Starfish: A Self-tuning System for Big Data Analytics

Herodotos Herodotou,

Harold Lim, Fei Dong, Shivnath Babu

Duke University

Analysis in the Big Data Era

Massive Data

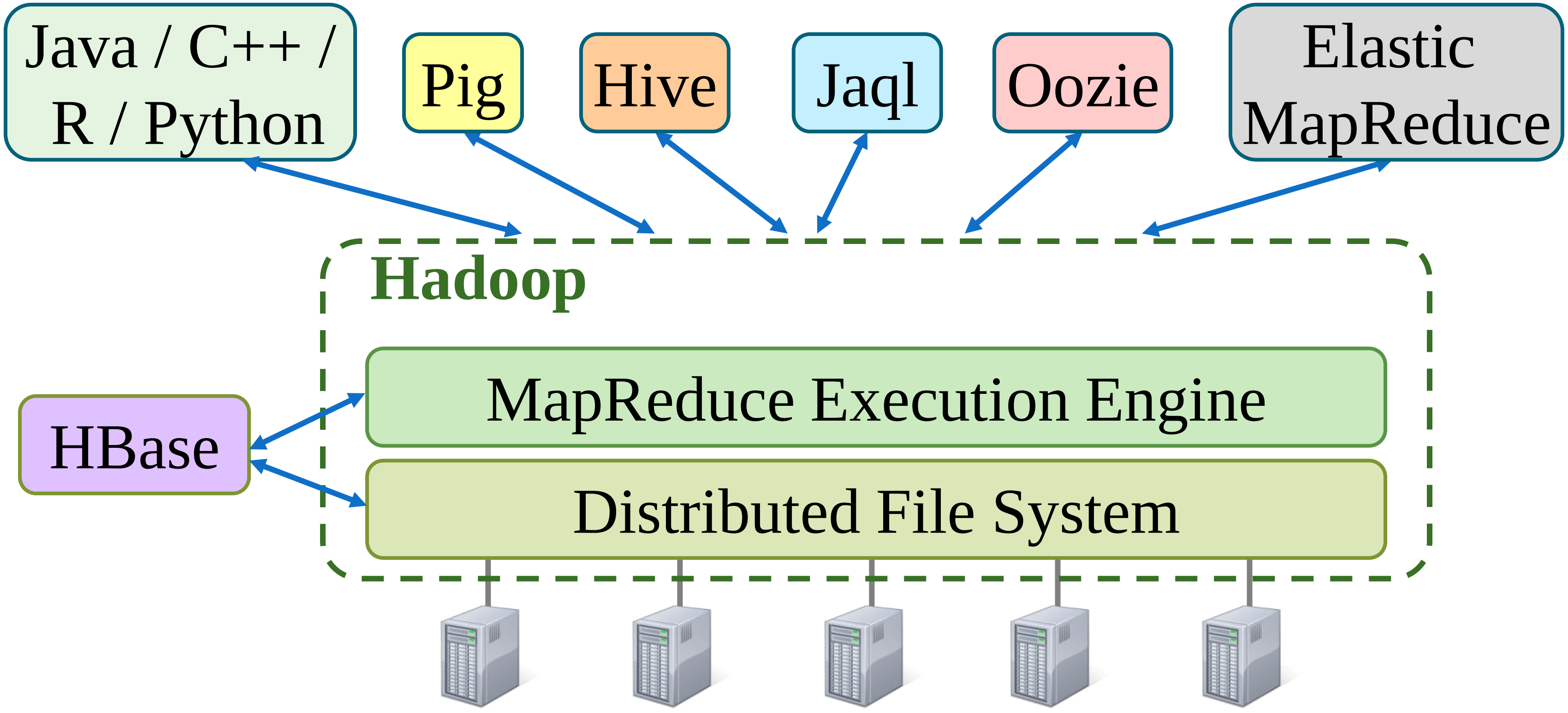
Data Analysis

Insight

Key to Success = Timely and Cost-Effective Analysis

Hadoop MapReduce Ecosystem

- Popular solution to Big Data Analytics



Practitioners of Big Data Analytics

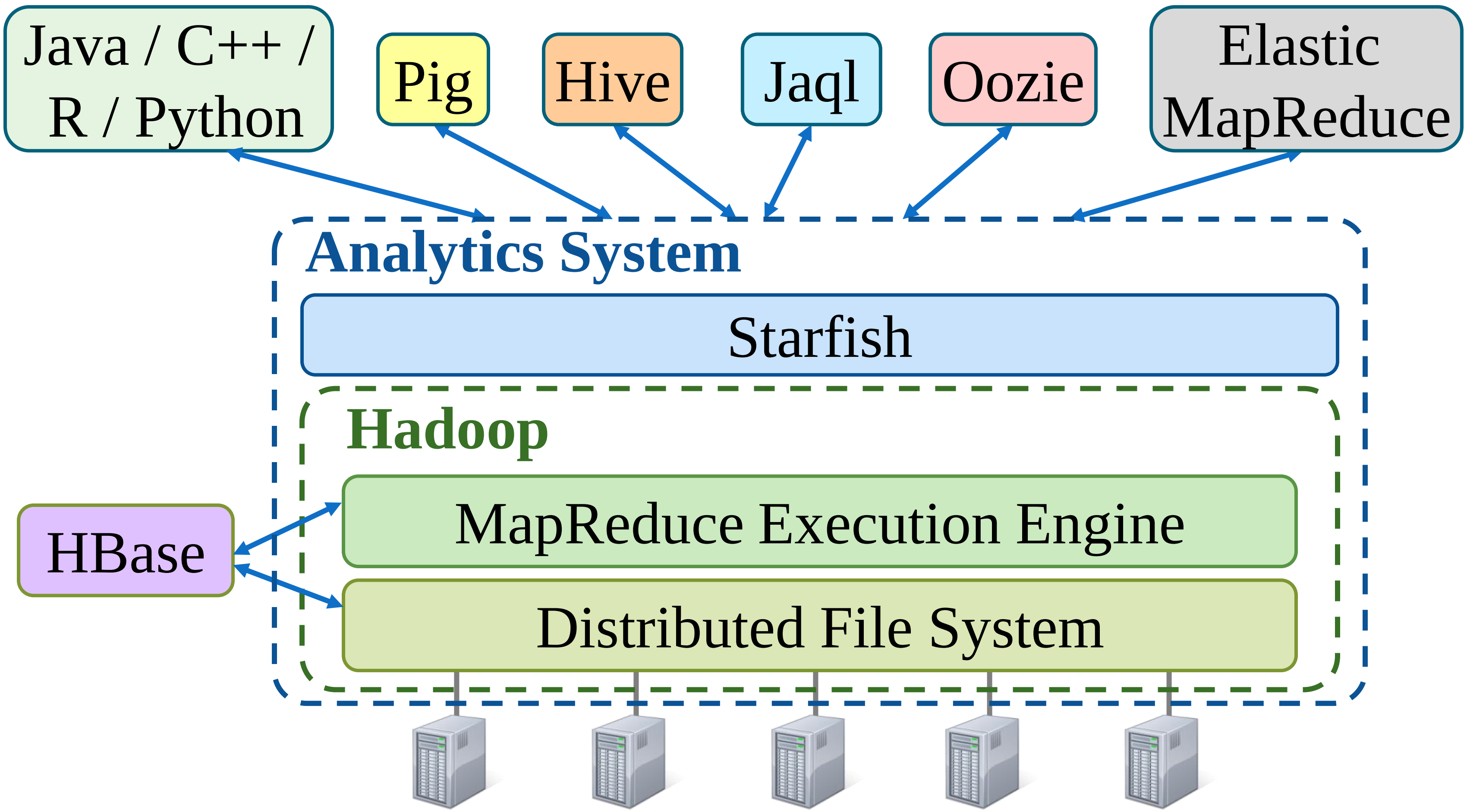
- Who are the users?
  - Data analysts, statisticians, computational scientists…
  - Researchers, developers, testers…
  - You!
- Who performs setup and tuning?
  - The users!
  - Usually lack expertise to tune the system

Tuning Challenges

- Heavy use of programming languages for MapReduce programs (e.g., Java/python)
- Data loaded/accessed as opaque files
- Large space of tuning choices
- Elasticity is wonderful, but hard to achieve (Hadoop has many useful mechanisms, but policies are lacking)
- Terabyte-scale data cycles

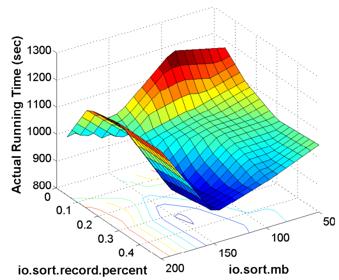
Starfish: Self-tuning System

- Our goal: Provide good performance automatically

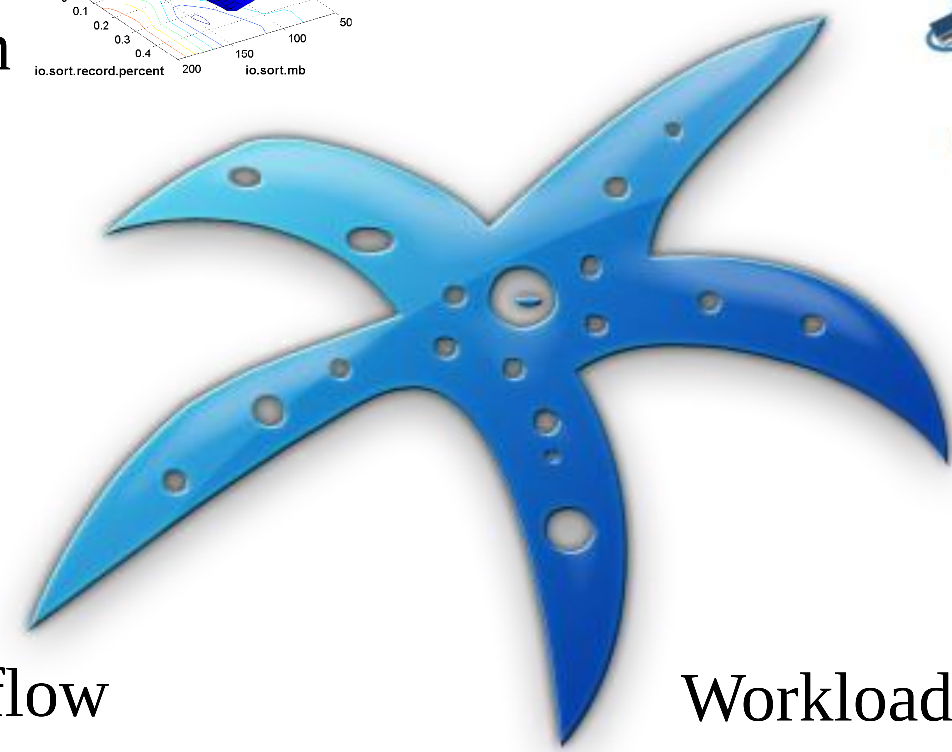


What are the Tuning Problems?

