VLDB 2019 Tutorial

Speedup your Analytics: Automatic Parameter Tuning for Databases and Big Data Systems

Jiaheng Lu, University of Helsinki
Yuxing Chen, University of Helsinki
Herodotos Herodotou, Cyprus University of Technology
Shivnath Babu, Duke University / Unravel Data Systems

Outline

Motivation and Background

History and Classification

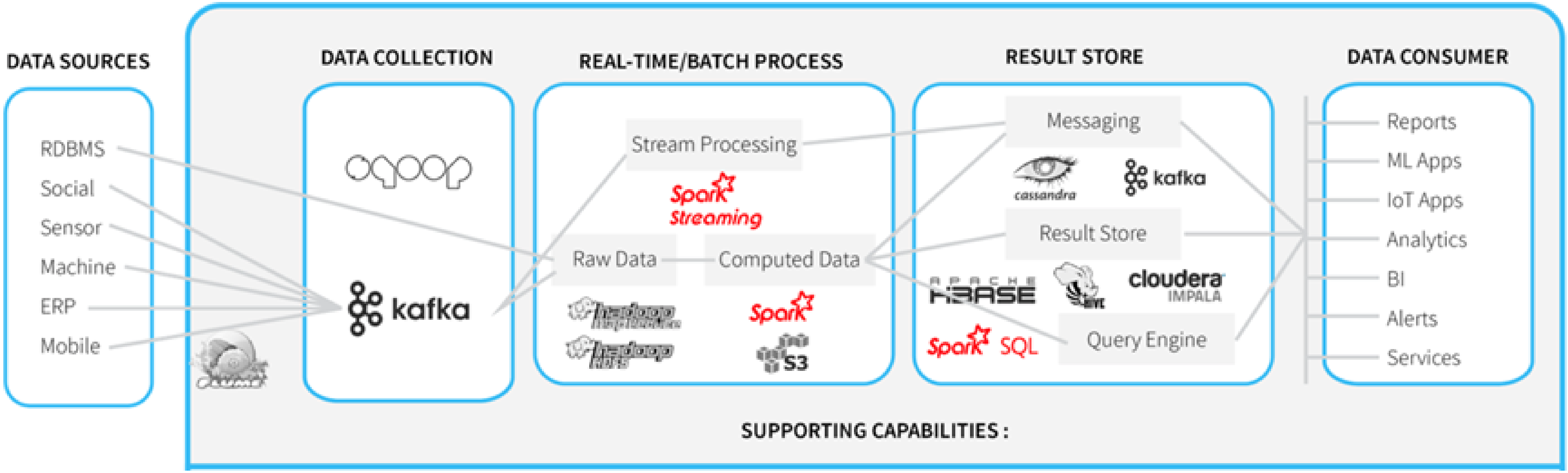
Parameter Tuning on Databases

Parameter Tuning on Big Data Systems

Applications of Automatic Parameter Tuning

Open Challenges and Discussion

Modern applications are being built on a collection of distributed systems



But:
Running distributed applications reliably & efficiently is hard

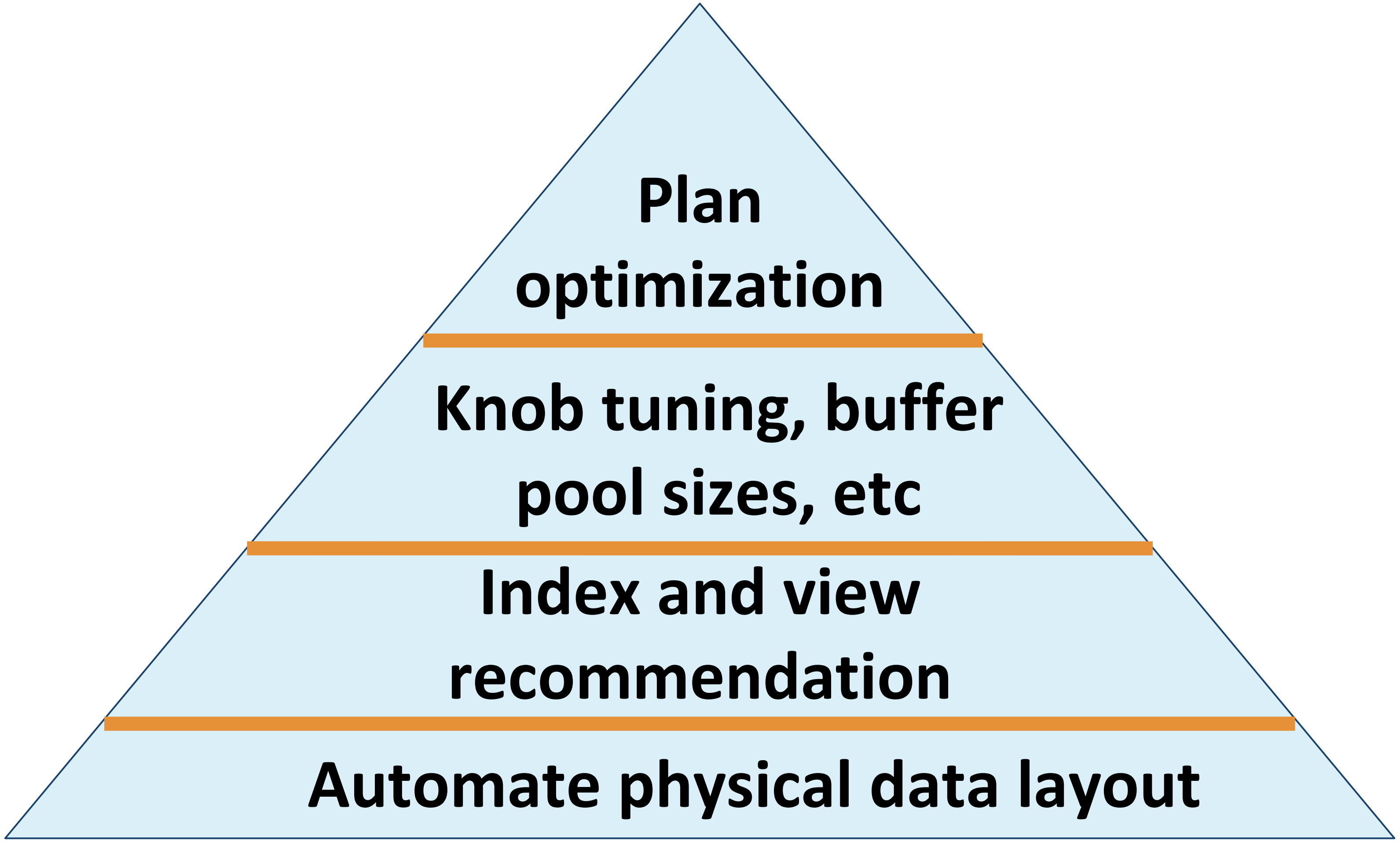
My app failed

My data pipeline is missing SLA

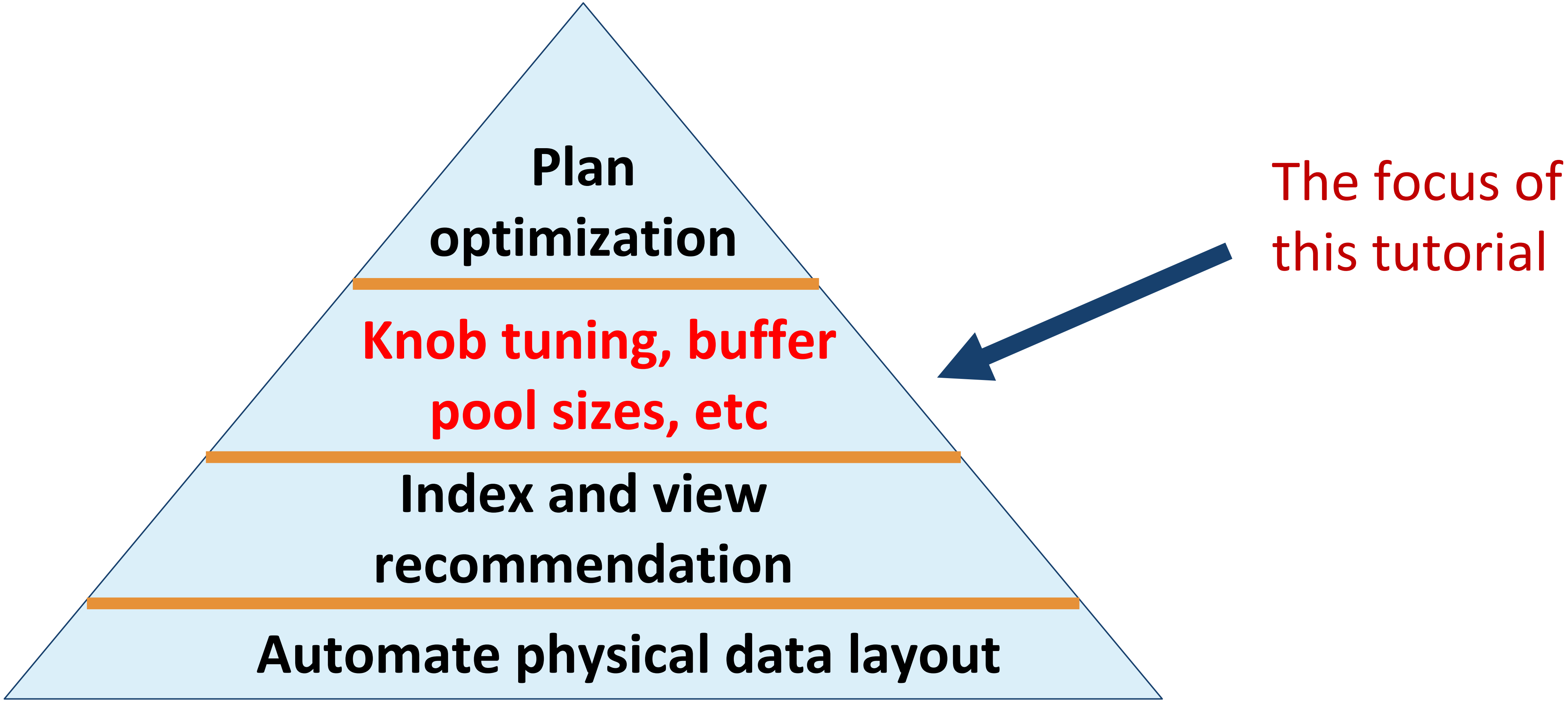
My cloud cost is out of control

There are many challenges

Self-driving Systems



Different Optimization Levels of Self-driving Systems



Effectiveness of Knob (Parameter) Tuning

	Tuned vs. Default
Running time	Often 10x
System resource utilization	Often 10x
Others	Well tuned jobs may avoid failures like OOM, out of disk, job time out, etc.

Good performance after tuning

Selected Performance-aware Parameters in PostgreSQL

Parameter Name	Description	Default value
bgwriter_lru_maxpages	Max number of buffers written by the background writer	100
checkpoint_segments	Max number of log file segments between WAL checkpoints	3
checkpoint_timeout	Max time between automatic WAL checkpoints	5 min
deadlock_timeout	Waiting time on locks for checking for deadlocks	1 sec
effective_cache_size	Size of the disk cache accessible to one query	4 GB
effective_io_concurrency	Number of disk I/O operations to be executed concurrently	1 or 0
shared_buffers	Memory size for shared memory buffers	128MB

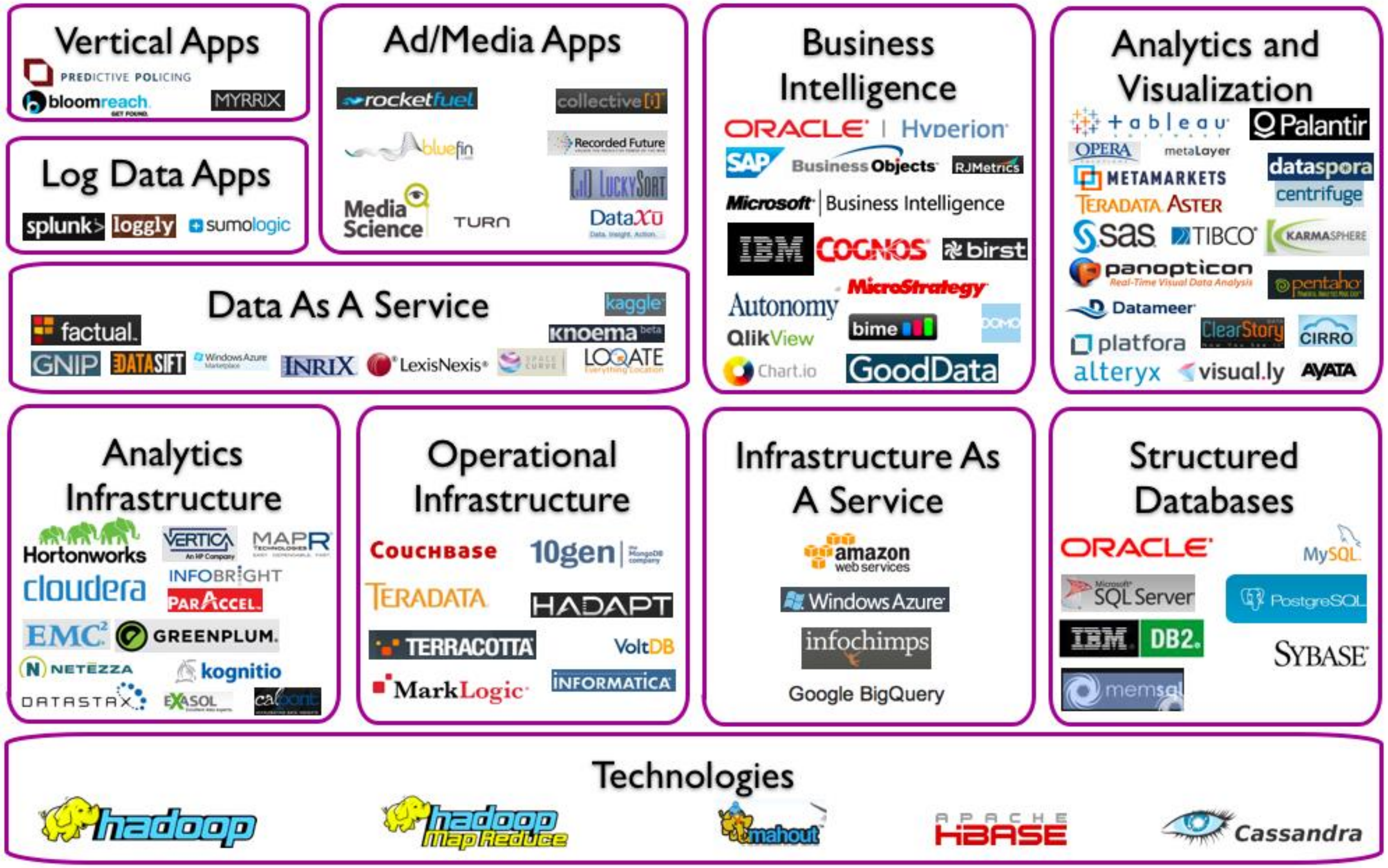
Selected Performance-aware Parameters in Hadoop

Parameter Name	Description	Default value
dfs.block.size	The default block size for files stored HDFS	128MB
mapreduce.map.tasks	Number of map tasks	2
mapreduce.reduce.tasks	Number of reduce tasks	1
mapreduce.job.reduce .slowstart.completedmaps	Min percent of map tasks completed before scheduling reduce tasks	0.05
mapreduce.map.combine .minspills	Min number of map output spill files present for using the combine function	3
mapreduce.reduce.merge.in mem.threshold	Max number of shuffled map output pairs before initiating merging during the shuffle	1000

Selected Performance-aware Parameters in Spark

Parameter Name	Description	Default value
spark.driver.cores	Number of cores used by the Spark driver process	1
spark.driver.memory	Memory size for driver process	1 GB
spark.sql.shuffle.partitions	Number of tasks	200
spark.executor.cores	The number of cores for each executor	1
spark.files.maxPartitionBytes	Max number of bytes to group into one partition during file reading	128MB
spark.memory.fraction	Fraction for execution and storage memory. It may cause frequent spills or cached data eviction if given a low fraction	0.6

