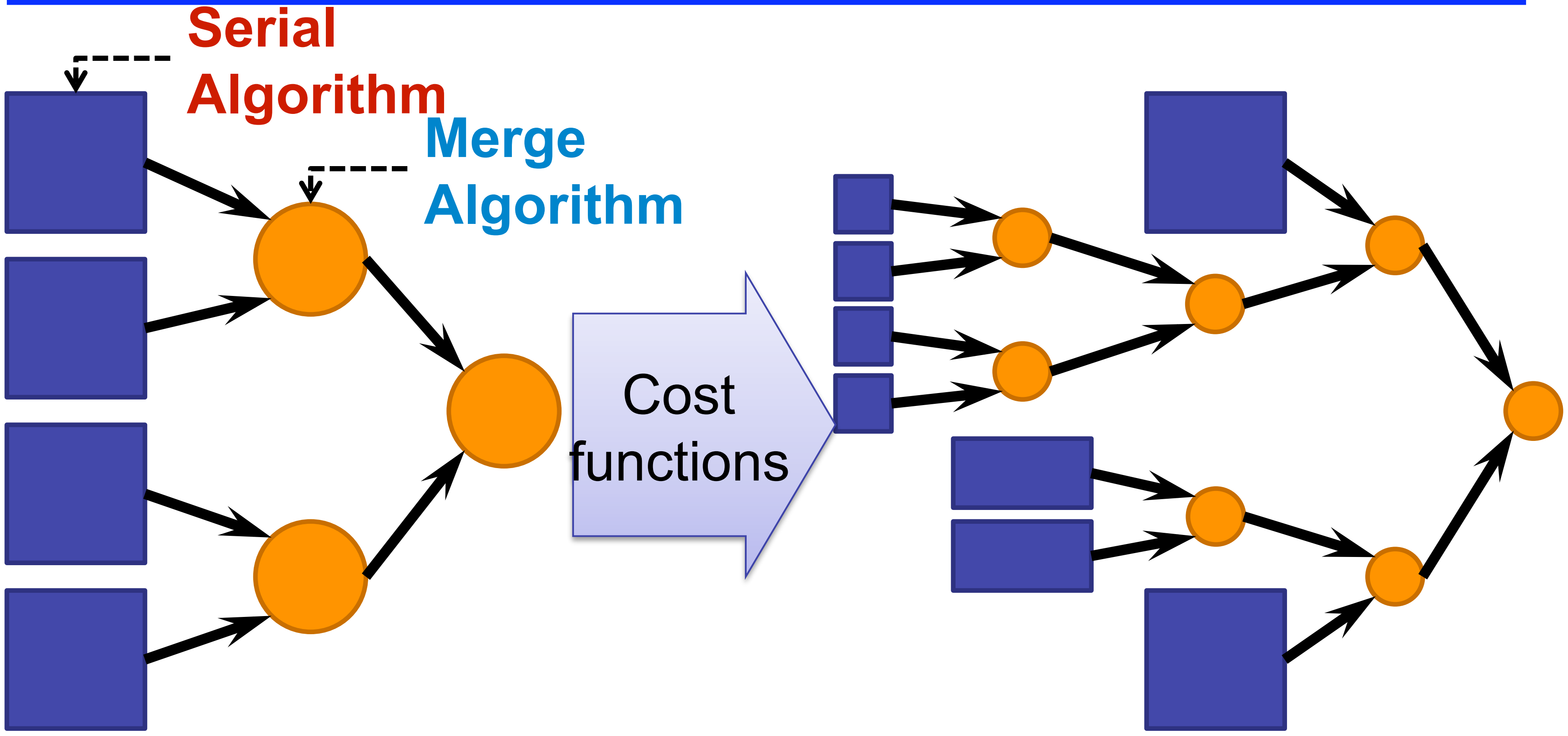


Can we find a good partitioning plan without trial and error?

