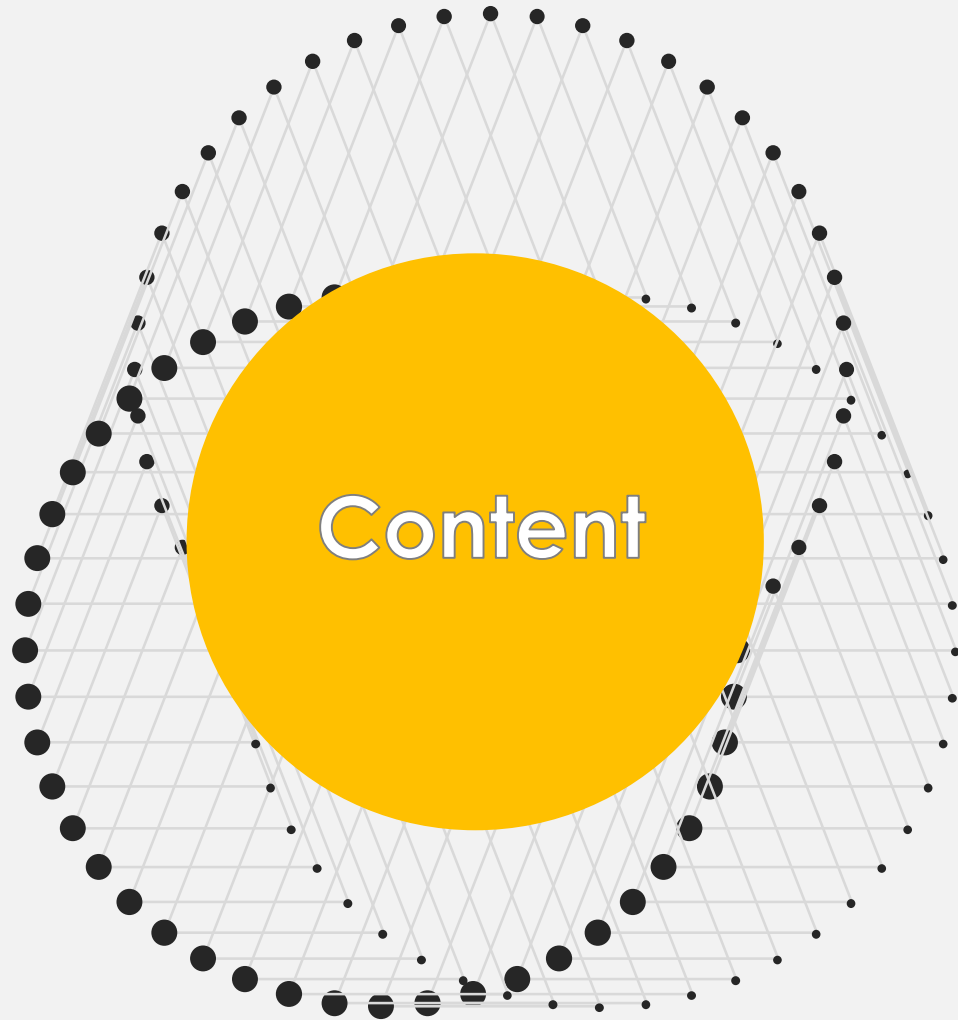


Starfish: A Self-tuning System for Big Data Analytics



Authors: Herodotos Herodotou, Harold Lim, Fei Dong, Shivnath Babu

Presented by: Chen Zou, Kai Zou

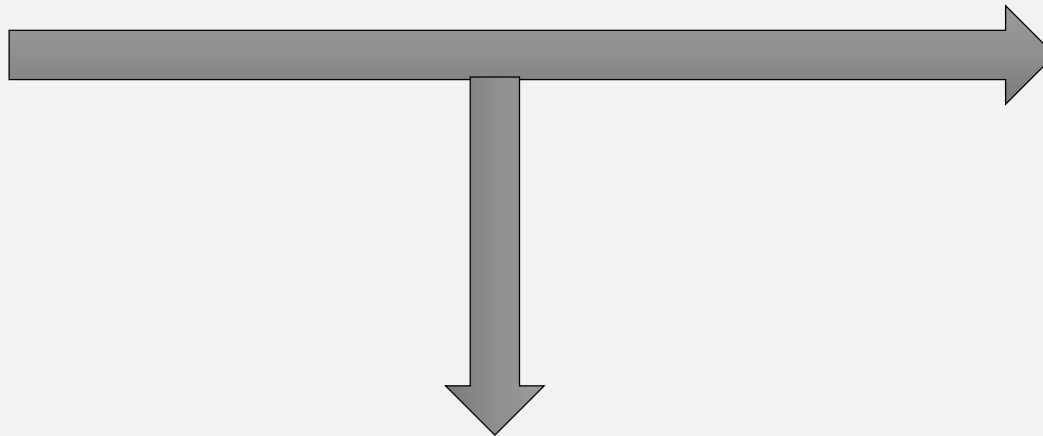
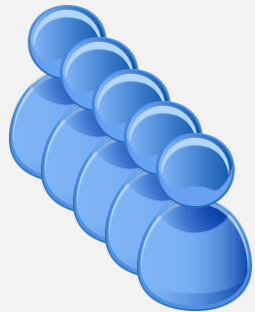


- Part 1 Introduction
- Part 2 Overview
- Part 3 Job-level Tuning
- Part 4 Workflow-aware tuning
- Part 5 Workload-level Tuning
- Part 6 Critique

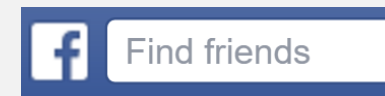
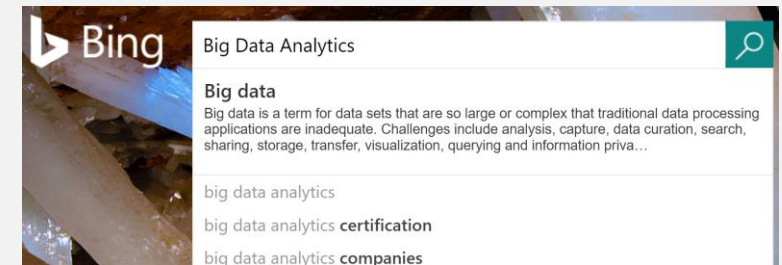
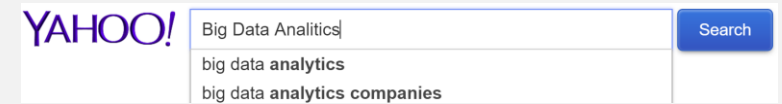
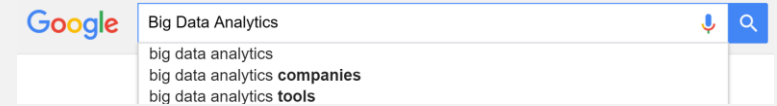


1 Introduction

Big Data is EVERYWHERE!

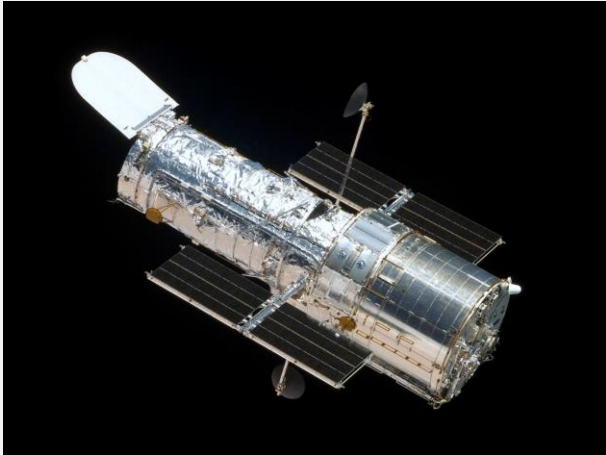


Large amount of data is captured and analyzed



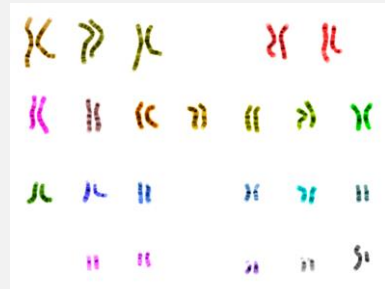
Big Data is EVERYWHERE!

Hubble Space Telescope

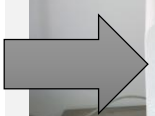


Adopt from https://en.wikipedia.org/wiki/Hubble_Space_Telescope

DNA Sequencer

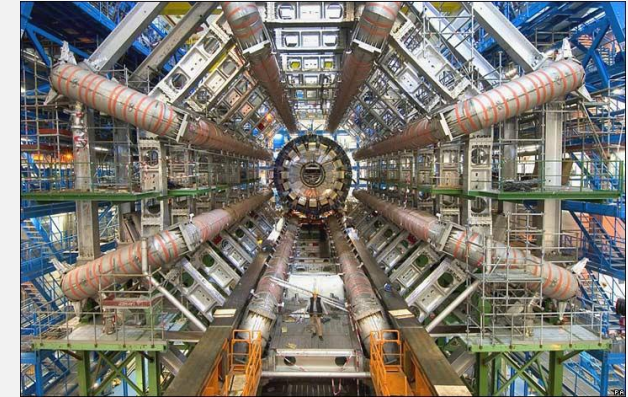


Adopt from <https://en.wikipedia.org/wiki/Genome>



Adopt from https://en.wikipedia.org/wiki/DNA_sequencing

Particle Accelerator



Adopt from <https://www.flickr.com/photos/11304375@N07/2046228644>

**Generating massive
amount of data**

Big Data is EVERYWHERE!

Hubble Space Telescope



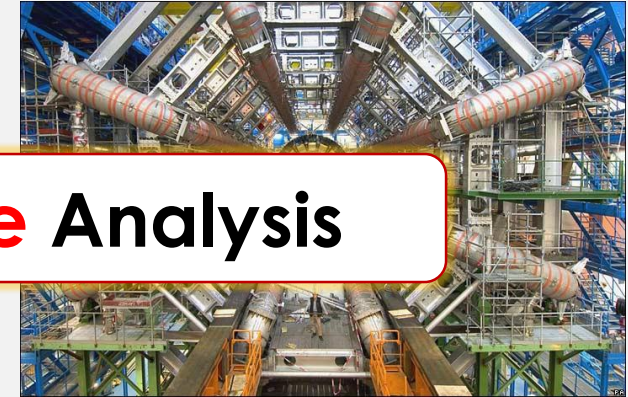
Adopt from https://en.wikipedia.org/wiki/Hubble_Space_Telescope

DNA Sequencer



Adopt from https://en.wikipedia.org/wiki/DNA_sequencing

Particle Accelerator



Adopt from <https://www.flickr.com/photos/11304375@N07/2046228644>

Key to **Success** = **Timely** and **Cost-Effective** Analysis

Generating massive
amount of data

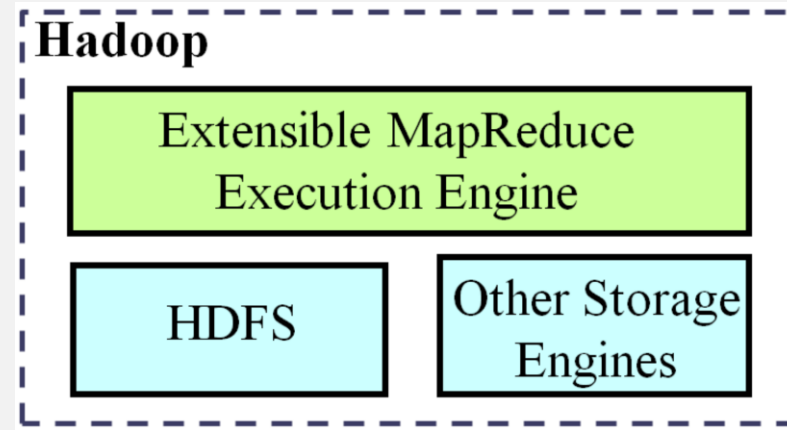
What we need is a MAD system

Magnetism “Attracts” or welcomes all sources of data, regardless of structure, values, etc.

