Big Data Analytics

徐 天棋
13M54027

Starfish: A Self-tuning System for Big Data Analytics

Herodotos Herodotou, Harold Lim, Gang Luo, Nedyalko Borisov, Liang Dong,
Fatma Bilgen Cetin, Shivnath Babu
Department of Computer Science
Duke University

OUTLINE

1. Introduction
2. MapReduce
3. Overview of Starfish
4. Just-In-Time Job Optimization
5. Workflow-Aware Scheduling
6. Optimization and Provisioning for Hadoop Workloads
7. Summary
8. Starfish’s Visualizer
9. My Impression

INTRODUCTION

Timely and cost-effective analytics over “Big Data” has emerged as a key ingredient for success

- business
- science
- Engineering
- government

MAD SYSTEM

- **Magnetism:**
  A magnetic system attracts all sources of data irrespective of issues like possible presence of outliers, unknown schema or lack of structure, and missing values that keep many useful data sources out of conventional data warehouses.
- **Agility:**
  An agile system adapts in sync with rapid data evolution.
- **Depth:**
  A deep system supports analytics needs that go far beyond conventional rollups and drilldowns to complex statistical and machine-learning analysis.

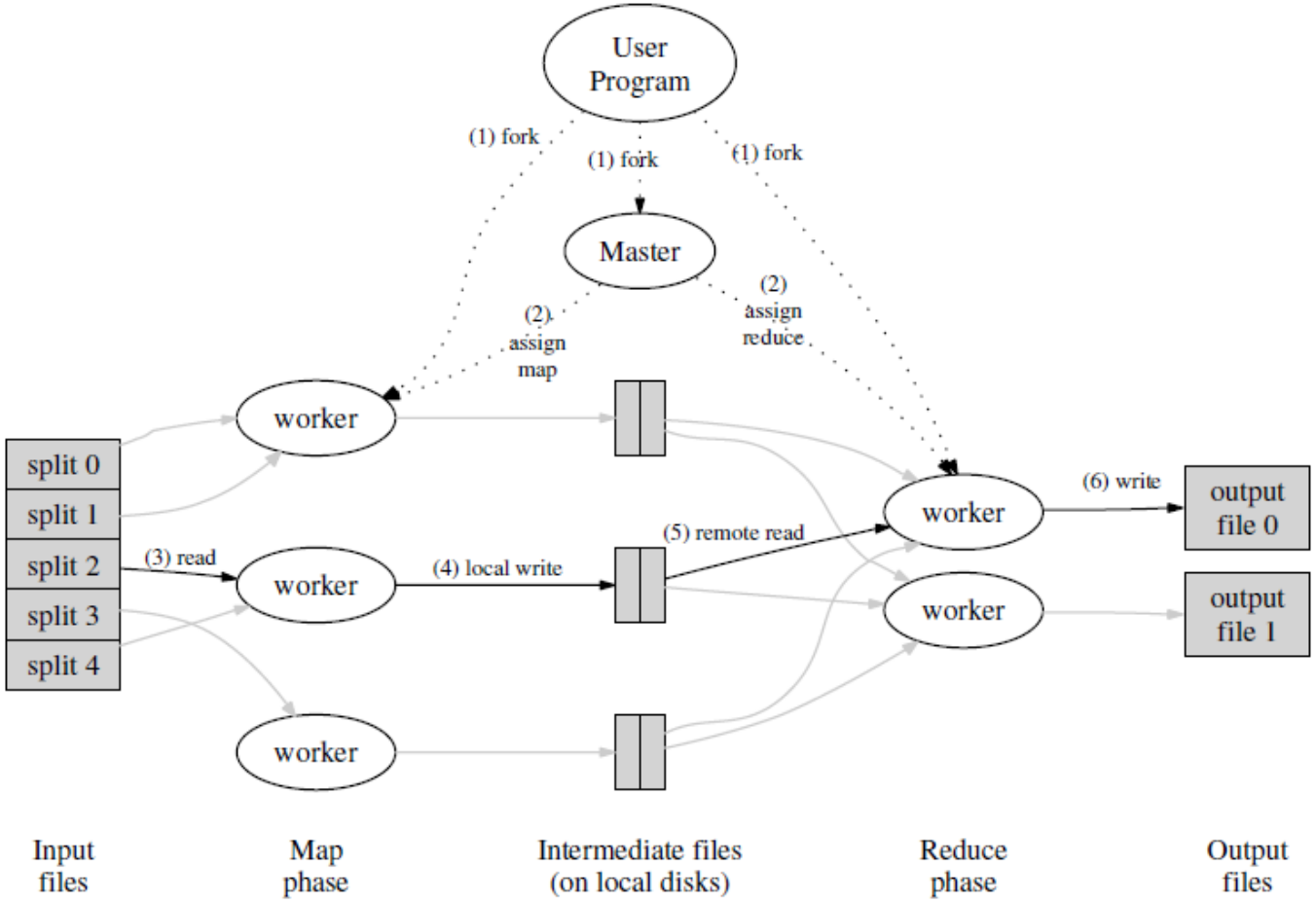
HADOOP'S MADness

- Copying files into the distributed filesystem is all it takes to get data into Hadoop.
- The MapReduce methodology is to interpret data (lazily) at processing time, and not (eagerly) at loading time.
- MapReduce computations in Hadoop can be expressed directly in programming languages
  - general-purpose programming languages: Java, Python
  - domain-specific languages: R
  - SQL-like declarative languages: HiveQL, Pig Latin

MADDER SYSTEM

- ***Data-lifecycle awareness***
  A data-lifecycle-aware system goes beyond query execution to optimize the movement, storage, and processing of big data during its entire lifecycle.
- ***Elasticity***
  An elastic system adjusts its resource usage and operational costs to the workload and user requirements.
- ***Robustness***
  A robust system continues to provide service in the face of undesired events like hardware failures, software bugs, and data corruption.

MAPREDUCE

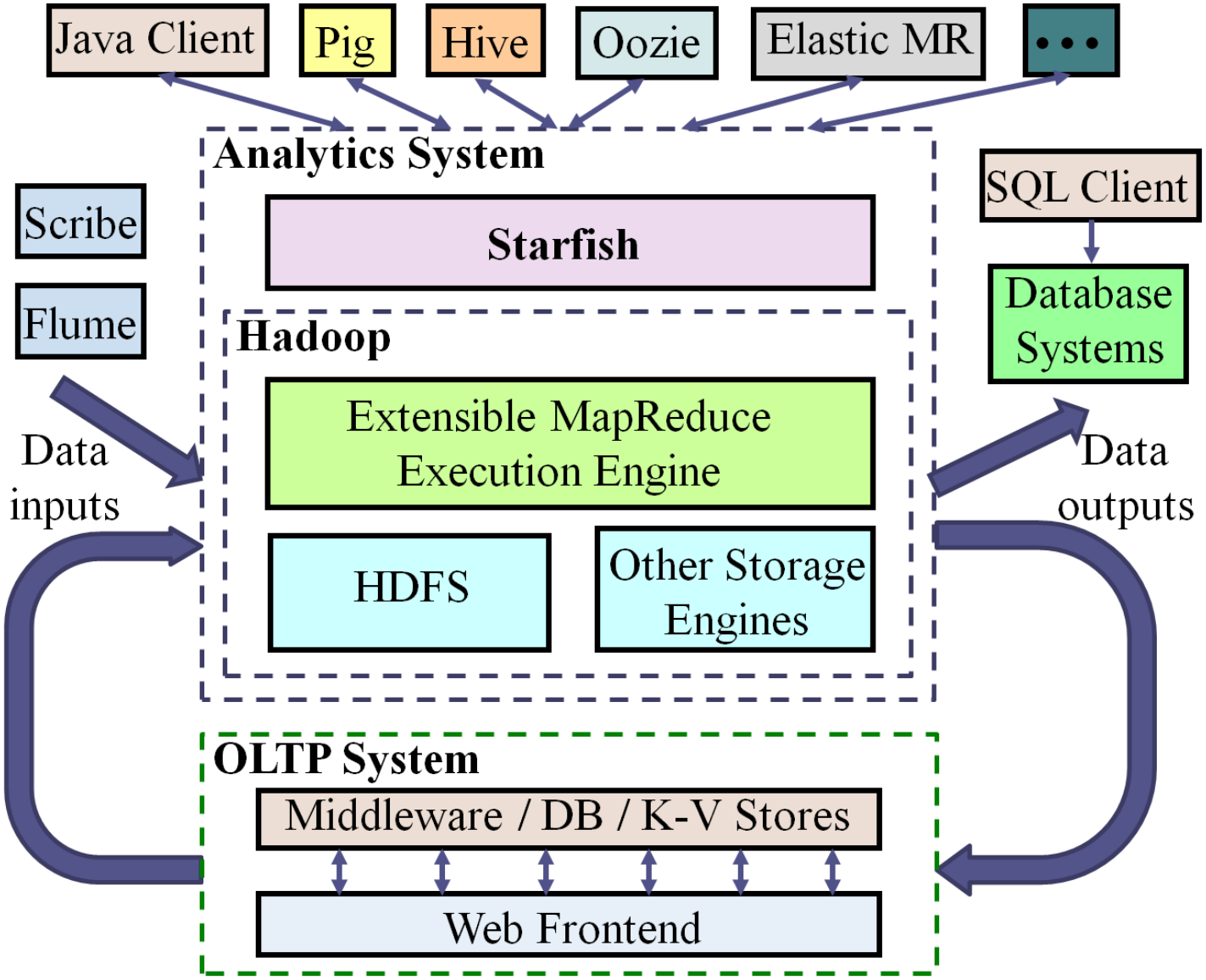

User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

MAP SHUFFLE AND REDUCE

- Map： takes an input pair and produces a set of intermediate key/value pairs.
- Shuffle： assign each key/value pair to a reduce worker.
- Reduce： accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values.