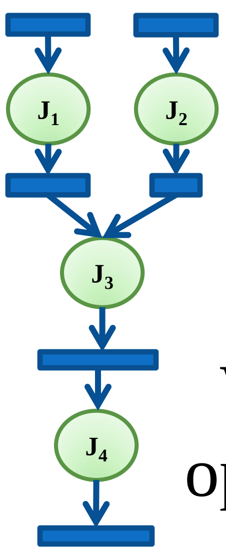
Job-level MapReduce configuration



Cluster sizing



Data layout tuning



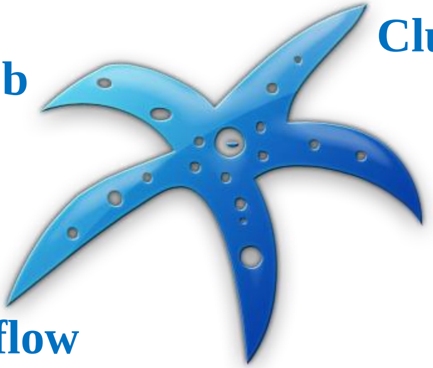
Workflow optimization

Workload management

Starfish's Core Approach to Tuning

Optimizers

Search through space of tuning choices



Profiler

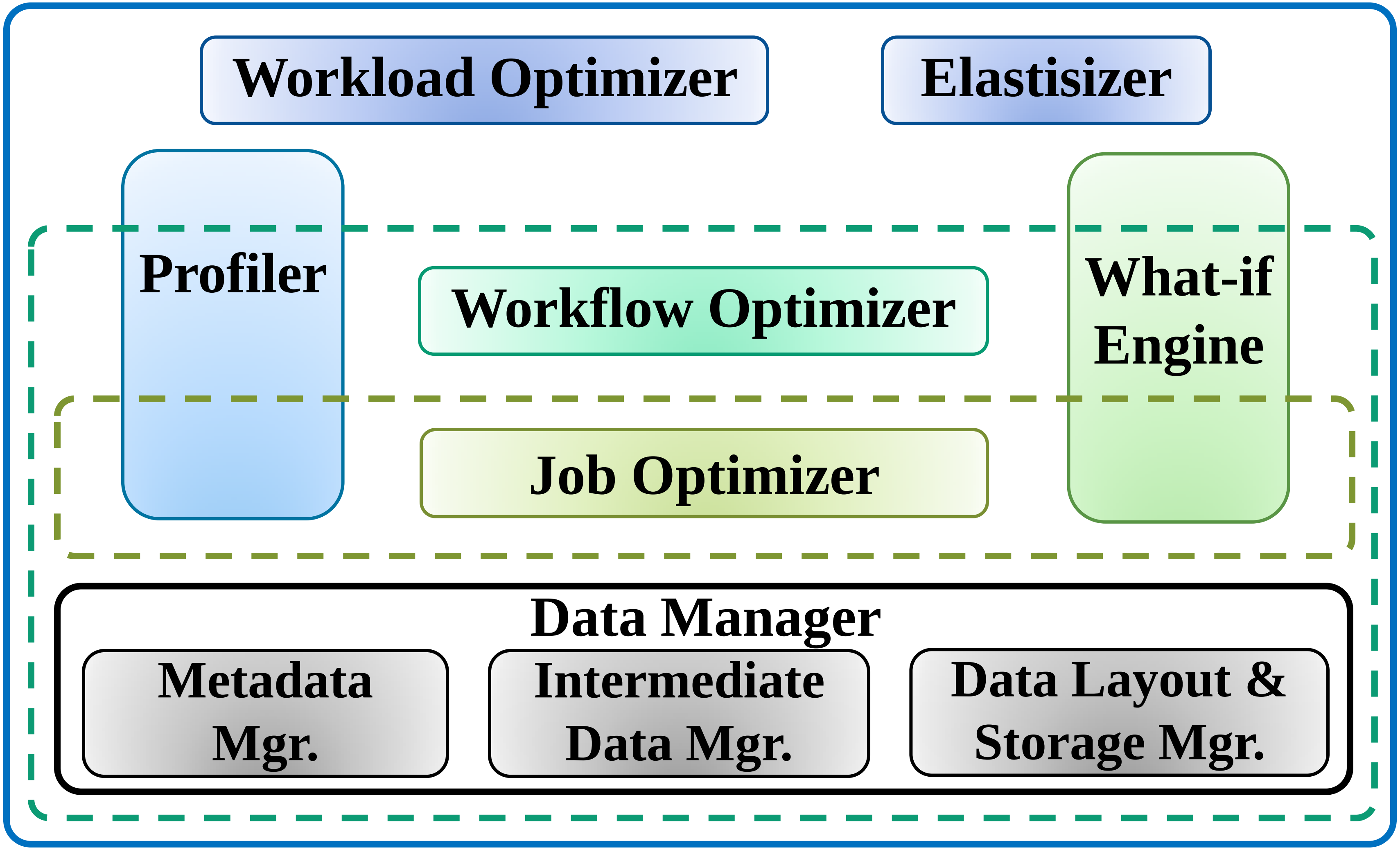
Collects concise summaries of execution

What-if Engine

Estimates impact of hypothetical changes on execution

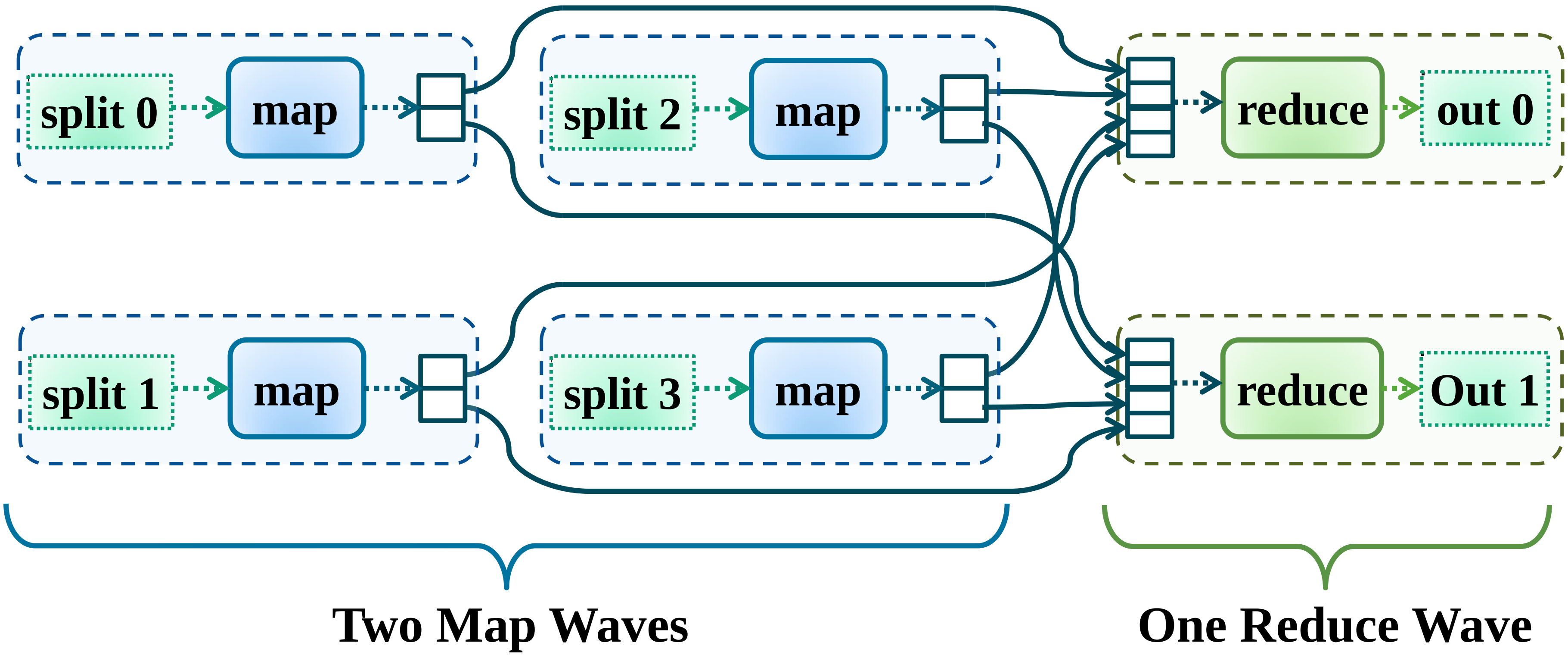
1) if Δ(conf. parameters) then what …?
2) if Δ(data properties) then what …?
3) if Δ(cluster properties) then what …?