Agility Adaptive, remains in sync with rapid data evolution and modification

Depth More than just your typical analytics, we need to support complex operations like statistical analysis and machine learning

Hadoop is MAD



- Copying files into the Distributed File System is all it takes to get data into Hadoop
 - Interpret data at processing time, and not at loading time
 - Supports different kinds of programming languages
- **Magnetism**
Agility
- **Depth**

Want best performance? Non-trivial!

```
mapreduce.jobtracker.jobhistory.location
mapreduce.jobtracker.jobhistory.task.number.progress.splits
mapreduce.job.userhistory.location
mapreduce.jobtracker.jobhistory.completed.location
mapreduce.job.committer.setup.cleanup.enabled
mapreduce.task.io.sort.factor
mapreduce.task.io.sort.mb
mapreduce.map.sort.spill.percent
mapreduce.jobtracker.address
mapreduce.local.clientfactory.class.name
mapreduce.jobtracker.http.address
mapreduce.jobtracker.handler.count
mapreduce.tasktracker.report.address
mapreduce.cluster.local.dir
mapreduce.jobtracker.system.dir
mapreduce.jobtracker.staging.root.dir
mapreduce.cluster.temp.dir
mapreduce.tasktracker.local.dir.minspacart
mapreduce.tasktracker.local.dir.minspacek
mapreduce.jobtracker.expire.trackers.interval
mapreduce.tasktracker.instrumentation
mapreduce.tasktracker.resourcecalculator.login
mapreduce.tasktracker.taskmemorymanager.monitoringinterval
mapreduce.tasktracker.tasks.sleepbefore.resigkill
mapreduce.job.maps
mapreduce.job.reduce
mapreduce.jobtracker.restart.recover
mapreduce.jobtracker.jobhistory.block.size
mapreduce.jobtracker.taskscheduler
mapreduce.job.running.map.limit
mapreduce.job.running.reduce.limit
mapreduce.job.reducer.preempt.delay.seconds
mapreduce.job.max.split.locations
mapreduce.job.split.metainfo.maxsize
mapreduce.jobtracker.taskscheduler.maxrunningtasks.per.job
mapreduce.map.maxattempts
mapreduce.reduce.maxattempts
mapreduce.reduce.shuffle.fetch.retry.enabled
mapreduce.reduce.shuffle.fetch.retry.interval.ms
mapreduce.reduce.shuffle.fetch.retry.timeout.ms
mapreduce.reduce.shuffle.retry.delay.max.ms
mapreduce.reduce.shuffle.parallelcopies
mapreduce.reduce.shuffle.connect.timeout
mapreduce.reduce.shuffle.read.timeout
mapreduce.shuffle.connection.keep.alive.enabled
mapreduce.shuffle.connection.keep.alive.timeout
mapreduce.task.timeout
mapreduce.tasktracker.map.tasks.maximum
mapreduce.tasktracker.reduce.tasks.maximum
mapreduce.map.memory.mb
mapreduce.map.cpu.vcores
mapreduce.reduce.memory.mb
mapreduce.reduce.cpu.vcores
mapreduce.jobtracker.retry.failed.jobs.cache.size
mapreduce.tasktracker.outofband.heartbeat
mapreduce.jobtracker.jobhistory.lru.cache.size
mapreduce.jobtracker.instrumentation
mapred.child.java.opts
mapred.child.env
mapreduce.admin.user.env
mapreduce.map.log.level
mapreduce.reduce.log.level
mapreduce.map.cpu.vcores
mapreduce.reduce.cpu.vcores
mapreduce.reduce.merge.inmem.threshold
mapreduce.reduce.shuffle.merge.percent
mapreduce.reduce.shuffle.input.buffer.percent
mapreduce.reduce.shuffle.memory.limit.percent
mapreduce.shuffle.ssl.enabled
mapreduce.shuffle.ssl.file.buffer.size
mapreduce.shuffle.max.connections
mapreduce.shuffle.max.threads
mapreduce.shuffle.transfer.to.allowed
mapreduce.shuffle.transfer.buffer.size
mapreduce.reduce.markreset.buffer.percent
mapreduce.map.speculative
```

```
mapreduce.reduce.speculative
mapreduce.job.speculative.speculative.capacity.running.tasks
mapreduce.job.speculative.speculative.capacity.total.tasks
mapreduce.job.speculative.minimum.allowed.tasks
mapreduce.job.speculative.retry.after.no.speculate
mapreduce.job.speculative.retry.after.speculate
mapreduce.job.map.output.collector.class
mapreduce.job.speculative.slowtaskthreshold
mapreduce.job.yvm.memtasks
mapreduce.job.ubertask.enable
mapreduce.job.ubertask.maxmaps
mapreduce.job.ubertask.maxreduces
mapreduce.job.ubertask.maxbytes
mapreduce.job.emit.time.line.data
mapreduce.input.fileinputformat.split.min.size
mapreduce.input.fileinputformat.list.status.num.threads
mapreduce.jobtracker.maxtasks.per.job
mapreduce.input.lininputformat.linesper.map
mapreduce.client.submit.file.replication
mapreduce.tasktracker.dns.interface
mapreduce.tasktracker.dns.nameserver
mapreduce.tasktracker.http.threads
mapreduce.tasktracker.http.address
mapreduce.task.files.preserve.failedtasks
mapreduce.output.fileoutputformat.compress
mapreduce.output.fileoutputformat.compress.type
mapreduce.output.fileoutputformat.compress.codec
mapreduce.map.output.compress
mapreduce.map.output.compress.codec
mapreduce.task.userlog.limit.kb
yarn.app.mapreduce.am.container.log.limit.kb
yarn.app.mapreduce.task.container.log.backup
yarn.app.mapreduce.am.container.log.backup
yarn.app.mapreduce.shuffle.log.separate
yarn.app.mapreduce.shuffle.log.limit.kb
yarn.app.mapreduce.shuffle.log.backups
mapreduce.job.userlog.retain.hours
mapreduce.jobtracker.hosts.filename
mapreduce.jobtracker.hosts.exclude.filename
mapreduce.jobtracker.heartbeats.in.second
mapreduce.job.acl.modify-job
mapreduce.job.acl.view-job
mapreduce.tasktracker.indexcache.mb
mapreduce.job.token.tracking.ids.enabled
mapreduce.job.token.tracking.ids
mapreduce.task.merge.progress.records
mapreduce.task.combine.progress.records
mapreduce.job.reduce.slowstart.completed.maps
mapreduce.job.complete.cancel.delegation.tokens
mapreduce.jobtracker.persist.jobstatus.hours
mapreduce.jobtracker.persist.jobstatus.dir
mapreduce.task.profile
mapreduce.task.profile.maps
mapreduce.task.profile.reduces
mapreduce.task.profile.params
mapreduce.task.profile.map.params
mapreduce.task.profile.reduce.params
mapreduce.task.skip.start.attempts
mapreduce.map.skip.proc.count.autoincr
mapreduce.reduce.skip.proc.count.autoincr
mapreduce.job.skip.outdir
mapreduce.map.skip.maxrecords
mapreduce.reduce.skip.maxgroups
mapreduce.ifile.readahead
mapreduce.ifile.readahead.bytes
mapreduce.jobtracker.taskcache.levels
mapreduce.job.queue.name
mapreduce.job.tags
mapreduce.cluster.acls.enabled
mapreduce.job.acl.modify-job
mapreduce.job.acl.view-job
mapreduce.tasktracker.indexcache.mb
mapreduce.job.token.tracking.ids.enabled
mapreduce.job.token.tracking.ids
mapreduce.task.merge.progress.records
mapreduce.task.combine.progress.records
mapreduce.job.reduce.slowstart.completed.maps
mapreduce.job.complete.cancel.delegation.tokens
```

```
mapreduce.tasktracker.taskcontroller
mapreduce.tasktracker.group
mapreduce.shuffle.port
mapreduce.job.reduce.shuffle.consumer.pluggin.class
mapreduce.tasktracker.healthchecker.script.path
mapreduce.tasktracker.healthchecker.interval
mapreduce.tasktracker.healthchecker.script.timeout
mapreduce.job.counters.limit
mapreduce.framework.name
yarn.app.mapreduce.am.staging.dir
mapreduce.am.max.attempts
mapreduce.job.end.notification.url
mapreduce.job.end.notification.retry.attempts
mapreduce.job.end.notification.retry.interval
mapreduce.job.end.notification.max.attempts
mapreduce.job.log4j.properties.file
mapreduce.job.end.notification.max.retry.interval
yarn.app.mapreduce.am.env
yarn.app.mapreduce.am.admin.user.env
yarn.app.mapreduce.am.command.opts
yarn.app.mapreduce.am.admin.command.opts
yarn.app.mapreduce.am.job.task.listener.thread.count
mapreduce.jobhistory.done.dir
mapreduce.jobhistory.cleaner.enable
mapreduce.jobhistory.cleaner.interval.ms
mapreduce.jobhistory.max.age.ms
mapreduce.jobhistory.client.thread.count
mapreduce.jobhistory.daterstring.cache.size
mapreduce.jobhistory.joblist.cache.size
mapreduce.jobhistory.loadedjobs.cache.size
mapreduce.jobhistory.move.interval.ms
mapreduce.jobhistory.move.thread.count
mapreduce.jobhistory.store.class
mapreduce.jobhistory.miniicluster.fixed.ports
mapreduce.jobhistory.admin.address
mapreduce.jobhistory.admin.acl
mapreduce.jobhistory.recovery.enable
mapreduce.jobhistory.recovery.store.class
mapreduce.jobhistory.recovery.store.fs.uri
mapreduce.jobhistory.recovery.store.levelb.path
mapreduce.jobhistory.http.policy
yarn.app.mapreduce.am.containerlauncher.threadpool.initial.size
mapreduce.tasktracker.taskcontroller
mapreduce.tasktracker.group
mapreduce.shuffle.port
mapreduce.job.reduce.shuffle.consumer.pluggin.class
mapreduce.tasktracker.healthchecker.script.path
mapreduce.tasktracker.healthchecker.interval
mapreduce.tasktracker.healthchecker.script.timeout
mapreduce.job.counters.limit
mapreduce.framework.name
yarn.app.mapreduce.am.staging.dir
mapreduce.am.max.attempts
mapreduce.job.end.notification.url
mapreduce.job.end.notification.retry.attempts
mapreduce.job.end.notification.retry.interval
mapreduce.job.end.notification.max.attempts
mapreduce.job.log4j.properties.file
mapreduce.job.end.notification.max.retry.interval
yarn.app.mapreduce.am.env
yarn.app.mapreduce.am.admin.user.env
yarn.app.mapreduce.am.command.opts
yarn.app.mapreduce.am.admin.command.opts
yarn.app.mapreduce.am.job.task.listener.thread.count
yarn.app.mapreduce.am.job.client.port.range
yarn.app.mapreduce.am.job.committer.cancel.timeout
yarn.app.mapreduce.am.job.committer.commit.window
mapreduce.fileoutputcommitter.algorithm.version
yarn.app.mapreduce.am.scheduler.heartbeat.interval.ms
yarn.app.mapreduce.client.am.ipc.max.retries
yarn.app.mapreduce.client.am.ipc.max.retries.on.timeouts
yarn.app.mapreduce.client.max.retries
yarn.app.mapreduce.am.resource.mb
yarn.app.mapreduce.am.resource.cpu.vcores
yarn.app.mapreduce.am.hard.kill.timeout.ms
yarn.app.mapreduce.client.job.max.retries
yarn.app.mapreduce.client.job.retry.interval
mapreduce.application.classpath
mapreduce.app.submission.cross.platform
mapreduce.application.framework.path
mapreduce.job.classloader
mapreduce.job.classloader.system.classes
mapreduce.jobhistory.address
mapreduce.jobhistory.webapp.address
mapreduce.jobhistory.keystab
mapreduce.jobhistory.principal
mapreduce.jobhistory.intermediate.done.dir
```

Want best performance? Non-trivial!

Over **190** configuration parameters...

mapreduce.jobtracker.jobhistory.location	mapreduce.job.reduces	mapreduce.job.speculative.retry-after-no-speculative	mapreduce.map.output.compress.codec	mapreduce.task.profile.params	mapreduce.jobhistory.done-dir
mapreduce.jobtracker.jobhistory.task.num	mapreduce.jobtracker.restart.recover	mapreduce.job.speculative.retry-after-speculative	map.sort.class	mapreduce.task.profile.map.params	mapreduce.jobhistory.cleaner.enable
mapreduce.jobtracker.jobhistory.block.size	mapreduce.jobtracker.jobhistory.block.size	mapreduce.job.speculative.retry-after-speculative	mapreduce.task.userlog.limit.kb	mapreduce.task.profile.reduce.params	mapreduce.jobhistory.cleaner.interval.ms
mapreduce.jobtracker.task.scheduler	mapreduce.jobtracker.task.scheduler	mapreduce.job.speculative.retry-after-speculative	yarn.app.mapreduce.am.container.log.limit.kb	mapreduce.task.skip.start.attempts	mapreduce.jobhistory.max-age.ms
mapreduce.job.user.history.location	mapreduce.job.max.split.locations	mapreduce.job.speculative.retry-after-speculative	yarn.app.mapreduce.am.container.log.limit.kb	mapreduce.task.skip.proc.count.autoincr	mapreduce.jobhistory.client.thread-count
mapreduce.jobtracker.jobhistory.complete.d.location	mapreduce.job.split.metainfo.maxsize	mapreduce.job.speculative.retry-after-speculative	yarn.app.mapreduce.task.container.log.backup.kb	mapreduce.reduce.skip.proc.count.autoincr	mapreduce.jobhistory.daterange.cache.size
mapreduce.job.committer.setup.cleanup.enabled	mapreduce.job.split.metainfo.maxsize	mapreduce.job.speculative.retry-after-speculative	yarn.app.mapreduce.task.container.log.backup.kb	mapreduce.job.skip.outdir	mapreduce.jobhistory.joblist.cache.size
mapreduce.task.io.sort.factor	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	yarn.app.mapreduce.shuffle.log.separate	mapreduce.map.skip.maxrecords	mapreduce.jobhistory.joblist.cache.size
mapreduce.task.io.sort.mb	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	yarn.app.mapreduce.shuffle.log.limit.kb	mapreduce.reduce.skip.maxgroups	mapreduce.jobhistory.loadedjobs.cache.size
mapreduce.map.sort.spill.percent	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	yarn.app.mapreduce.shuffle.log.backups	mapreduce.ifile.readahead	mapreduce.jobhistory.move.interval.ms
mapreduce.jobtracker.address	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.userlog.retain.hours	mapreduce.jobtracker.taskcache.levels	mapreduce.jobhistory.move.thread-count
mapreduce.local.clientfactory.class.name	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.jobtracker.taskcache.levels	mapreduce.job.queuename	mapreduce.jobhistory.move.thread-count
mapreduce.jobtracker.http.address	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.tags	mapreduce.job.tags	mapreduce.jobhistory.store.class
mapreduce.jobtracker.handler.count	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.cluster.acls.enabled	mapreduce.job.acl-modify-job	mapreduce.jobhistory.store.class
mapreduce.tasktracker.report.address	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.acl-view-job	mapreduce.job.acl-view-job	mapreduce.jobhistory.store.level
mapreduce.cluster.local.dir	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.jobtracker.system.dir	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.jobtracker.staging.root.dir	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.cluster.temp.dir	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.tasktracker.local.dir.minspaceart	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.tasktracker.local.dir.minspacekill	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.jobtracker.expire.trackers.interval	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.tasktracker.instrumentation	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.tasktracker.resourcecalculator.login	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.tasktracker.taskmemorymanager.monitoringinterval	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.tasktracker.tasks.sleep.time.before.kill	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
mapreduce.job.maps	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level
	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.speculative.retry-after-speculative	mapreduce.job.token.tracking.ids.enabled	mapreduce.job.token.tracking.ids.enabled	mapreduce.jobhistory.store.level