STARFISH IN THE HADOOP ECOSYSTEM

Starfish is a MADDER and self-tuning system for analytics on big data.
An important design is to build Starfish on the Hadoop stack



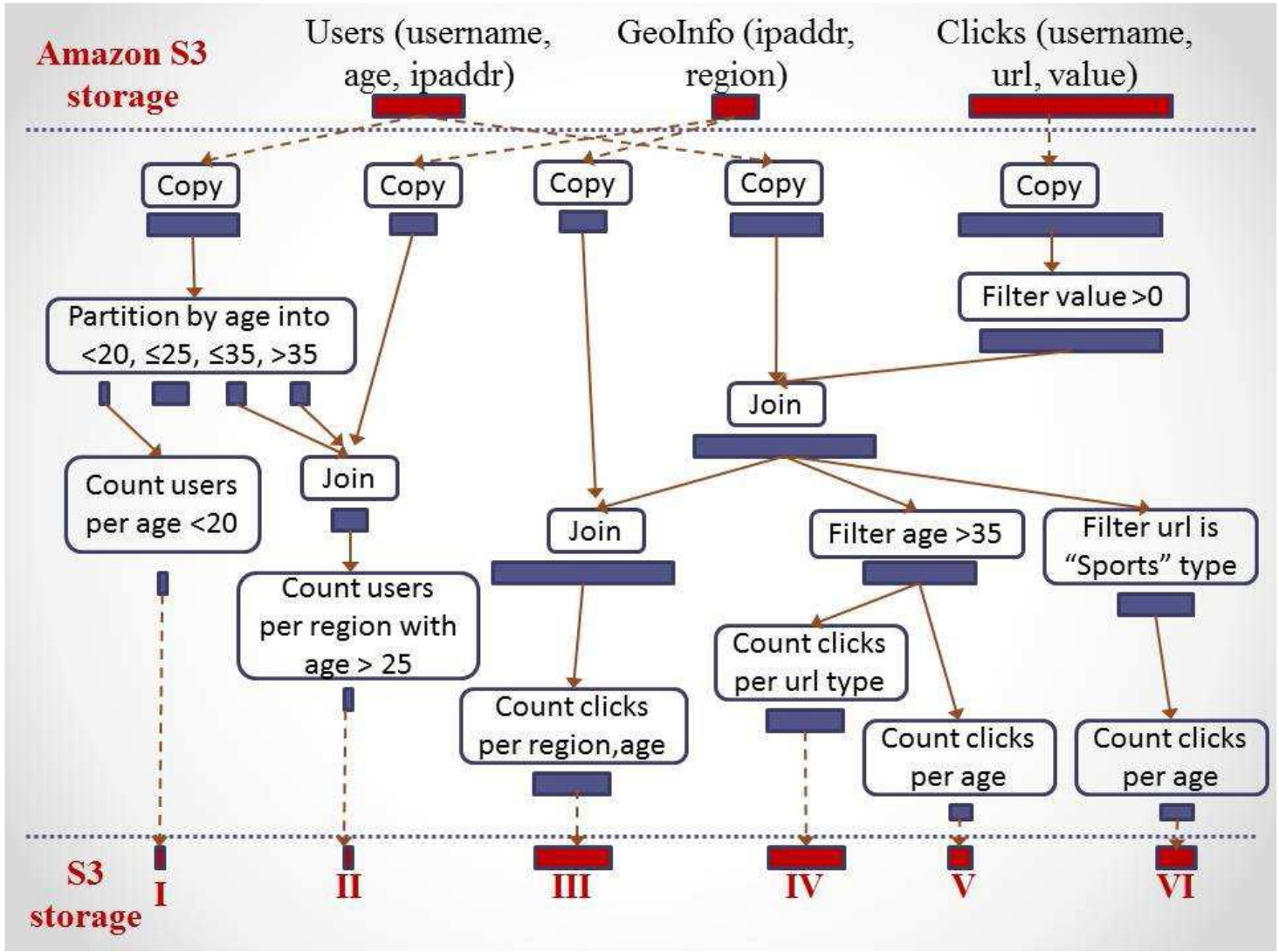
STARFISH GOAL

- A number of ongoing projects aim to improve Hadoop's peak performance
- Regular users may rarely see performance close to this peak
- Starfish's goal is to enable Hadoop users and applications to get good performance automatically throughout the data lifecycle in analytics; without any need on their part to understand and manipulate the many tuning knobs available.

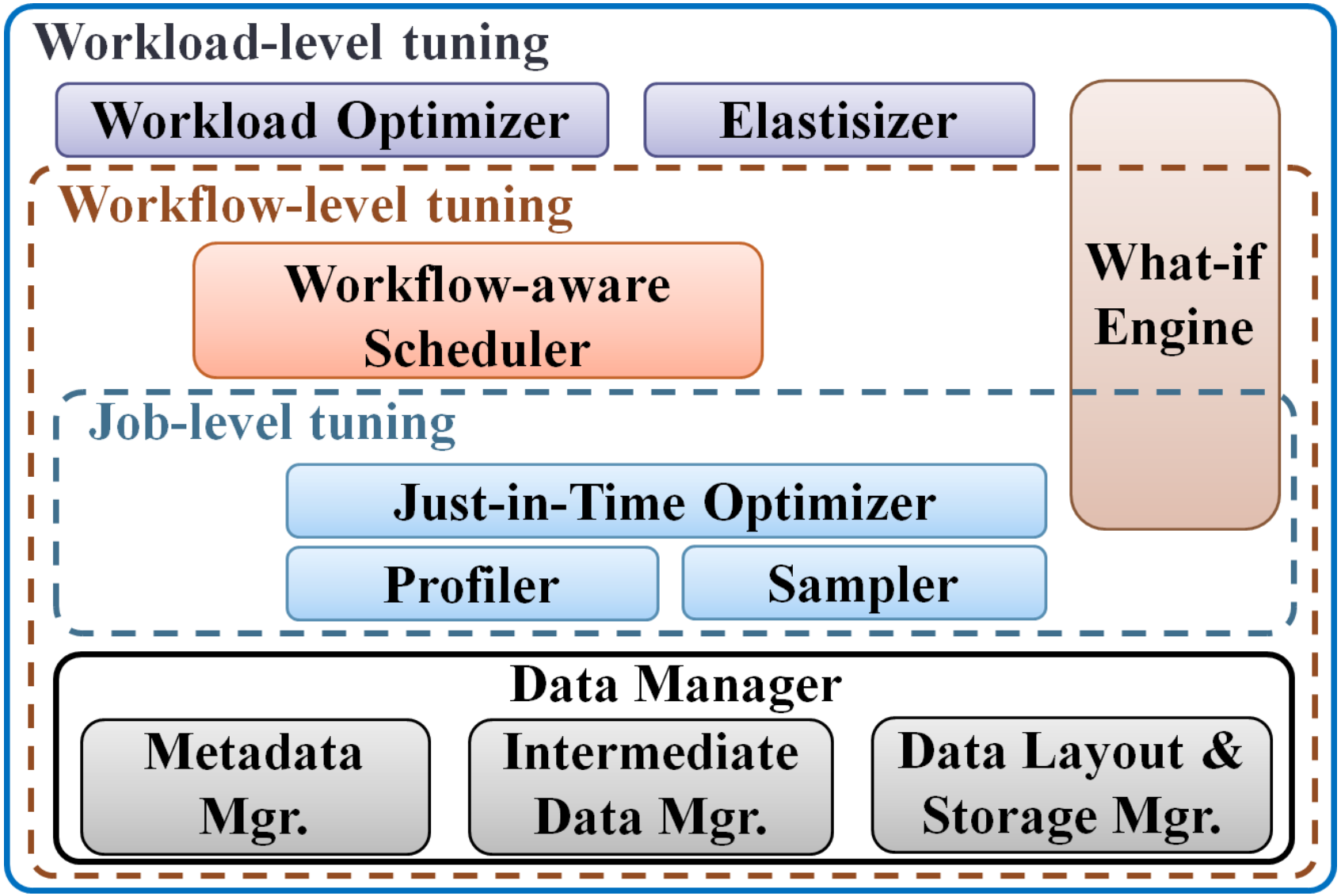
EXAMPLE ANALYTICS WORKLOAD

A simple website analytics

The input data are collected by a personalized Web-site like my.yahoo.com



STARFISH ARCHITECTURE



OVERVIEW OF STARFISH

- ***Job level Tuning***
  - ✓***Just-in-Time Optimizer***
    - ➢Profiler
    - ➢Sampler
- ***Workflow level Tuning***
- ***Workload level Tuning***
  - ➢Data-flow sharing
  - ➢Materialization
  - ➢Reorganization

JOB LEVEL TUNING

- The behavior of a MapReduce job in Hadoop is controlled by the settings of more than 190 configuration parameters.
- Good settings for these parameters depend on job, data, and cluster characteristics.
- The optimizer takes the help of the *Profiler* and the *Sampler*.