Starfish Architecture



MapReduce Job Execution

job *j* = < program *p*, data *d*, resources *r*, configuration *c* >



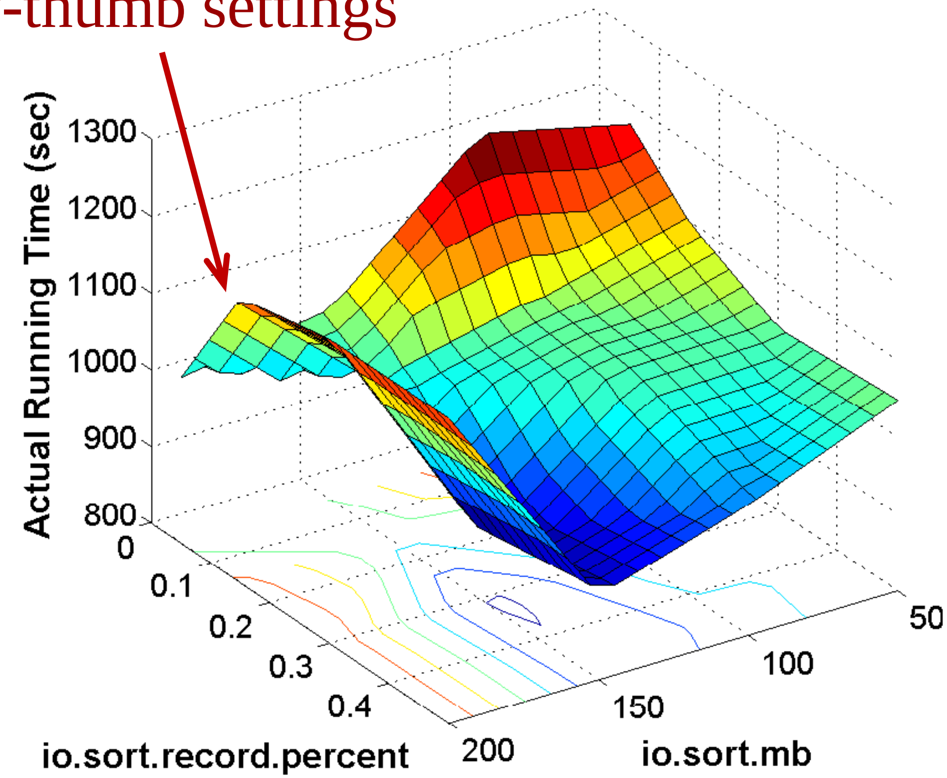
What Controls MR Job Execution?

job j = < program p, data d, resources r, configuration c >

- Space of configuration choices:
  - Number of map tasks
  - Number of reduce tasks
  - Partitioning of map outputs to reduce tasks
  - Memory allocation to task-level buffers
  - Multiphase external sorting in the tasks
  - Whether output data from tasks should be compressed
  - Whether combine function should be used

Effect of Configuration Settings

Rules-of-thumb settings



Two-dimensional projection of a multi-dimensional surface (Word Co-occurrence MapReduce Program)

- Use defaults or set manually (rules-of-thumb)
- Rules-of-thumb may not suffice

MapReduce Job Tuning in a Nutshell

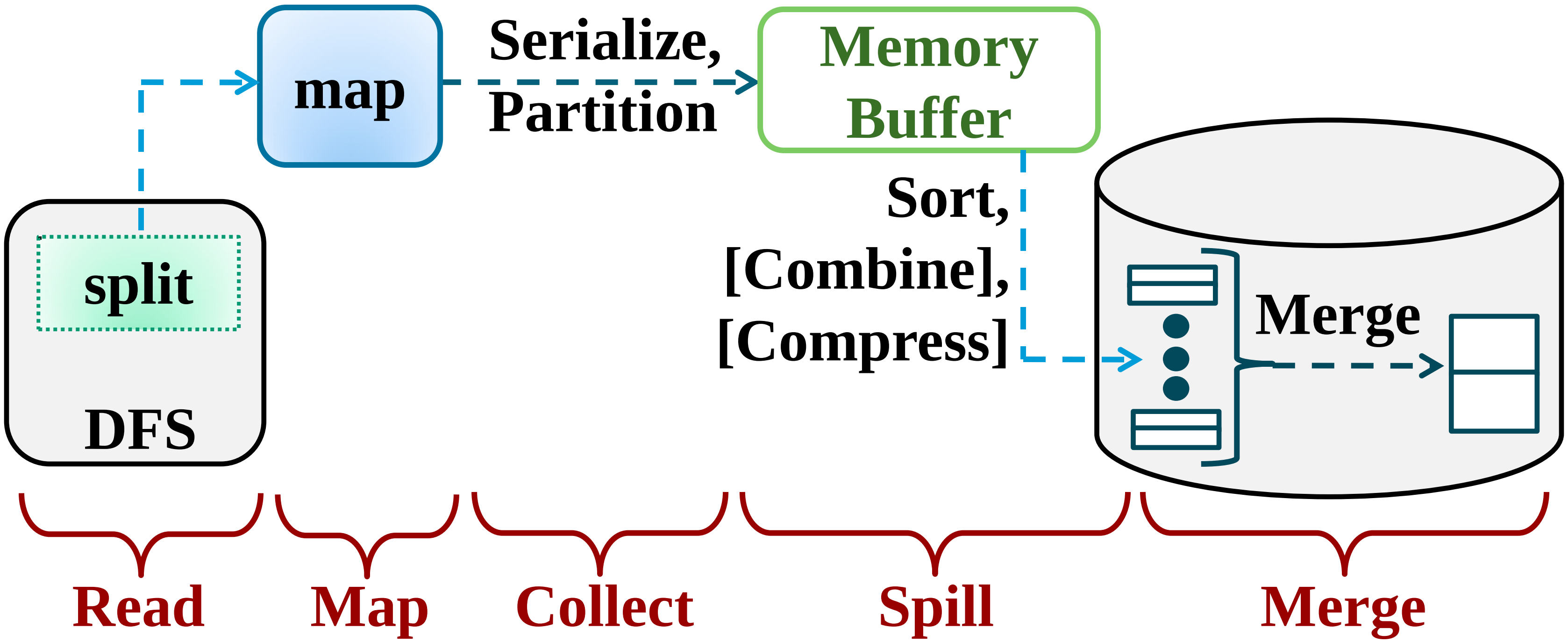
- Goal: $perf = F(p, d, r, c)$

$$c_{opt} = \underset{c \in S}{\arg\min}\, F(p, d, r, c)$$

- Challenges: *p* is an arbitrary MapReduce program; *c* is high-dimensional; …
- Profiler — Runs *p* to collect a ***job profile*** (concise execution summary) of <p, d_1, r_1, c_1>
- What-if Engine — Given profile of <p, d_1, r_1, c_1>, estimates ***virtual profile*** for <p, d_2, r_2, c_2>
- 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

Dataflow: amount of data flowing through task phases
Map output bytes
Number of spills
Number of records in buffer per spill
⋮

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
⋮

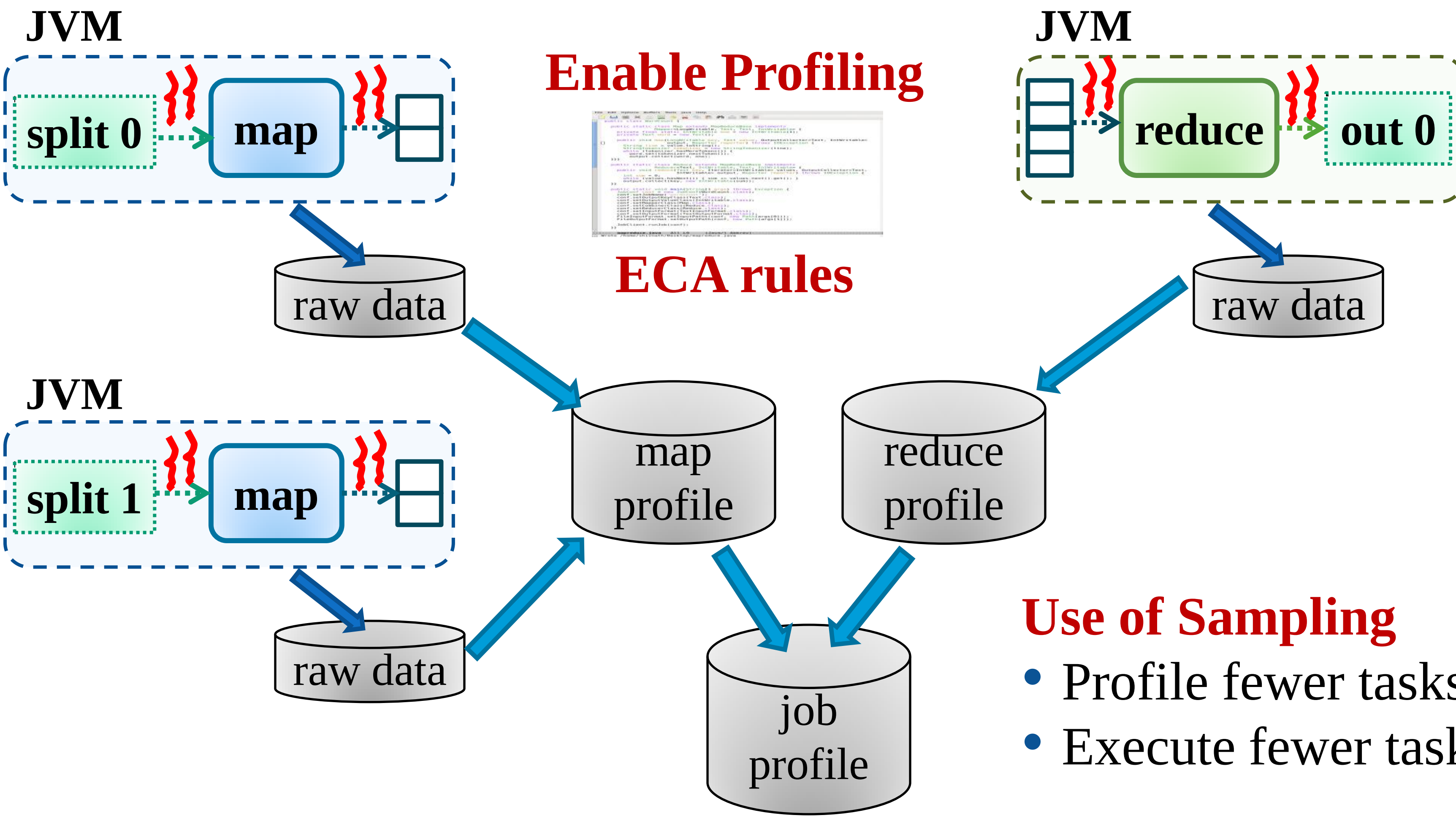
Dataflow Statistics: statistical information about dataflow
Width of input key-value pairs
Map selectivity in terms of records
Map output compression ratio
⋮