Want best performance? Non-trivial!

Over **190** configuration parameters...

Lack the expertise to tune system internals

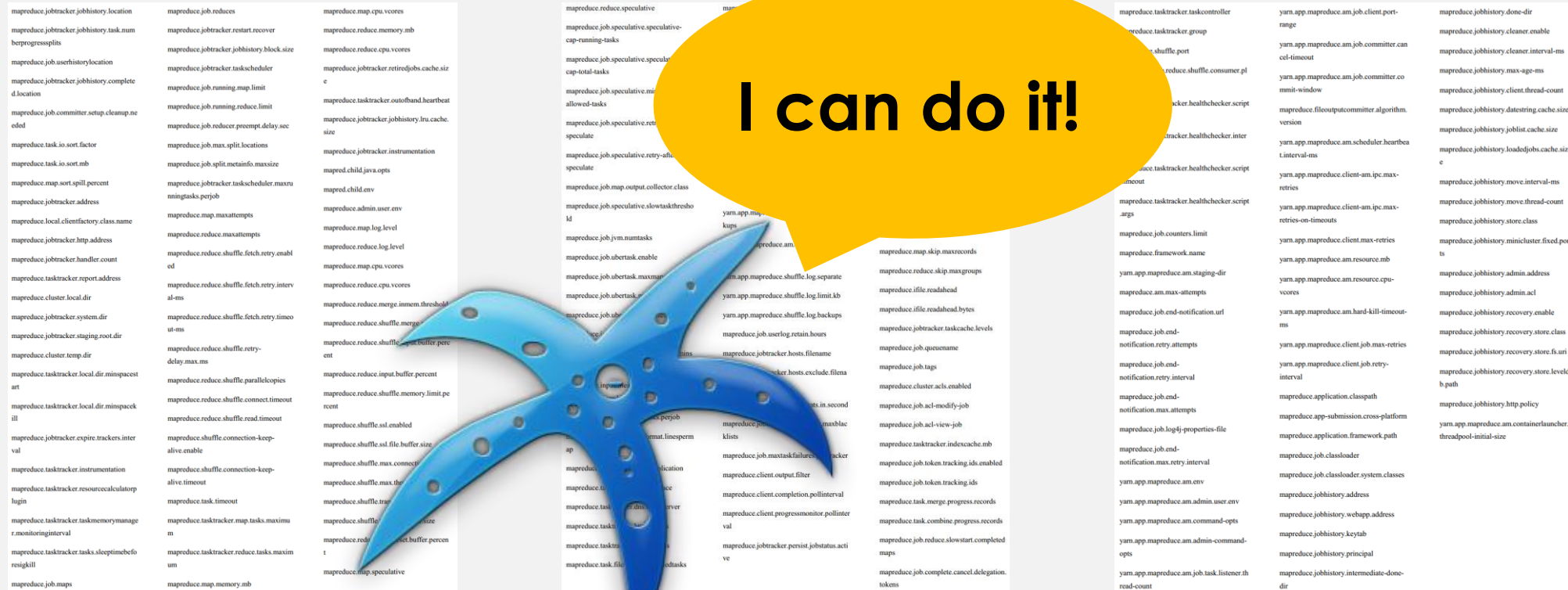
Want best performance? Non-trivial!

Over **190** configuration parameters...

Lack the expertise to tune system internals

What if there is a **Self-tuning** System?

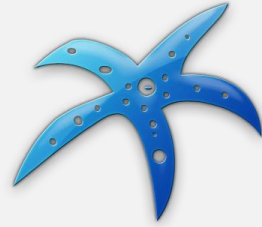
Want best performance? Non-trivial!



I can do it!

The background consists of a grid of small, illegible text snippets, likely representing configuration parameters or code snippets related to Hadoop MapReduce. A large yellow speech bubble with the text "I can do it!" is centered over the grid. A blue starburst graphic is positioned at the bottom center, overlapping the grid.

Starfish is **MADDER**



Data-lifecycle Awareness Do more than just queries, optimize the movement, storage, and processing of big data

Elasticity Dynamically adjust resource usage and operational costs based on workload and user requirements

Robustness Provide storage and querying services even in the event of some failures

Goal of Starfish



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.

Purpose of this paper

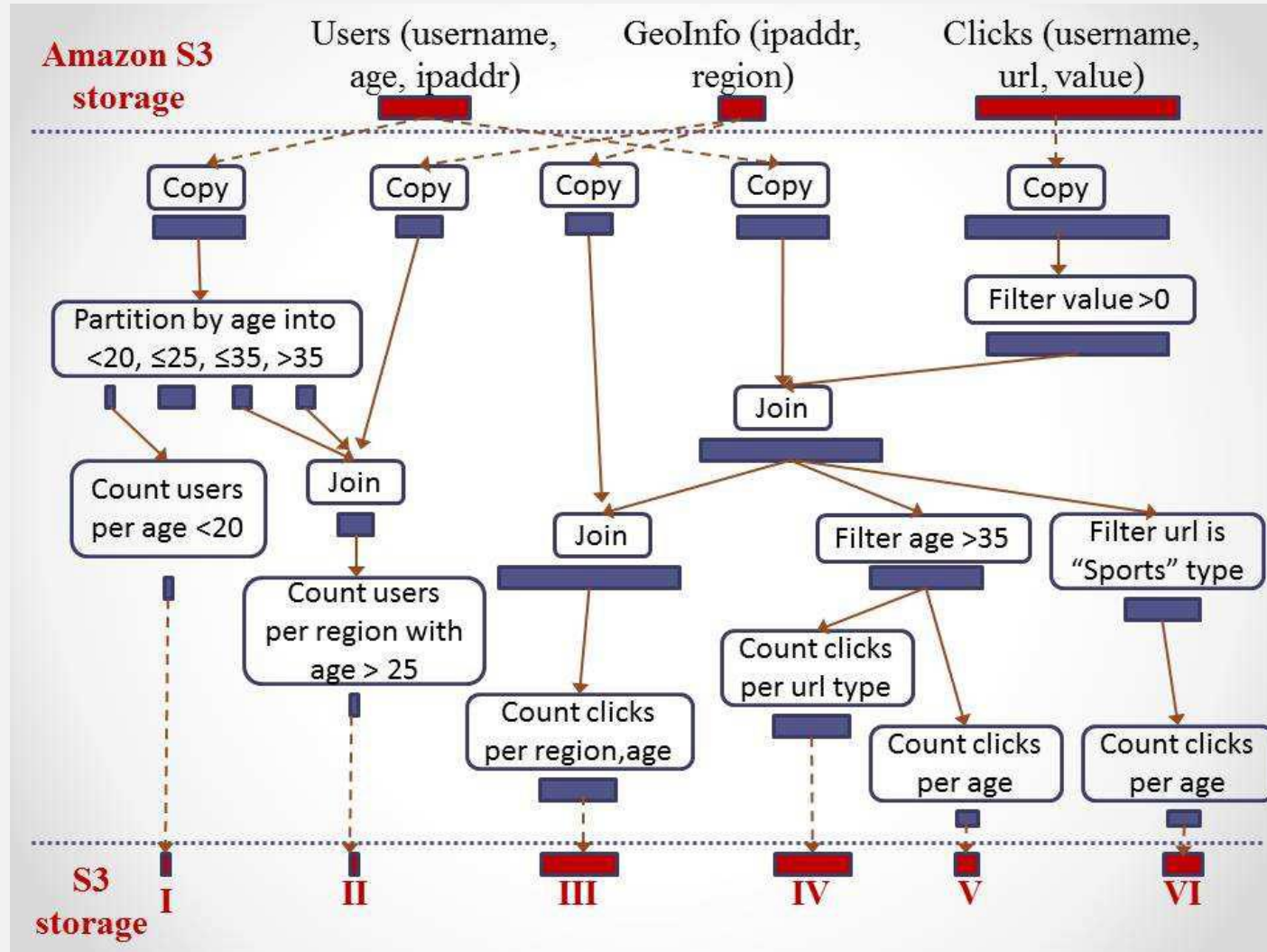
Using **experimental results** to illustrate the challenges in each component and to motivate Starfish's solution approach.