PROFILER

- The Profiler uses a technique called *dynamic instrumentation* to learn performance models, called *job profiles*

SAMPLER

- The Sampler collects statistics efficiently about the input, intermediate, and output *key-value spaces* of a MapReduce job
- A unique feature of the Sampler is that it can sample the execution of a MapReduce job in order to enable the Profiler to collect approximate job profiles at a fraction of the full job execution cost.

WORK-FLOWLEVEL TUNING

- some critical and unanticipated interactions between the MapReduce task scheduler and the underlying distributed filesystem.
- moving the computation to the data
  - the data layout across nodes in the cluster constrains how tasks can be scheduled
- Avoiding cascading reexecution under node failure or data corruption
- Ensuring power proportional computing
- Adapting to imbalance in load or cost of energy across geographic regions and time at the datacenter level.

WORKLOAD-LEVEL TUNING

- ***Data-flow sharing***
  a single MapReduce job performs computations for multiple and potentially different logical nodes belonging to the same or different workflows.
- ***Materialization***
  intermediate data in a workflow is stored for later reuse in the same or different workflows. Effective use of materialization has to consider the cost of materialization and its potential to avoid cascading reexecution of tasks under node failure or data corruption.
- ***Reorganization***
  new data layouts and storage engines are chosen automatically and transparently to store intermediate data so that downstream jobs in the same or different workflows can be executed very efficiently.

LASTWORD

- ✓ *Starfish's Language for Workloads and Data*
- ✓ *Lastword is not a language that humans will have to interface with directly.*
- ✓ Starfish provides language translators to automatically convert workloads specified in these higher-level languages to Lastword.
- ✓ physical workflows, logical workflows, hybrid workflows
- ✓ *Support for expressing metadata along with the tasks for execution*
- ✓ *Lastword enables Starfish to be used as a recommendation engine in these environments.*

JUST-IN-TIME JOB OPTIMIZATION

- VS Rules of Thumb for Parameter Tuning

io.sort.mb

io.sort.record.percent

=0.9 times the total number of reduce slots

mapred.reduce.tasks

$$= \frac{16}{16 - avg\_recode\_size}$$

	WordCount		TeraSort	
	Rules of Thumb	Based on Job Profile	Rules of Thumb	Based on Job Profile
io.sort.spill.percent	0.80	0.80	0.80	0.80
io.sort.record.percent	0.50	0.05	0.15	0.15
io.sort.mb	200	50	200	200
io.sort.factor	10	10	10	100
mapred.reduce.tasks	27	2	27	400
Running Time (sec)	785	407	891	606

Table 1: Parameter settings from rules of thumb and recommendations from job profiles for WordCount and TeraSort

RESPONSE SURFACES

the impact of various job configuration parameter settings on the running time of two MapReduce programs in Hadoop.

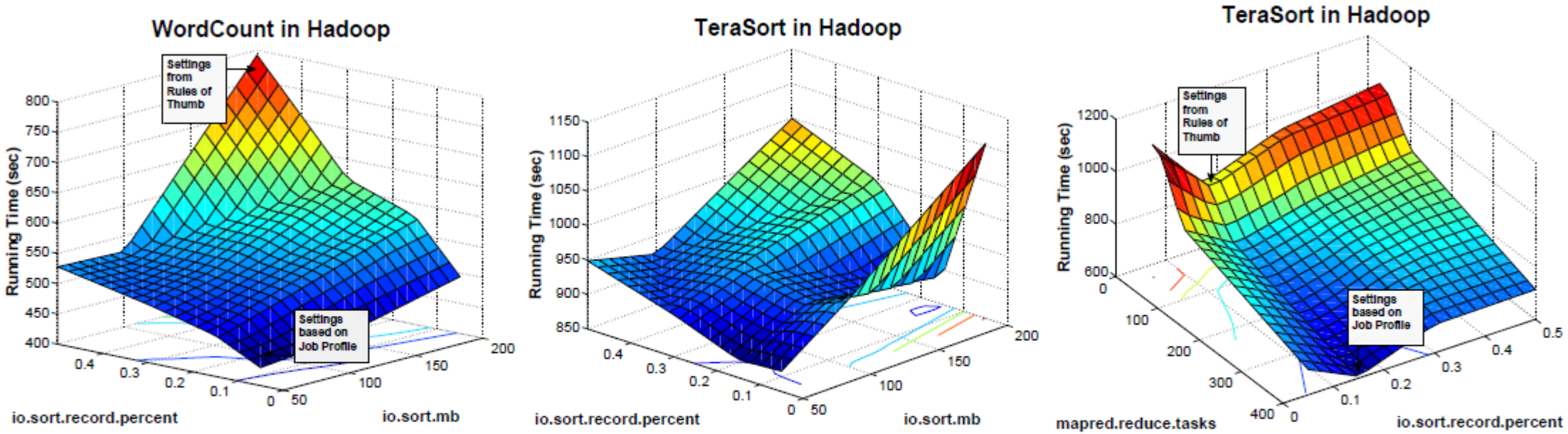


Figure 4: Response surfaces of MapReduce programs in Hadoop: (a) WordCount, with io.sort.mb $\in$ $[50, 200]$ and io.sort.record.percent $\in$ $[0.05, 0.5]$ (b) TeraSort, with io.sort.mb $\in$ $[50, 200]$ and io.sort.record.percent $\in$ $[0.05, 0.5]$ (c) TeraSort, with io.sort.record.percent $\in$ $[0.05, 0.5]$ and mapred.reduce.tasks $\in$ $[27, 400]$

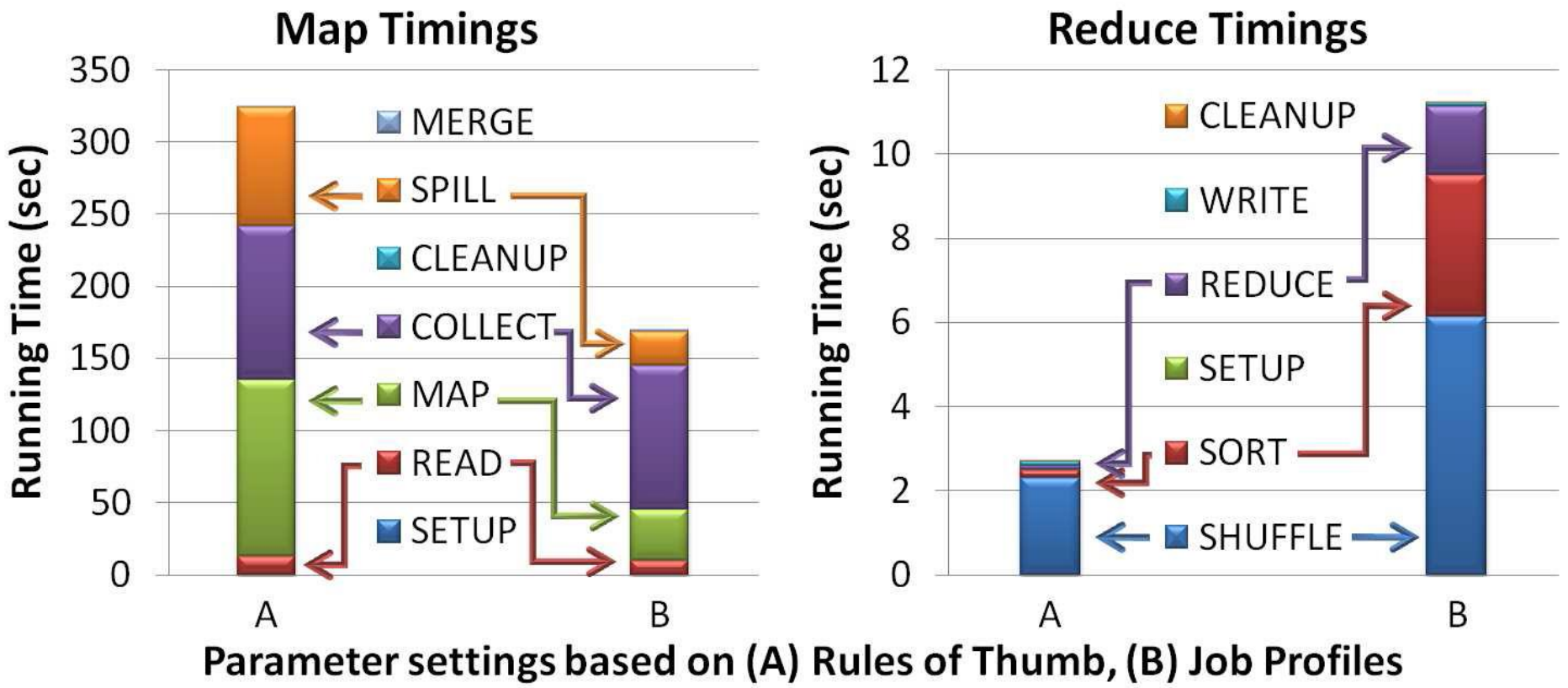
THREE VIEWS OF JOB PROFILE

- ***Timings view***
  - *Giving the breakdown of how wallclock time was spent in the various subphases.*
- ***Data-flow view***
  - *Giving the amount of data processed in terms of bytes and number of records during the various subphases.*
- ***Resource-level view***
  - *Capturing the usage trends of CPU, memory, I/O, and network resources during the various subphases of the job's execution.*

COMPARISON OF TWO SETTING

Combiner in Job A was processing an extremely large number of records, causing high CPU contention.