SkewReduce

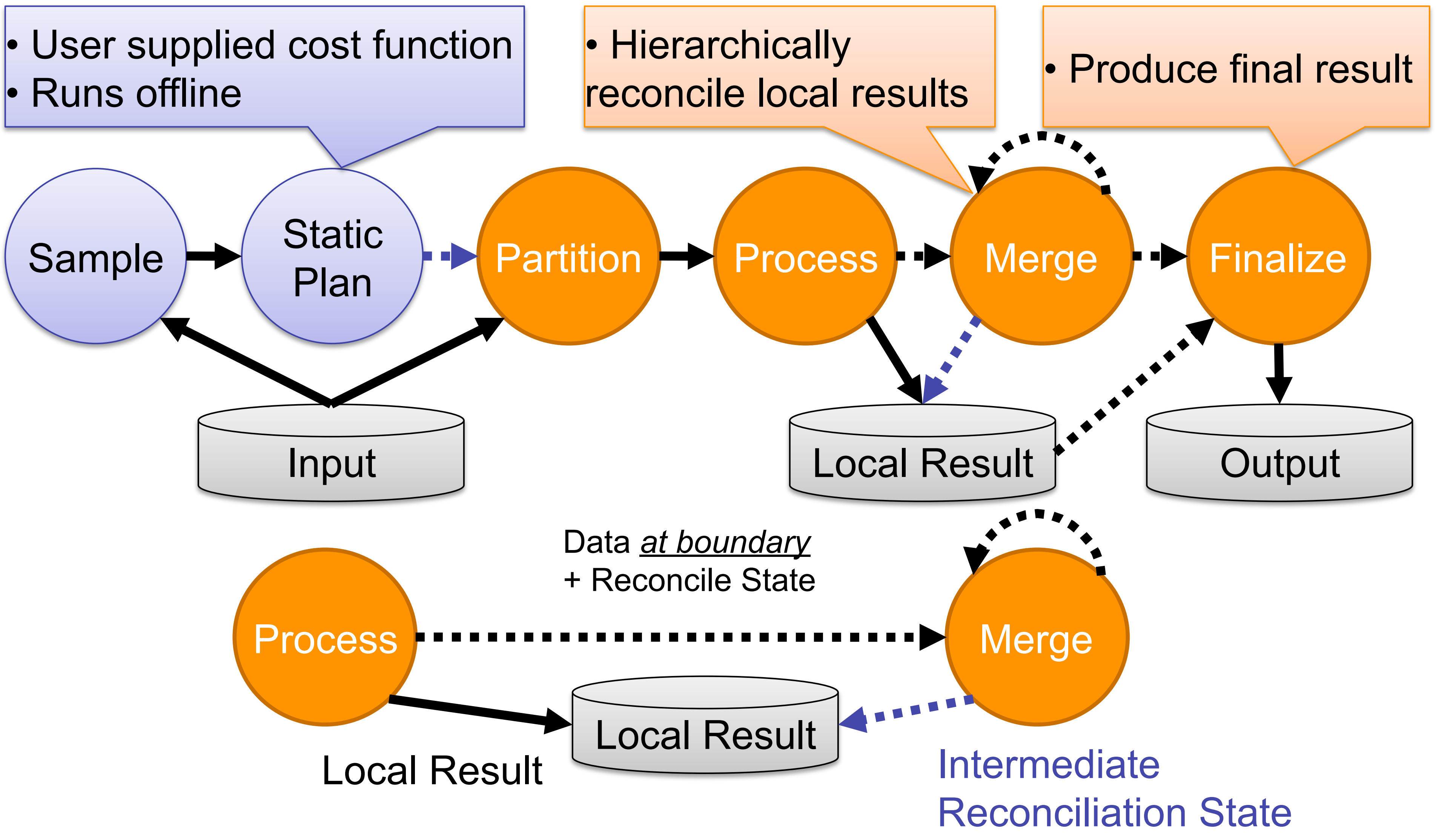
- Starts with a simple API
  - Process :: < Seq of T > -> <F, Seq of S>
  - Merge :: < F, F > -> <F, Seq of S>
  - Finalize :: <F, S> -> <Seq of Z>
- Add to this two cost functions
  - C_p :: (S, alpha, B) -> R
  - C_m :: (S, alpha, B) x (S, alpha, B) -> R