2

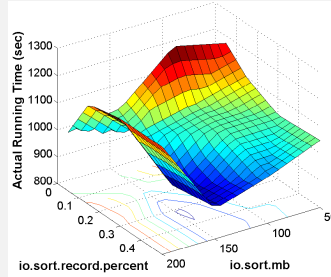
Overview

Example analytics workload

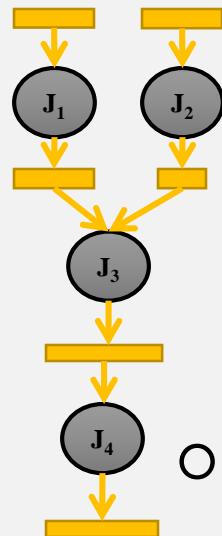


Tuning problems

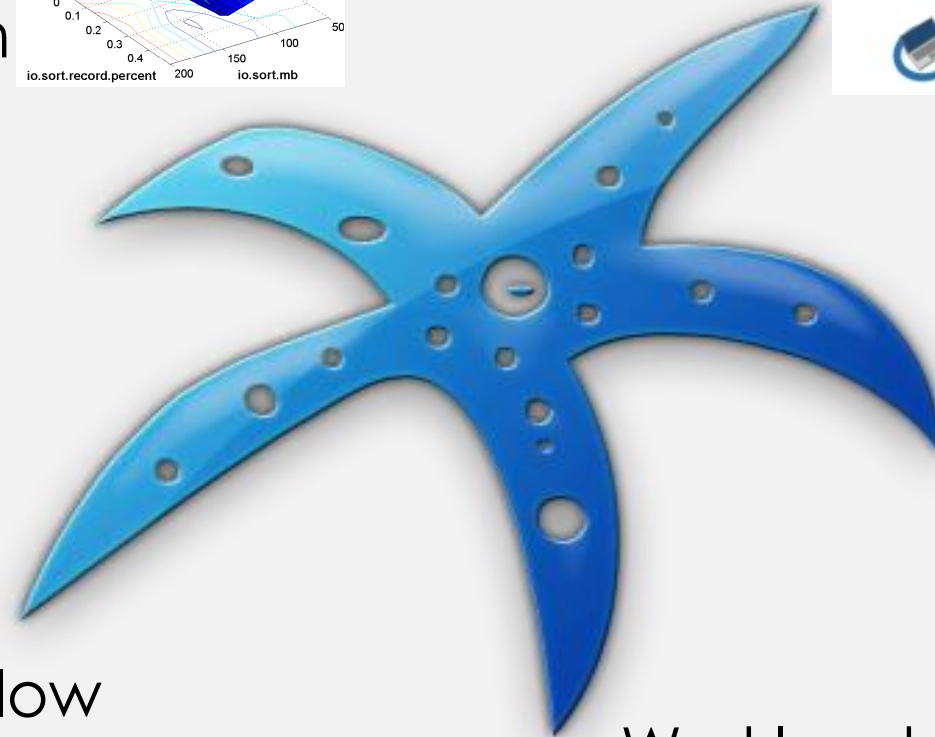
Job-level
MapReduce
configuration



Cluster sizing



Workflow
optimization



Workload
management

Data
layout
tuning

Core Approaches to Tuning

Profiler

Collects concise summaries of execution

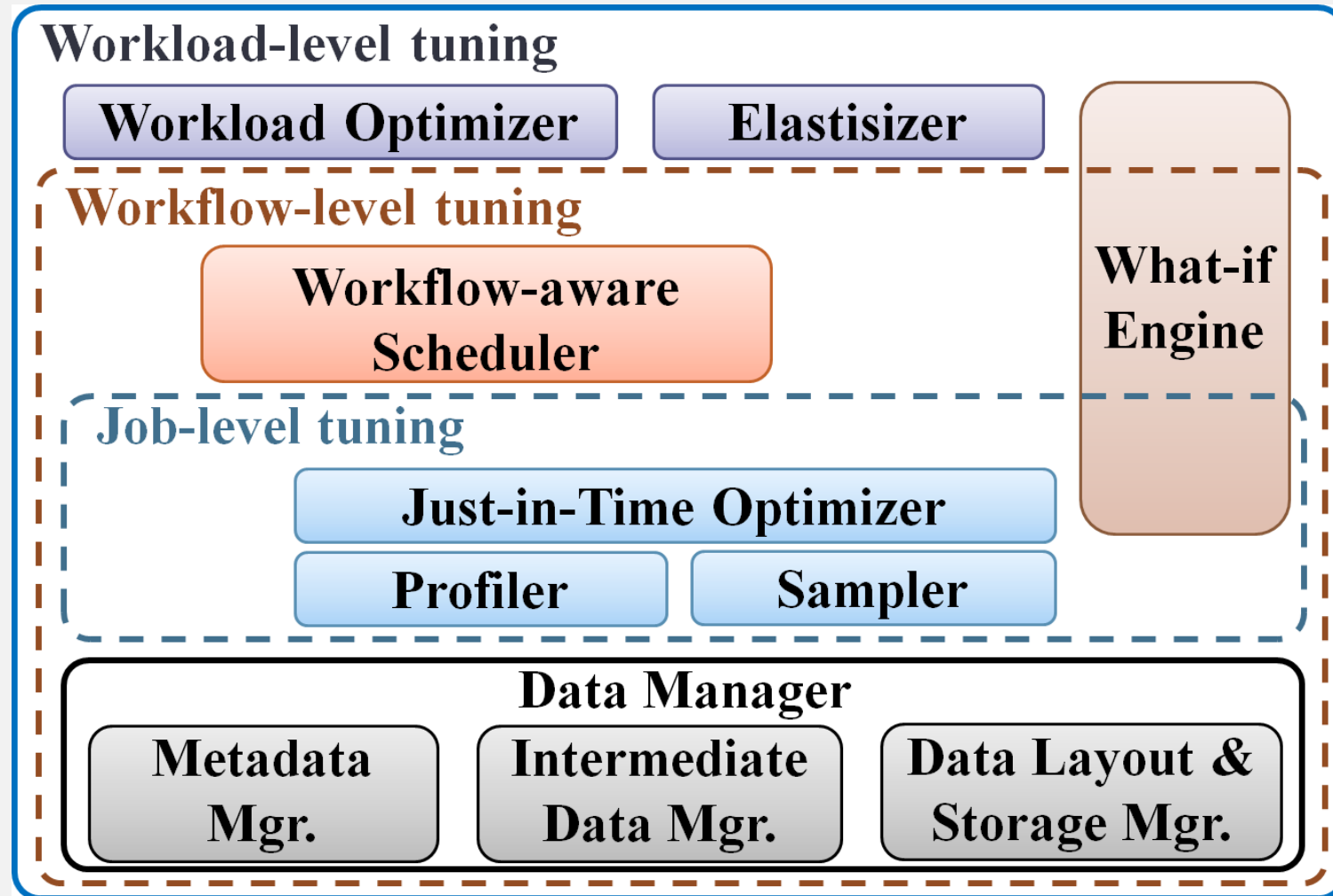
What-if Engine

Estimates impact of hypothetical changes
on execution

Optimizers

Search through space of tuning choices

Architecture



Basic idea of each level

Job-level Tuning

The ***Just-in-Time Optimizer*** optimizes parameters based on the results of the ***Profiler***, the ***Sampler***, and the ***What-if Engine***.

Workflow-level Tuning

Workflow-aware Scheduler addresses concerns, such as unbalanced data layout, in conjunction with the ***What-if Engine*** and the ***Data Manager***.

Workload-level Tuning

The ***Workload Optimizer*** generates optimized workflows with the help of the ***Elastisizer*** and the ***What-if Engine***.

LASTWORD:

Language for **Starfish Workloads** and **Data**

Clients submit WORKLOADS expressed in Lastword. It is **not a human-interactable language**.

Starfish provides **language translators** to convert workloads specified in high level languages to Lastword.



Job-level Tuning

Three Components



Just-in-Time Optimizer

Automatically selects efficient execution techniques for MapReduce jobs.

Profiler

Collects detailed summaries of jobs on a task-by-task basis.

Sampler

Collects statistics about input, intermediate, and output data of a MapReduce job.

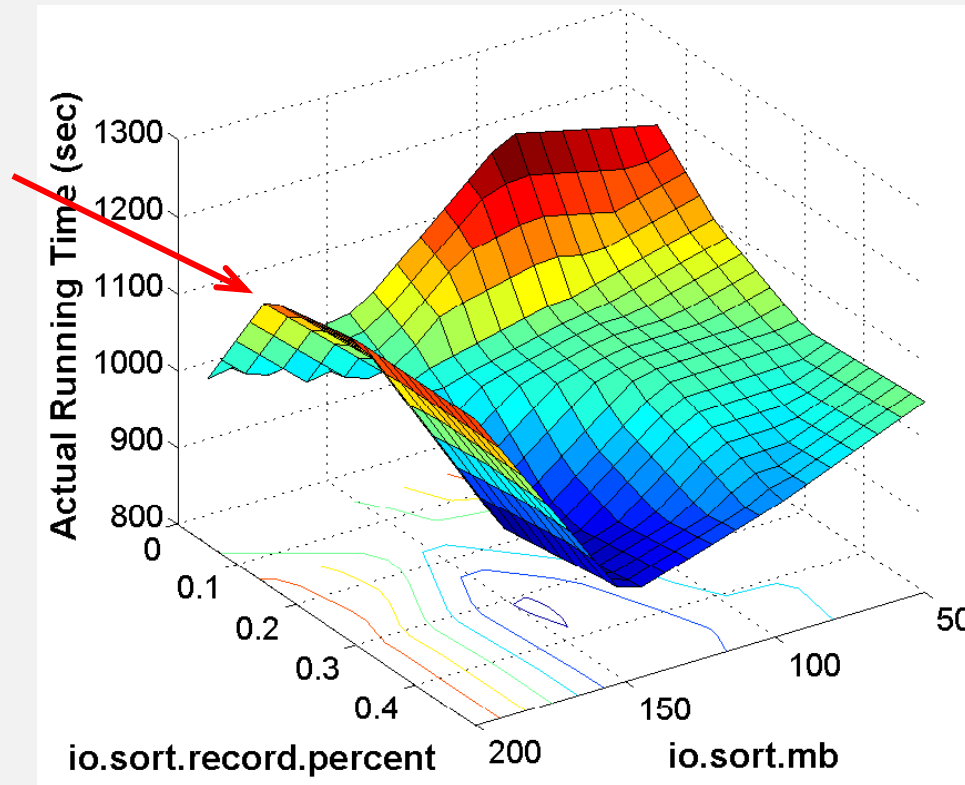
What Controls the Execution of a MR Job?

- Number of map tasks
- Number of reduce tasks
- Whether combine function should be used
- Partitioning of map outputs to reduce tasks
- Memory allocation to task-level buffers
- Whether output data from tasks should be compressed
- ...

$\text{job } j = \langle \text{program } p, \text{ data } d, \text{ resources } r, \text{ configuration } c \rangle$

What's Wrong with the Rule-of-thumb Settings?

Rule-of-thumb
settings



Response surface of a
Word Co-occurrence
MapReduce program

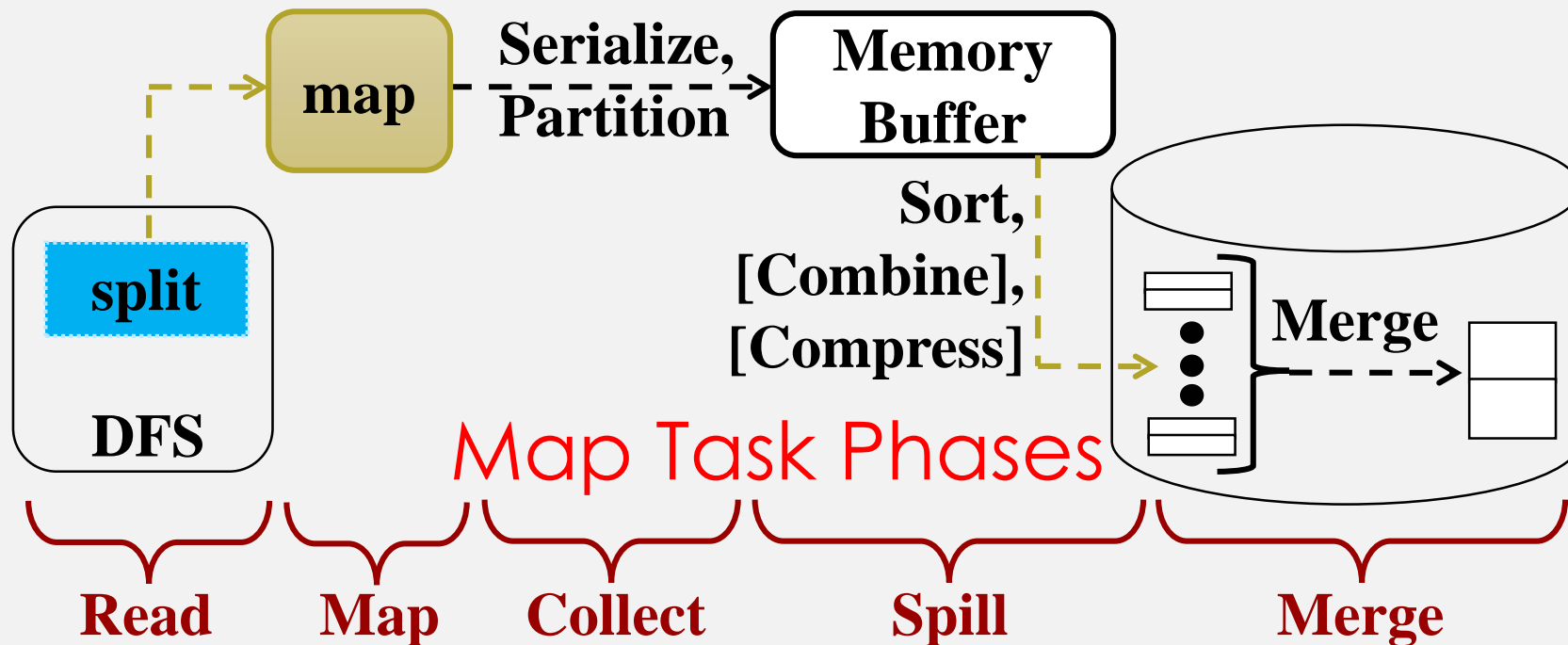
Dataset: 10 GB Wikipedia

MapReduce Job Tuning in a Nutshell

- **Goal:** $perf = F(p, d, r, c)$, $c_{opt} = \arg \min_{c \in S} F(p, d, r, c)$
- **Challenges:** p is an arbitrary MapReduce program; c is high-dimensional; ...
- **Profiler:** Runs p to collect a **job profile** (concise execution summary) of $\langle p, d_1, r_1, c_1 \rangle$
- **What-if Engine:** Given profile of $\langle p, d_1, r_1, c_1 \rangle$, estimates **virtual profile** for $\langle p, d_2, r_2, c_2 \rangle$
- **Optimizer:** Enumerates and searches through the **optimization space S** efficiently

Job Profile

- Concise representation of program execution as a job
- Records information at the level of “task phases”
- Generated by Profiler through measurement or by the What-if Engine through estimation



Job Profile Fields

1

Dataflow: amount of data flowing through task phases

- Map output bytes
- Number of spills
- Number of records in buffer per spill
- ...

2

Dataflow Statistics: statistical information about dataflow

- Width of input key-value pairs
- Map selectivity in terms of records
- Map output compression ratio
- ...

3

Costs: execution times at the level of task phases

- Read phase time in the map task
- Map phase time in the map task
- Spill phase time in the map task
- ...