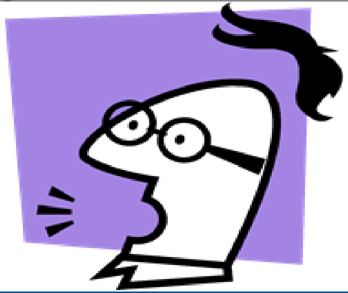
Challenge 1: A Huge Number of Systems



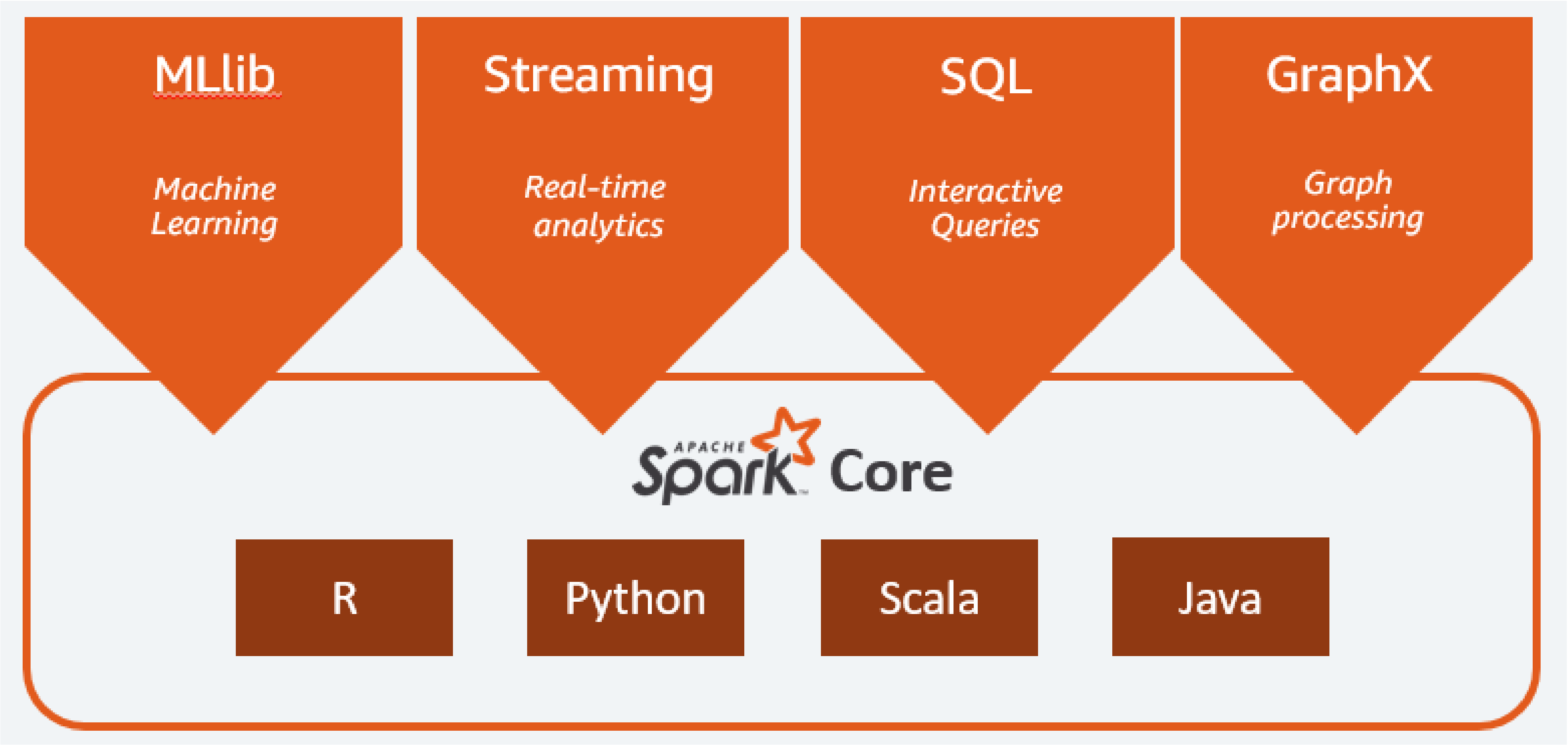
Challenge 2: Many Parameters in a Single System

Symbol	Parameter	Groups
N_m	mapred.map.tasks	Map input split
N_r	mapred.reduce.tasks	Reduce output
s_{min}	mapred.min.split.size	Map input split
s_{max}	mapred.max.split.size	Map input split
B_m	io.sort.mb	Map output buffer
r_{rec}	io.sort.record.percent	Map output buffer
c	mapred.compress.map.output	Map output compr.
N_{copy}	mapred.reduce.parallel. copies	Reduce copy
N_{sf}	io.sort.factor	Reduce input
B_r	mapred.job.reduce .input.buffer.percent	Reduce input
S_{OB}	dfs.block.size	Reduce output
$N_{ms}(i)$	mapred.tasktracker .map.tasks.maximum	Set by HAC
$N_{rs}(i)$	mapred.tasktracker .reduce.tasks.maximum	Set by HAC

There are more than **190 parameters** in Hadoop!



Challenge 3: Diverse Workloads and System Complexity



An example of Spark framework

Franco Pepe Chef:

"There is no pizza recipe. Every time the dough was made there were no scales, recipes, machinery."

There is no knob tuning recipe. Every time, we need to configure the parameters based on the bottleneck of different jobs and environment.

-- VLDB 2019 tutorial

Running Examples of Parameter Tuning (Hadoop)*

Workloads

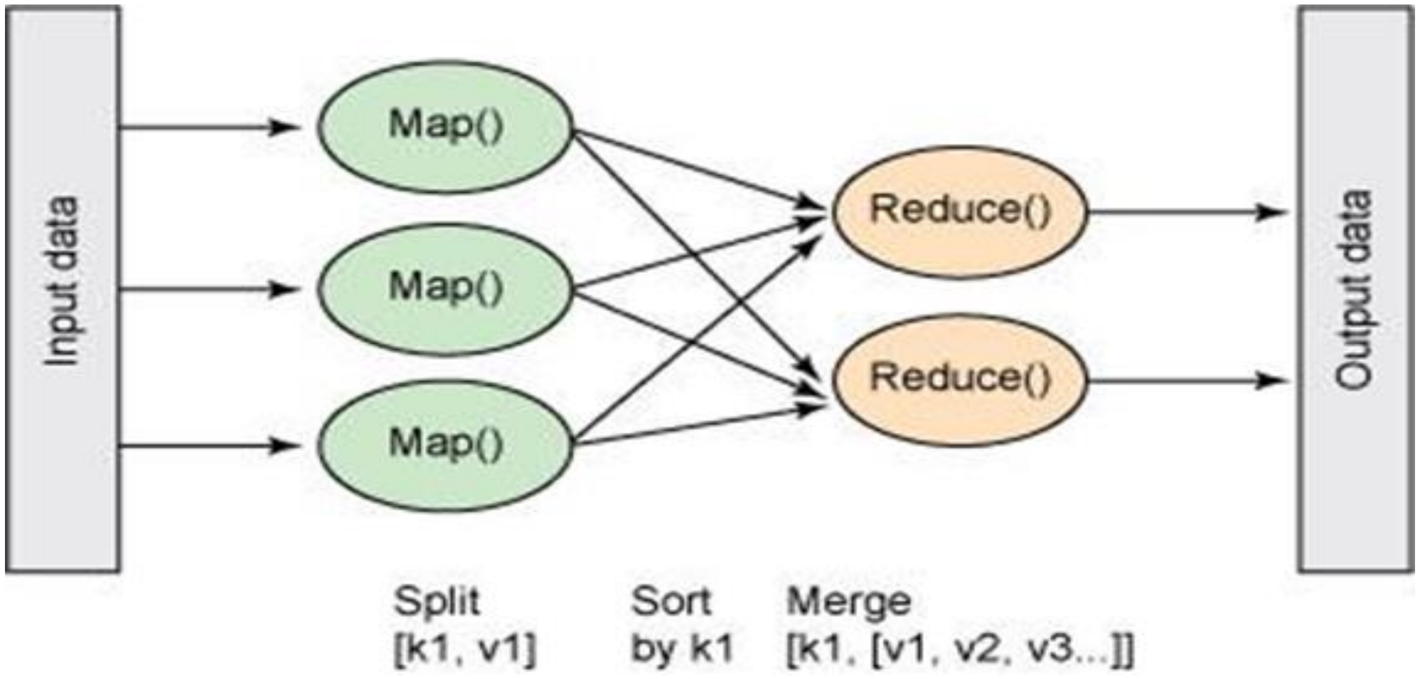
- Terasort: Sort a terabyte of data
- N-gram: Compute the inverted list of N-gram data
- PageRank: Compute pagerank of graphs

Hadoop platform with MapReduce

* Juwei Shi, Jia Zou, Jiaheng Lu, Zhao Cao, Shiqiang Li, Chen Wang: MRTuner: A Toolkit to Enable Holistic Optimization for MapReduce Jobs. PVLDB 7(13): 1319-1330 (2014)

Running Examples of Parameter Tuning (Hadoop)

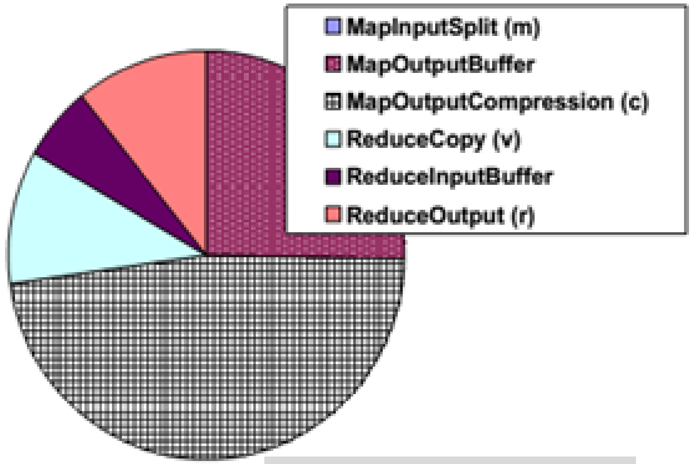
Problem: Given a MapReduce (or Spark) job with input data and running cluster, we want to find the setting of parameters that optimize the execution time of the job (i.e., minimize the job execution time)



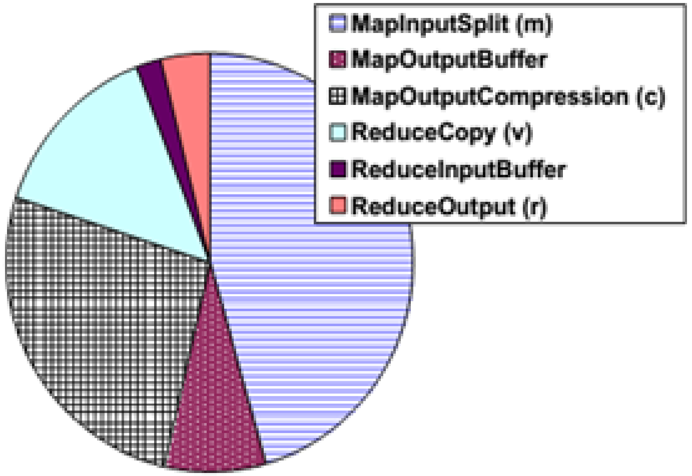
Tuned Key Parameters in Hadoop

Parameter Name	Description
MapInputSplit	Split number for map jobs
MapOutputBuffer	Buffer size of map output
MapOutputCompression	Whether the map output data is compressed
ReduceCopy	Time to start the copy in Reduce phase
ReduceInputBuffer	Input buffer size of Reduce
ReduceOutput	Reduce output block size

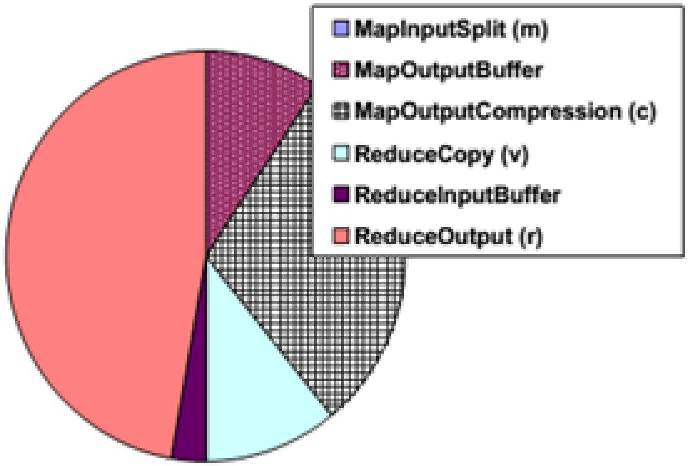
Impact of Parameters on Selected Jobs



TeraSort Job



N-gram Job



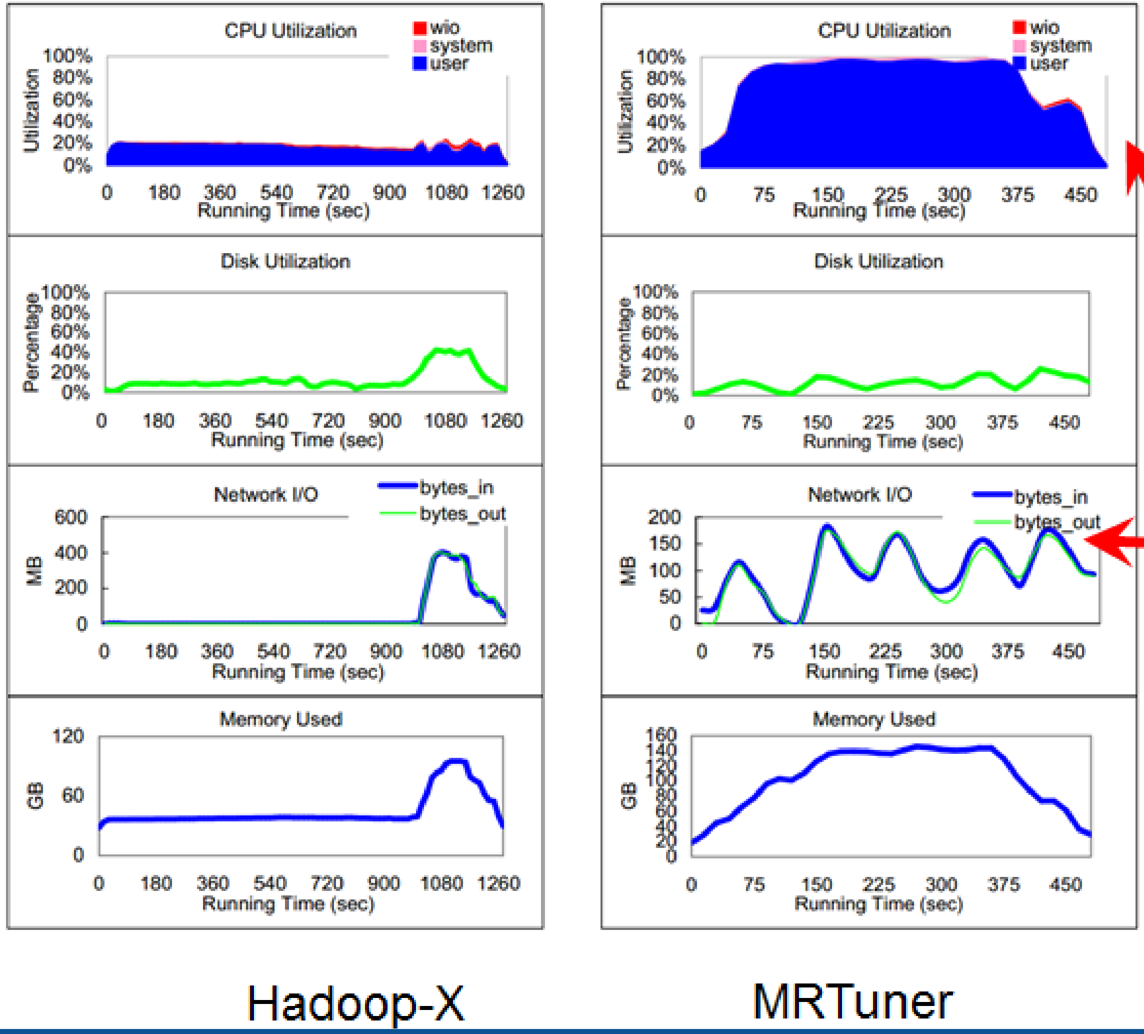
PageRank Job

Compression is important

Map task # is important

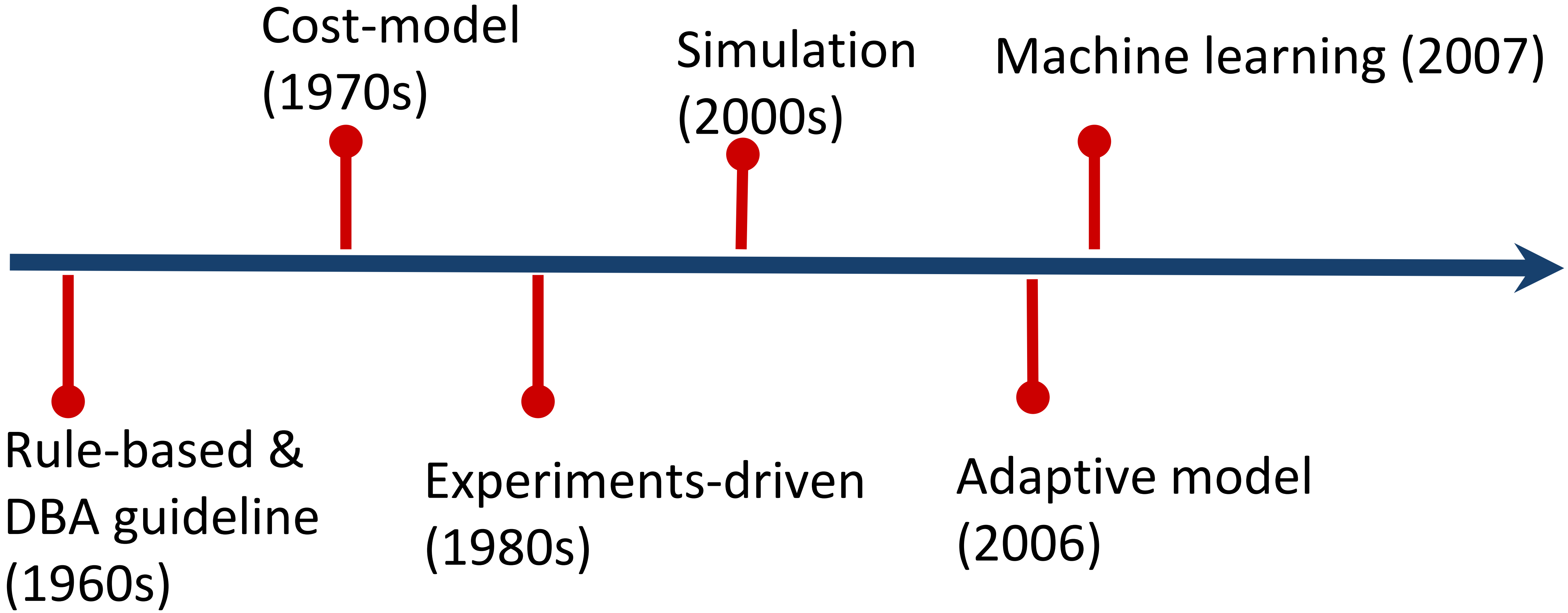
Reduce task # is important

Comparison between Hadoop-X and MRTuner with Different Parameters



CPU and Network utilizations are higher.