On the other hand, the Combiner drastically decreases the amount of intermediate data



PREDICTING JOB PERFORMANCE

- Using What-if Engine to predict the performance
- 4 Inputs to generate ***virtual job profile***
  1. The job profile generated for J by the Profiler
  2. The new setting S of the job configuration parameters using which Job J will be run
  3. The size, layout, and compression information of the input dataset on which Job J will be run
  4. The cluster setup and resource allocation that will be used to run Job J

OVERHEAD AND RELATIVE ERROR

The main challenges is in developing an efficient strategy to search through the high dimensional space of parameter settings.

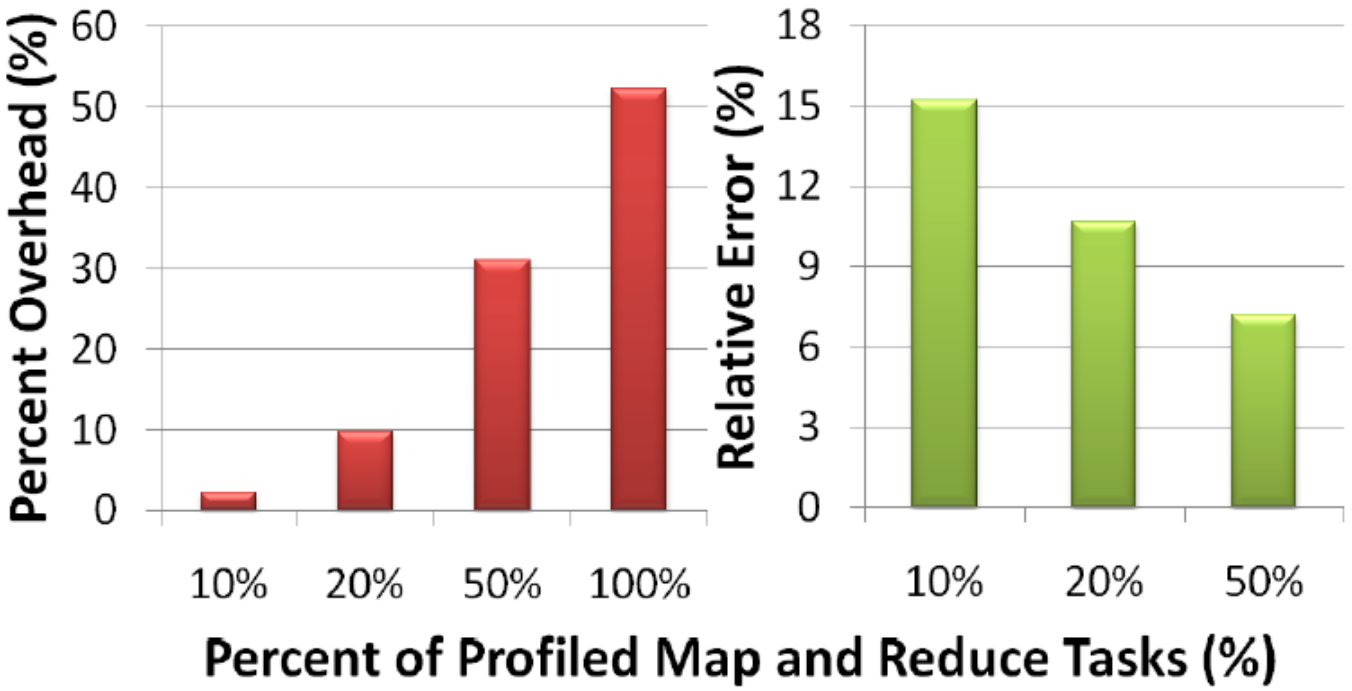


Figure 6: (a) Relative job slowdown, and (b) relative error in the approximate views generated as the percentage of profiled tasks in a job is varied

WORKFLOW-AWARE SCHEDULING

- **Problem**
  - Unbalanced Data Layouts
  - Collocating two or more datasets
- **Solution**
  - New block placement policies

UNBALANCED DATA LAYOUT

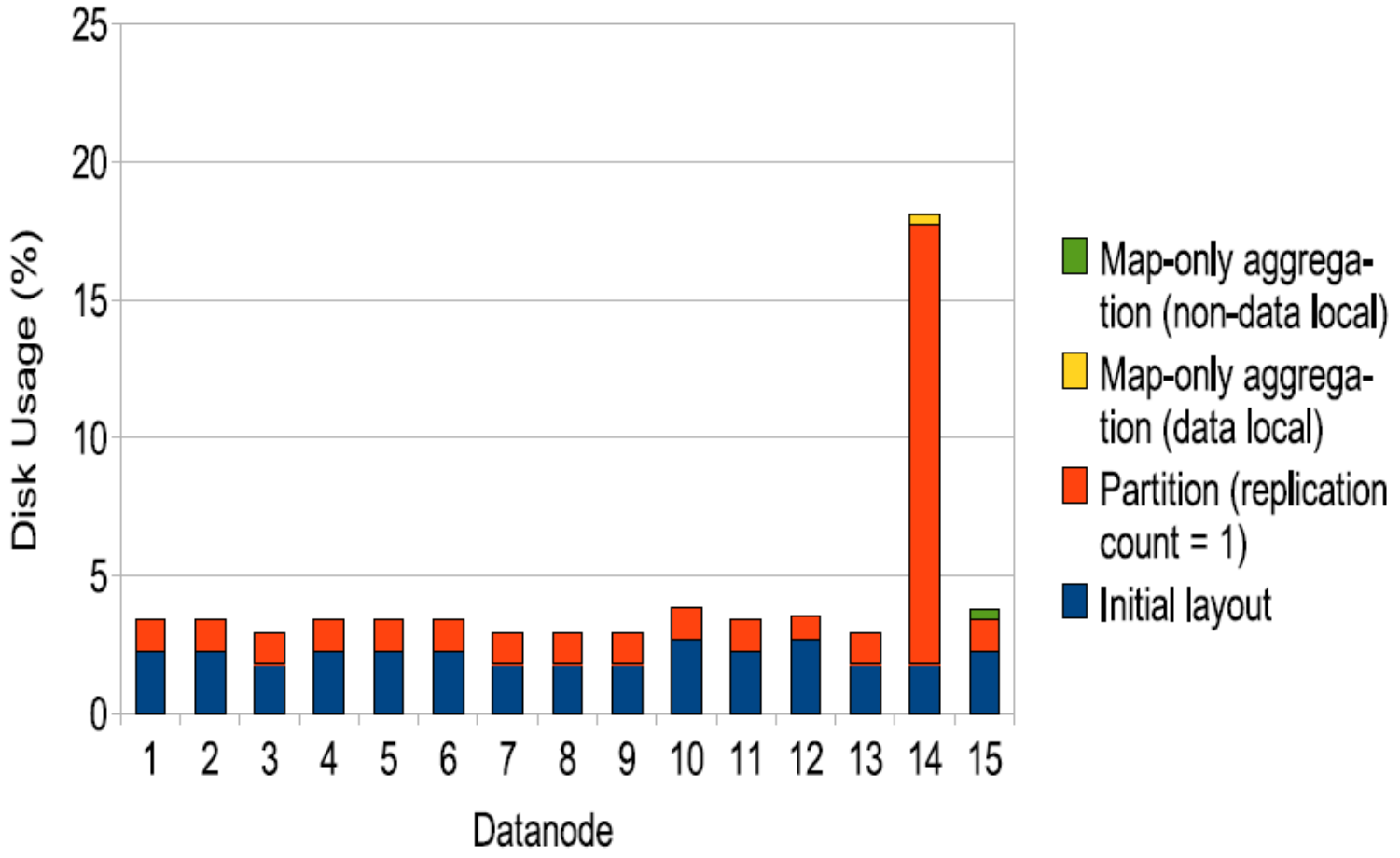


Figure 7: Unbalanced data layout

UNBALANCED DATA LAYOUT

- **A number of causes can lead to unbalanced data layouts rapidly or over time:**
  - Skewed data
  - Scheduling of tasks in a data-layout-unaware manner as done by the Hadoop schedulers available today
  - Addition or dropping of nodes without running costly data rebalancing operations

DILEMMA FOR DATA-LOCALITY-AWARE SCHEDULERS

Performance degradation due to reduced parallelism, and worse, making the data layout further unbalanced because new outputs will go to the over-utilized nodes.	Non-data-local scheduling incurs the overhead of data movement.

2x SLOWDOWN ON UNBALANCED LAYOUT

a partitioning MapReduce job that partitions a 100GB TPC-H Lineitem table into four partitions relevant to downstream workflow nodes.