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
CPU cost for uncompressing the input per byte
⋮

Generating Profiles by Measurement

- Goals
  - Have zero overhead when profiling is turned off
  - Require no modifications to Hadoop
  - Support unmodified MapReduce programs written in Java or Hadoop Streaming/Pipes (Python/Ruby/C++)
- Approach: Dynamic (on-demand) instrumentation
  - Event-condition-action rules are specified (in Java)
  - Leads to run-time instrumentation of Hadoop internals
  - Monitors task phases of MapReduce job execution
  - We currently use Btrace (Hadoop internals are in Java)

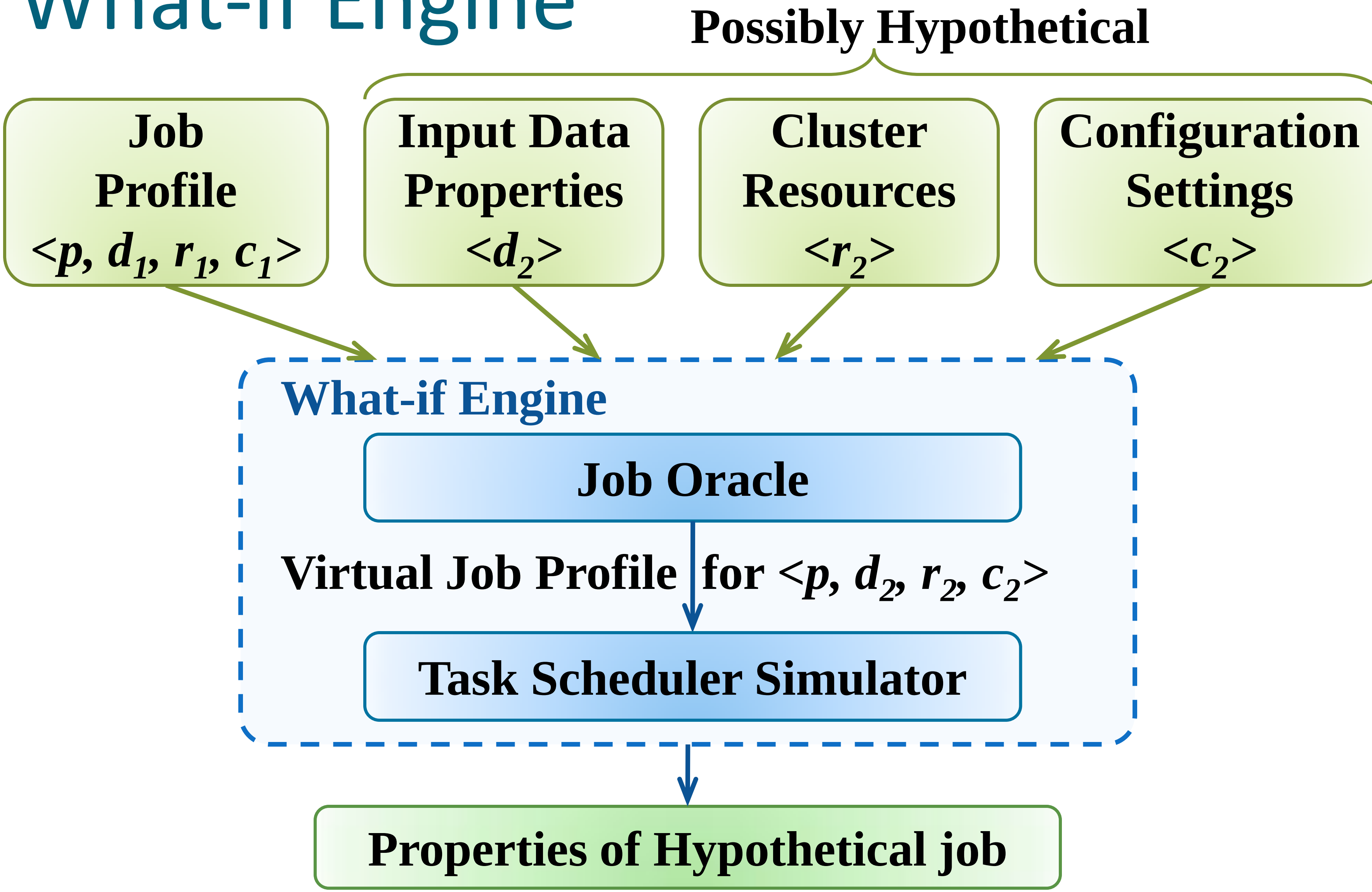
Generating Profiles by Measurement



Use of Sampling
- Profile fewer tasks
- Execute fewer tasks

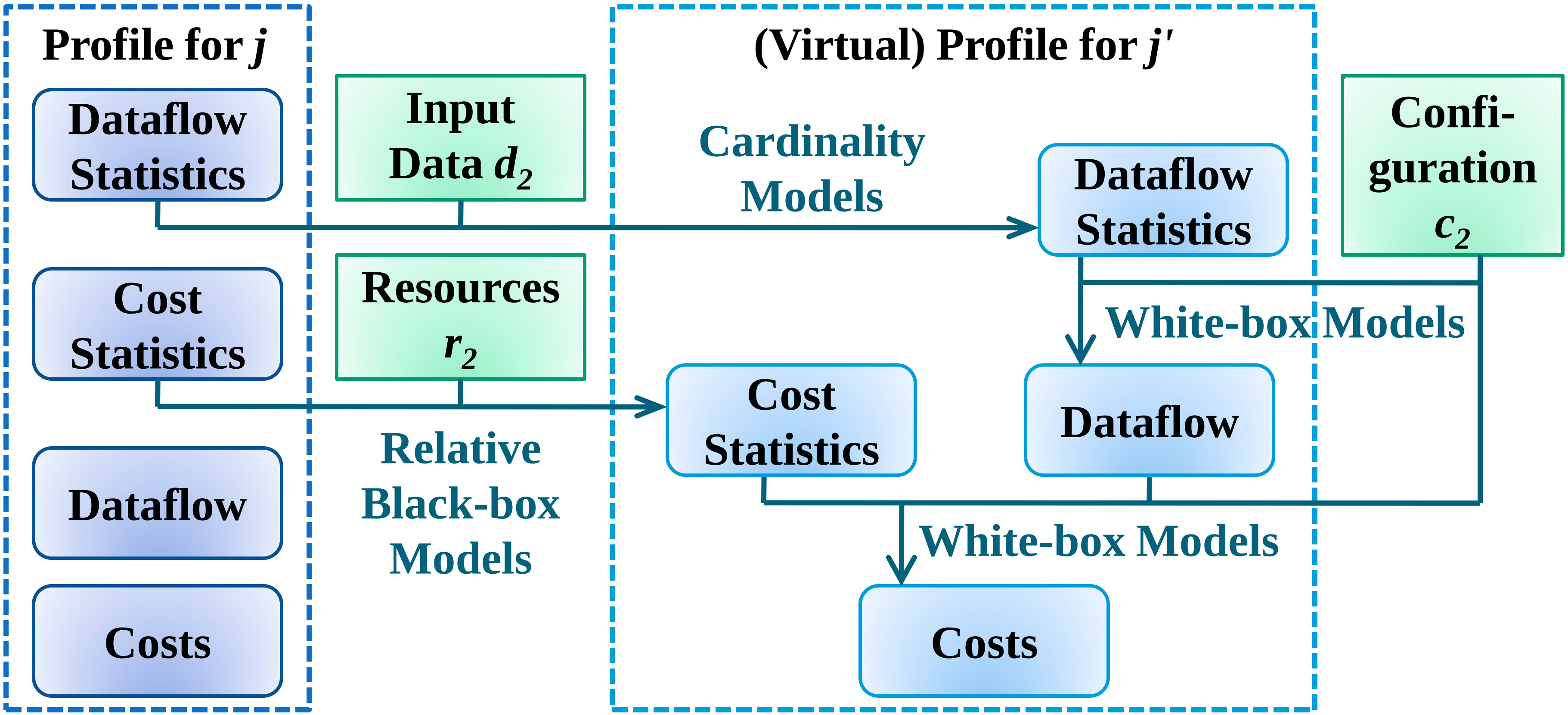
JVM = Java Virtual Machine, ECA = Event-Condition-Action