50 Years of Knob Tuning



Classification of Existing Approaches

Approach	Main Idea
Rule-based	Based on the experience of human experts
Cost Modeling	Using statistical cost functions
Simulation-based	Modular or complete system simulation
Experiment-driven	Execute an experiment with different parameter settings
Machine Learning	Employ machine learning methods
Adaptive	Tune configuration parameters adaptively

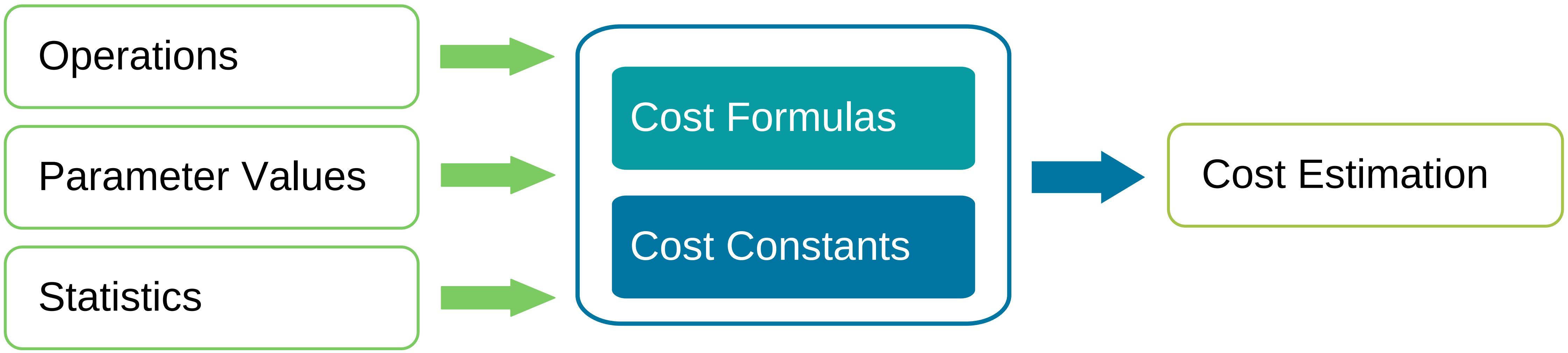
Rule-based Approach

- Assist users based on the experience of human experts

Parameter Name	***Default***	***Description***	***Recommendation***
dfs.replication in HDFS	*3*	*Lower it to reduce replication cost. Higher replication can make data local to more workers, but more space overhead*	*IF (Running time is the critical goal and enough space) Set 5 Otherwise Set 3*

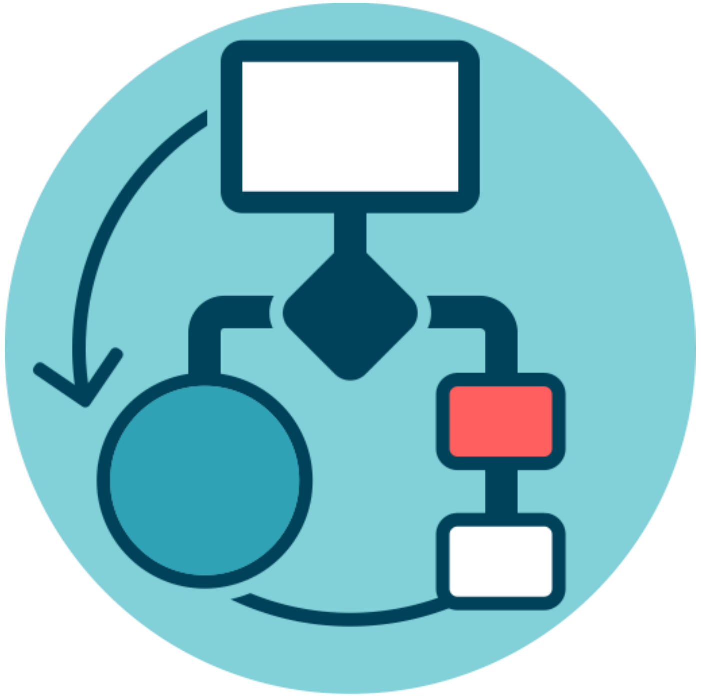
Cost Modeling Approach

- Build performance prediction models by using statistical cost functions



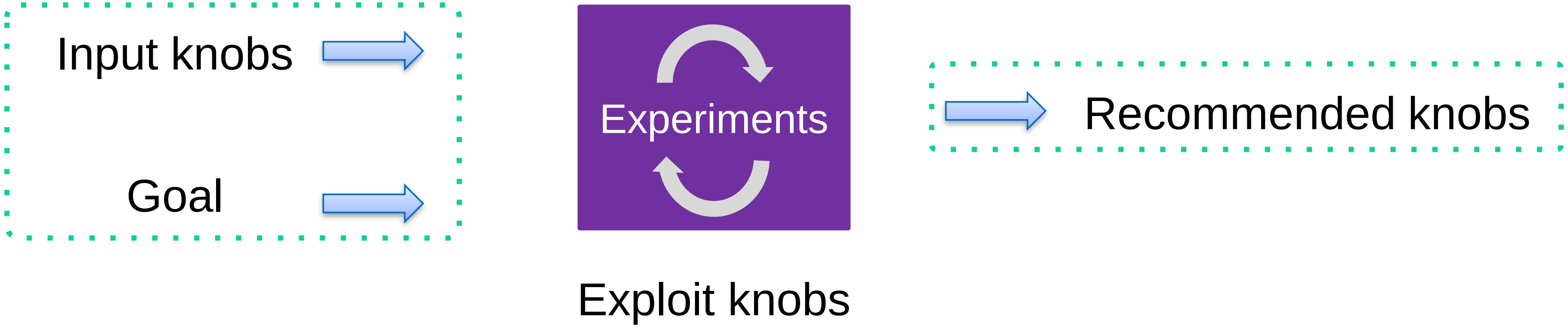
Simulation-based Approach

- Build performance models based on modular or complete simulation



Experiment-driven Approach

- Execute the experiments repeatedly with different parameter settings, guided by a search algorithm



Machine Learning Approach

- ➢ Establish performance models by employing machine learning methods
- ➢ Consider the complex system as a whole and assume no knowledge of system

Adaptive Approach

- Tune parameters adaptively while an application is running
- Adjust the parameter settings as the environment changes

CLOT (2006) strategy

Outline

Motivation and Background

History and Classification

Parameter Tuning on Databases

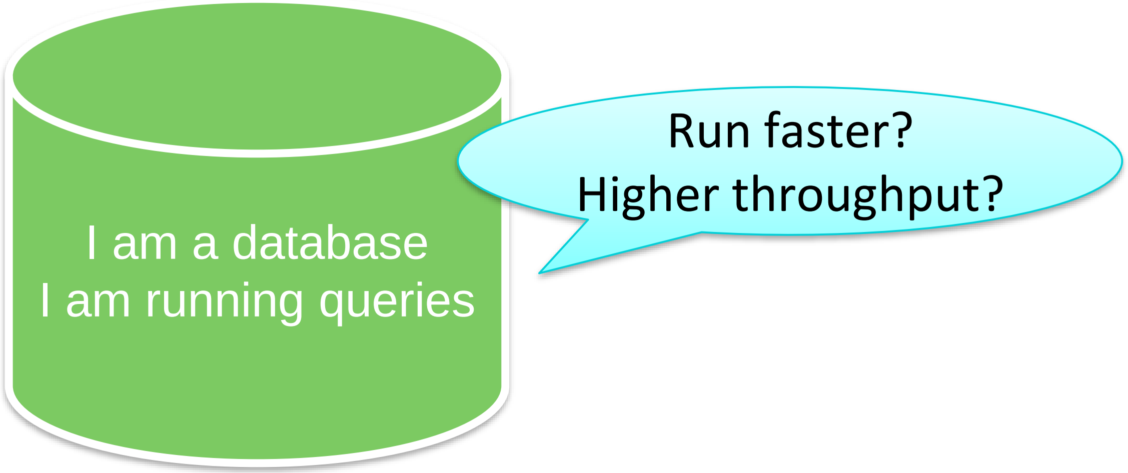
Parameter Tuning on Big Data Systems

Applications of Automatic Parameter Tuning

Open Challenges and Discussion

What and How to Tune?

- What to configure?
  - Which parameters (knobs)?
  - Which are most important?



- How to tune (to best throughput)?
  - Increase buffer size?
  - More parallelism on writing?

Figure. Tuning guitar knobs to right notes (frequencies)

What to Tune – Some Important Knobs for throughput

Threads
Timeout Settings
Memory Cache

Parameter Name	Brief Description and Use	Deafult
bgwriter_delay	Background writer's delay between activity rounds	200ms
bgwriter_lru_maxpages	Max number of buffers written by the background writer	100
checkpoint_segments	Max number of log file segments between WAL checkpoints	3
checkpoint_timeout	Max time between automatic WAL checkpoints	5min
deadlock_timeout	Waiting time on locks for checking for deadlocks	1s
default_statistics_target	Default statistics target for table columns	100
effective_cache_size	Effective size of the disk cache accessible to one query	4GB
shared_buffers	Memory size for shared memory buffers	128MB

What are the Important Parameters and How to Choose

- Affect the performance most (manually)
  - Based on expert experiences
  - Default documentation

Parameters have strong correlation to performance are important!

Performance-sensitive parameters are important!

If you want higher throughput, better tuning memory-related parameters

What are the Important Parameters and How to Choose

- Affect the performance most
- **Strongest correlation** between parameters and objective function (model)
  - Linear regression model for independent parameters:
    - Regularized version of least squares – Lasso (*OtterTune 2017)*
      - Interpretable, stable, and computationally efficient with higher dimensions
  - Deep learning model (*CBDTune 2019)*
    - The important input parameters will gain higher **weights** in training

$$Y = \underbrace{\beta_0 + \beta_1}_{\text{Weights}} \underbrace{X_1}_{\text{Knobs}} + \ldots + \beta_N X_N$$

How to Tune – Key Tuning Goals

- **Avoidance:** to identify and avoid error-prone configuration settings
- **Ranking:** to rank parameters according to the performance impact
- **Profiling:** to classify and store useful log information from previous runs
- **Prediction:** to predict the database or workload performance under hypothetical resource or parameter changes
- **Tuning:** to recommend parameter values to achieve objective goals

How to Tune – Tuning Methods

Methods	Approach	Methodology	Target Level
Rule-based	SPEX (2013)	Constraint inference	Avoidance
	Xu (2015)	Configuration navigation	Ranking
Cost-model	STMM (2006)	Cost model	Tuning
Simulation-based	Dushyanth (2005)	Trace-based simulation	Prediction
	ADDM (2005)	DAG model & simulation	Profiling, tuning
Experiment driven	SARD (2008)	P&B statistical design	Ranking
	iTuned (2009)	LHS & Guassian Process	Profiling, tuning
Machine Learning	Rodd (2016)	Neural Networks	Tuning
	OtterTune (2017)	Guassian Process	Ranking, tuning
	CDBTune (2019)	Deep RL	Tuning
Adaptive	COLT (2006)	Cost Vs. Gain analysis	Profiling, tuning

Relational Database Tuning Methods

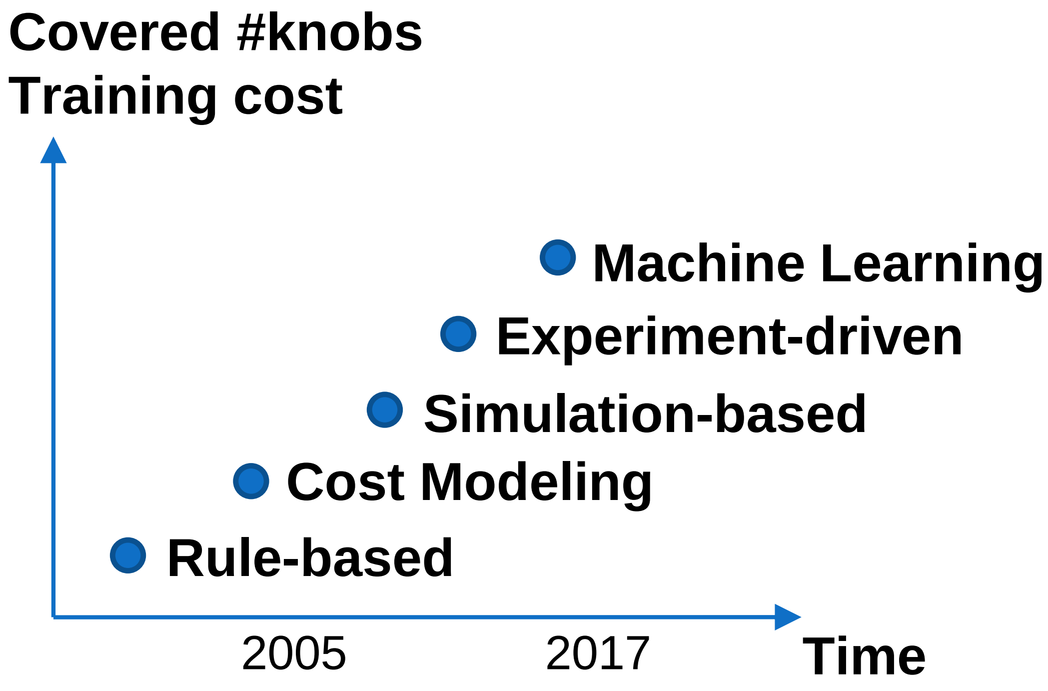


Figure. Developing trend: putting more training cost to uncover more knobs

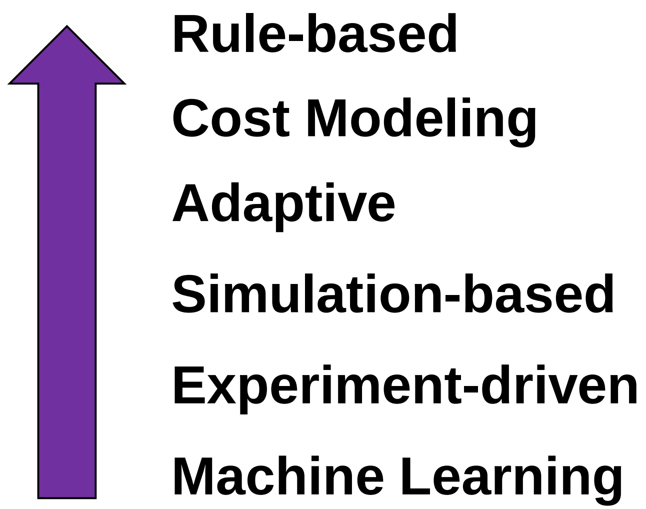
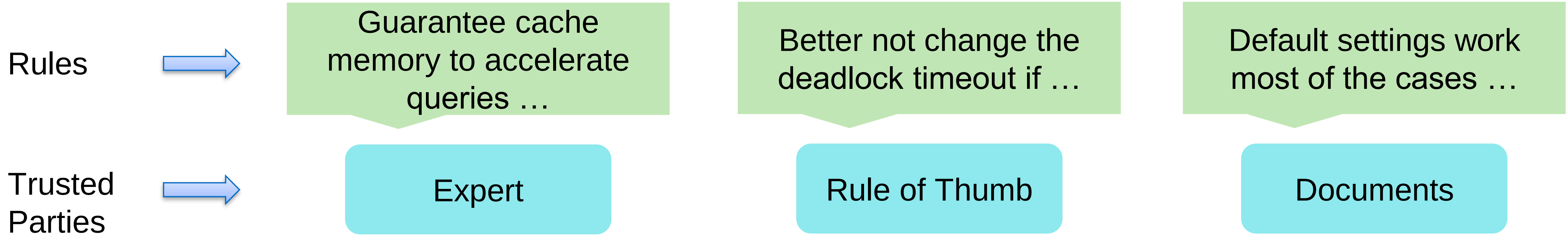