SkewReduce Approach



- Two algorithms: Serial/Merge algorithm
- Two cost functions for each algorithm
- Find a good partition plan and schedule

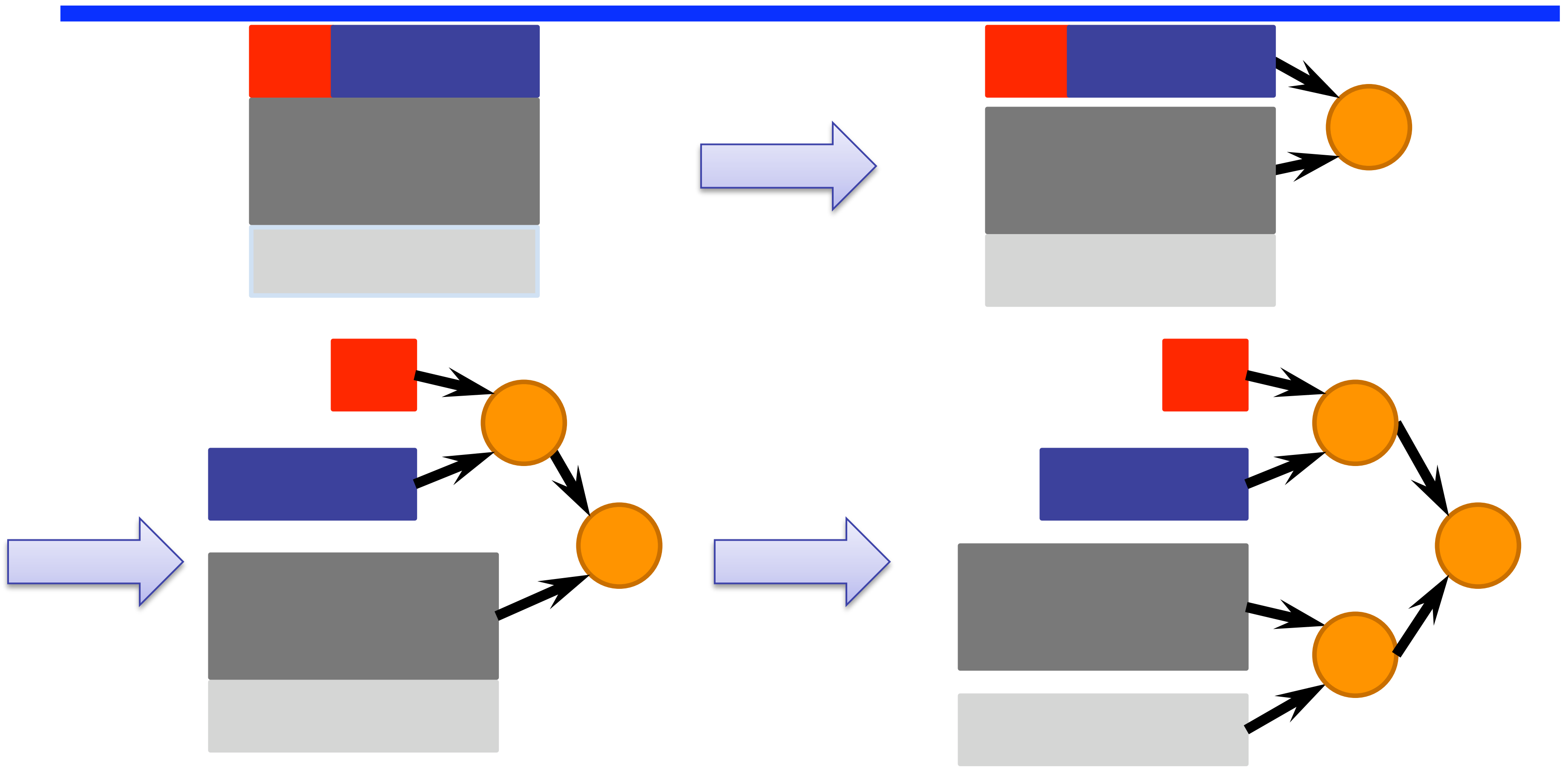
Skew Reduce Framework



SkewReduce: Approach

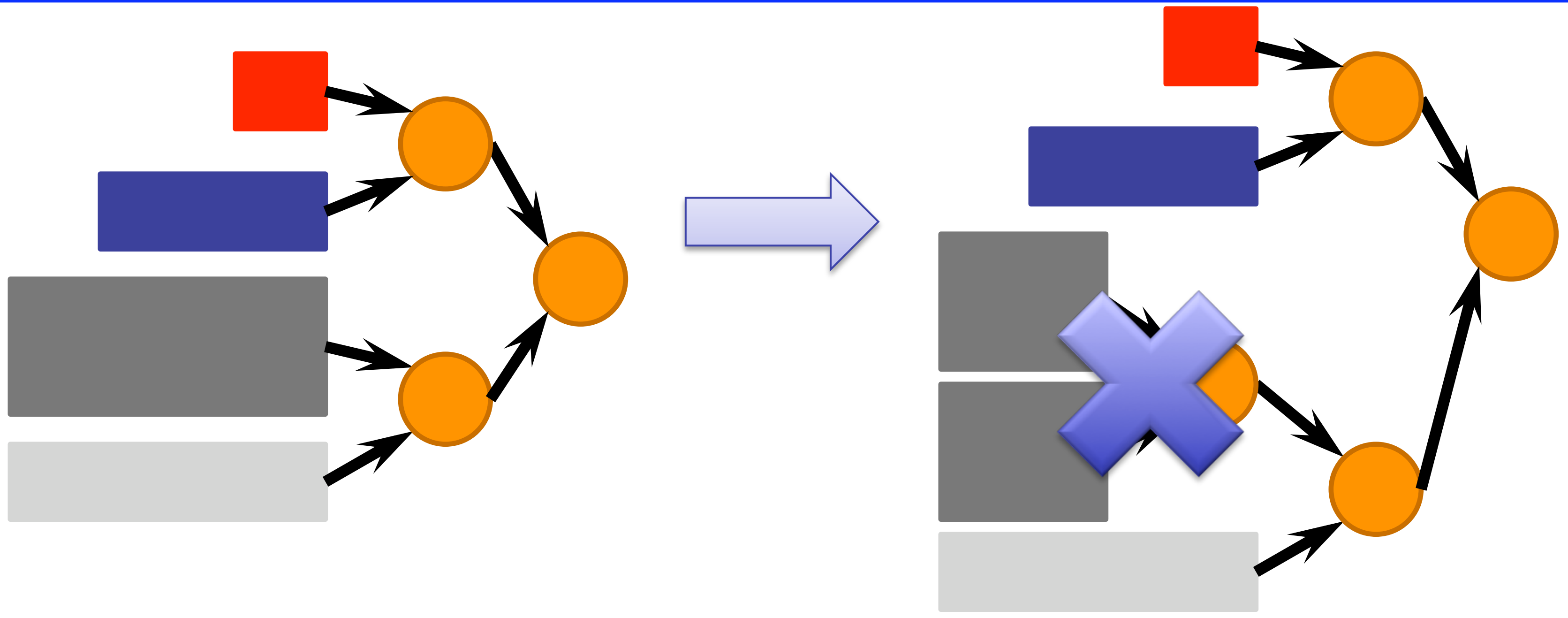
- How to decompose space?
  - Bisect space such that subspaces have **identical cost**
- How to schedule?
  - In decreasing order of cost (Longest Processing Time)
- How to prevent Out of Memory?
  - Estimate the size of input data
  - ***Setaside*** unnecessary data before merge

Static Plan



- Search for both partitioning axes and point
  - Guided by cost functions
  - Two subpartitions have roughly the same cost