What-if Engine

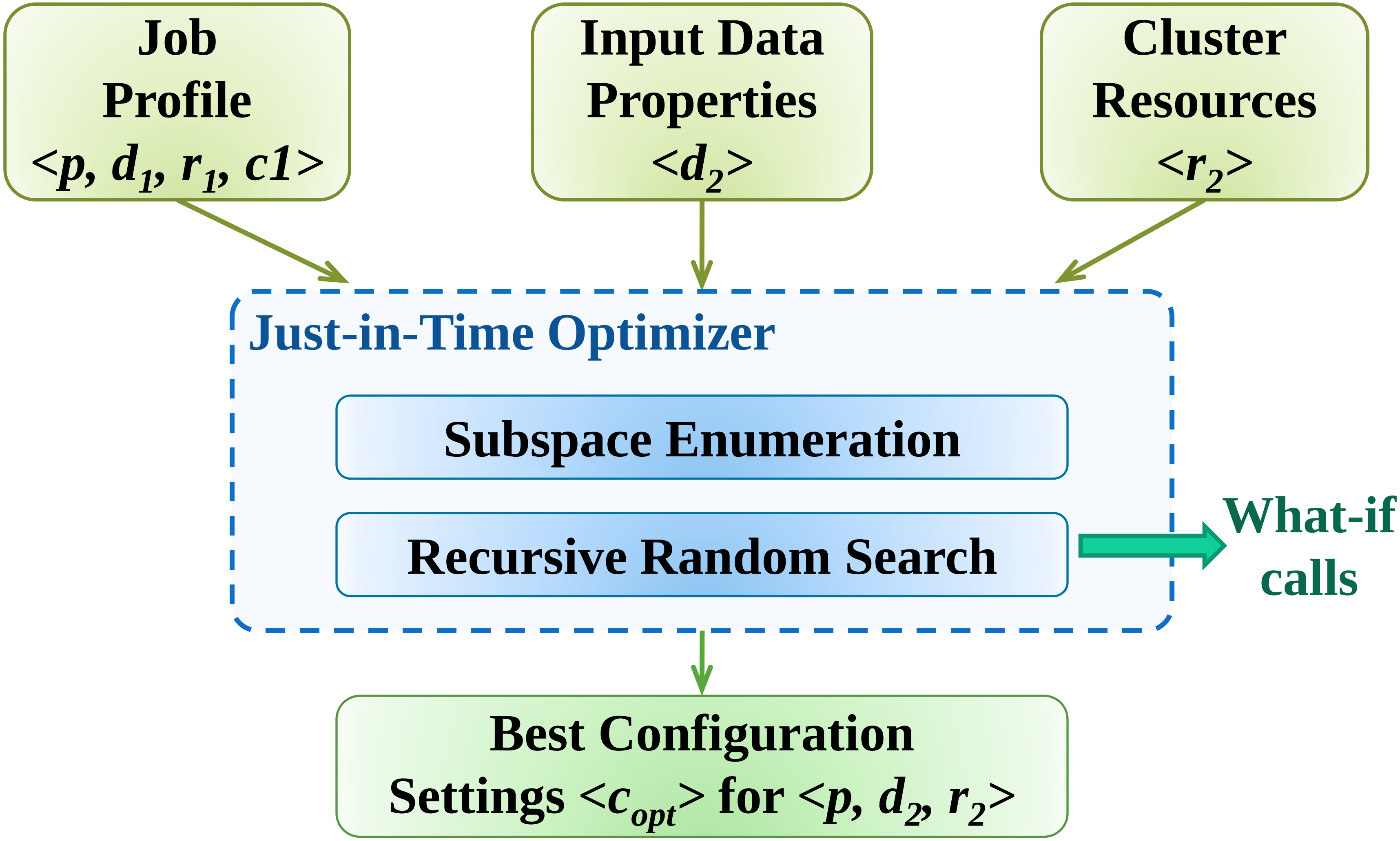


Virtual Profile Estimation

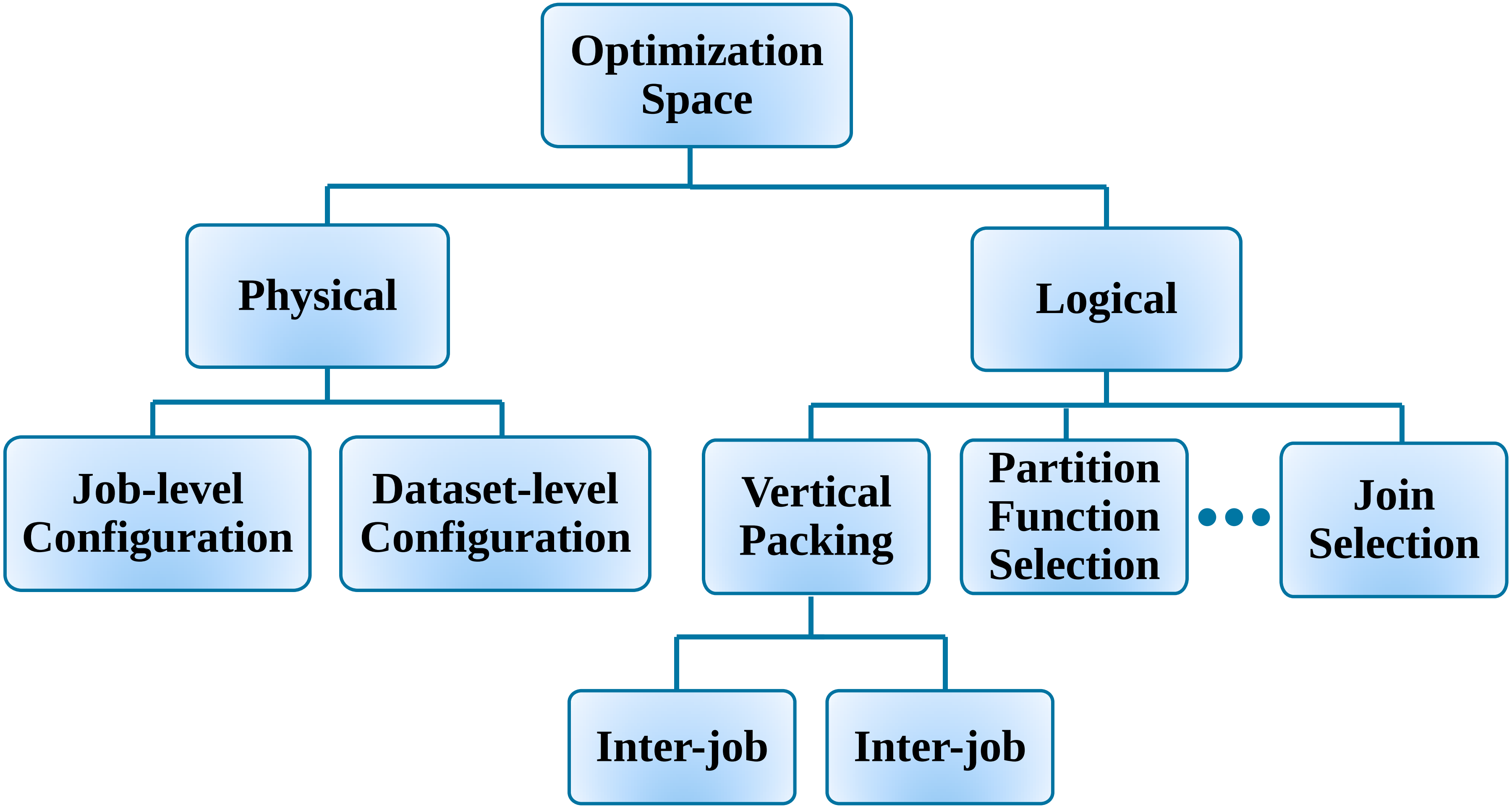
Given profile for job $j = <p, d_1, r_1, c_1>$
estimate profile for job $j' = <p, d_2, r_2, c_2>$