The data properties are such that one partition is much larger than the others.

performance degradation due to reduced parallelism

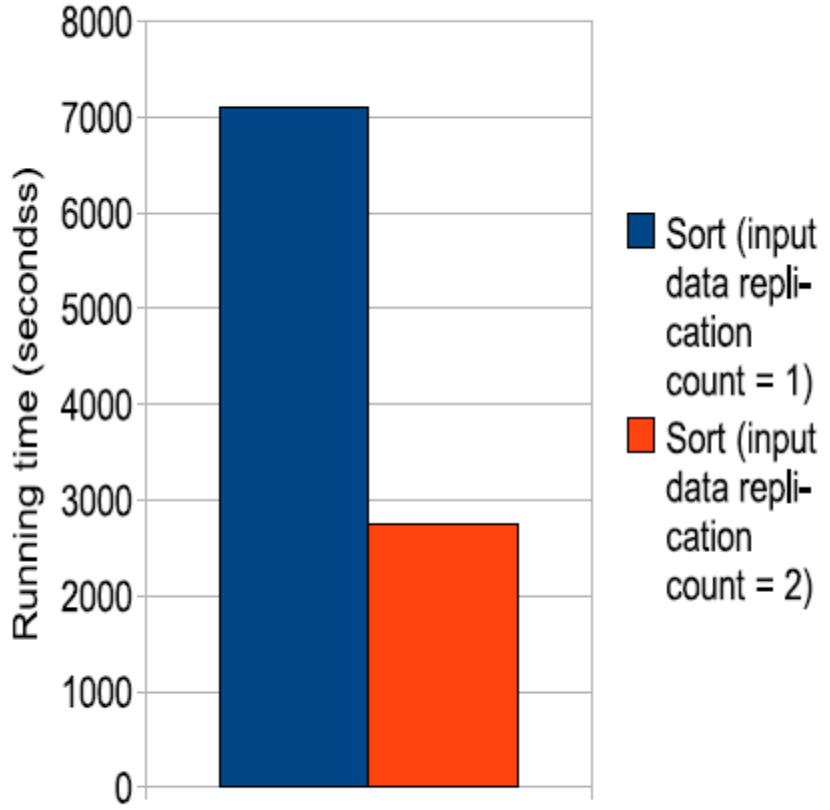


Figure 8: Sort running time on the partitions

PARTITION CREATION TIME

with a replication factor of two for the partitions.

HDFS places the second replica of each block of the partitions on a randomly-chosen node.

The overall layout is still unbalanced, but the time to sort
the partitions improved significantly because the second copy of
the data is spread out over the cluster

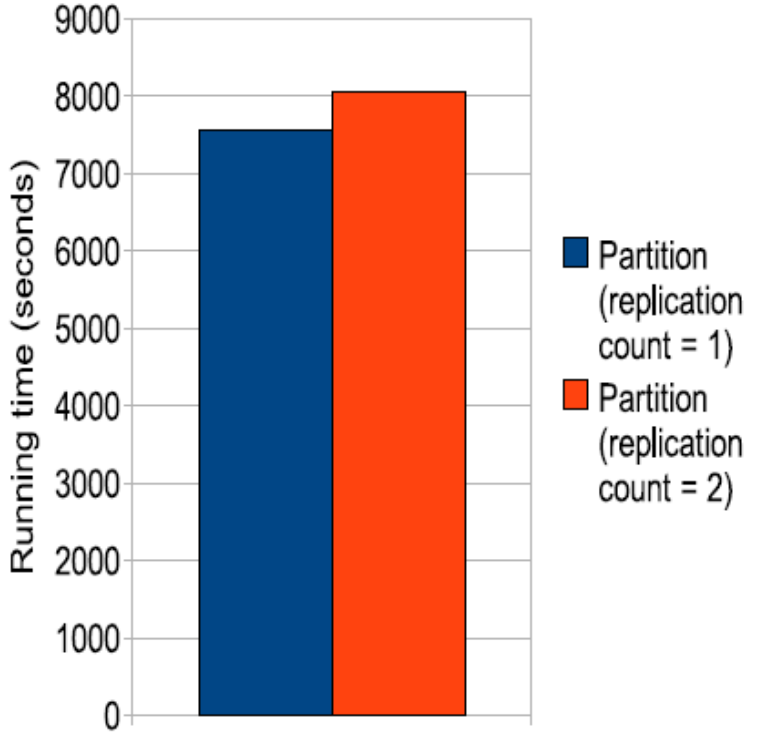


Figure 9: Partition creation time

COLLOCATING TWO OR MORE DATASETS

- HDFS does not provide the ability to collocate the joining partitions, so a join job run in Hadoop will have to do non-data-local reads for one of its inputs

NEW BLOCK PLACEMENT POLICY

A new block placement policy in HDFS that enables collocation of two or more datasets.

The new policy gives a 22% improvement in the running time of a partition-wise join job by collocating the joining partitions.

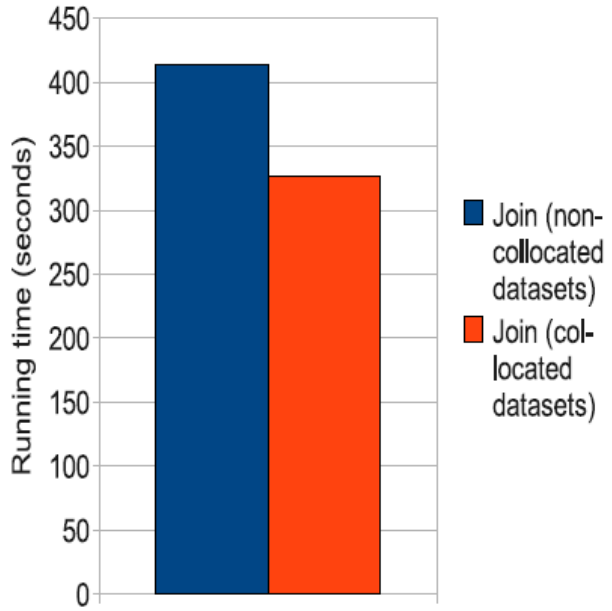


Figure 10: Respective execution times of a partition-wise join job with noncollocated and collocated input partitions

NEED FOR A WORKFLOW-AWARE SCHEDULER

- A Workflow-aware Scheduler that can run jobs in a workflow such that the overall performance of the workflow is optimized.
- Workflow performance can be measured in terms of running time, resource utilization in the Hadoop cluster, and robustness to failure and transient issues(reacting to the slowdown of a node due to temporary resource contention)

ANALYZING RELATIONSHIPS

- What parts of the data output by a job are used by downstream jobs in the workflow?
- What is the unit of data-level parallelism in each job that reads the data output by a job?

EXAMPLE

1) File1 forms the input to Job C1, while *File1* and *File2* form the input to Job C2. Since *File3* is not used by any of the downstream jobs, a Workflow-aware Scheduler can configure Job P to avoid generating *File3*.

2) For *File2* all blocks in the file should be placed on the same node to ensure data-local computation

For *File1* The data-level parallelism is at the block-level in Job C1, but at the file-level in Job C2.to spread *File1*'s blocks across the nodes so that C1's maptasks can run in parallel across the cluster. However, the optimal layout of *File1* from Job C2's perspective is to place all blocks on the same node.

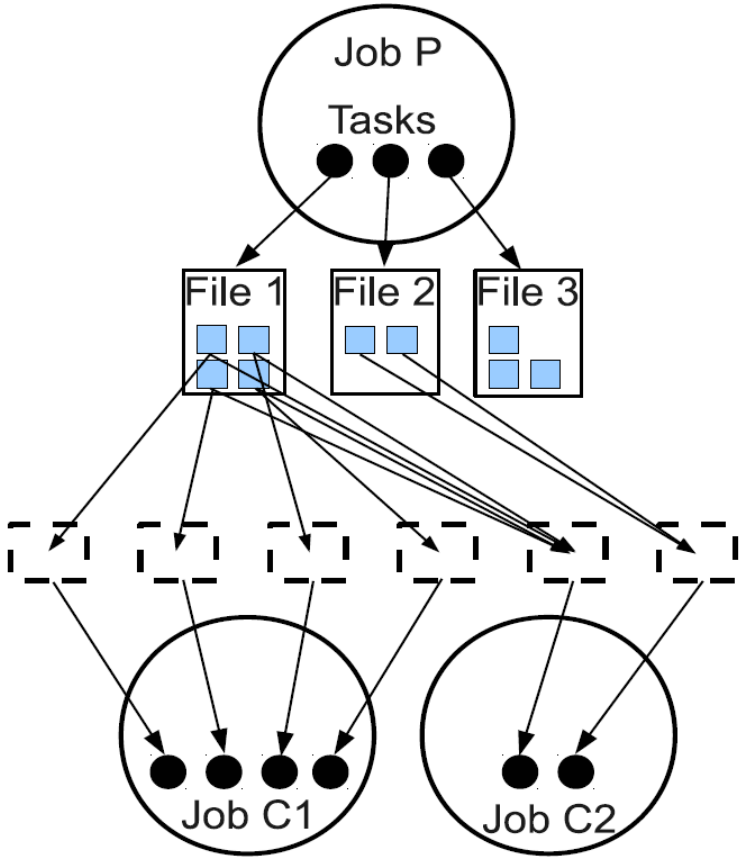


Figure 11: Part of an example workflow showing producer-consumer relationships among jobs

CHOICES FOR DATA LAYOUT

- What block placement policy to use in the distributed filesystem for the output file of a job?
  - *Round Robin*
  - local write
- How many replicas to store—called the ***replication factor***—for the blocks of a file?
- What size to use for blocks of a file?
- Should a job's output files be compressed for storage?

WHAT-IF QUESTIONS

- What is the expected running time of Job P if the Round Robin block placement policy is used for P's output files?
- What will the new data layout in the cluster be if the Round Robin block placement policy is used for P's output files?
- What is the expected running time of Job C1 (C2) if its input data layout is the one in the answer to Question 2?

WHAT-IF QUESTIONS

- What are the expected running times of Jobs C1 and C2 if they are scheduled concurrently when Job P completes?
- Given the Local Write block placement policy and a replication factor of 1 for Job P's output, what is the expected increase in the running time of Job C1 if one node in the cluster were to fail during C1's execution?