Figure. Required expert knowledge on system

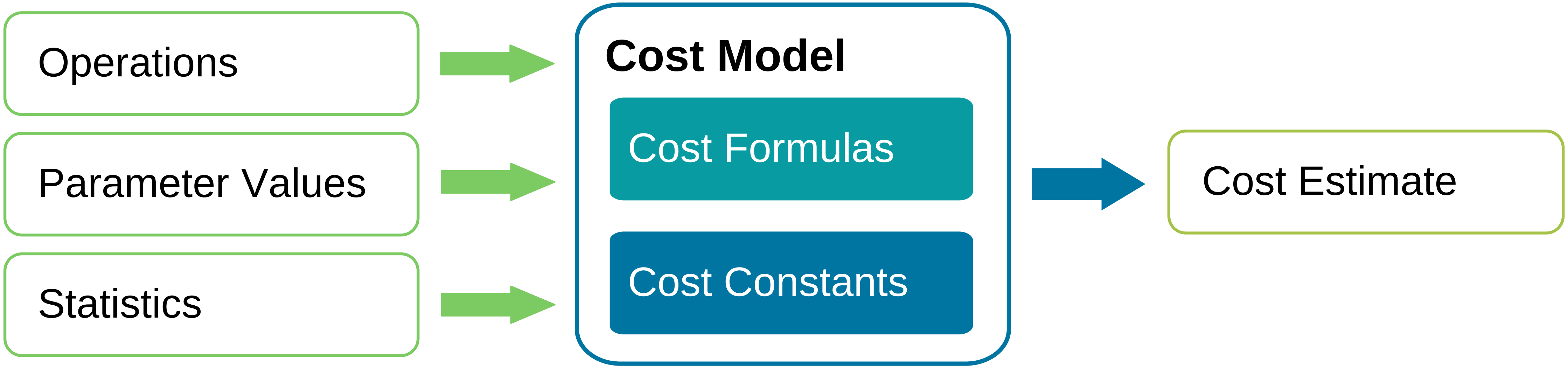
Tuning Method: Rule-based

- Tuning based on rules derived from DBAs' expertise, experience, and knowledge, or Rule of Thumb default recommendation



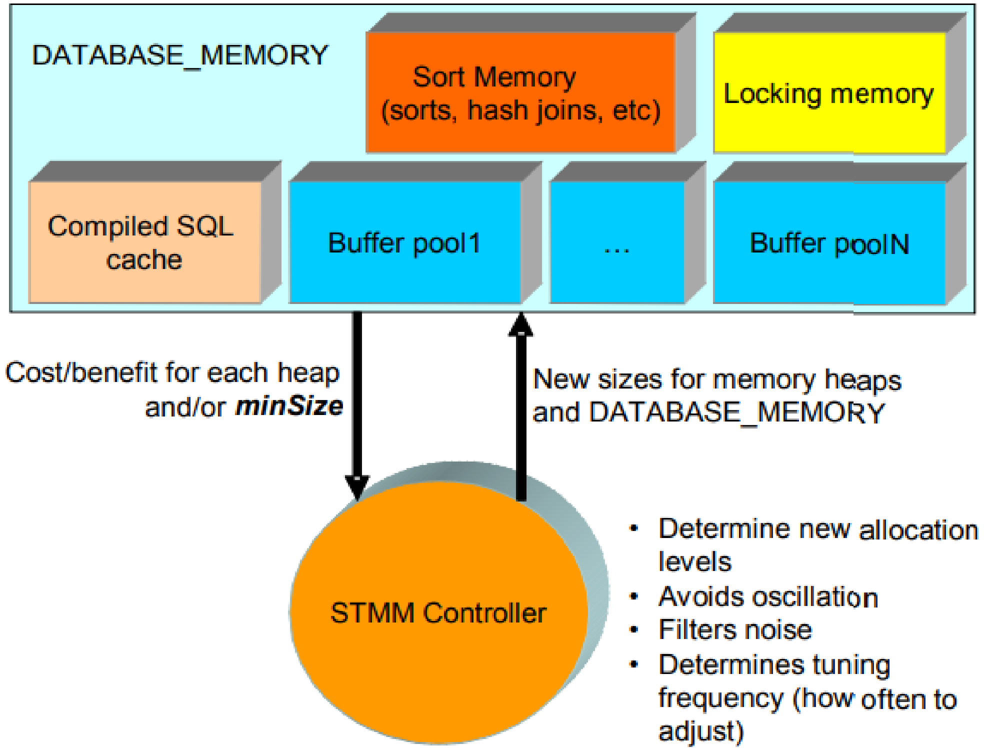
Tuning Method: Cost Modeling

➢ A cost model establishes a performance model by cost functions based on the deep understanding of system components



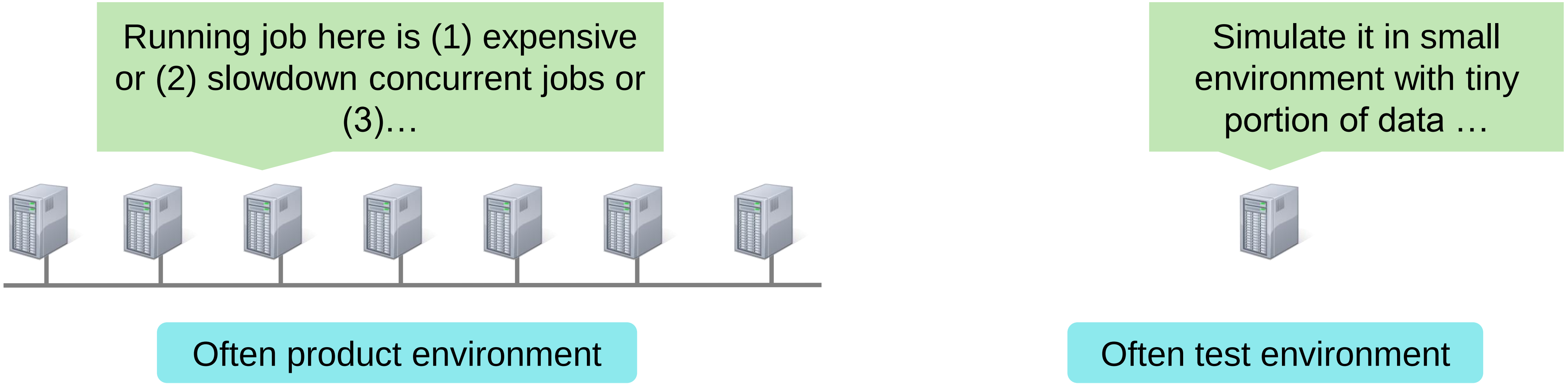
Tuning Method: Cost Modeling (STMM)

- STMM: Adaptive Self-Tuning Memory in DB2 (2006)
  - Reallocates memory for several critical components(e.g., compiled statement cache, sort, and buffer pools)



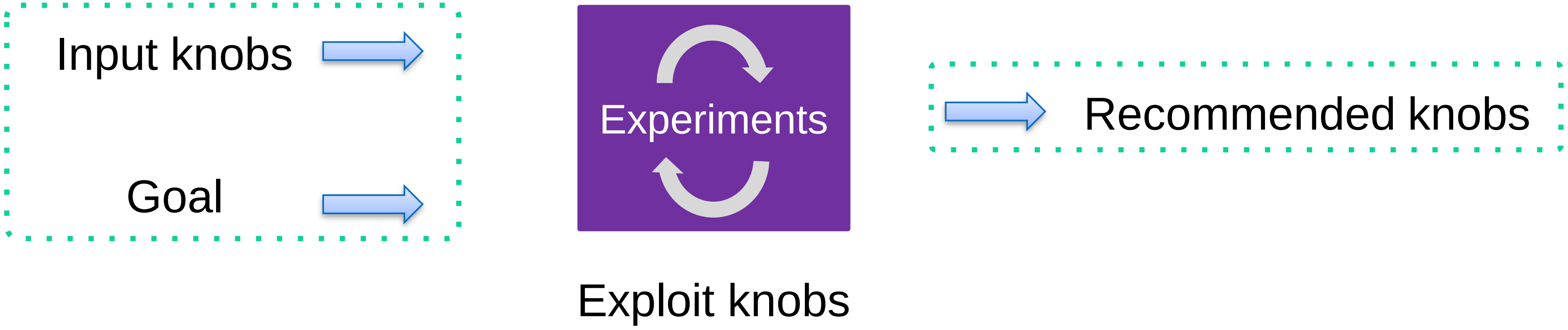
Tuning Method: Simulation-based

- A simulation-based approach simulates workloads in one environment and learns experience or builds models to predict the performance in another.



Tuning Method: Experiment-driven

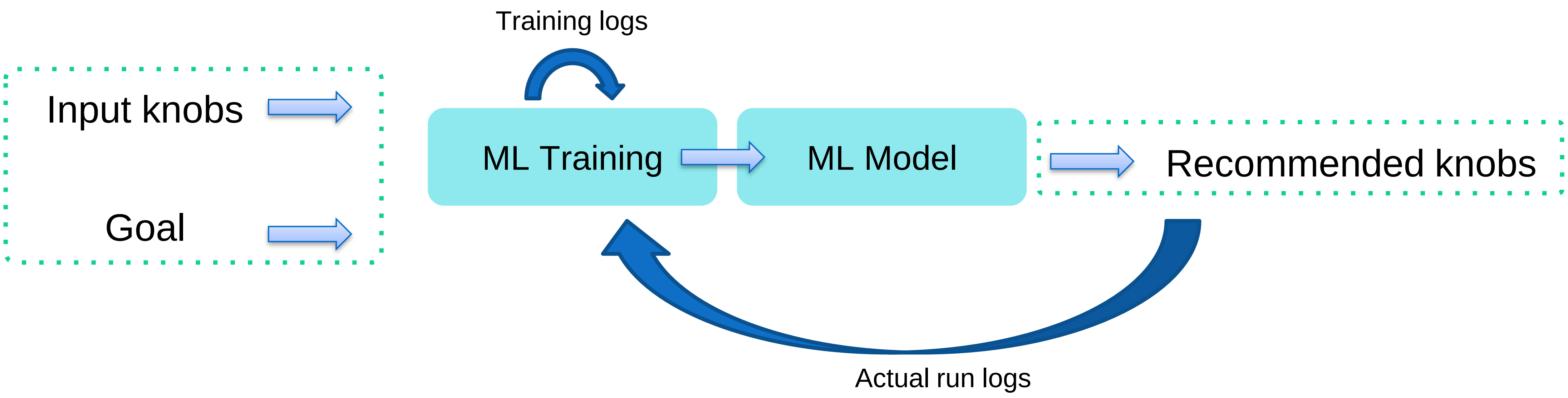
➢ An experiment-driven approach relies on repeated executions of the same workload under different configuration settings towards tuning parameter values



Classic paper: Tuning Database Configuration Parameters with iTuned. 2009

Tuning Method: Machine Learning

- Machine Learning (ML) approaches aim to tune parameters automatically by taking advantages of ML methods.



Tuning Method: Machine Learning (OtterTune 2017)

- **Factor Analysis:** transform high dimension parameters to few factors
- **Kmeans:** Cluster distinct metrics
- **Lasso:** Rank parameters
- **Gaussian Process:** Predict and tune performance

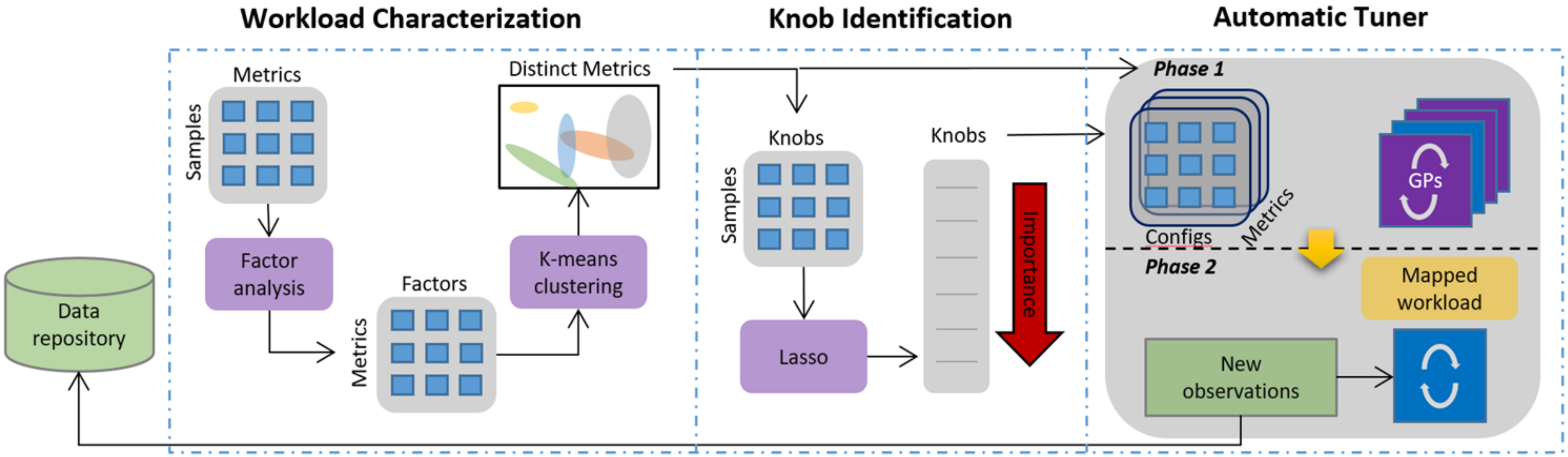


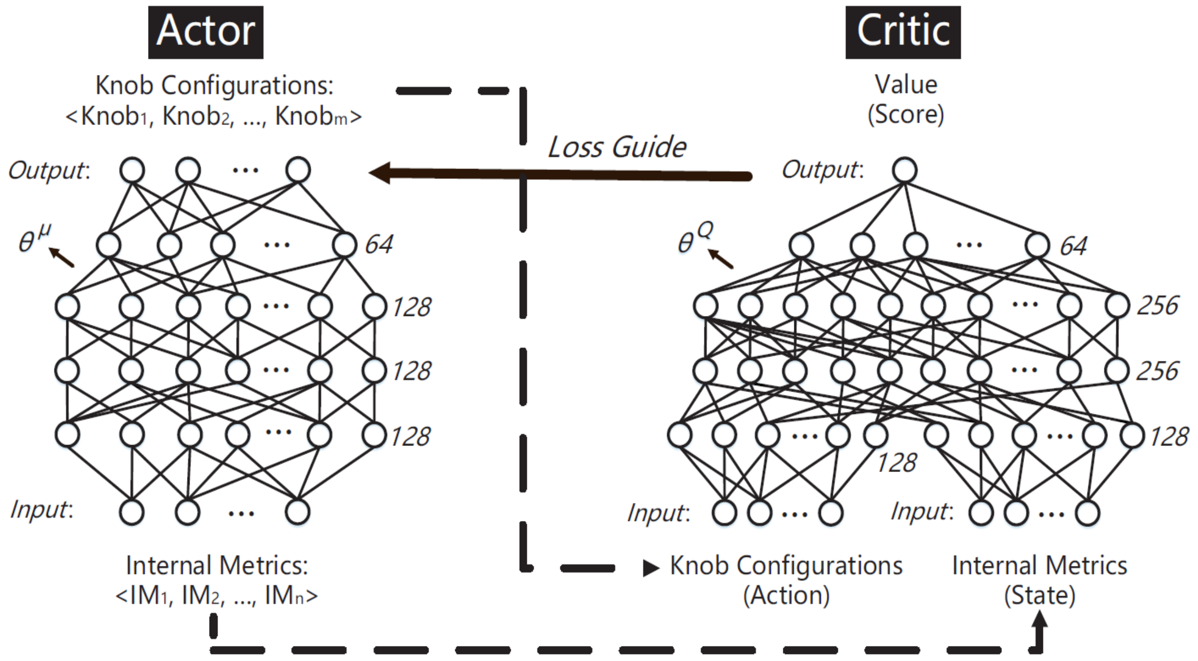
Figure. OtterTune system architecture

Tuning Method: Machine Learning (CDBTune 2019)

- **Reinforcement learning**
  - **State:** knobs and metrics
  - **Reward:** performance change
  - **Action:** recommended knobs
  - **Policy**: Deep Neural network
- **Key idea**
  - Feedback: try-and-error method
    - Recommend -> good/bad
  - Deep deterministic policy gradient
    - Actor critic algorithm

Reward: Throughput and latency performance change Δ from time t − 1 and the initial time to time t

Figure. CDBTune Deep deterministic policy gradient



Tuning Method: Adaptive

- An adaptive approach changes parameter configurations online as the environment or query workload changes

Figure. CLOT (2006) strategy

The Differences of Tuning Database & Big Data Systems in research papers

	Relational Database	Big Data System
Parameters	More parameters on memory	More parameters on vcores
Resource	Often fixed resources	Now more varying resources
Scalability	Often single machine	Often many machines in a distributed environment
Metrics	Throughput, latency	Time, resource cost

References (1/1)