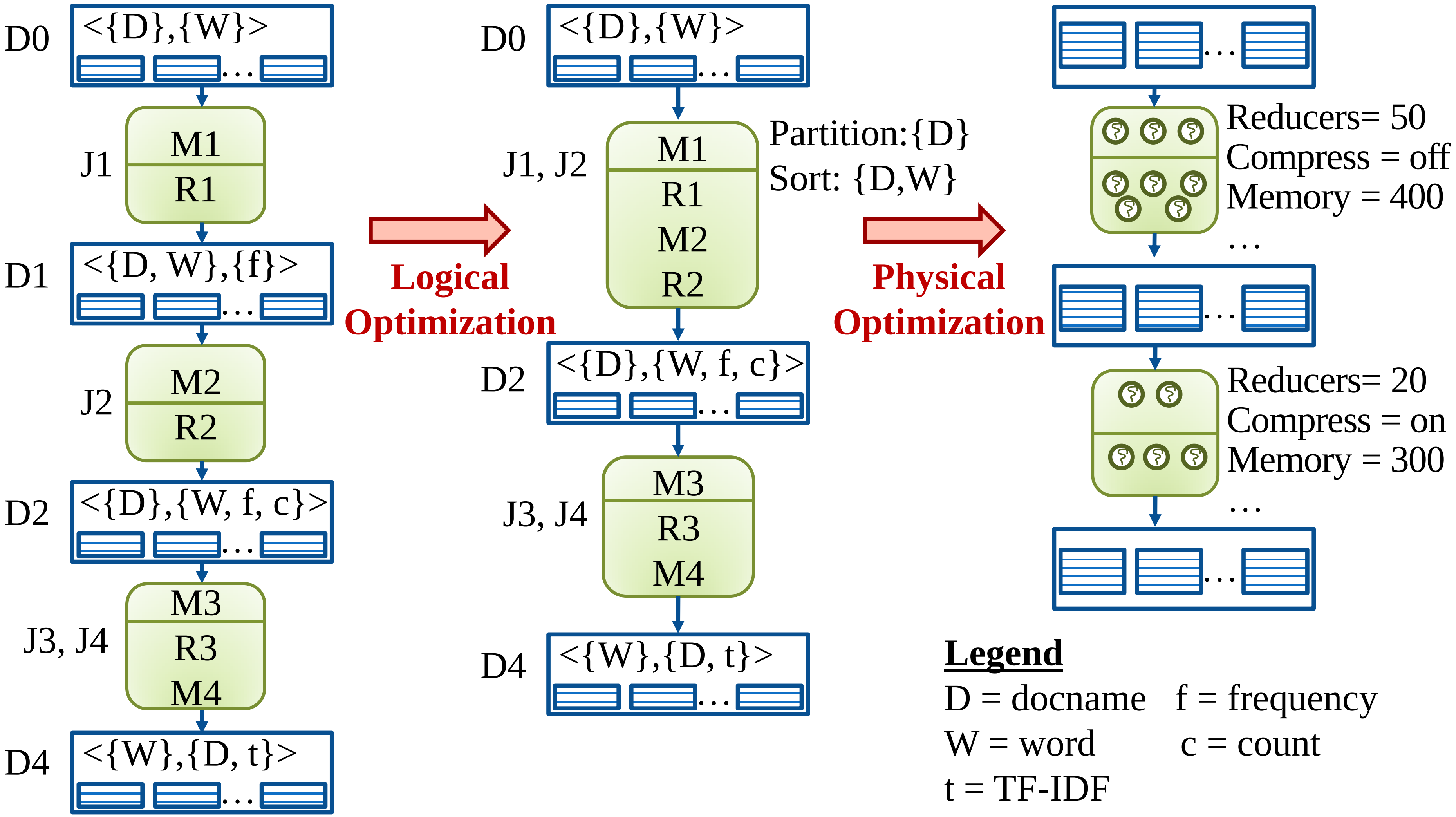
Job Optimizer



Workflow Optimization Space



Optimizations on TF-IDF Workflow



Legend

D = docname f = frequency

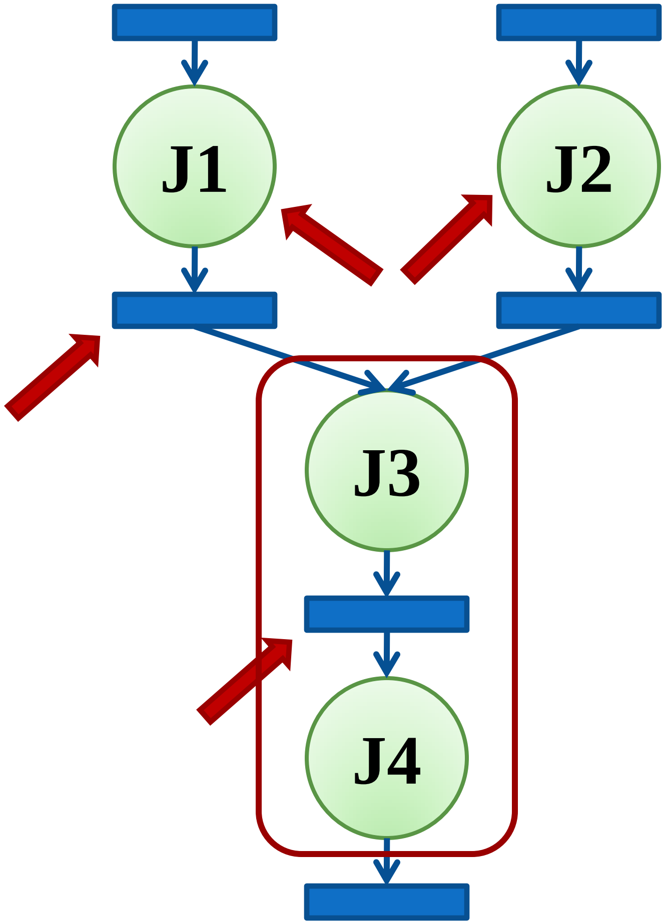
W = word c = count

t = TF-IDF

New Challenges

- What-if challenges:
  - Support concurrent job execution
  - Estimate intermediate data properties
- Optimization challenges
  - Interactions across jobs
  - Extended optimization space
  - Find good configuration settings for individual jobs

Workflow

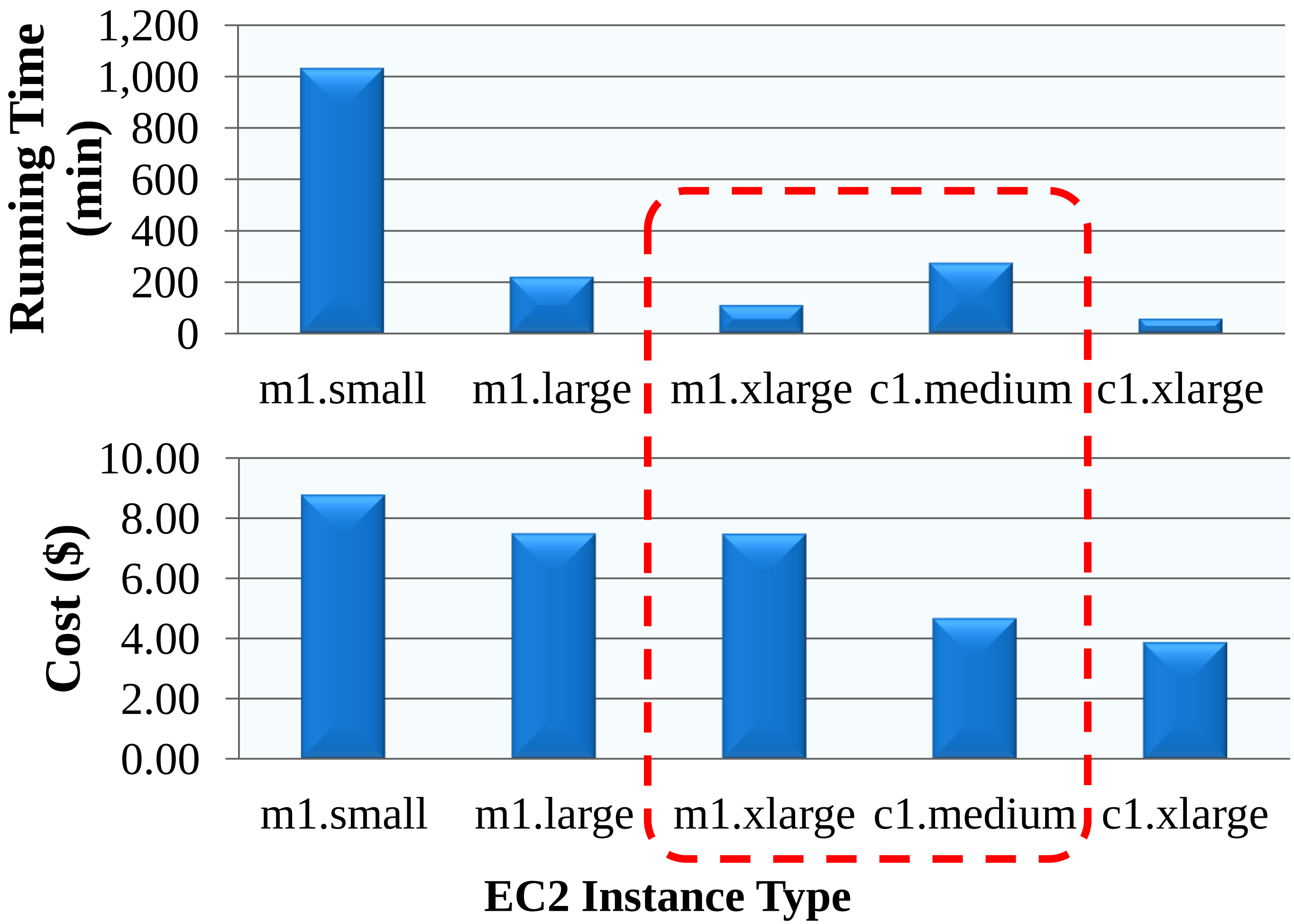


Cluster Sizing Problem

- Use-cases for cluster sizing
  - Tuning the cluster size for elastic workloads
  - Workload transitioning from development cluster to production cluster
  - Multi-objective cluster provisioning
- Goal
  - Determine cluster resources & job-level configuration parameters to meet workload requirements

Multi-objective Cluster Provisioning

- Cloud enables users to provision clusters in minutes



Experimental Evaluation

- Starfish (versions 0.1, 0.2) to manage Hadoop on EC2
- Different scenarios: Cluster × Workload × Data

EC2 Node Type	CPU: EC2 units	Mem	I/O Perf.	Cost /hour	#Maps /node	#Reds /node	MaxMem /task
m1.small	1 (1 x 1)	1.7 GB	moderate	$0.085	2	1	300 MB
m1.large	4 (2 x 2)	7.5 GB	high	$0.34	3	2	1024 MB
m1.xlarge	8 (4 x 2)	15 GB	high	$0.68	4	4	1536 MB
c1.medium	5 (2 x 2.5)	1.7 GB	moderate	$0.17	2	2	300 MB
c1.xlarge	20 (8 x 2.5)	7 GB	high	$0.68	8	6	400 MB
cc1.4xlarge	33.5 (8)	23 GB	very high	$1.60	8	6	1536 MB