Static Plan

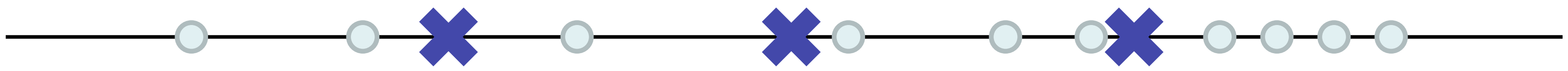


- Partition stop when there is no benefit
  - Partitioning is not free

Search Optimal Split

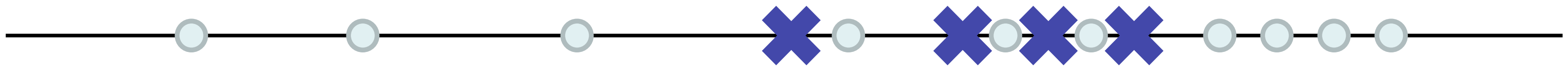
- Three strategies to find optimal split point
  - Find a spot where the costs are identical

Sampling



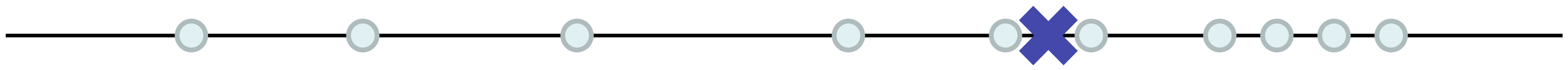
6:4. 3 cost evaluations

Binary Search



5:5. 4 cost evaluations

Incremental Update



5:5. 1 cost evaluation

Check improvement

- Partitioning is not free
  - Two subpartitions + Merge < Original partition
- Accept new partitions if
  - Original partition includes too much data
  - Expected runtime improves with respect to task scheduling algorithm
    - Longest Run Time algorithm by default