- Narayanan, Dushyanth, Eno Thereska, and Anastassia Ailamaki. "Continuous resource monitoring for self-predicting DBMS." 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. IEEE, 2005.
- Dias, K., Ramacher, M., Shaft, U., Venkataramani, V., & Wood, G.. Automatic Performance Diagnosis and Tuning in Oracle. In CIDR (pp. 84-94), 2005
- Schnaitter, K., Abiteboul, S., Milo, T., & Polyzotis, N. Colt: continuous on-line tuning. In Proceedings of the 2006 ACM SIGMOD international conference on Management of data (pp. 793-795). ACM, 2006. (**CLOT**)
- Storm, Adam J., et al. "Adaptive self-tuning memory in DB2." Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006. (**STMM**)
- Debnath, Biplob K., David J. Lilja, and Mohamed F. Mokbel. "SARD: A statistical approach for ranking database tuning parameters." 2008 IEEE 24th International Conference on Data Engineering Workshop. IEEE, 2008
- Babu, S., Borisov, N., Duan, S., Herodotou, H., & Thummala, V. Automated Experiment-Driven Management of (Database) Systems. In HotOS, 2009.
- Duan, Songyun, Vamsidhar Thummala, and Shivnath Babu. "Tuning database configuration parameters with iTuned." Proceedings of the VLDB Endowment 2.1: 1246-1257, 2009. (**iTuned**)
- Rodd, Sunil F., and Umakanth P. Kulkarni. "Adaptive tuning algorithm for performance tuning of database management system." arXiv preprint arXiv:1005.0972, 2010.
- Van Aken, D., Pavlo, A., Gordon, G. J., & Zhang, B.. Automatic database management system tuning through large-scale machine learning. In Proceedings of the 2017 ACM International Conference on Management of Data(pp. 1009-1024). ACM, 2017. (**OtterTune**)
- Zhang, J., Liu, Y., Zhou, K., Li, G., Xiao, Z., Cheng, B., & Ran, M. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In Proceedings of the 2019 International Conference on Management of Data (pp. 415-432). ACM, 2019. (**CBDTune**)

Outline

Motivation and Background

History and Classification

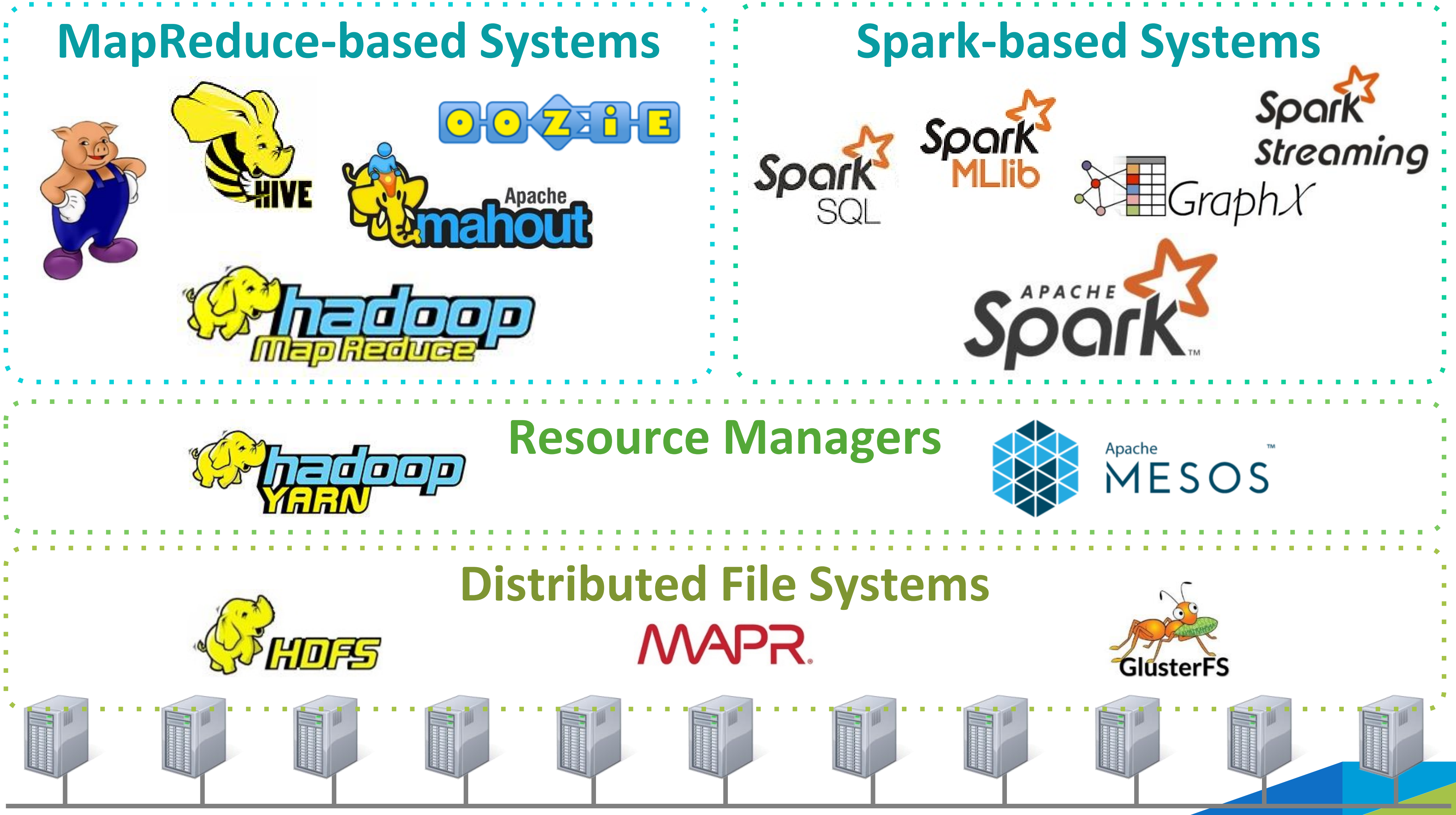
Parameter Tuning on Databases

Parameter Tuning on Big Data Systems

Applications of Automatic Parameter Tuning

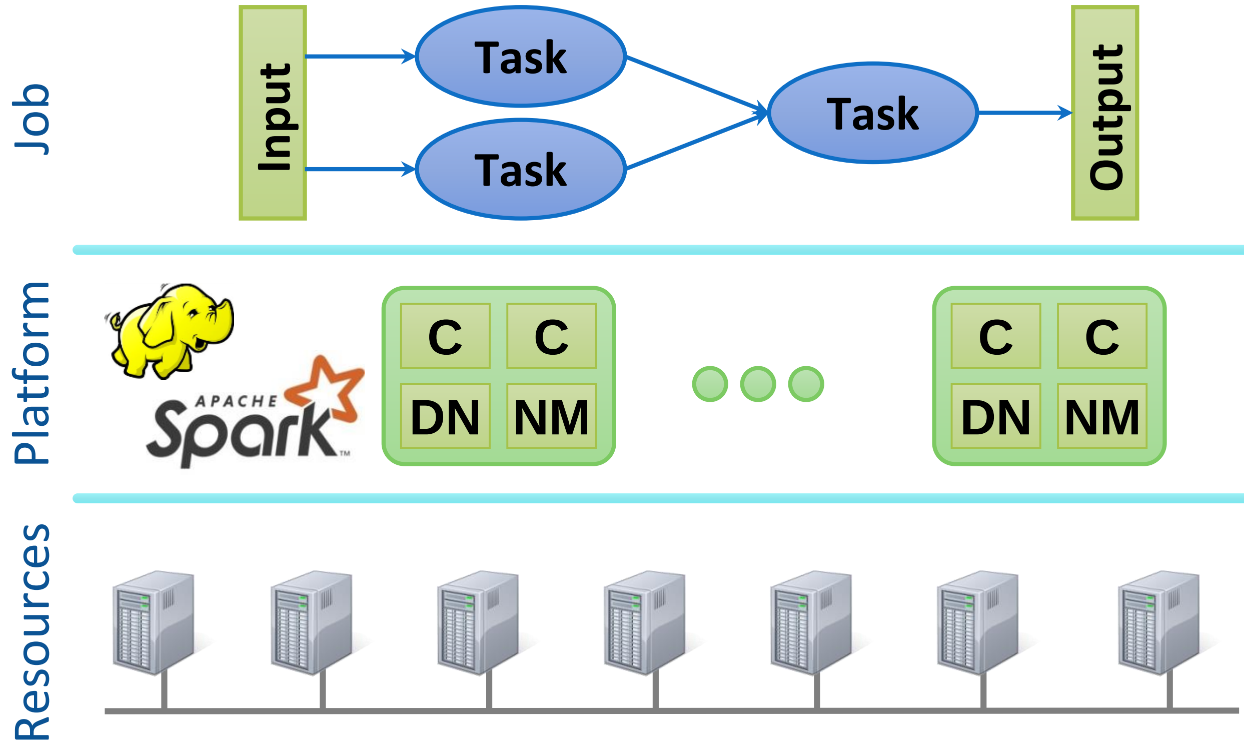
Open Challenges and Discussion

Ecosystems for Big Data Analytics



Executing Analytics Workloads

Goal:
Execute an analytics workload in < 2 hours



Decisions:

- Task parallelism
- Use compression
- …

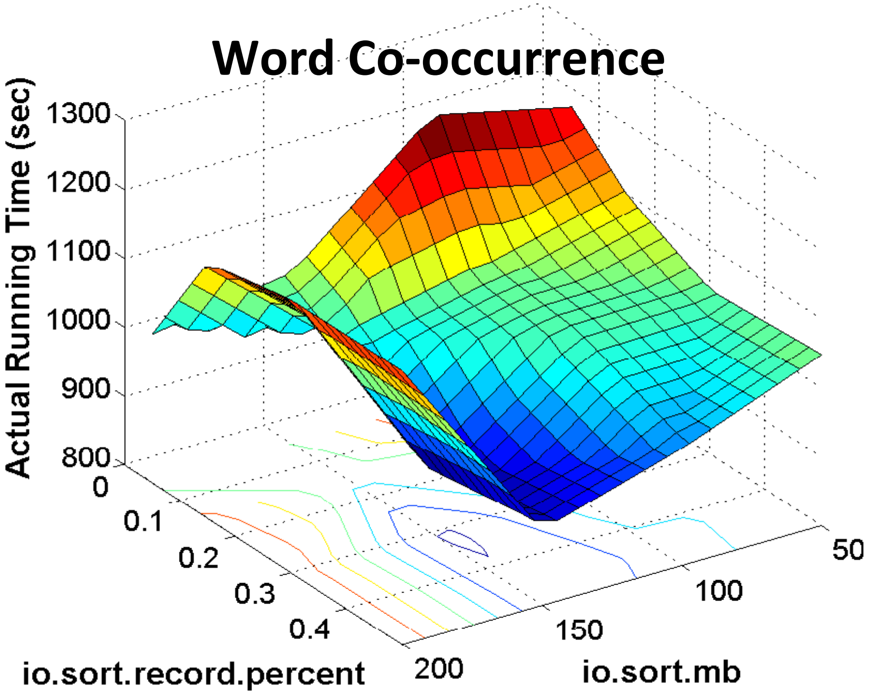
- Container settings
- Executor cores
- …

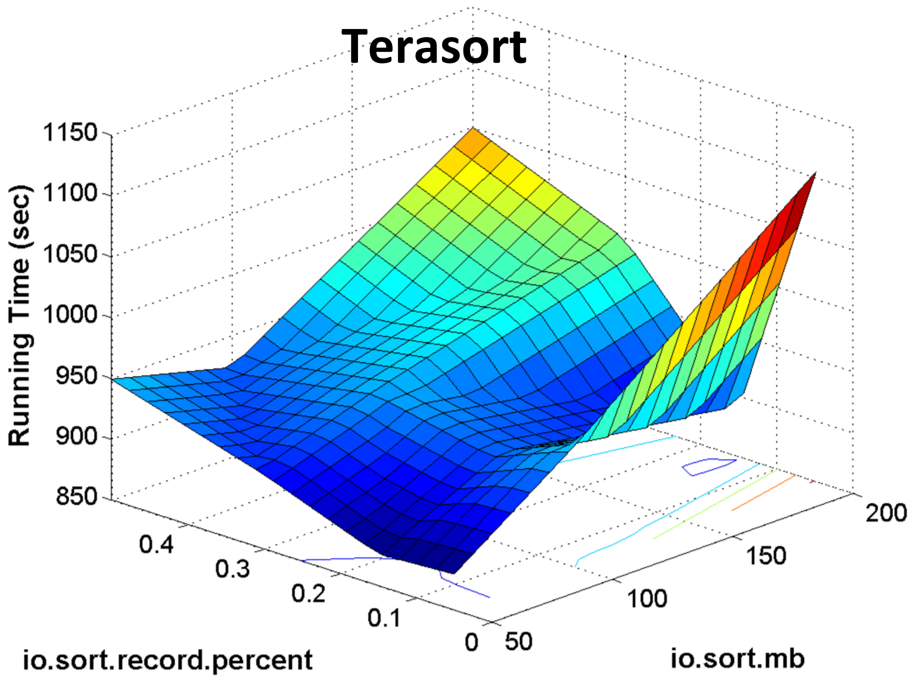
- Number of nodes
- Machine specs
- …

Effect of Job-level Configuration Parameters

- 190+ parameters in Hadoop, 15-20 impact performance
- 200+ parameters in Spark, 20-30 impact performance

Two-dimensional projections of a multi-dimensional surface





Scenario: 2 MapReduce jobs, 50GB, 16-node EC2 cluster

Tuning Challenges

- High-dimensional space of configuration parameters
- Non-linear effect of hardware/applications/parameters on performance
- Heavy use of programming languages (e.g., Java/Python)
- Lack of schema & statistics for input data residing in files
- Terabyte-scale data cycles

Applying Cost-based Optimization

- Goal: $$perf = F\left(G_p, \{d\}, r, \{c\}\right)$$ $$\left\{c_{opt}\right\} = \underset{\{c\} \in S}{\arg\min} F\left(G_p, \{d\}, r, \{c\}\right)$$

  - $G_p = (P, E)$
    - P : set of programs to execute
    - E : the program dependencies
  - $\{d\}$: The data set properties
  - r : The cluster resources
  - $\{c\}$: The set of configuration setting for each job

Applying Cost-based Optimization

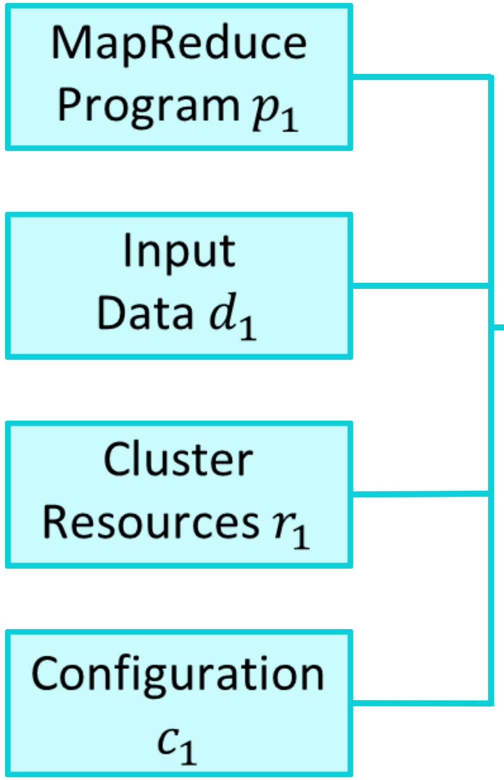
- **Goal:** $$perf = F\left(G_p, \{d\}, r, \{c\}\right)$$
  $$\{c_{opt}\} = \underset{\{c\} \in S}{\arg\min}\, F\left(G_p, \{d\}, r, \{c\}\right)$$

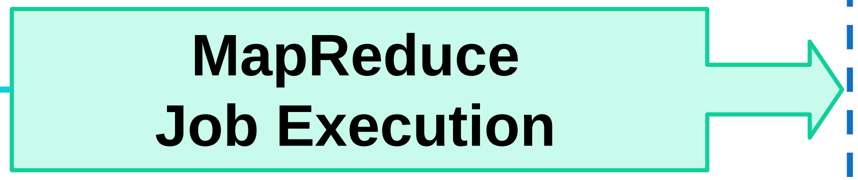
- Starfish line of work [Herodotou et. al., 2011-13] pioneered cost-based optimization for Hadoop configuration parameters

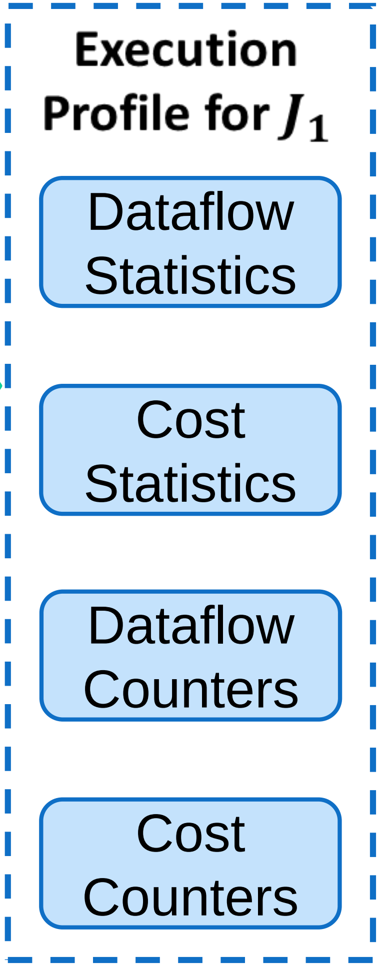
Profile → **Predict** → **Optimize**

Profile	Predict	Optimize
Collect concise & general summaries of execution	Estimate impact of hypothetical changes on execution	Enumerate & search through space S of tuning choices