PERFORMANCE COMPARISON

the local I/O within a node becomes the bottleneck before the parallel writes of data blocks to other storage nodes over the network.

(b)the performance of the sort job for different layouts of the output of the partition job.
the Round Robin policy spreads the blocks over the cluster so that maximum data-level parallelism of sort processing can be achieved while performing data-local computation.

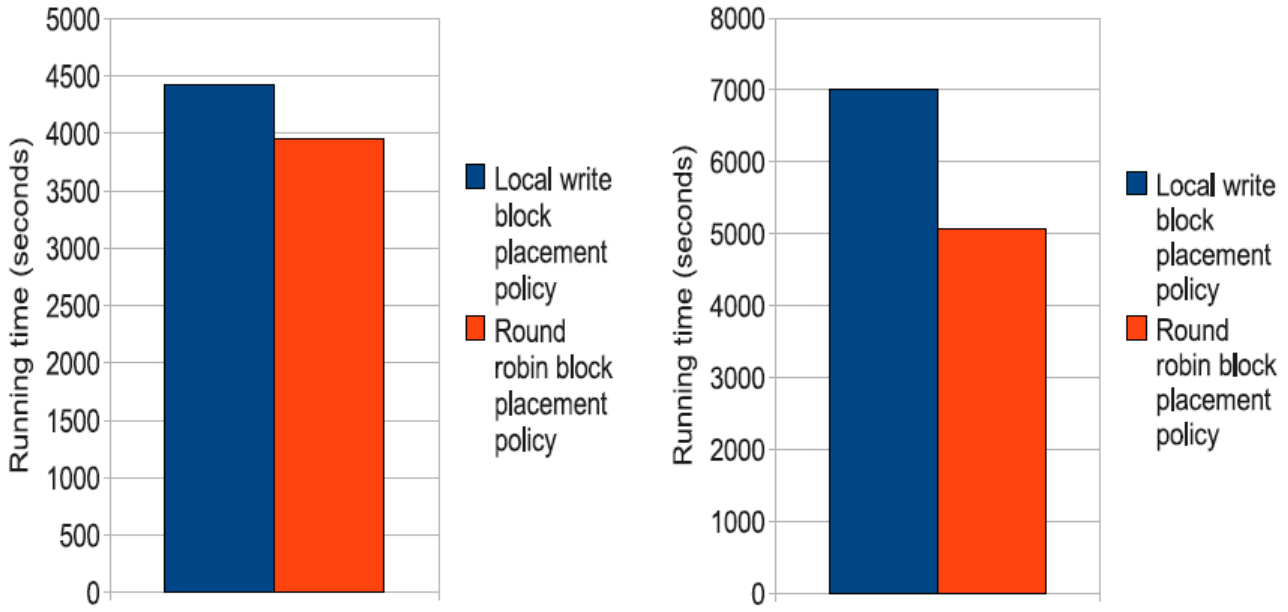


Figure 12: Respective running times of (a) a partition job and (b) a two-job workflow with the (default) Local Write and the Round Robin block placement policies used in HDFS

OPTIMIZATION AND PROVISIONING FOR HADOOP WORKLOADS

- **Workload Optimizer**
  - Represents the workload as a directed graph and a graph-to-graph transformations.
  - Use the What-if Engine to do a cost-based estimation of whether a graph-to-graph transformation will improve performance.
- **Elastisizer**
  - Money
  - Time

Jumbo operator

- ***SPA：***
  - Select-Project-Aggregate expression over the join
  - The results IV, V, and VI can each be represented as a Select-Project-Aggregate (SPA) expression over the join.
- ***Jumbo operator：***
  - Can process any number of logical SPA workflow nodes over the same table in a single MapReduce job
- Without the Jumbo operator, each SPA node will have to be processed as a separate job.

EXPERIMENTAL RESULT

results I, II, IV, and V. These four workflows have filter conditions on the age attribute in the Users dataset.

Partition pruning improves the performance of all MapReduce jobs in our experiment. At the same time, partition pruning decreases the performance benefits provided by the Jumbo operator.

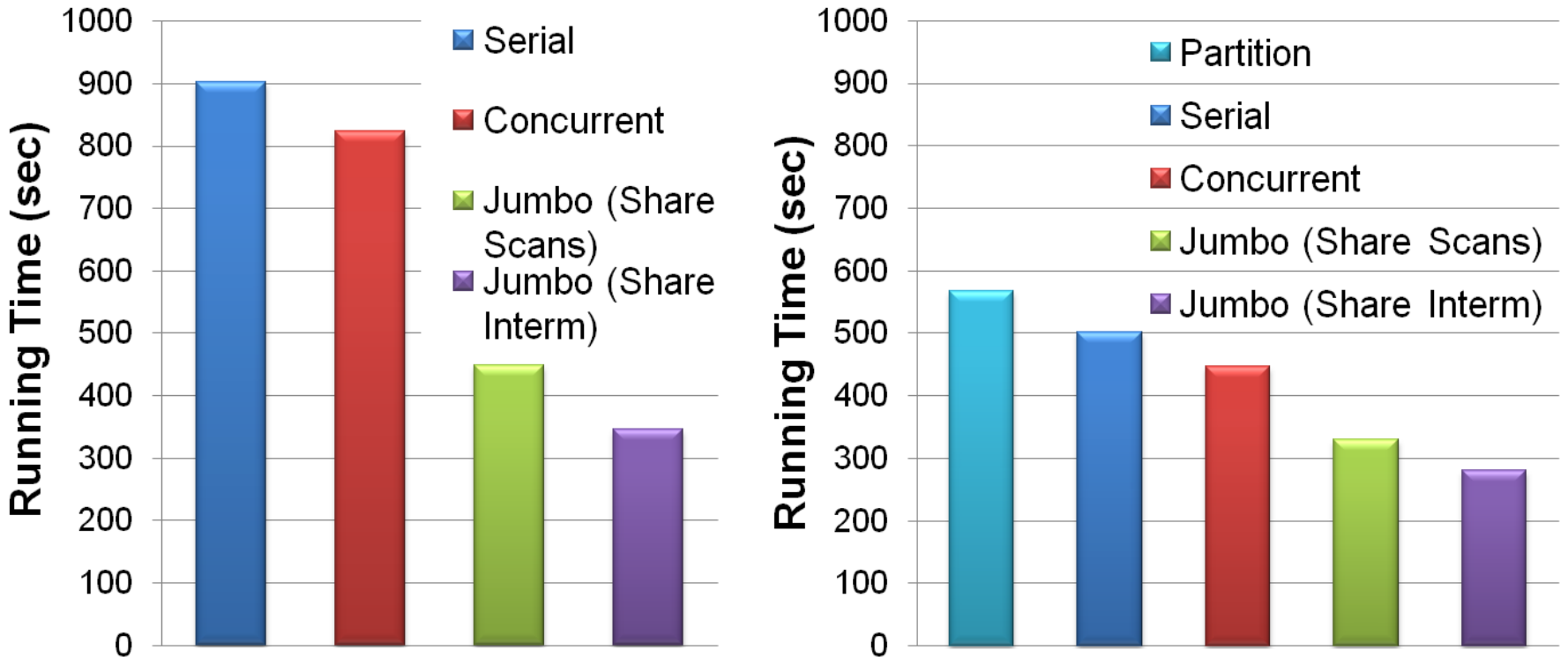


Figure 13: Processing multiple SPA workflow nodes on the same input dataset

Elastisizer

- Users can now leverage pay-as-you-go resources on the cloud to meet their analytics needs.
- The cluster can be released when the workflow completes, and the user pays for the resources used.
- Users have to specify the number and type of EC2 nodes (from among 10+ types) as well as whether to copy data from S3 into the in-cluster HDFS.
- One of the goals of Starfish's Elastisizer is to automatically determine the best cluster and Hadoop configurations to process a given workload subject to user-specified goals