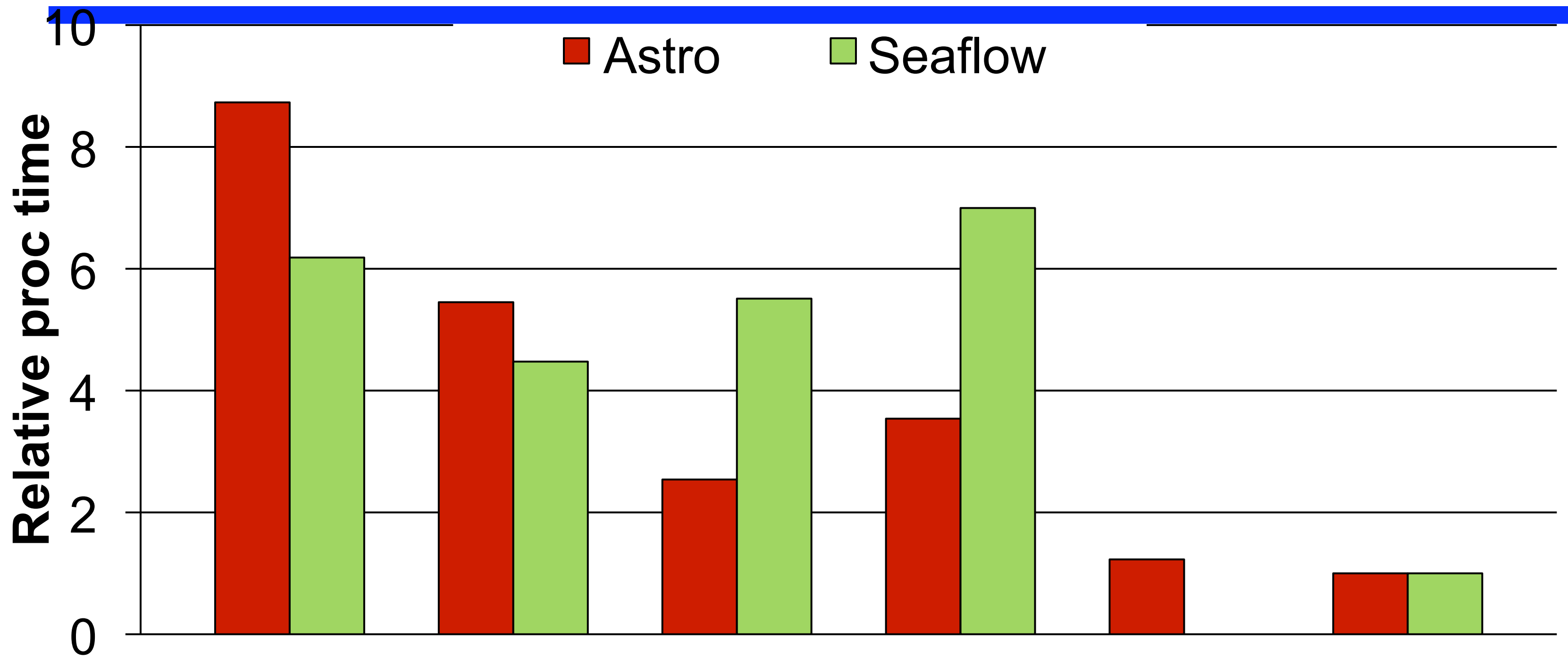
Static Plan: Summary

- Goal
  - Find a good partition plan and corresponding schedule with respect to cost functions and input data sample
- Approach
  - Greedily search best splits
  - Only accept partitions if runtime improves
  - Run offline using a data sample

Evaluation

- 8 node cluster
  - Dual quad core, 16 GB RAM
  - Hadoop 0.20.1 + custom patch in MapReduce API
- Dataset
  - Astro: simulation snapshot at step 92
  - Seaflow: flow cytometry survey

Does SkewReduce work?



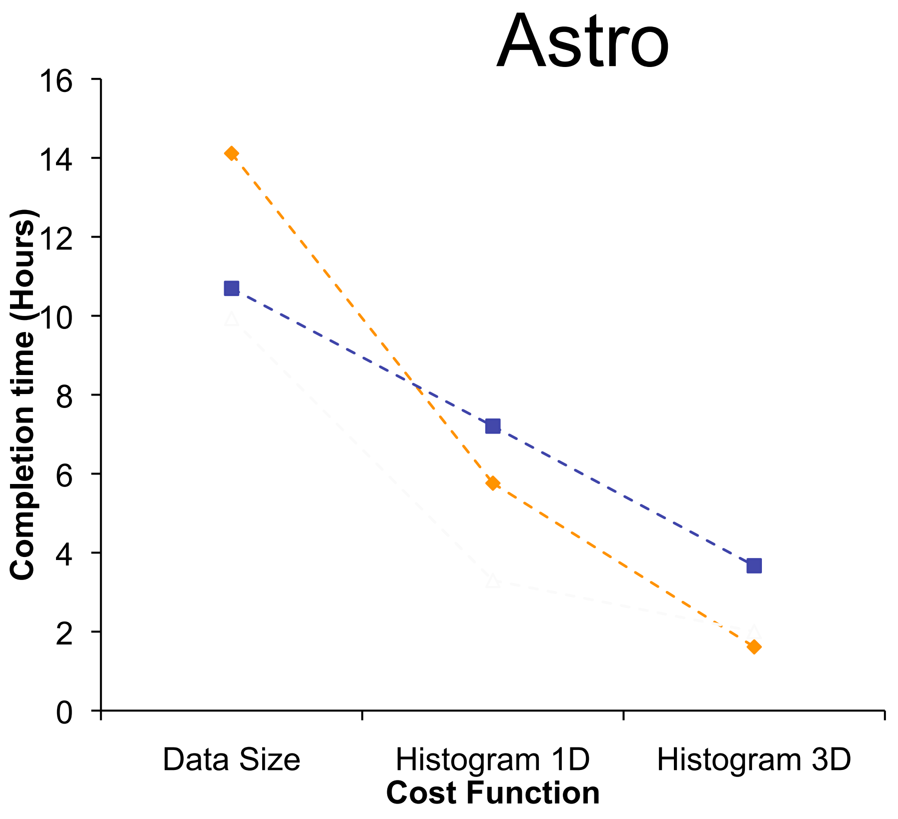
	Coarse	Fine	Finer	Finest	Manual	Opt
Hours	14.1	8.8	4.1	5.7	2.0	1.6
Minutes	87.2	63.1	77.7	98.7	-	14.1