Experimental Evaluation

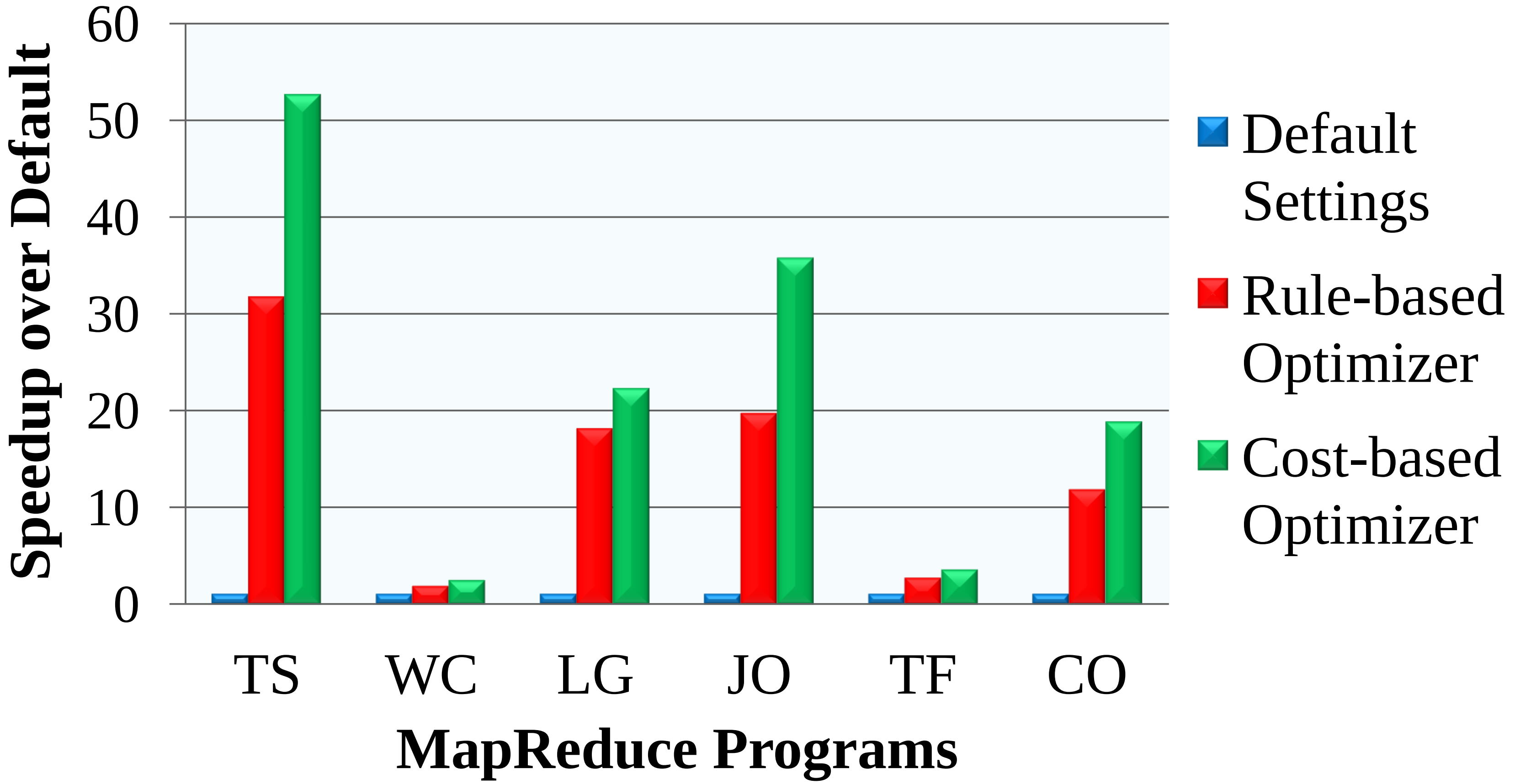
- Starfish (versions 0.1, 0.2) to manage Hadoop on EC2
- Different scenarios: Cluster × Workload × Data

Abbr.	MapReduce Program	Domain	Dataset
CO	Word Co-occurrence	Natural Lang Proc.	Wikipedia (10GB – 22GB)
WC	WordCount	Text Analytics	Wikipedia (30GB – 1TB)
TS	TeraSort	Business Analytics	TeraGen (30GB – 1TB)
LG	LinkGraph	Graph Processing	Wikipedia (compressed ~6x)
JO	Join	Business Analytics	TPC-H (30GB – 1TB)
TF	Term Freq. - Inverse Document Freq.	Information Retrieval	Wikipedia (30GB – 1TB)

Job Optimizer Evaluation

Hadoop cluster: 30 nodes, m1.xlarge

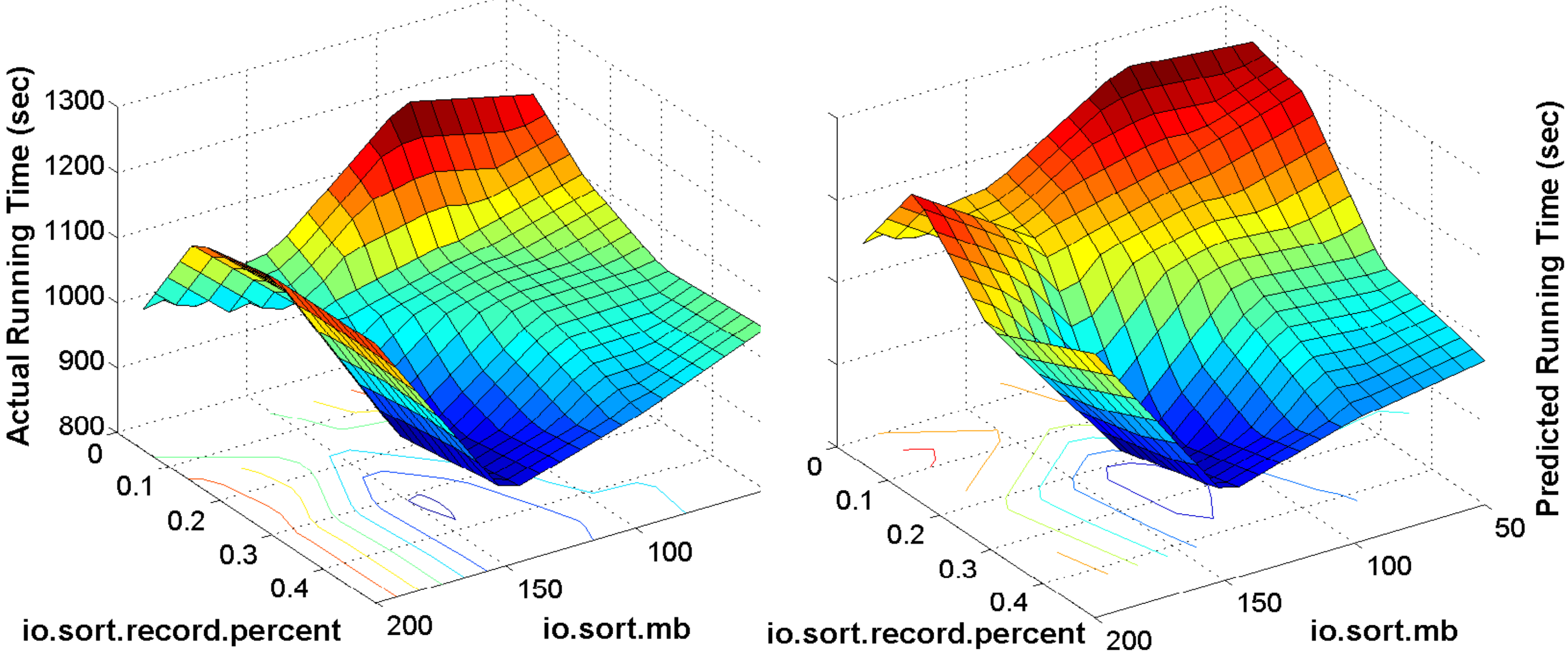
Data sizes: 60-180 GB



Estimates from the What-if Engine

Hadoop cluster: 16 nodes, c1.medium
MapReduce Program: Word Co-occurrence
Data set: 10 GB Wikipedia