WORKLOAD PERFORMANCE

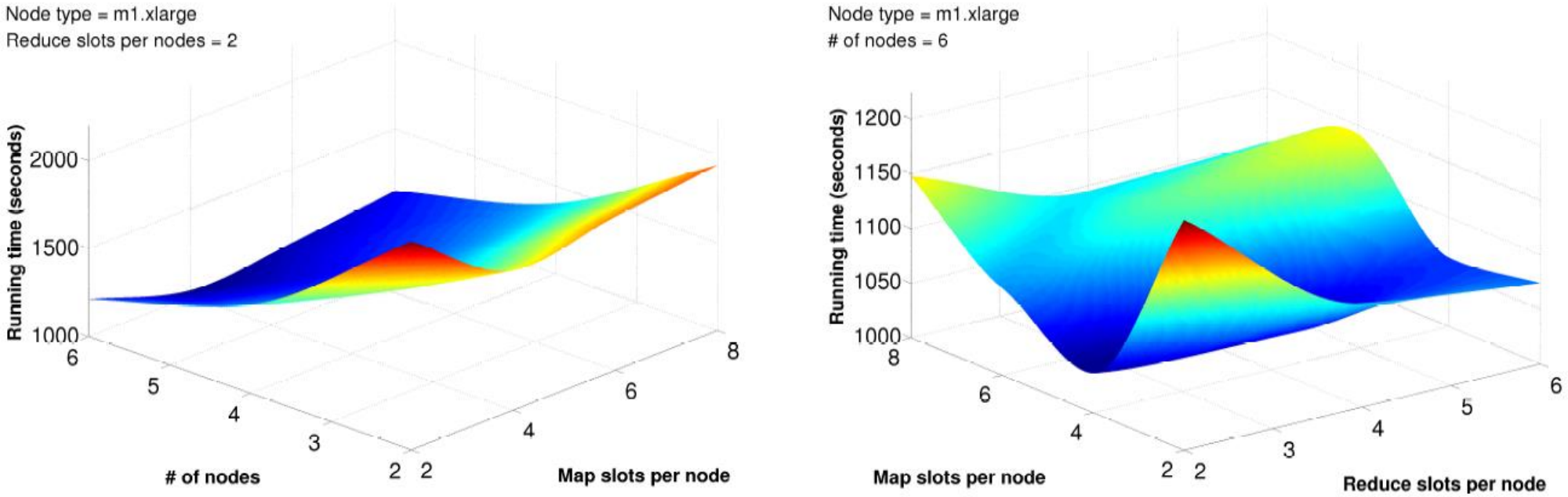
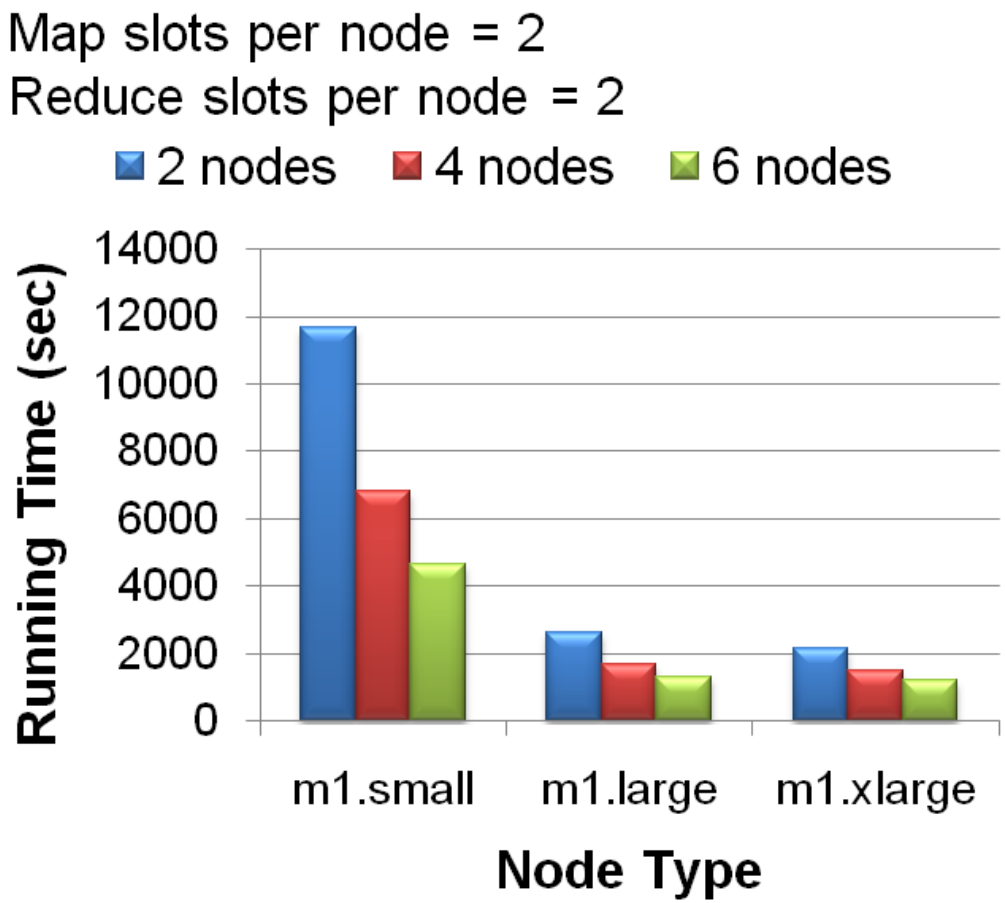
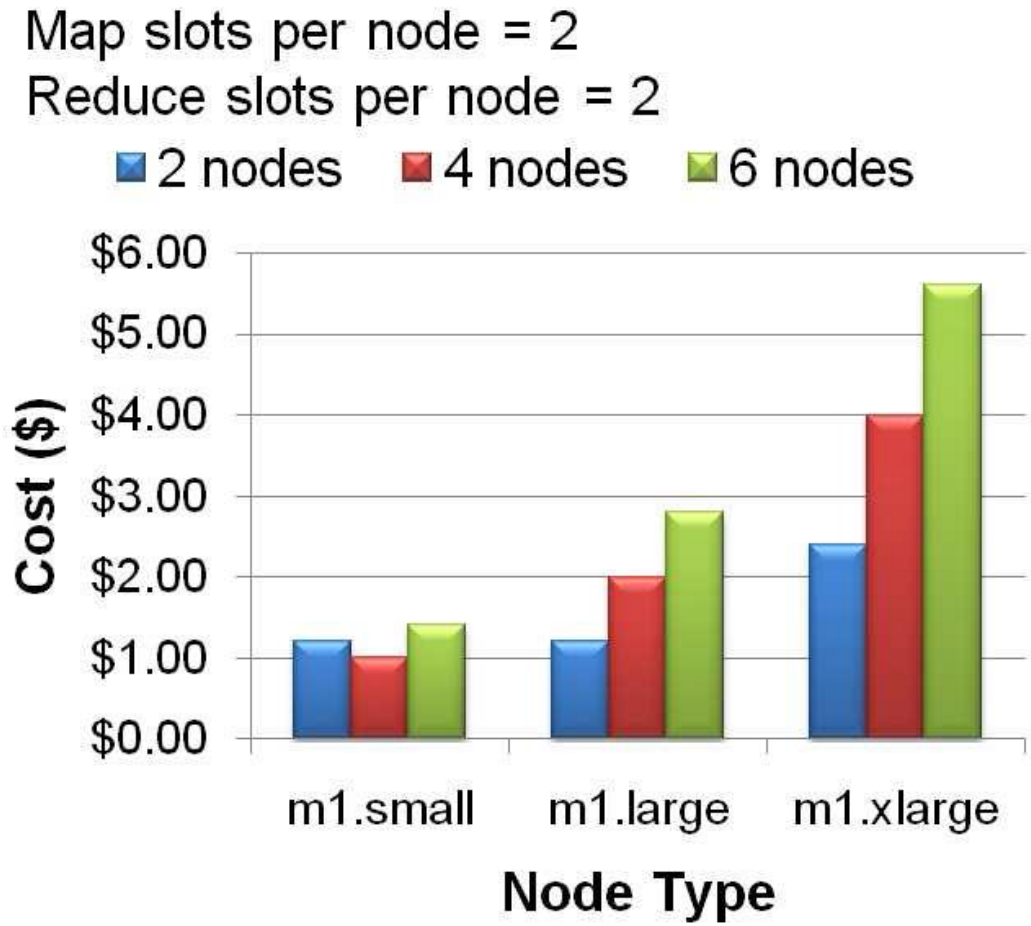


Figure 14: Workload performance under various cluster and Hadoop configurations on Amazon Elastic MapReduce