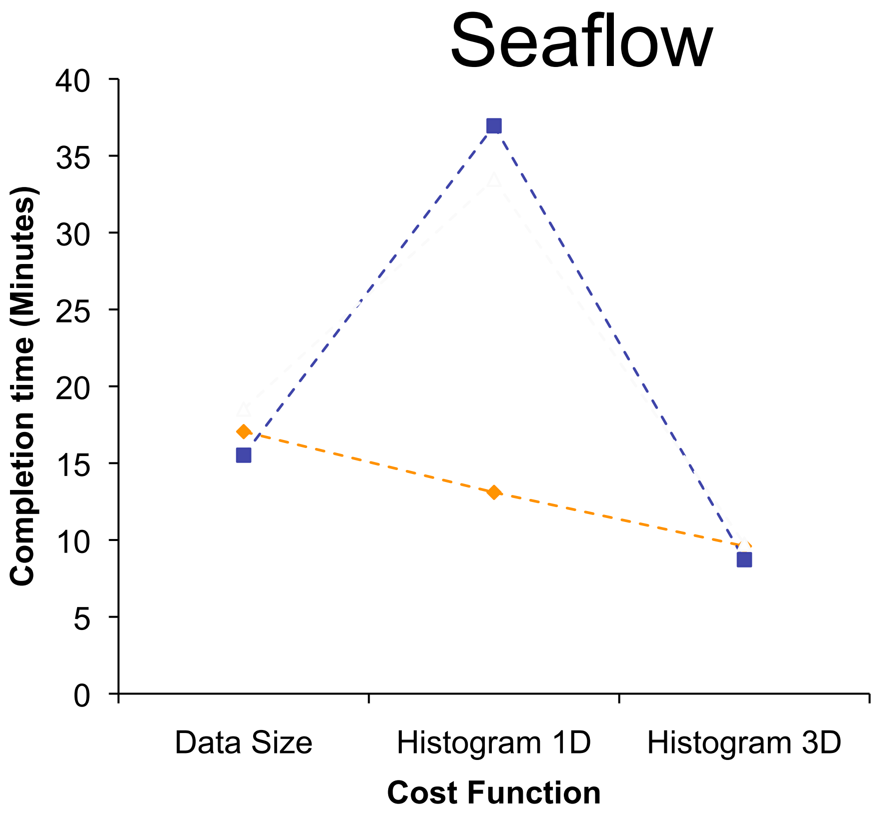
- Static plan yields 2 ~ 8 times faster running time

Cost Functions

- Data Size
  - the number of data items in a partition
- Histogram 3D
  - Model spatial index traversal pattern
  - Construct equi-width 3D histogram
  - Cost = sum of square of frequencies
- Histogram 1D
  - 1D version of Histogram 3D