True surface | Estimated surface

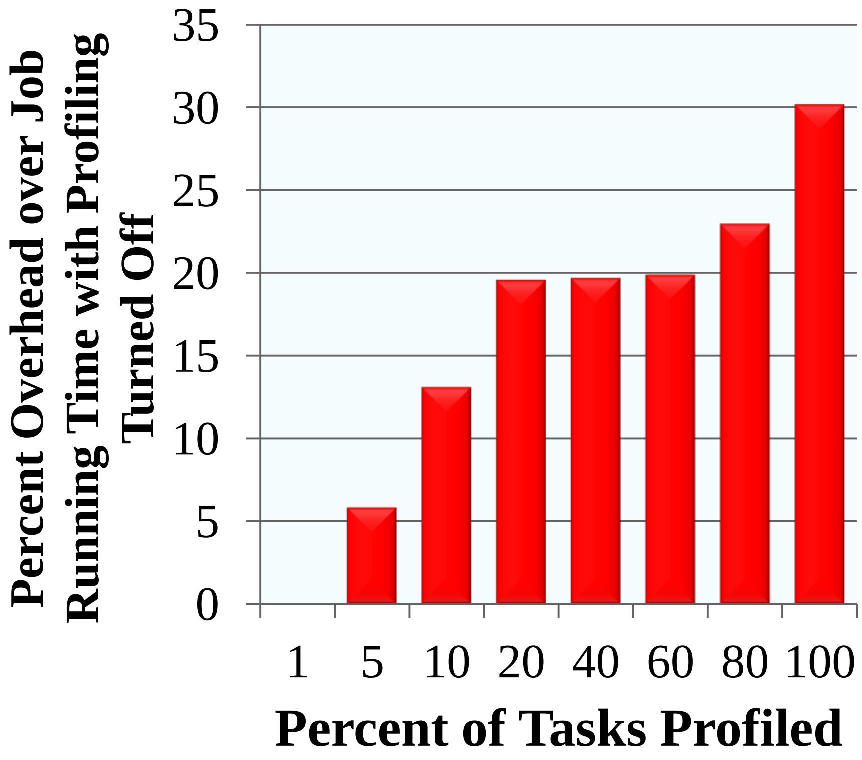


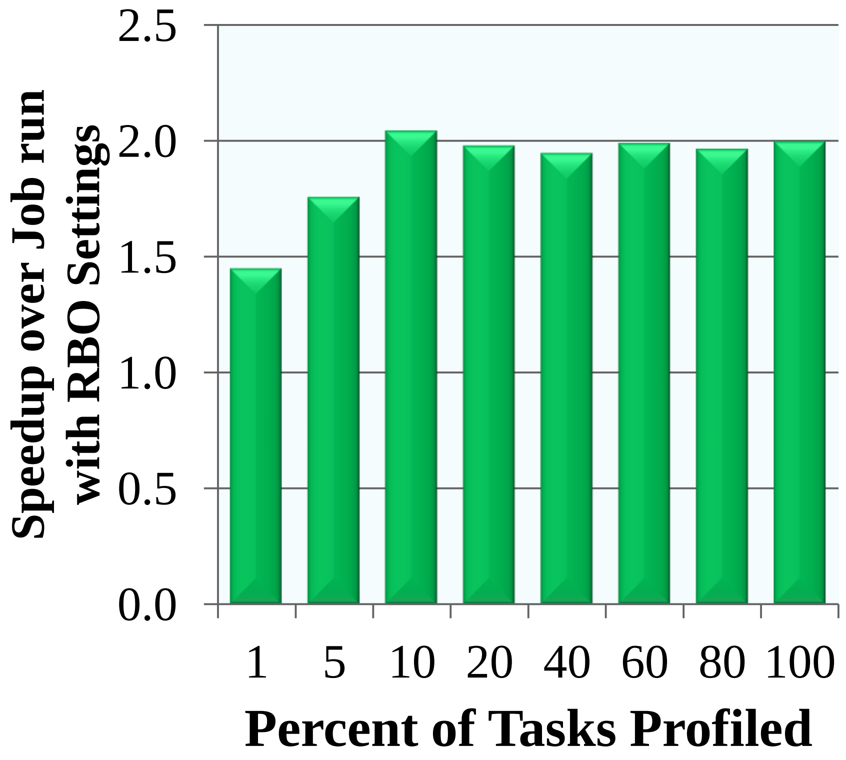
Profiling Overhead Vs. Benefit

Hadoop cluster: 16 nodes, c1.medium

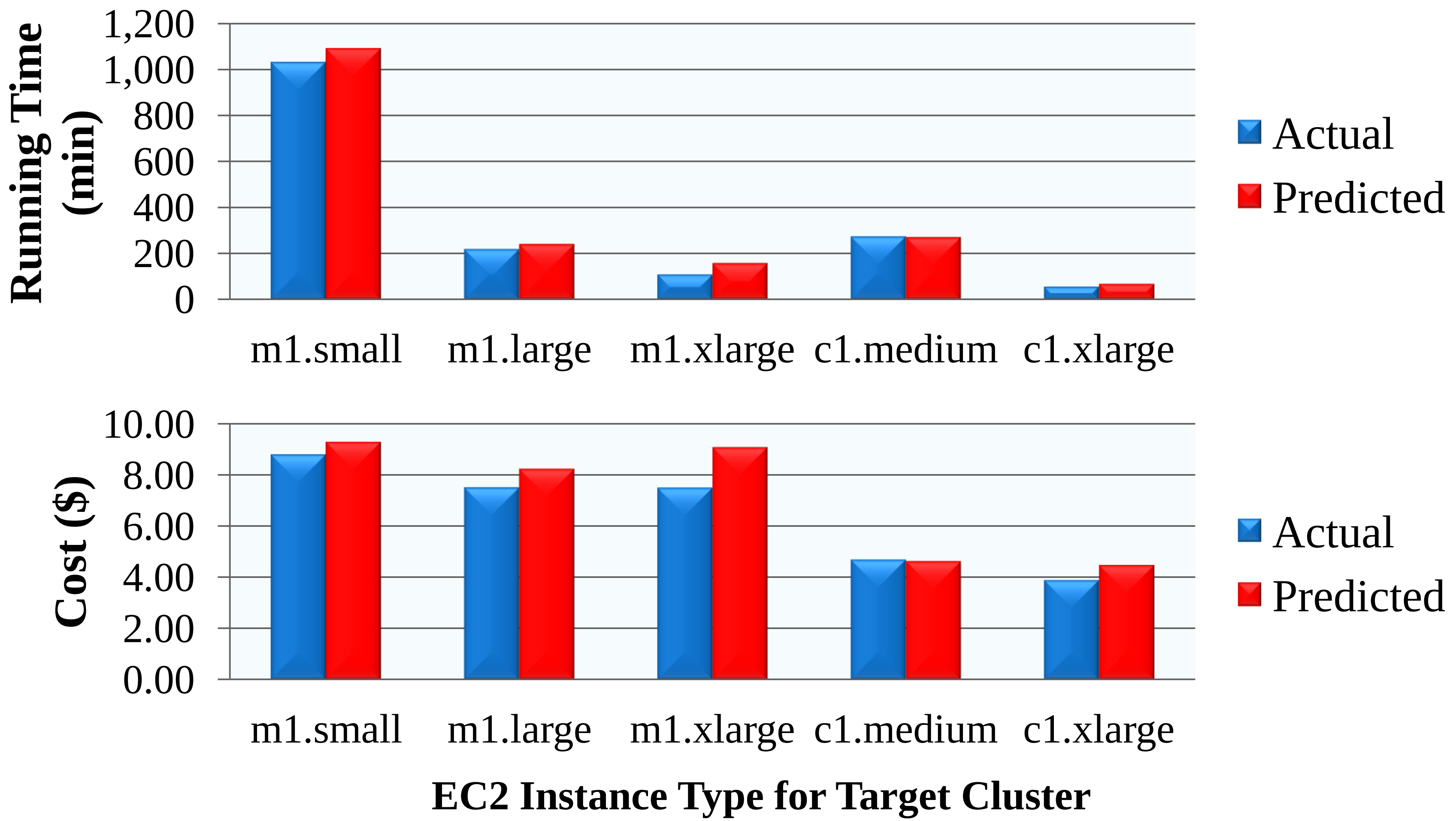
MapReduce Program: Word Co-occurrence

Data set: 10 GB Wikipedia





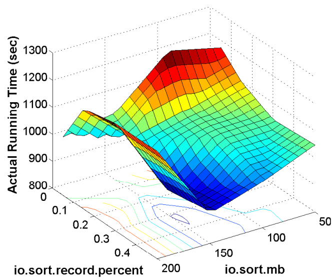
Multi-objective Cluster Provisioning



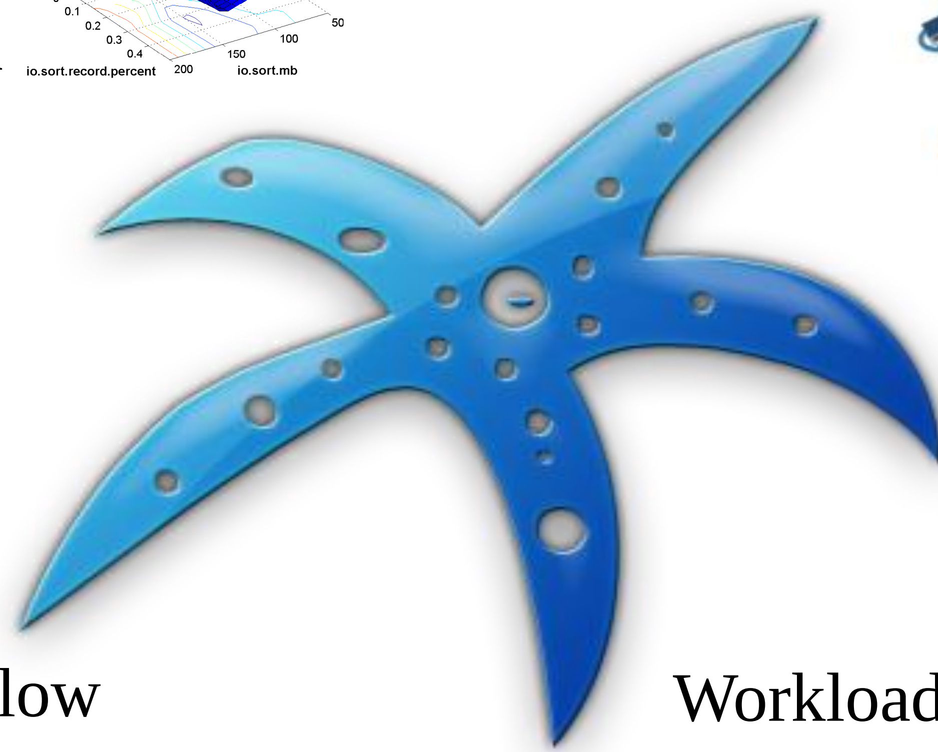
Instance Type for Source Cluster: m1.large

More info: www.cs.duke.edu/starfish

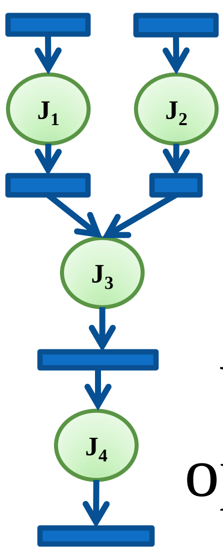
Job-level MapReduce configuration



Cluster sizing



Data layout tuning



Workflow optimization

Workload management