4

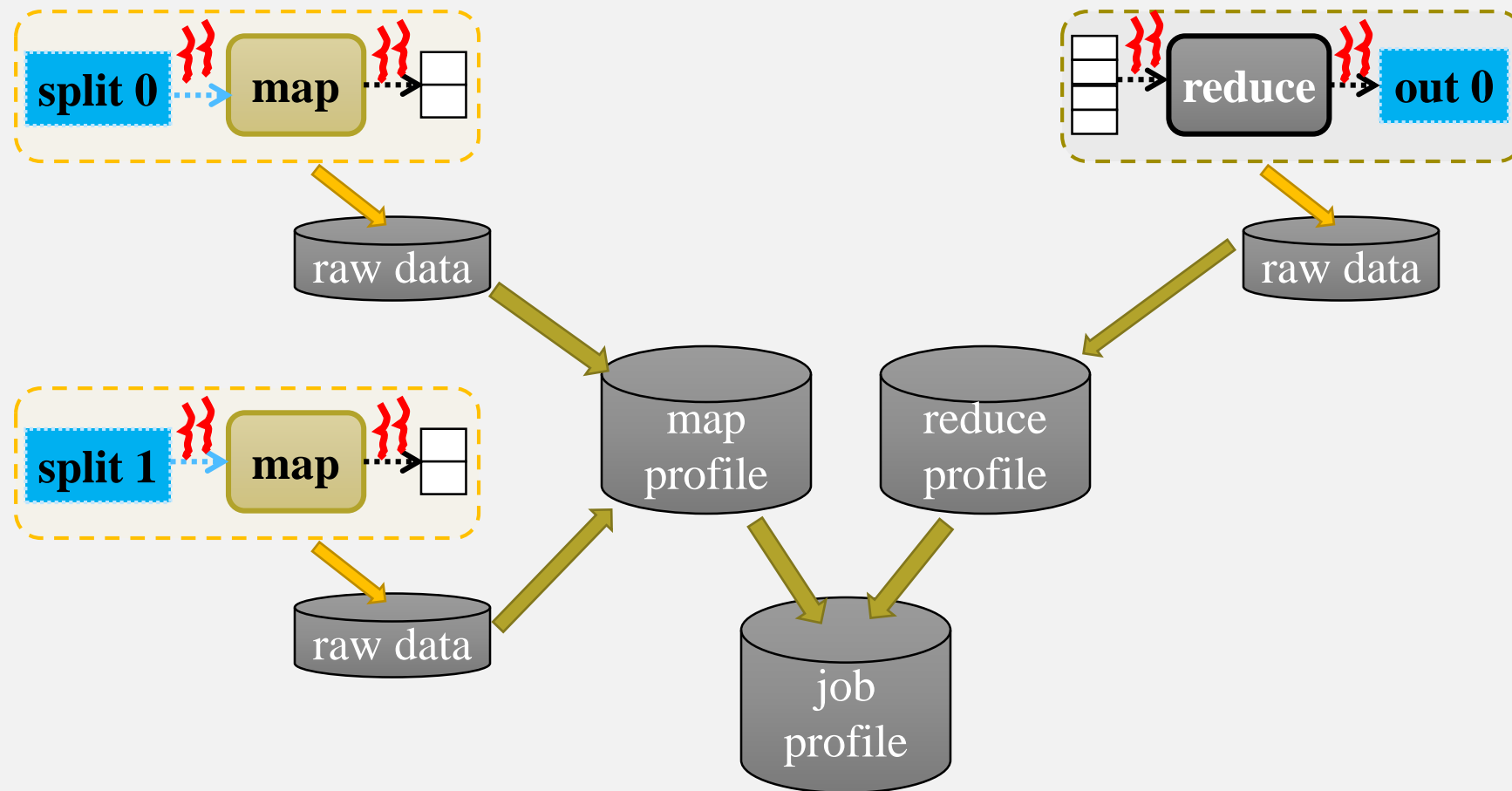
Cost Statistics: statistical information about resource costs

- I/O cost for reading from local disk per byte
- CPU cost for executing the Mapper per record
- ...

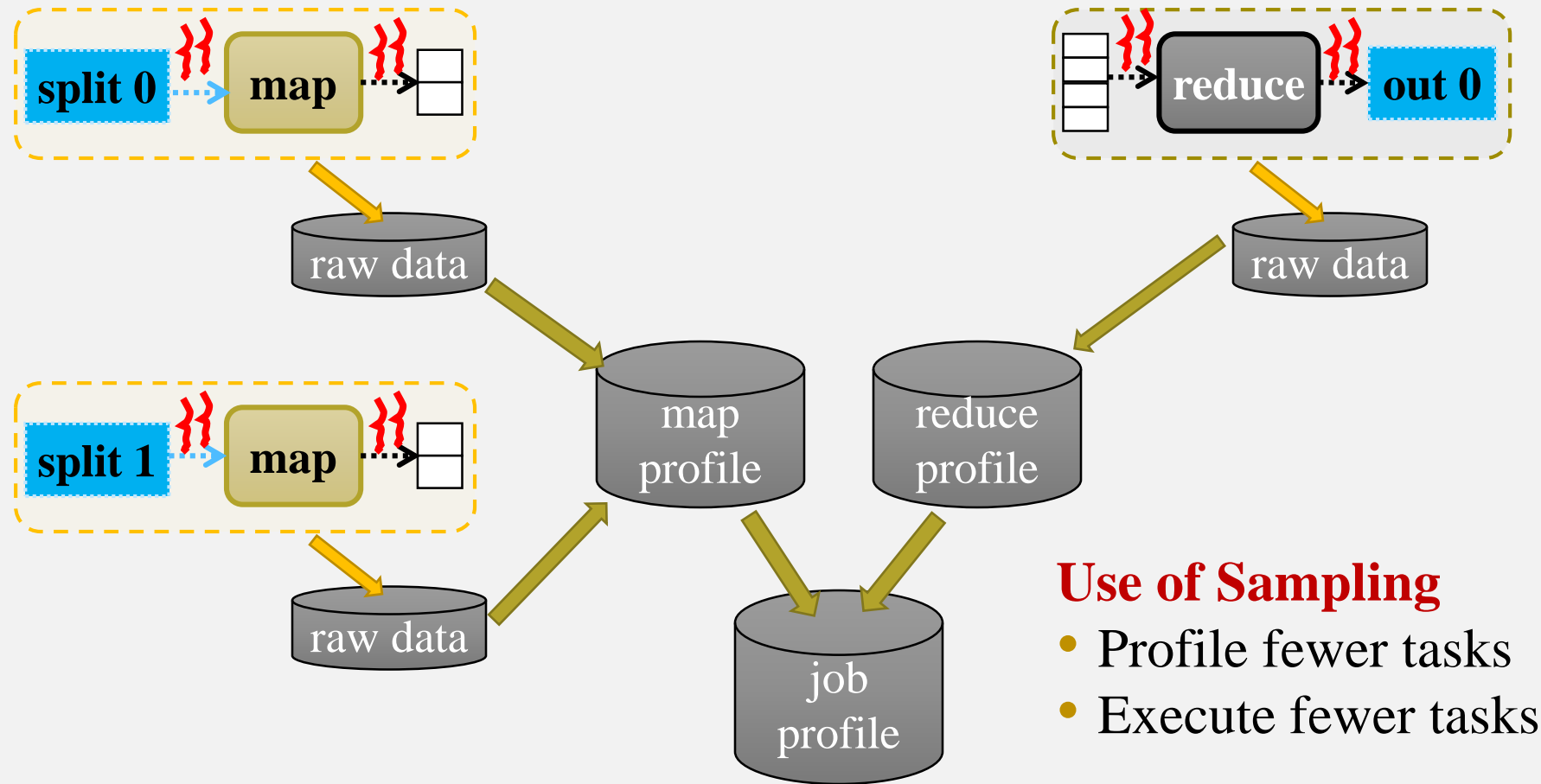
Profiling Using Dynamic Instrumentation

- **Dynamic Instrumentation** collects run-time monitoring information from unmodified MapReduce programs running on Hadoop.
- **BTrace** is the current implementation of the *Profiler*.
 - A safe, dynamic tracing tool for the Java platform
- Three views are exposed to capture the features of a job's execution:
 - **Timings view**: The amount of time spent in each sub-phases
 - **Data-flow view**: The amount of data processed during each sub-phases
 - **Resource-level view**: The usage trends of CPU, memory, I/O, and network resources during each sub-phases