Profiling MapReduce Job Execution

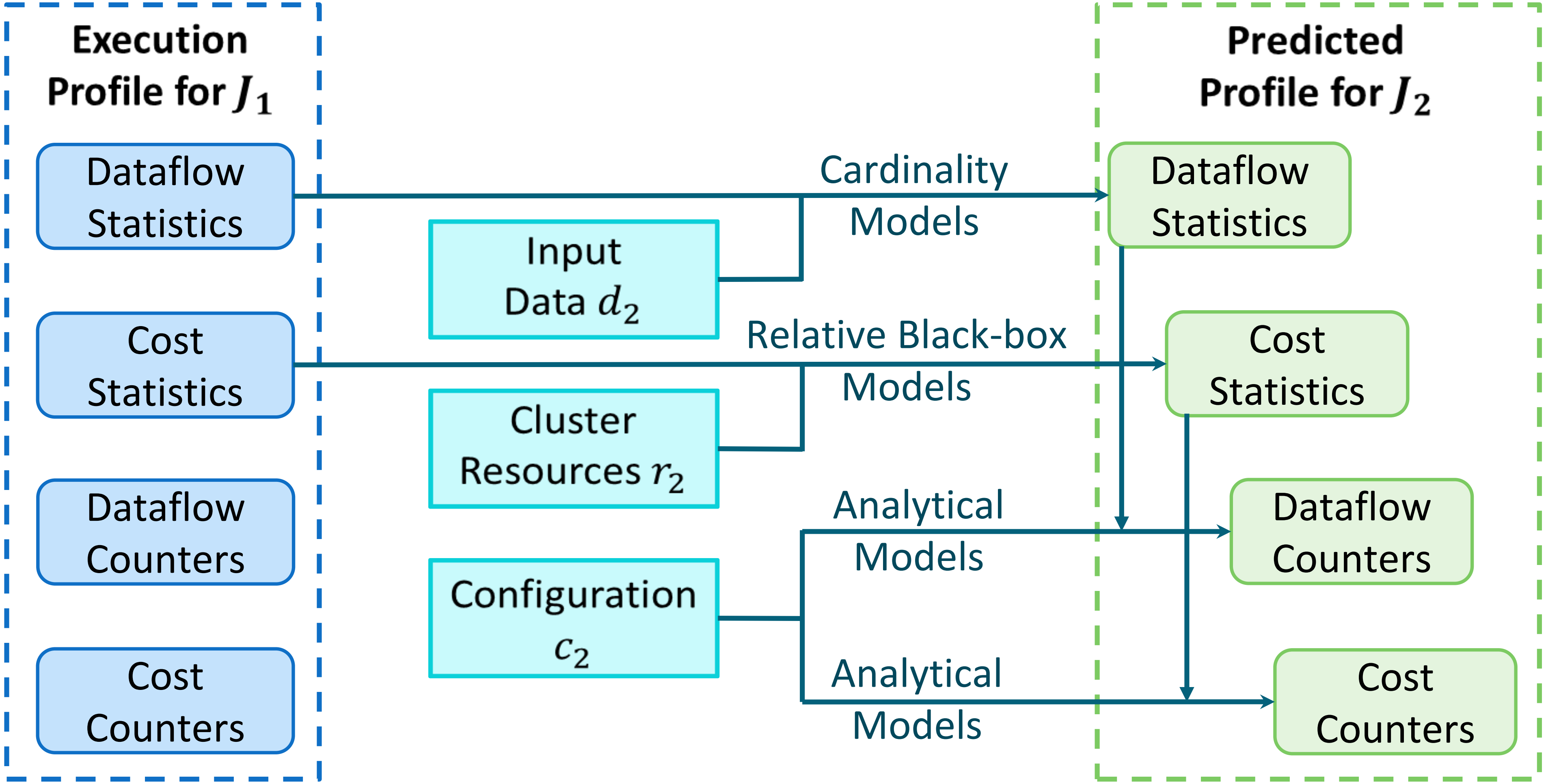




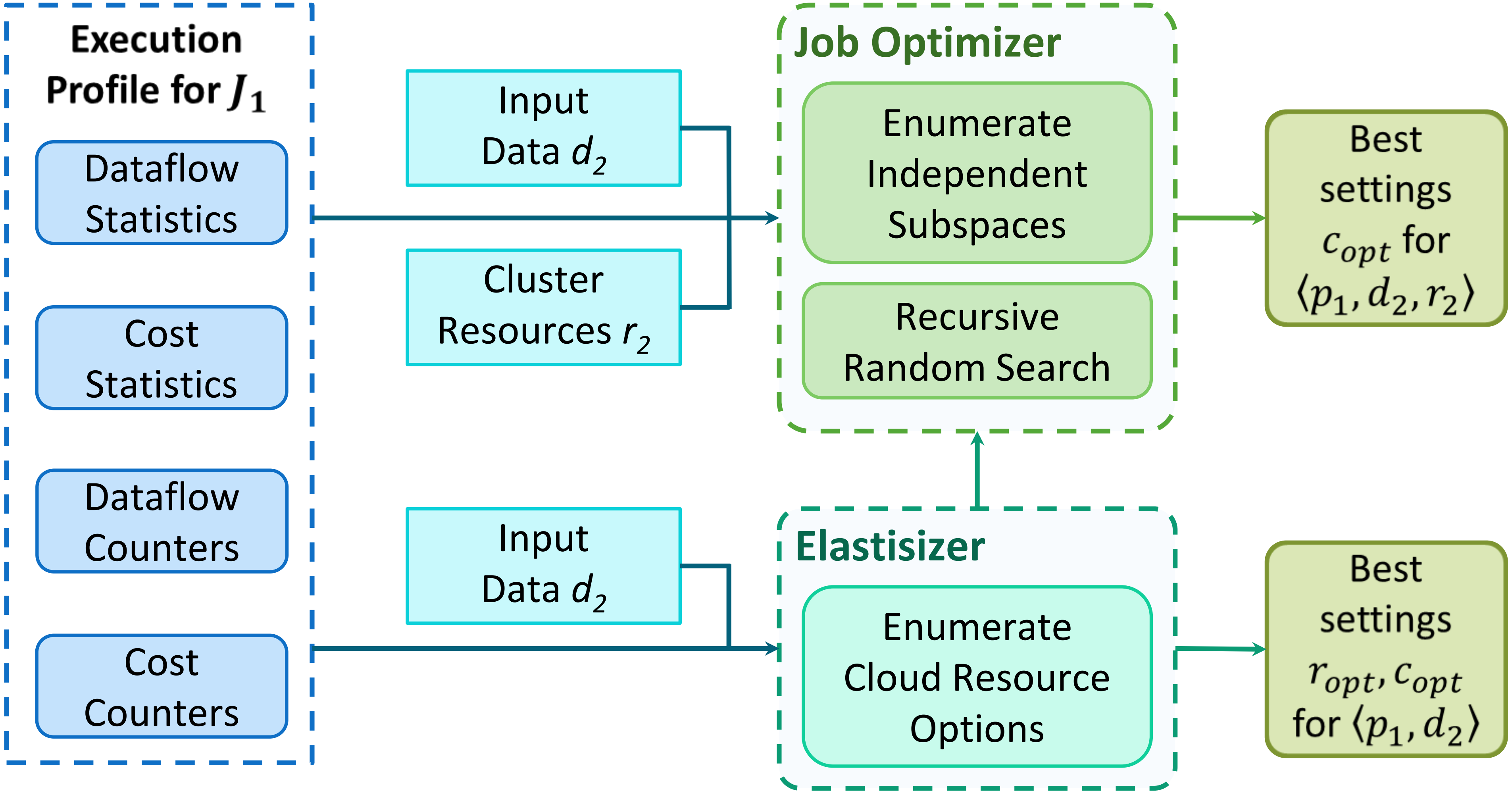


- Use dynamic instrumentation
  - Support unmodified MapReduce programs
- Use sampling techniques
  - Minimize overhead of profiling

Predicting Job Profiles in Starfish



Job Optimization & Resource Provisioning

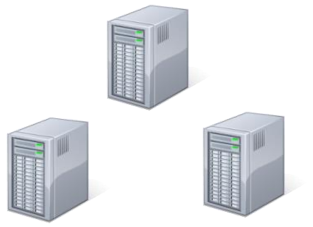


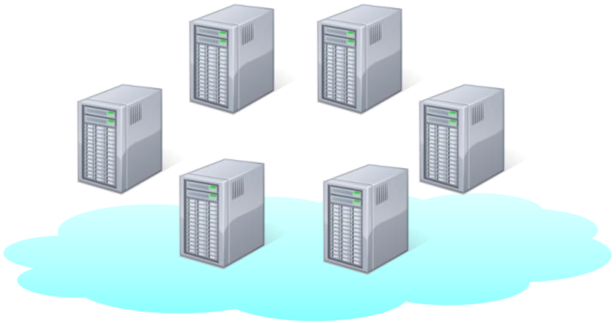
MapReduce Cost Modeling Approaches

Approach	Modeling	Optimization	Target Level
Starfish (2011-13)	Analytical & relative black box models	Recursive Random Search	Job, Platform, Cloud
ARIA (2011)	Analytical models	Lagrange Multipliers	Job, Platform
HPM (2011)	Scaling models & LR	Brute-force Search	Platform
Predator (2012)	Analytical models	Grid Hill Climbing	Job
MRTuner (2014)	PTC analytical models	Grid-based search	Job, Platform
CRESP (2014)	Analytical models & LR	Brute-force Search	Platform, Cloud
MR-COF (2015)	Analytical models & MRPerf simulation	Genetic Algorithm	Job
IHPM (2016)	Scaling models & LWLR	Lagrange Multipliers	Platform, Cloud

Spark Cost Modeling Approaches

- **Focus:**





Profile using small clusters, data samples → Predict performance on large clusters, full data

- **Unique features:**
  - Ernest (2016): Focus on machine learning Spark applications
  - Assurance (2017): Mixes white-box models with simulation
  - DynamiConf (2017): Optimizes degree of parallelism for tasks

Cost Modeling Approach: Pros & Cons

Pros	Very efficient for predicting performance
	Good accuracy in many (not complex) scenarios
Cons	Hard to capture complexity of system internals & pluggable components (e.g., schedulers)
	Models often based on simplified assumptions
	Not effective on heterogeneous clusters

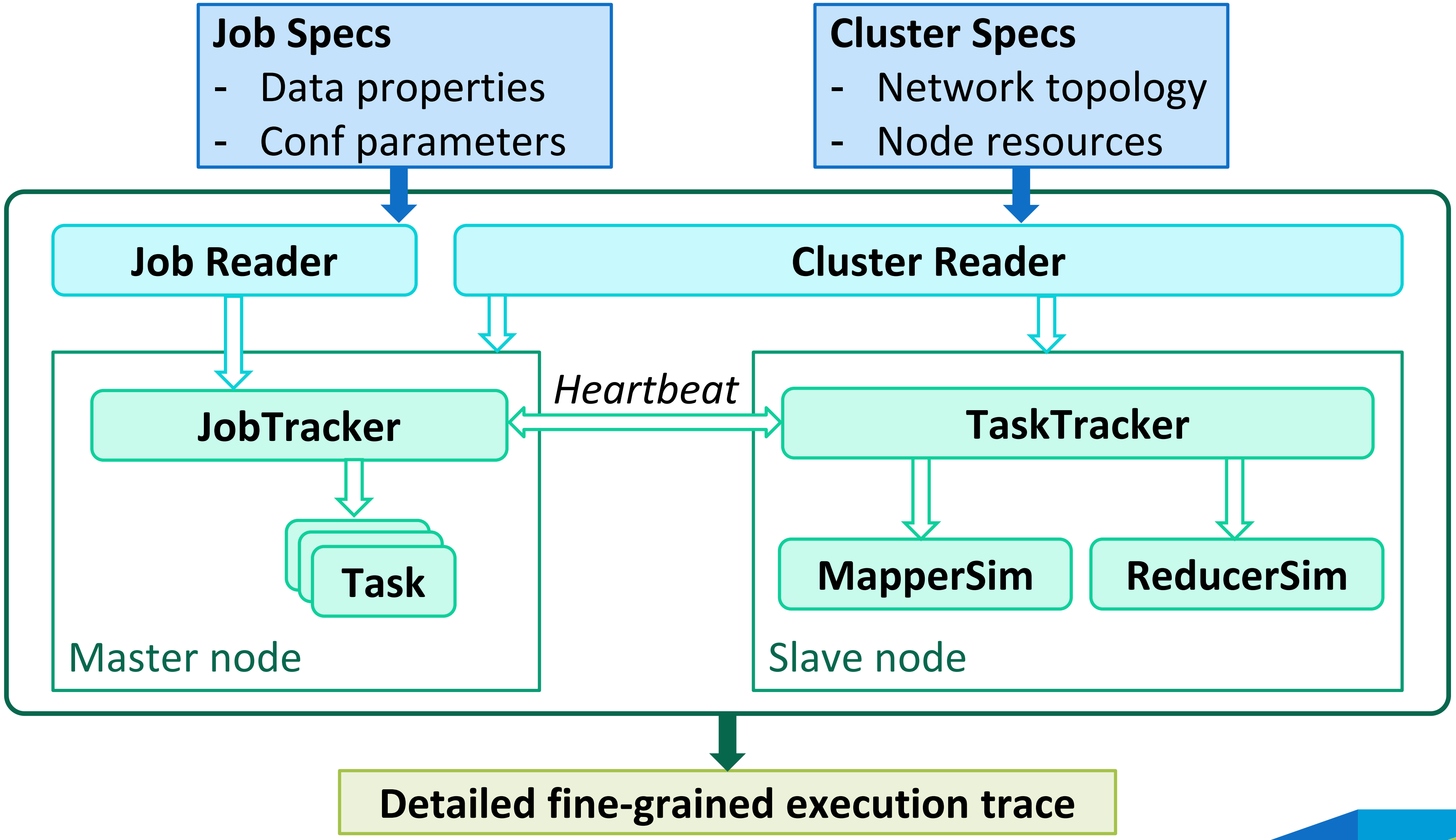
Simulation-based Approach

- Key Objective: Accurately predict MapReduce job performance at fine granularity
  - Sorry, **no** fully-fledged Spark simulator available at this point!

- Use cases:
  - Find optimal configuration settings
  - Find cluster settings based on user requirements
  - Identify performance bottlenecks
  - Test new pluggable components (e.g., schedulers)

- Common technique: discrete event simulation

HSim: Hadoop Simulator



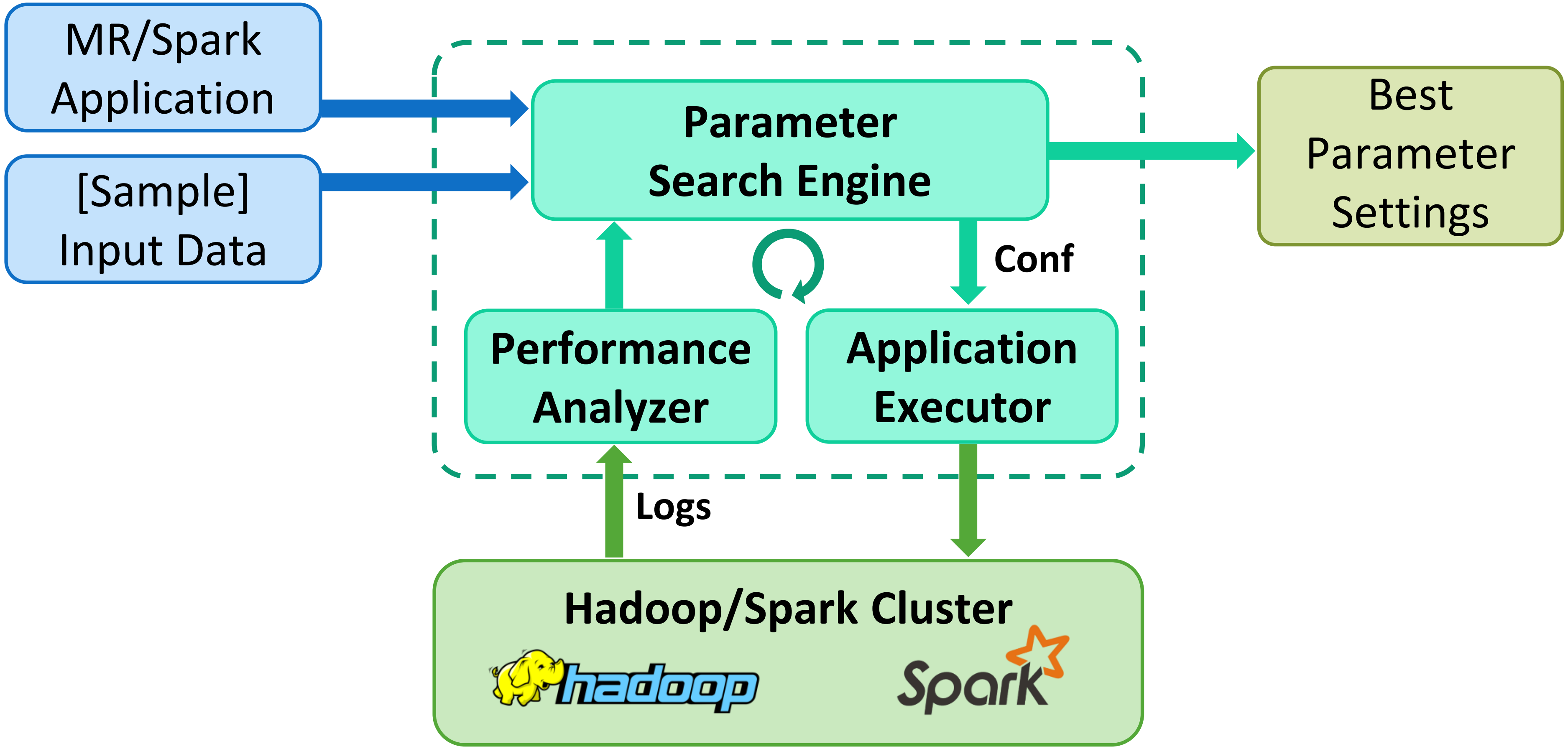
Comparison of Hadoop Simulators

Simulator	Network Traffic	Hardware Properties	MapReduce Execution	MapReduce Scheduling	Conf Parameters
MRPerf (2009)	Yes (ns-2)	Yes	Task sub-phases	No	Only few
MRSim (2010)	Yes (GridSim)	Yes	Task sub-phases	No	Several
Mumak (2009)	No	No	Only task level	No	Only few
SimMR (2011)	No	No	Task sub-phases	FIFO, Deadline	Only few
SimMapRed (2011)	Yes (GridSim)	Yes	Task sub-phases	Several	Only few
HSim (2013)	Yes (GridSim)	Yes	Task sub-phases	FIFO, FAIR	Several