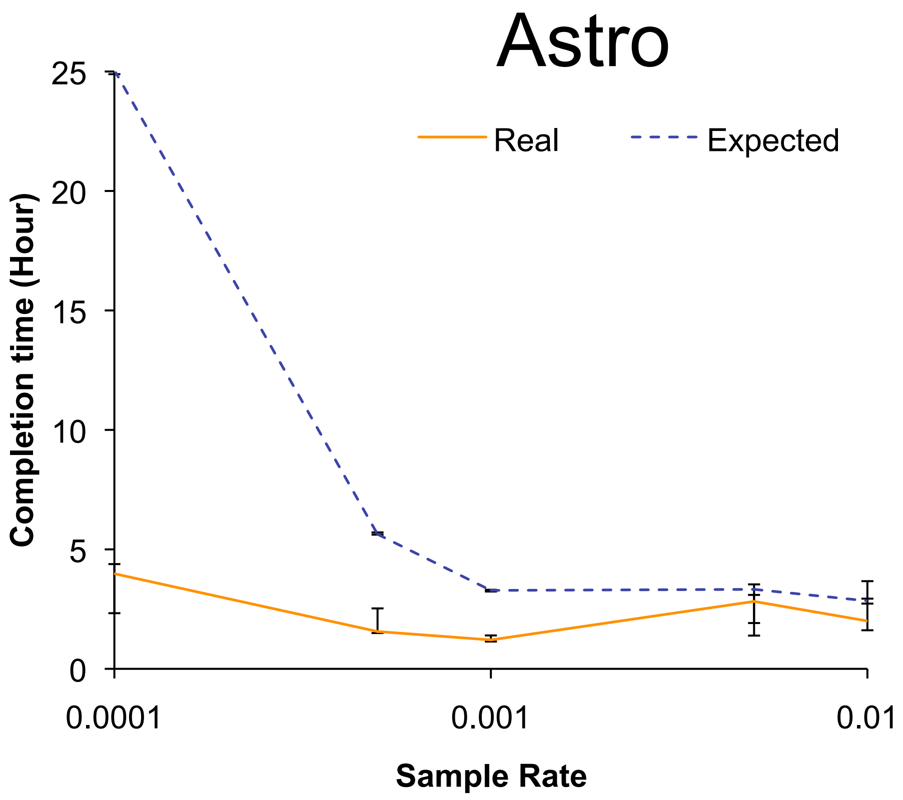
Fidelity of Cost Functions

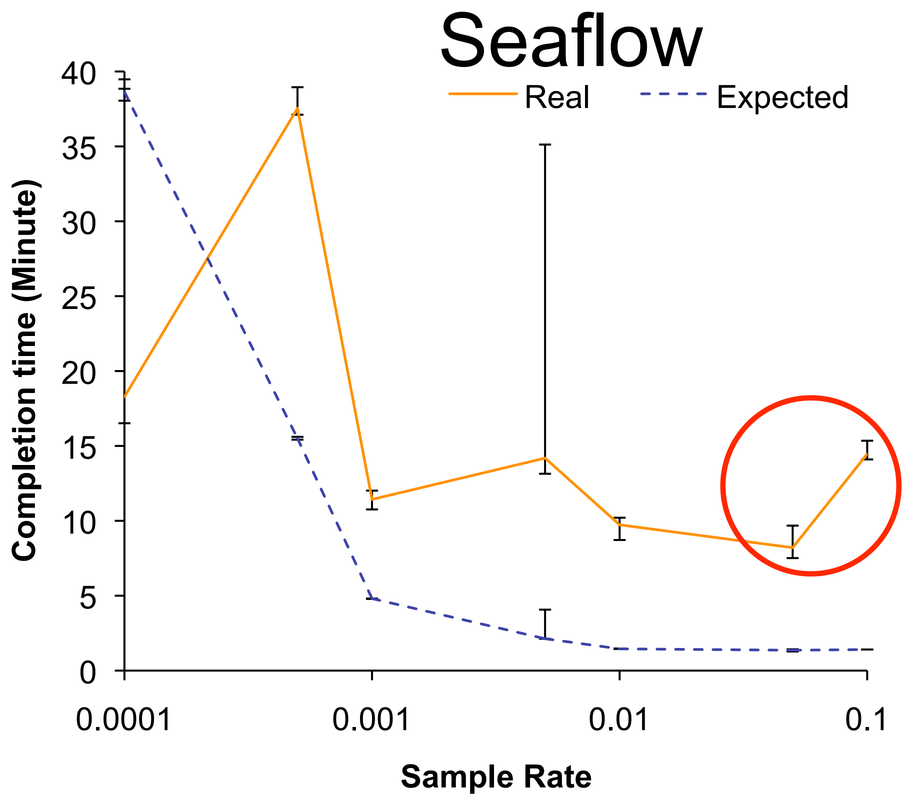




- Higher fidelity = Better performance
- Seaflow -- overestimation

Quality of Sample





- Varied sample rate
- Representativeness of sample affect the performance

Benefits and Limitations

- Is SkewReduce useful?
- Is this the right strategy?
- What are the limitations?
- Could this approach be generalized?