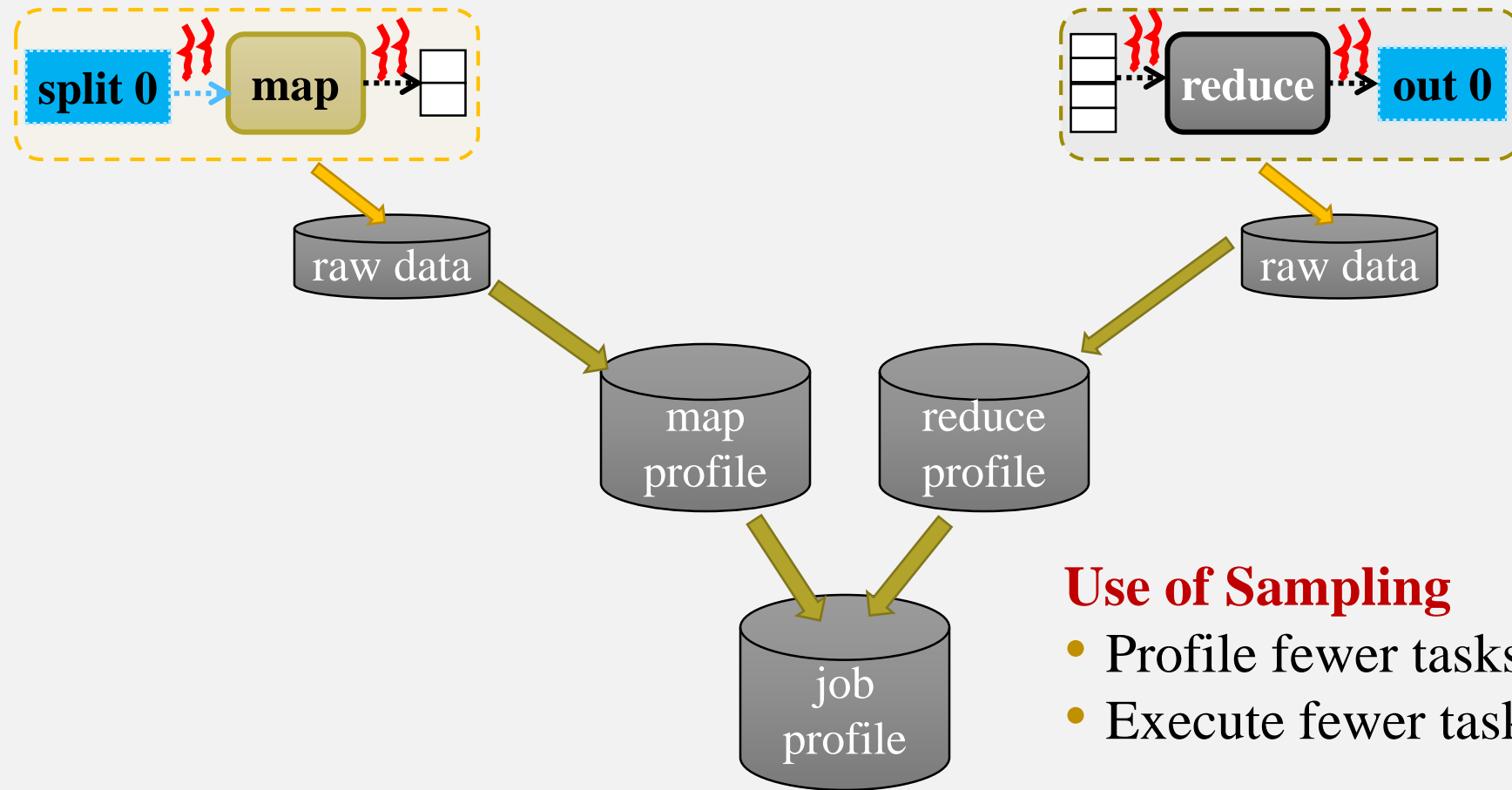
Generating Profiles



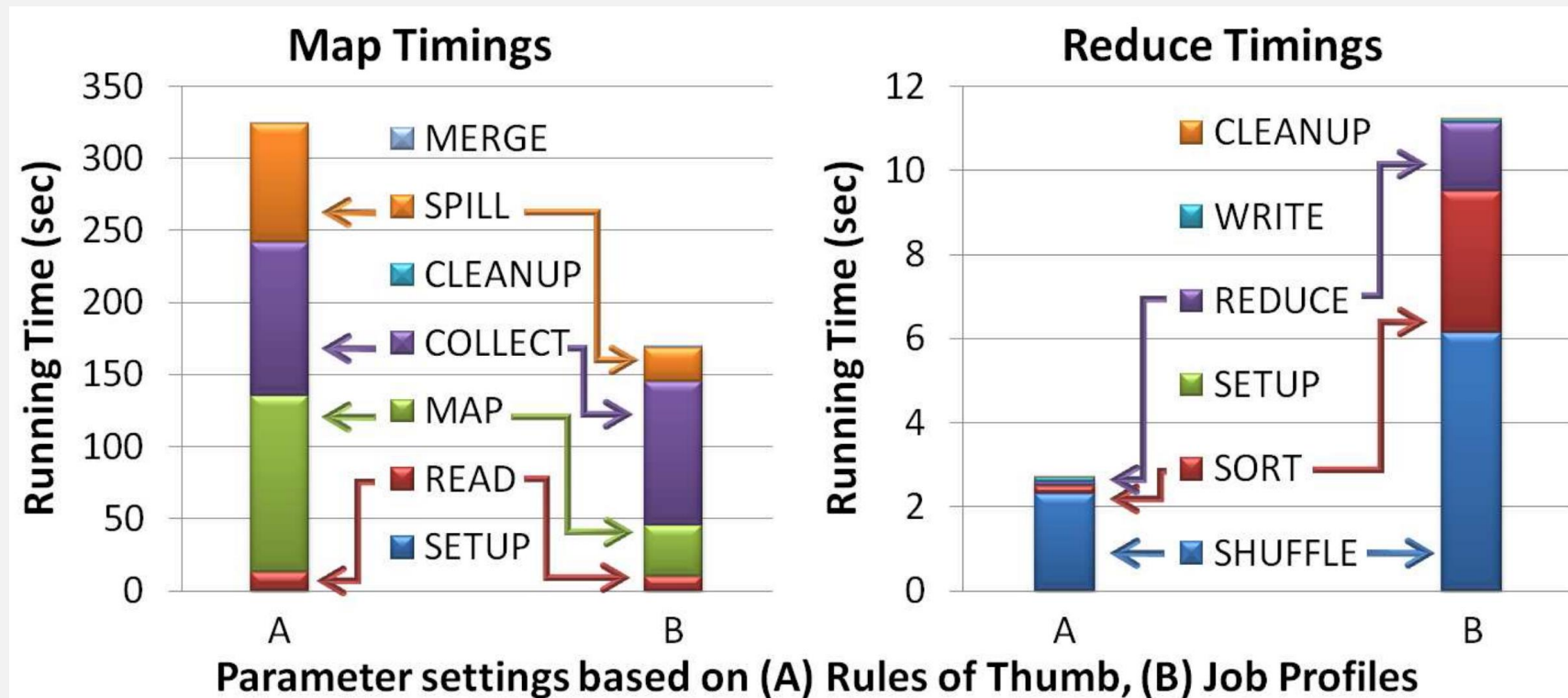
Generating Profiles



Generating Profiles



Example of a Timings View

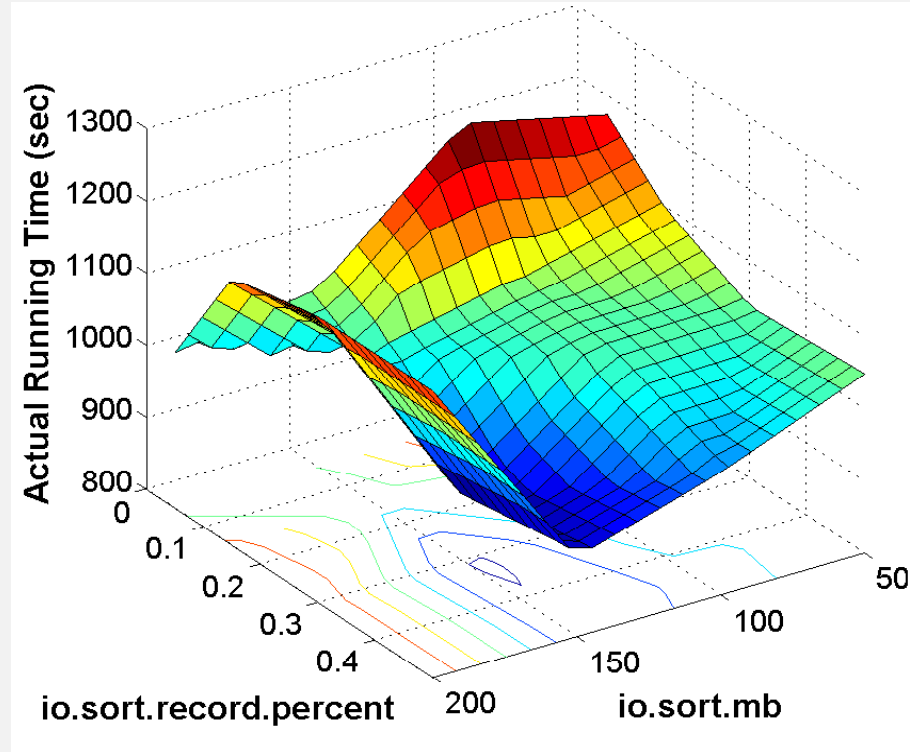


Predicting Job Performance in Hadoop

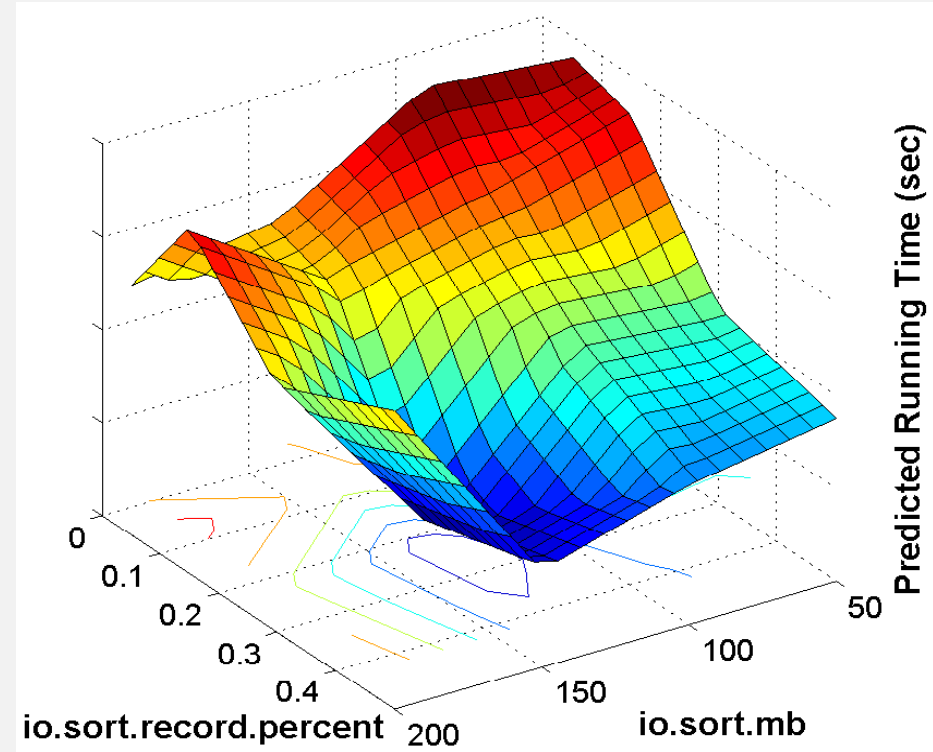
The What-if Engine

- Uses the **job profile** and a set of **performance models** to **estimate** the new profile if the job were to be run using S
- Four inputs are required:
 - The **job profile** $\langle p, d_1, r_1, c_1 \rangle$ generated for this job J by the *Profiler*
 - The **new setting** c_2 of the job configuration parameters using which Job J will be run
 - The **size, layout, and compression information** d_2 of the input dataset on which Job J will be run
 - The **cluster setup and resource allocation** r_2 that will be used to run Job J
- **Virtual job profile** $\langle p, d_2, r_2, c_2 \rangle$: Contains **the predicted Timings** and **Data-flow views** of the job when run with the new parameter settings

Estimate Result of the What-if Engine



True Surface



Estimated Surface

Experiment Results

	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

- Single-rack Hadoop cluster running on 16 Amazon EC2 nodes of the c1.medium type.
- Each node runs at most 3 map tasks and 2 reduce tasks concurrently.
- WordCount processes 30GB of data generated using the RandomTextWriter program in Hadoop.
- TeraSort processes 50GB of data generated using Hadoop's TeraGen program.



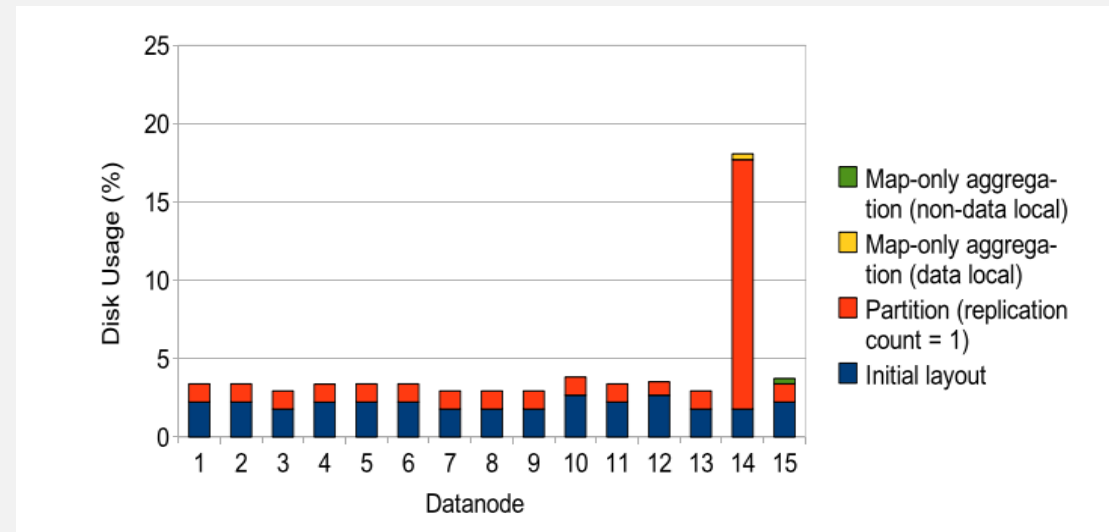
Workflow-aware Tuning

Workflow-aware Tuning

- **Skewed Data**
- **Data-Layerout-unaware Tasks Scheduling**
- **Addition or dropping of nodes Without data rebalancing**

- **An Example Easy to Understand**

Consequence of one execution of a large job



Main Problem: Unbalanced Data Layouts in Hadoop

A Slightly More Complicated Example Showing that Unbalanced Data Layout is Not Only A Coincidence

Tasks are usually scheduled where data is stored.



HDFS always write the first replica of any block on the same node the writer runs. (Overutilize)

Unbalanced data layouts are serious problems in big data analytics because they are prominent causes of tasks failure and performance degradation.

Dilemma faced:

Exploiting data locality pros:

1. Less disk I/O. Save shuffling time.

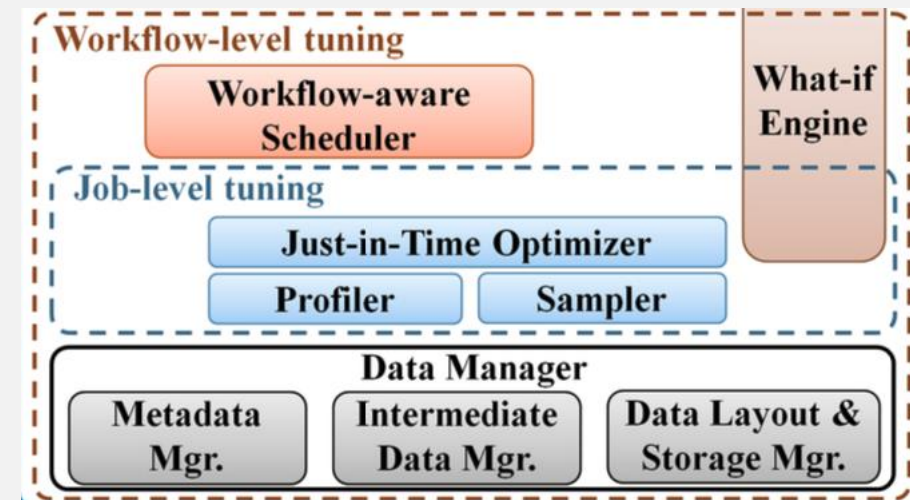
Exploiting data locality cons (in this context):

1. Performance degradation due to reduced parallelism.
2. Making data layout further unbalanced.

Need a tuning method to achieve automatic optimization.

Workflow-aware Tuning

A Workflow-aware Scheduler can ensure that job-level optimization and scheduling policies are **coordinated tightly** with the policies for data placement employed by the underlying distributed filesystem.