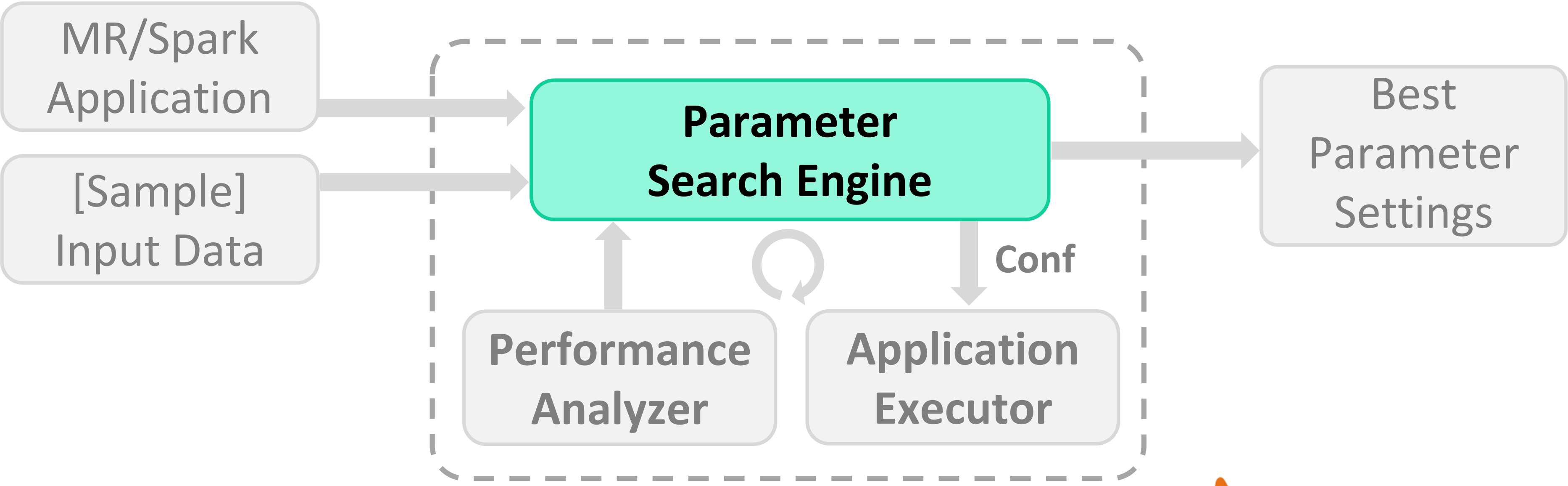
Simulation-based Approach: Pros & Cons

Pros ✓	High accuracy in simulating dynamic system behaviors
	Efficient for predicting fine-grained performance
Cons ✗	Hard to comprehensively simulate complex internal dynamics
	Unable to capture dynamic cluster utilization
	Not very efficient for finding optimal settings

General Experiment-driven Architecture



Experiment-driven Approaches



Panacea (2012)	Gunther (2013)	Petridis (2016)	AutoTune (2018)
• Grid search on independent subspaces	• Genetic search algorithm	• Search tree based on heuristics	• Multiple bound and search algorithm

Gounaris (2018) Exp-driven Approach

Use benchmarking applications (sort-by-key, shuffling, k-means)

Test parameter values separately and in pairs (117 runs for 15 parameters)

Create candidate configurations (9 complex parameter configurations)

Offline

Online

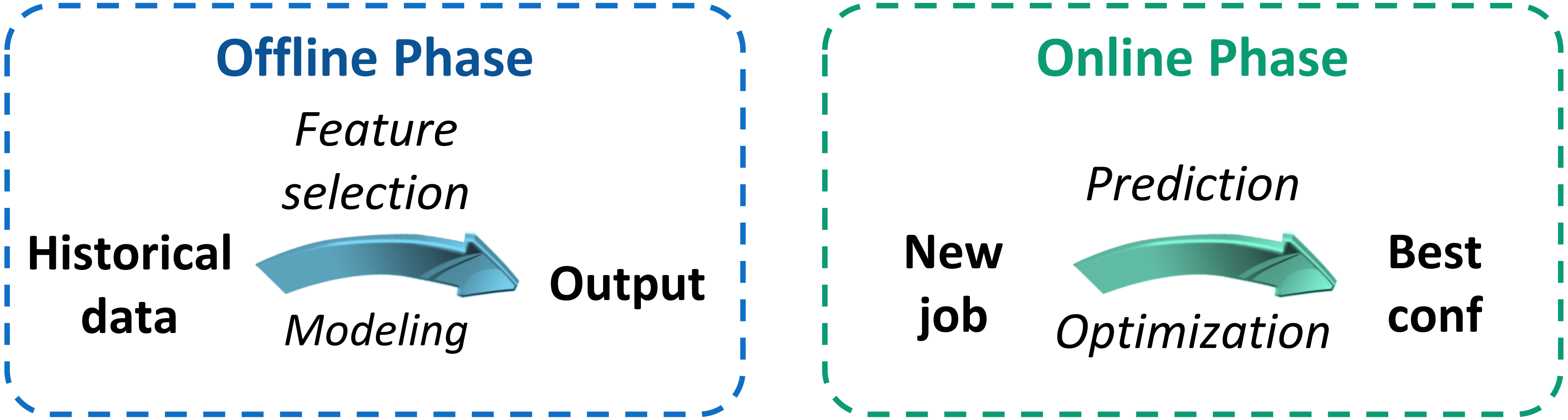
Test all candidate configurations and keep best one

Spark Application

Experiment-driven Approach: Pros & Cons

Pros ✔	Finds good settings based on real test runs on real systems
	Works across different system versions and hardware
Cons ✘	Very time consuming as it requires multiple actual runs
	Not cost effective for ad-hoc analytics applications

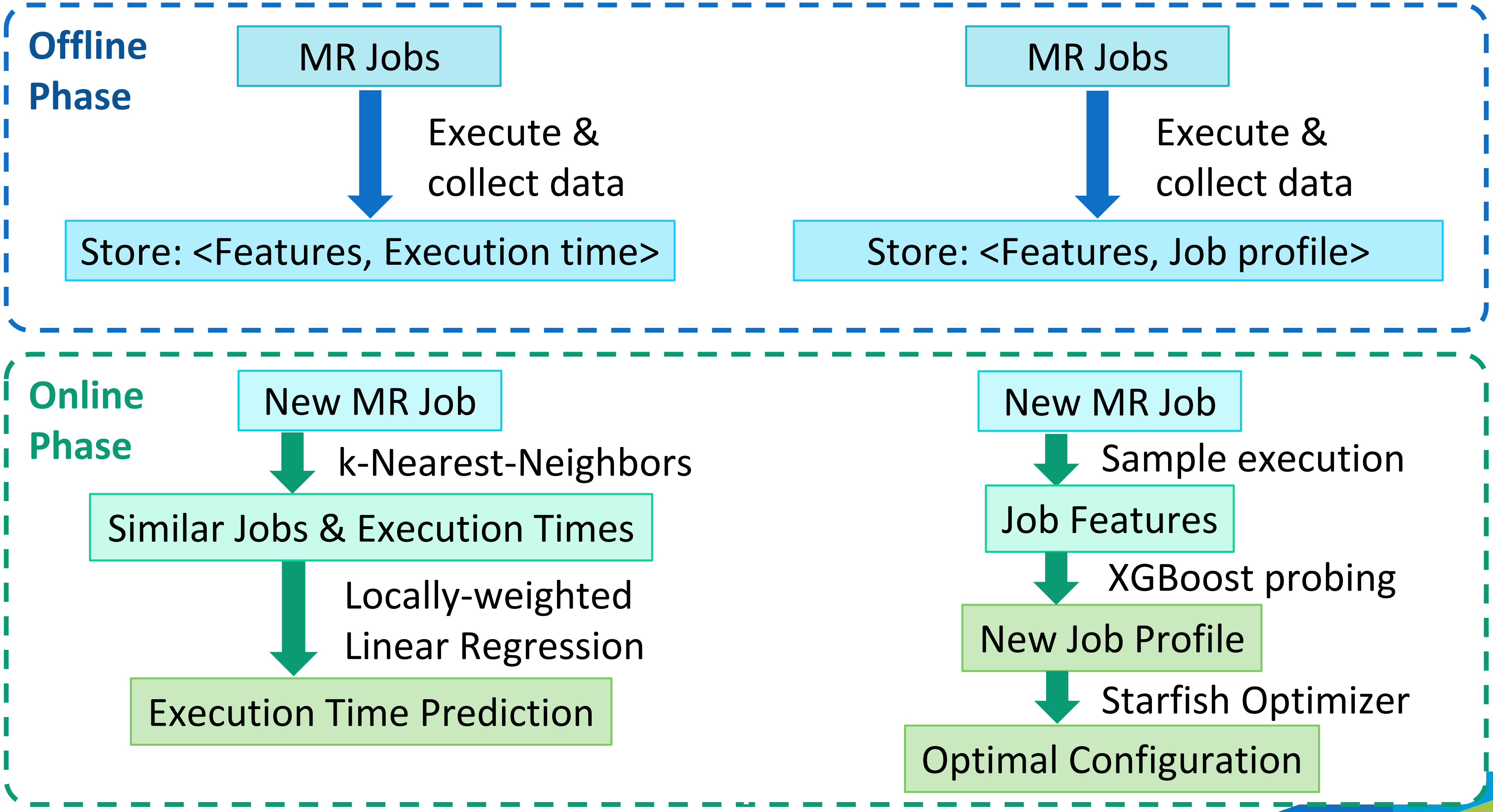
Machine Learning (ML) Approaches



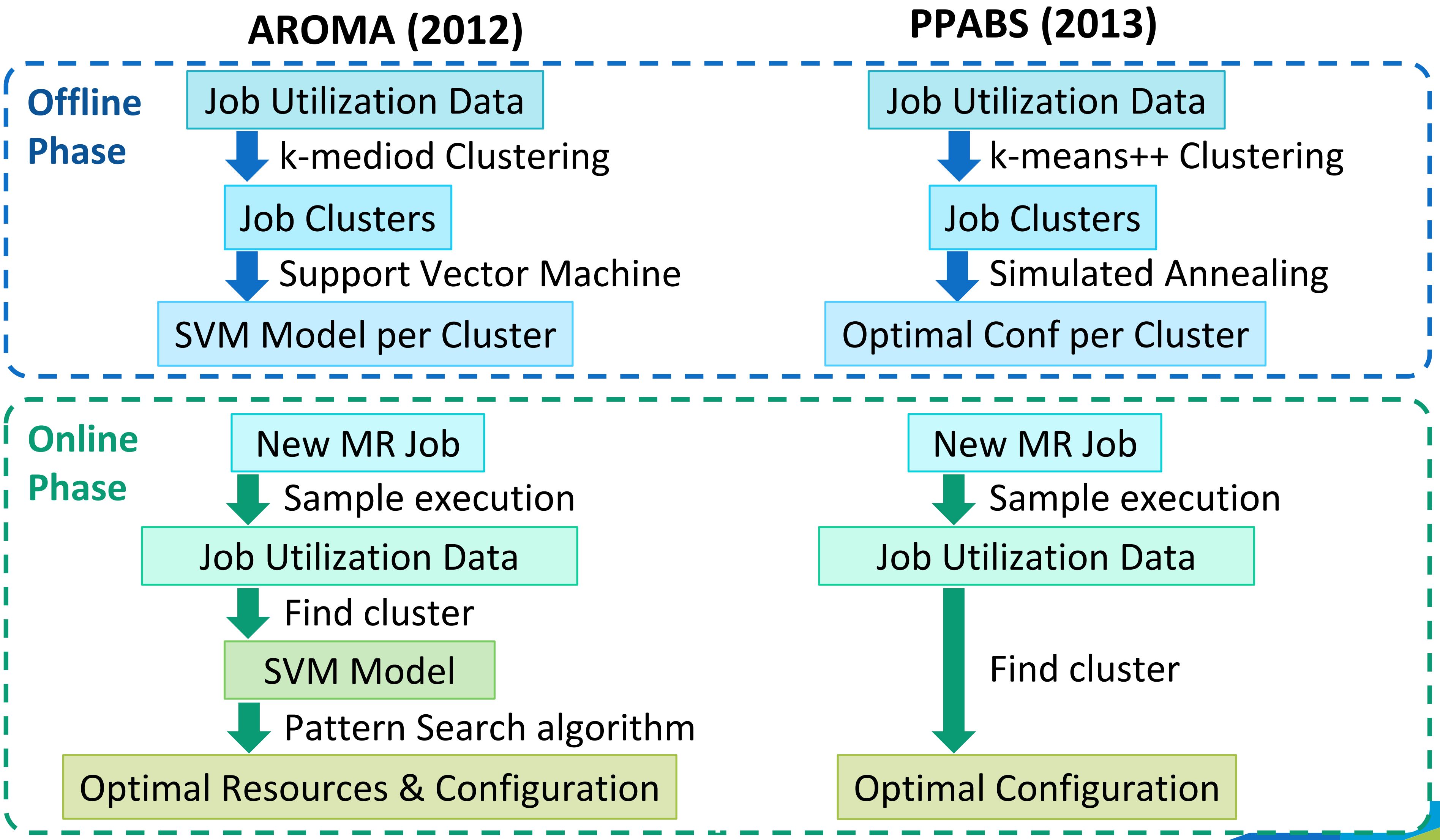
- Three categories of ML approaches:
  1) Build a historical store and use similarity measures
  2) Perform clustering and ML modeling per cluster
  3) Train and utilize a ML model per application

ML Approaches with Historical Stores

Kavulya (2010) **PStorm (2014)**



ML Approaches with Clustering



ML Approaches with App Modeling

Offline Phase

MR Job / Spark App

↓ Sample executions

Training Data

↓ Training model *

Machine Learning Model

Online Phase

MR Job / Spark App

↓ Employ ML model

Execution Time Prediction

↓ Search algorithm +

Optimal Configuration

hadoop

Chen (2015)	RFHOC (2016)
* Tree-based Regression + Random Hill Climbing	* Random-Forest Approach + Genetic Algorithm

Spark

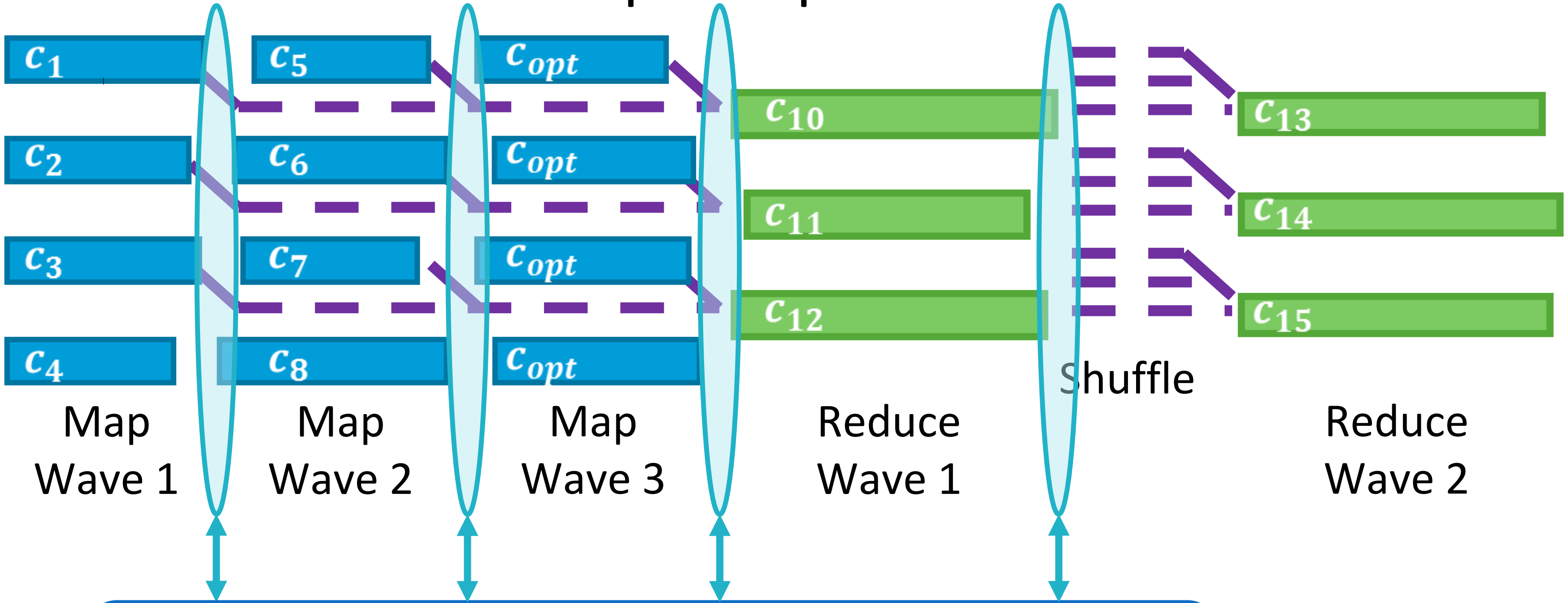
Guolu (2016)	Hernandez (2017)
* Decision Tree (C5.0) + Recursive Random Search	* Boosted Regression Trees + Heuristic algorithm