PERFORMANCE & COSTS





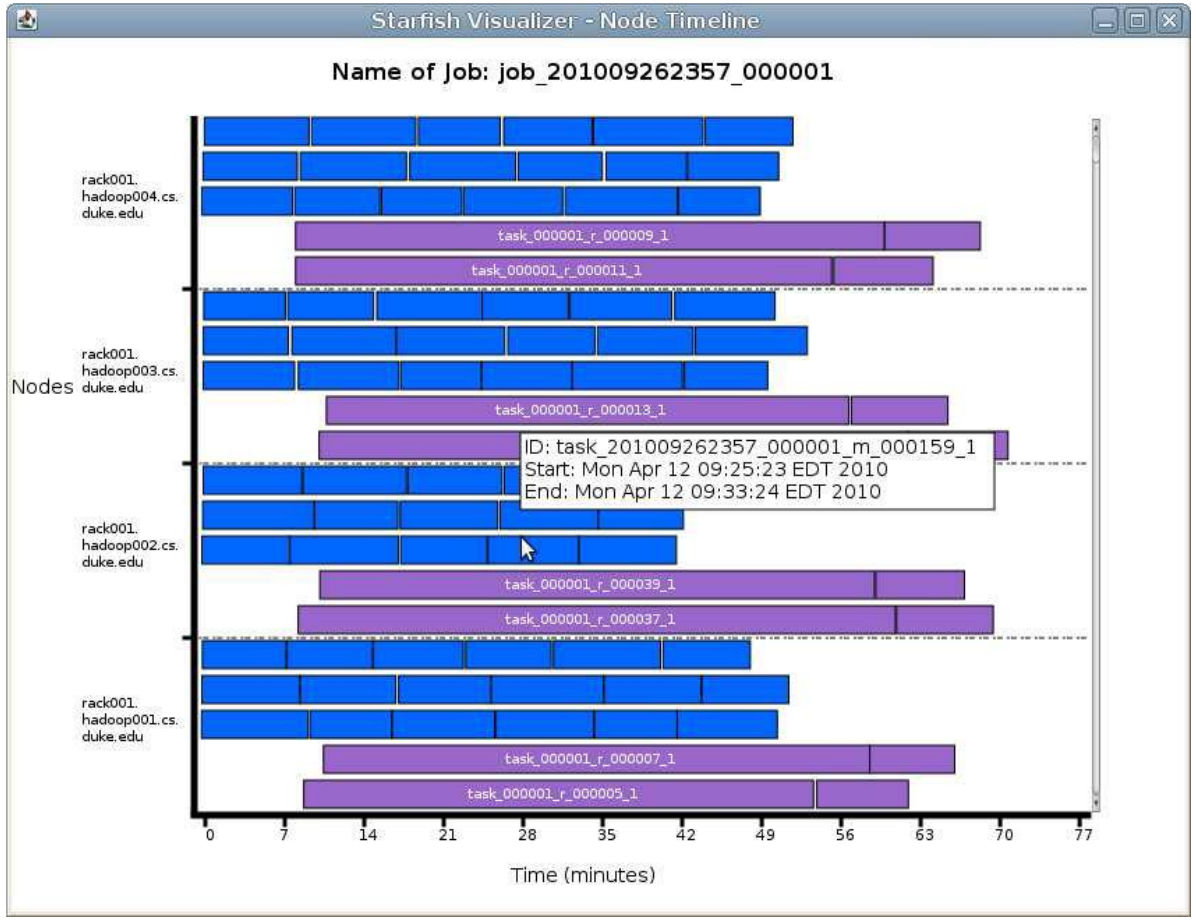
EXAMPLE

- Minimize the monetary cost incurred to run the workload, subject to a maximum tolerable workload completion time.
- If the user wants to minimize costs subject to a completion time of 30 minutes, then the Elastisizer should recommend a cluster of four m1.large EC2 nodes.
- If the user wants to minimize costs, then four m1.small nodes are best. However, the Elastisizer can suggest that by paying just 20% more, the completion time can be reduced by 2.6x.

SUMMARY

- Starfish fills a different void by enabling Hadoop users and applications to get good performance automatically throughout the data lifecycle in analytics; without any need to understand and manipulate it.
- The novelty in Starfish's approach comes from how it focuses *simultaneously* on different workload granularities—overall workload, workflows, and jobs as well as across various decision points—provisioning, optimization, scheduling, and data layout.

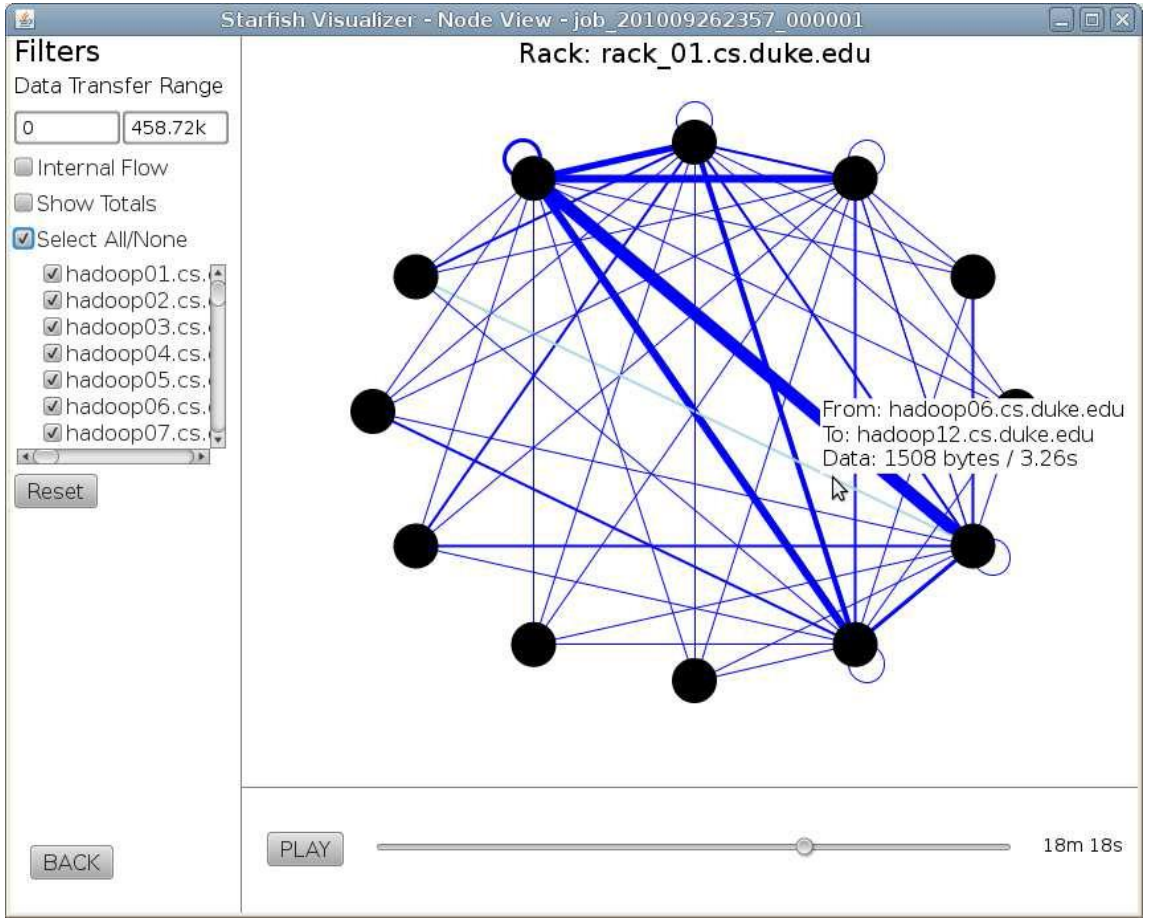
TIMELINE VIEWS



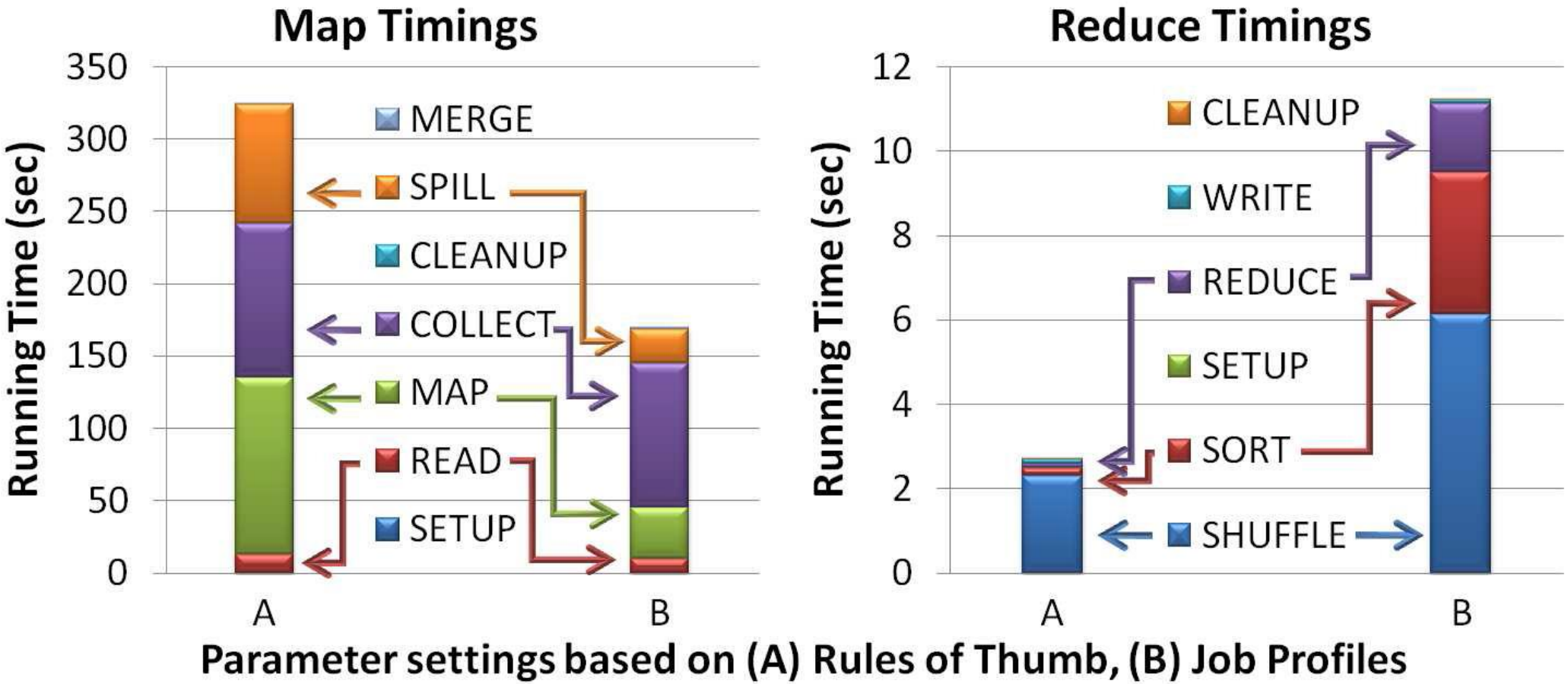
DATA-FLOW VIEWS

Video mode

allows users to play back a job execution from the past.



PROFILE VIEWS



MY IMPRESSION

PROS	CONS
✓ *A good idea to help these who are not familiar with Hadoop.* ✓ *Focus on the overall performance not the peak performance.* ✓ *Dynamic performance tuning on run time.*	X *Only typical job performance, haven't average performance for tens or hundreds jobs.* X *No details about starfish's overhead* X *All example are on one environment.*

THANK YOU