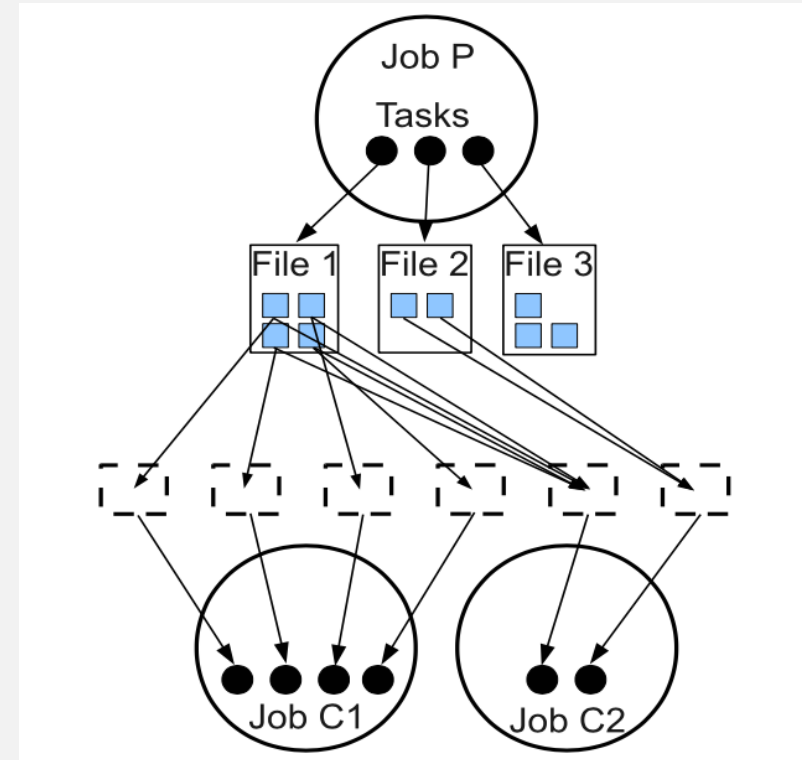
Rather than making decisions that are locally optimal for individual MapReduce jobs, Starfish's Workflow-aware Scheduler makes decisions by considering producer-consumer relationships among jobs in the workflow.

Workflow-aware Tuning

Some information this tuning needs:

1. What parts of the data output by a job are used by downstream jobs in the workflow?

In this example, both file 1 and file 2 will be used by downstream jobs



2. What is the unit of data-level parallelism in each job that reads the data output by a job?

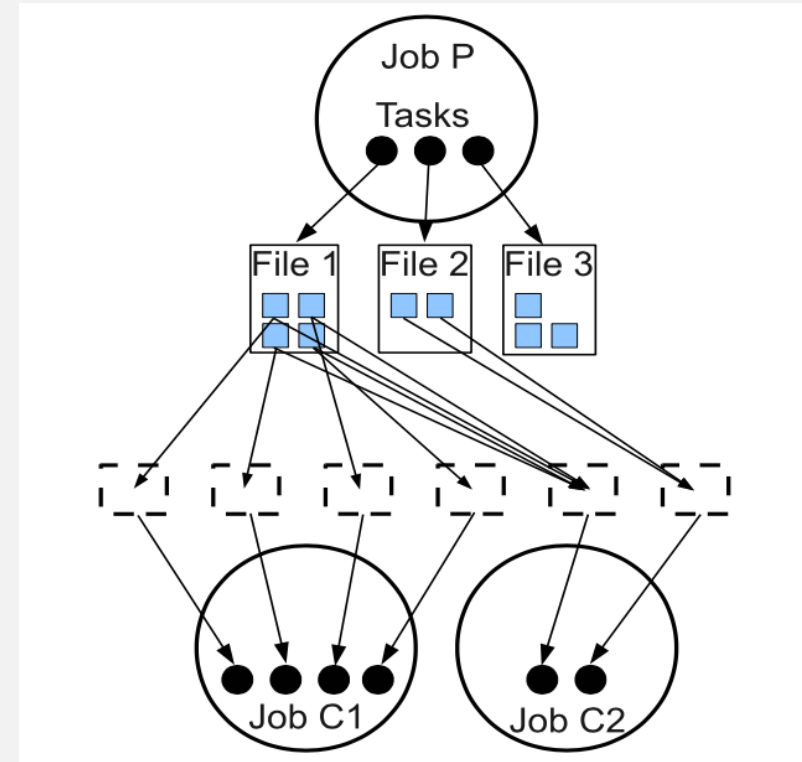
Data-parallel reader tasks of Job C1 read and process one data block each.

One file each for Job C2.

Workflow-aware Tuning

Some information this tuning needs (cont'd):

3. What is the expected running time of Job P if the Round Robin block placement policy is used for P's output files?
4. What will the new data layout in the cluster be if the Round Robin block placement policy is used for P's output files?
5. What is the expected running time of Job C1 (C2) if its input data layout is the one in the answer to Question 4?



...

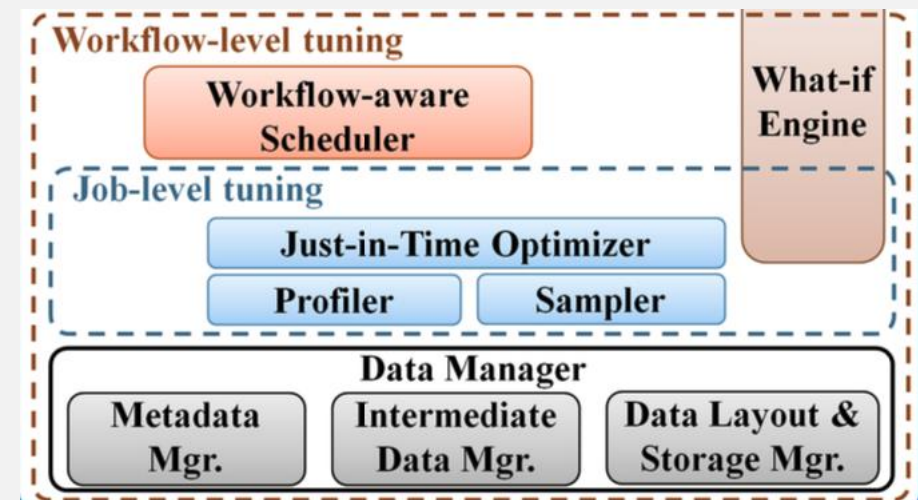
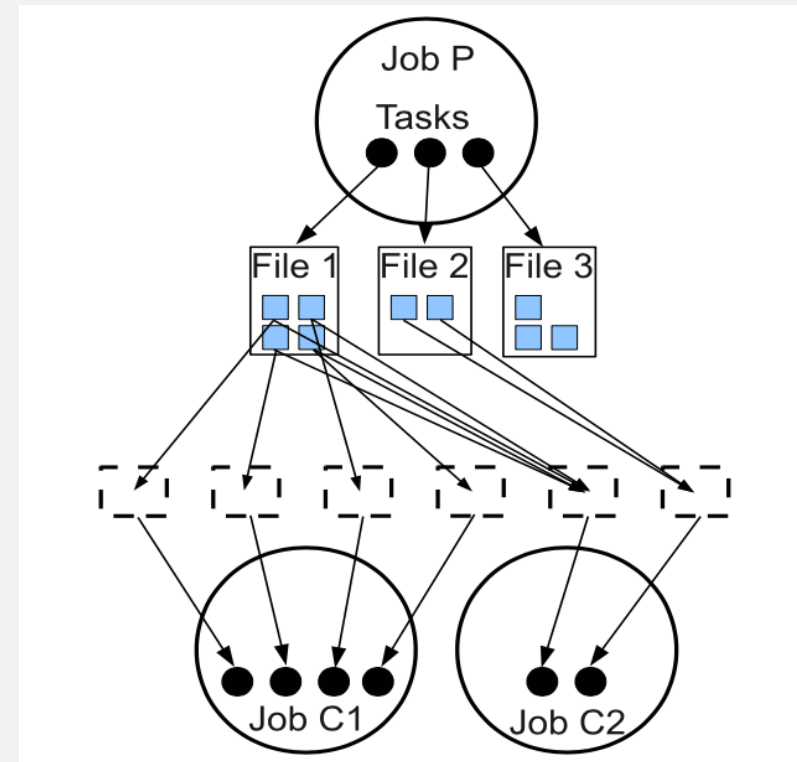
Workflow-aware Tuning

Some information this tuning needs (cont'd):

Questions in the previous slide are answered by the What-if Engine based on a simulation of the main aspects of workflow execution.

Action Involves Simulating:

- MapReduce job execution
- Task scheduling
- HDFS block placement policies



Workflow-aware Tuning

This system answers these questions:

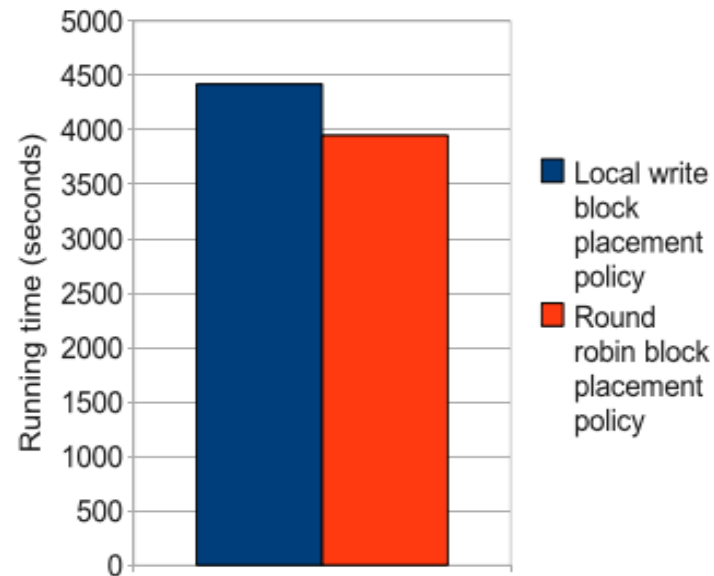
- What block placement policy to use in the distributed filesystem for the output file of a job?
Implemented a Round Robin block placement policy in HDFS.
- How many replicas to store for the blocks of a file?
Replication helps tasks including heavily-accessed files
- What size to use for blocks of a file?
- Should a job's output files be compressed for storage?
Transfer additional CPU cost for saving local disks I/O and network cost.

Workflow-aware Tuning

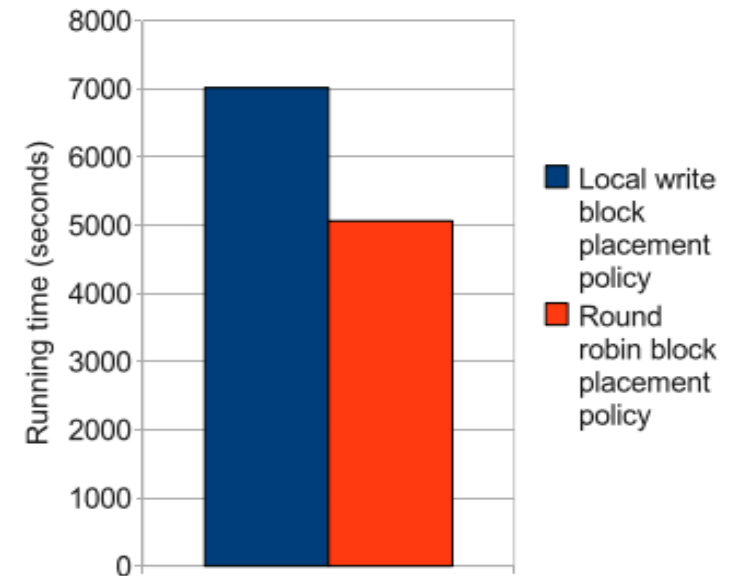
Experiment Result

Parameters:

- Replication Factor: 1
- Block Size: 128MB
- Compression: None



A partition job



A two-job workflow

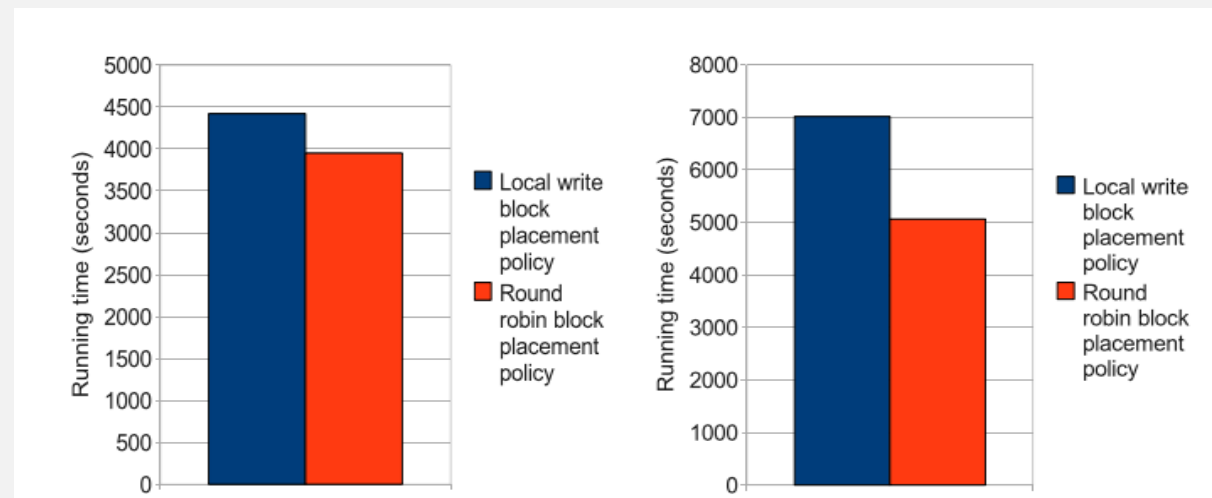
Workflow-aware Tuning

Reason for the winning of Round Robin

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

PLUS

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.



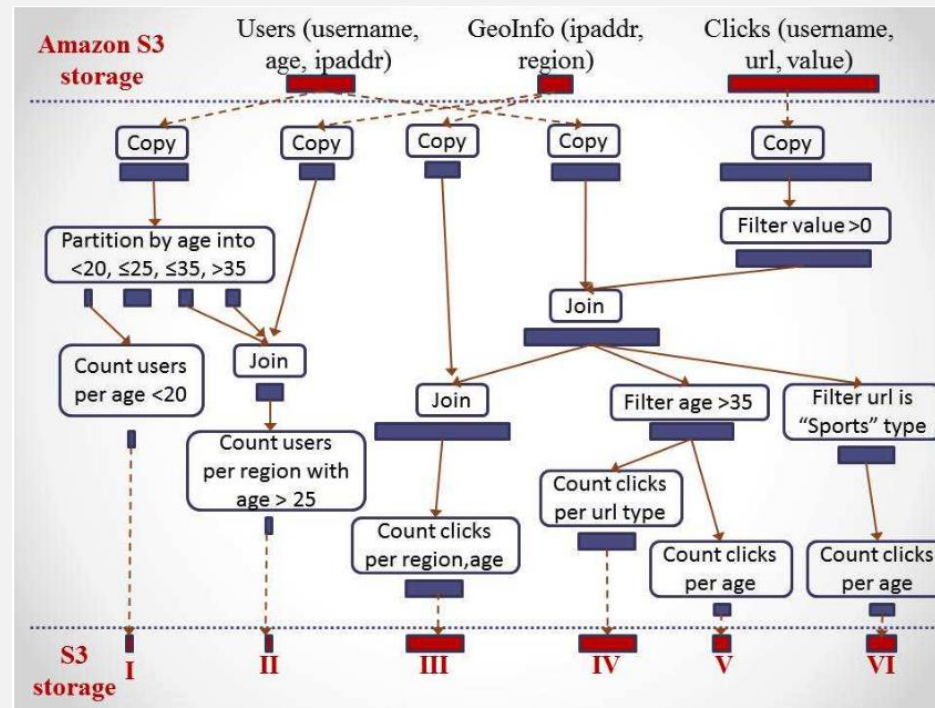


5 Workload-level Tuning

Workload-level Tuning

Workload Optimizer

This optimizer uses the What-if Engine to do a cost-based estimation of whether the graph-to-graph transformation will improve performance.



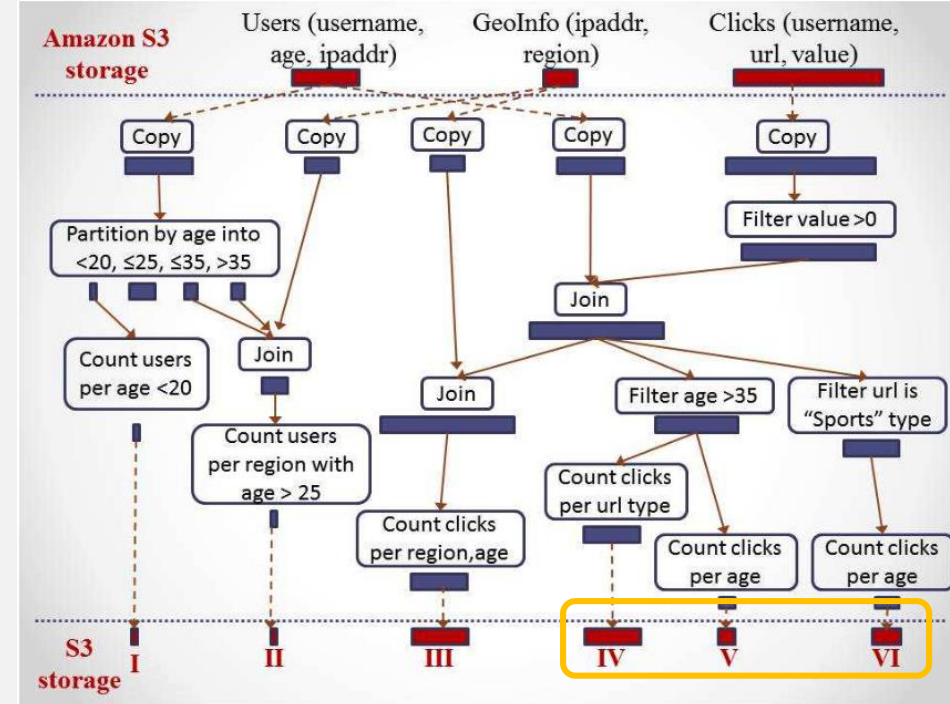
Workload-level Tuning

Workload Optimizer

Introducing Jumbo Operator

The results IV, V and VI can be represented as a Select-Project-Aggregation (SPA) expression over the join.

Jumbo operator can process any number of logical SPA Nodes over the same table in a single job.



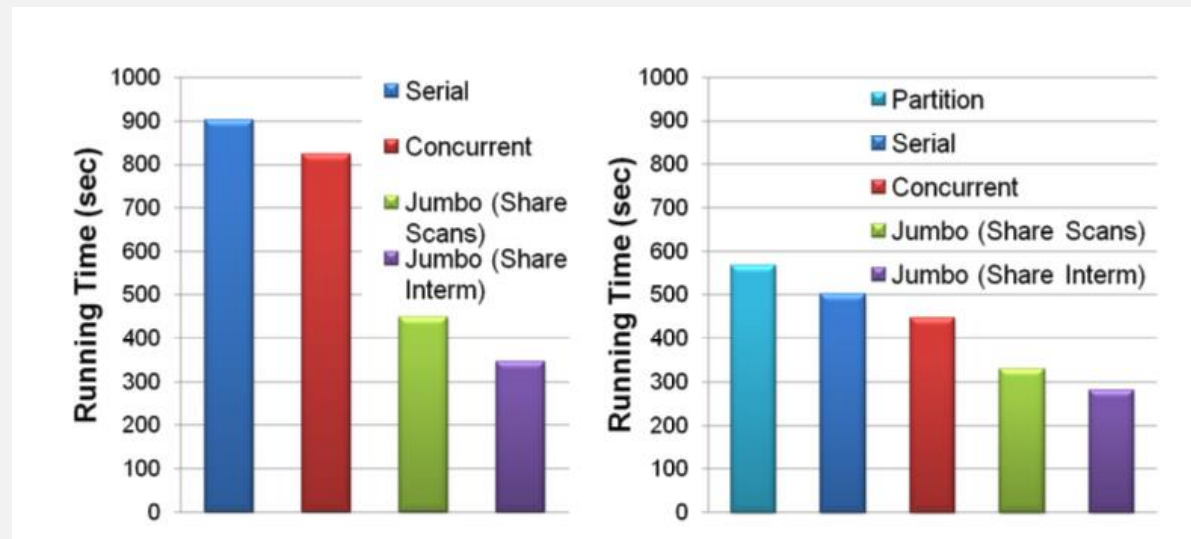
The Jumbo operator enables sharing of all or some of the map-side scan and computation, sorting and shuffling, as well as the reduce-side scan, computation, and output generation.

At the same time, the Jumbo operator can help the scheduler to better utilize the bounded number of map and reduce task slots in a Hadoop cluster.

Workload-level Tuning

Workload Optimizer

If possible, Running a MapReduce job to partition Users based certain attribute will enable the four workflows to prune out irrelevant partitions efficiently.



Processing multiple SPA workflow nodes on the same input dataset

Generating the partitions has significant. But possibilities exist to hide or reduce this overhead by combining partitioning with a previous job like data copying. 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.

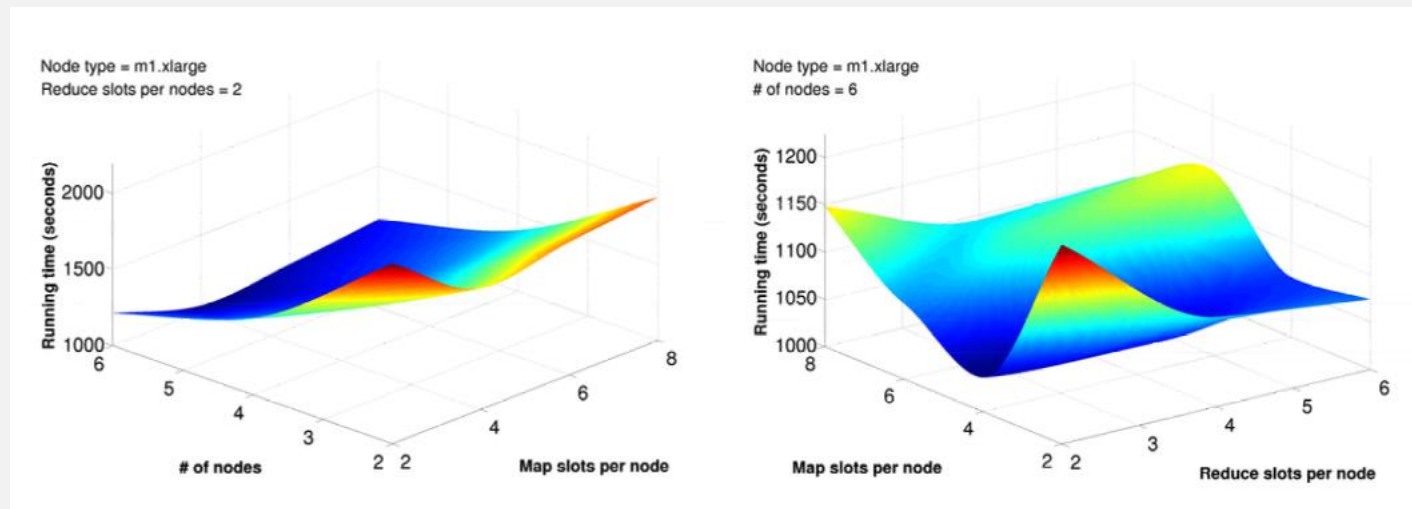
Workload-level Tuning

Elastisizer

Introducing an especially made optimizer for Amazon Web Service.

It automatically free users from the burden of setting the number and type of EC2 nodes.

These parameters are Hadoop-level configuration parameters.

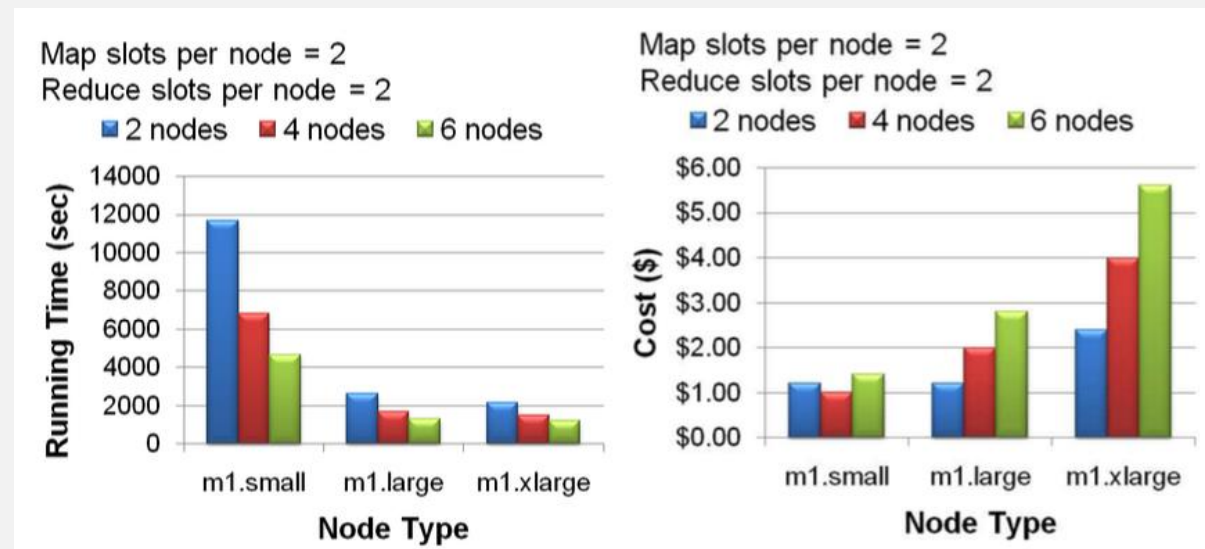


Workload performance under various cluster and Hadoop configurations
on Amazon Elastic MapReduce

Workload-level Tuning

Elastisizer

The user could have multiple preferences and constraints for the workload, which poses a multi-objective optimization problem. For example, the goal may be to minimize the monetary cost incurred to run the workload, subject to a maximum tolerable workload completion time.



Performance Vs. costs for a workload on Amazon Elastic MapReduce



Critique

Critique of this paper

Good:

- Able to See the impact of various settings
- Good Visualization Tools

Bad:

- This paper talks a lot about what starfish can do but not how it can achieve that
- Experiments constrained many variants (possibly related to each other) fixed.



**THANK YOU FOR
WATCHING**