Machine Learning Approach: Pros & Cons

Pros ✔	Ability to capture complex system dynamics
	Independence from system internals and hardware
	Learning based on real observations of system performance
Cons ✖	Requires large training sets, which are expensive to collect
	Training from history logs leads to data under-fitting
	Typically low accuracy for unseen analytics applications

Adaptive Approach

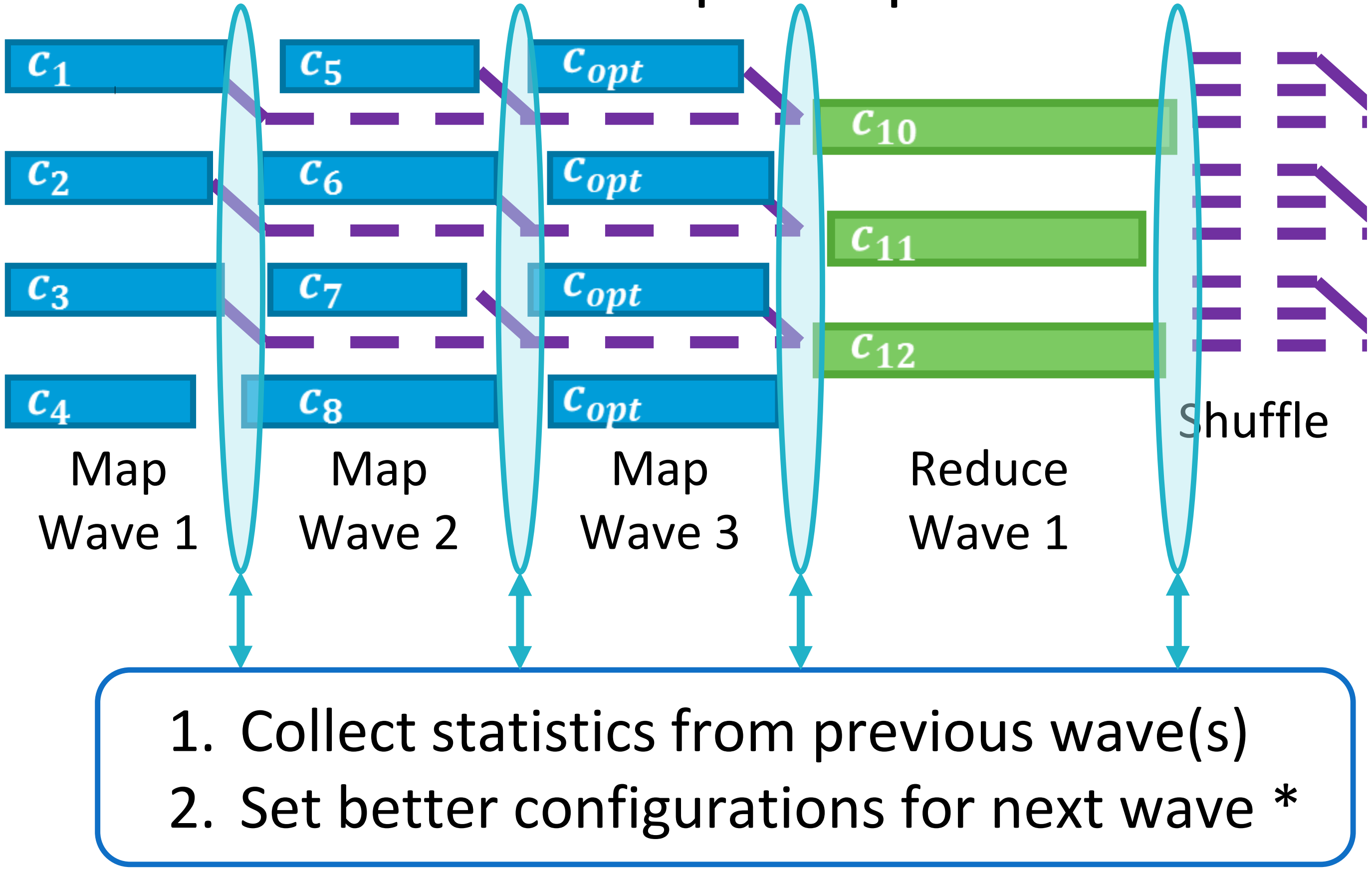
- **Key idea:** Track execution of a job and change its configuration in an online fashion in order to improve performance



1. Collect statistics from previous wave(s)
2. Set better configurations for next wave *

Adaptive Approach

- **Key idea:** Track execution of a job and change its configuration in an online fashion in order to improve performance

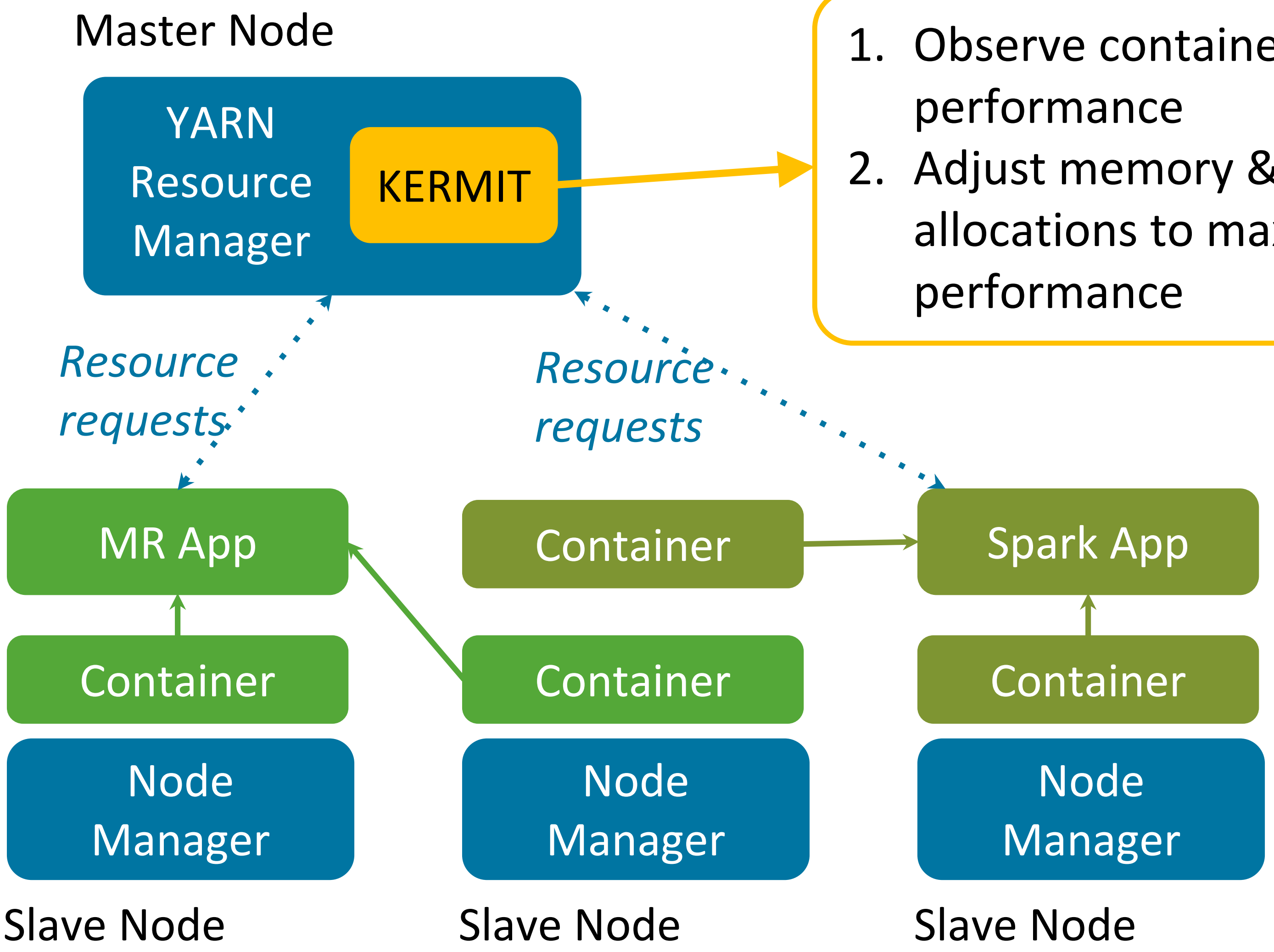


MROnline (2014)
* Gray-box hill climbing

Ant (2014)
* Genetic Algorithm

JellyFish (2015)
* Model-based hill climbing

The KERMIT (2016) Approach



1. Observe container performance
2. Adjust memory & CPU allocations to maximize performance

Adaptive Approach: Pros & Cons

Pros	Finds good settings based on actual task runs
	Able to adjust to dynamic runtime status
	Works well for ad-hoc analytics applications
Cons	Only applies to long-running analytics applications
	Inappropriate configuration can cause issues (e.g., stragglers)
	Neglects efficient resource utilization in the whole system

Outline

Motivation and Background

History and Classification

Parameter Tuning on Databases

Parameter Tuning on Big Data Systems

Applications of Automatic Parameter Tuning

Open Challenges and Discussion

Auto Parameter Tuning in Database Systems

- **Oracle Self-driving Database**
  - Automatically set various memory parameters and use of compression using machine learning
- **IBM DB2 Self-tuning Memory Manager**
  - Dynamically distributes available memory resources among buffer pools, locking memory, package cache, and sort memory
- **Azure SQL Database Automatic Tuning**
  - Memory buffer settings, index management, plan choice correction

Auto Parameter Tuning in Big Data Systems

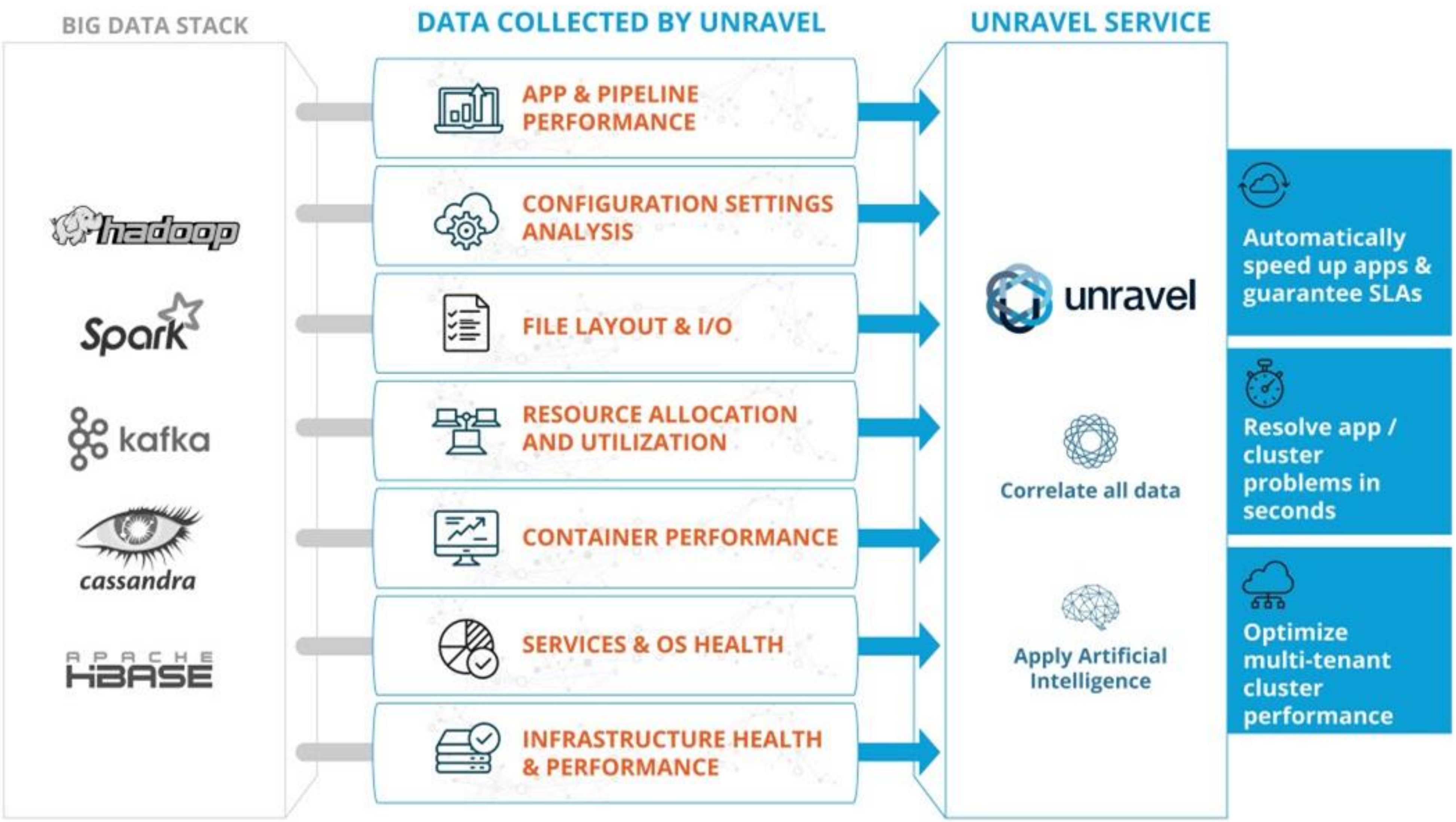
- **Databricks Optimized Autoscaling**
  - Automatically scale number of executors in Spark up and down
- **Spotfire Data Science Autotuning**
  - Automatically set Spark parameters for number and memory size of executors
- **Sparklens: Qubole's Spark Tuning Tool**
  - Automatically set memory of Spark executors

databricks

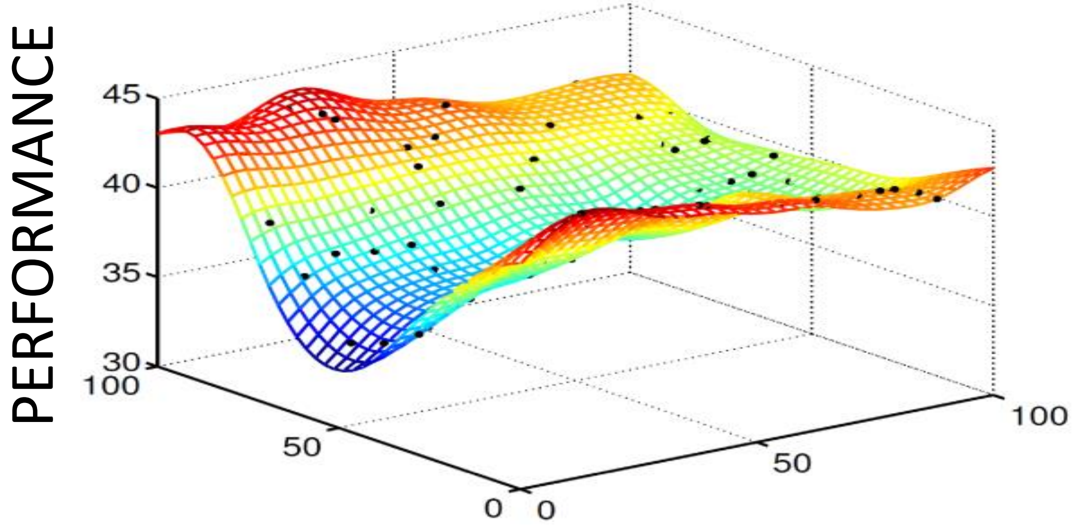
TIBCO Spotfire Data Science

Qubole ELEMENTAL TO BIG DATA

Auto Parameter Tuning with Unravel



spark.driver.cores	2
spark.executor.cores **…**	10
spark.sql.shuffle.partitions	300
spark.sql.autoBroadcastJoinThreshold	20MB
…	
SKEW('orders', 'o_custId')	true
spark.catalog.cacheTable(“orders")	true
…	



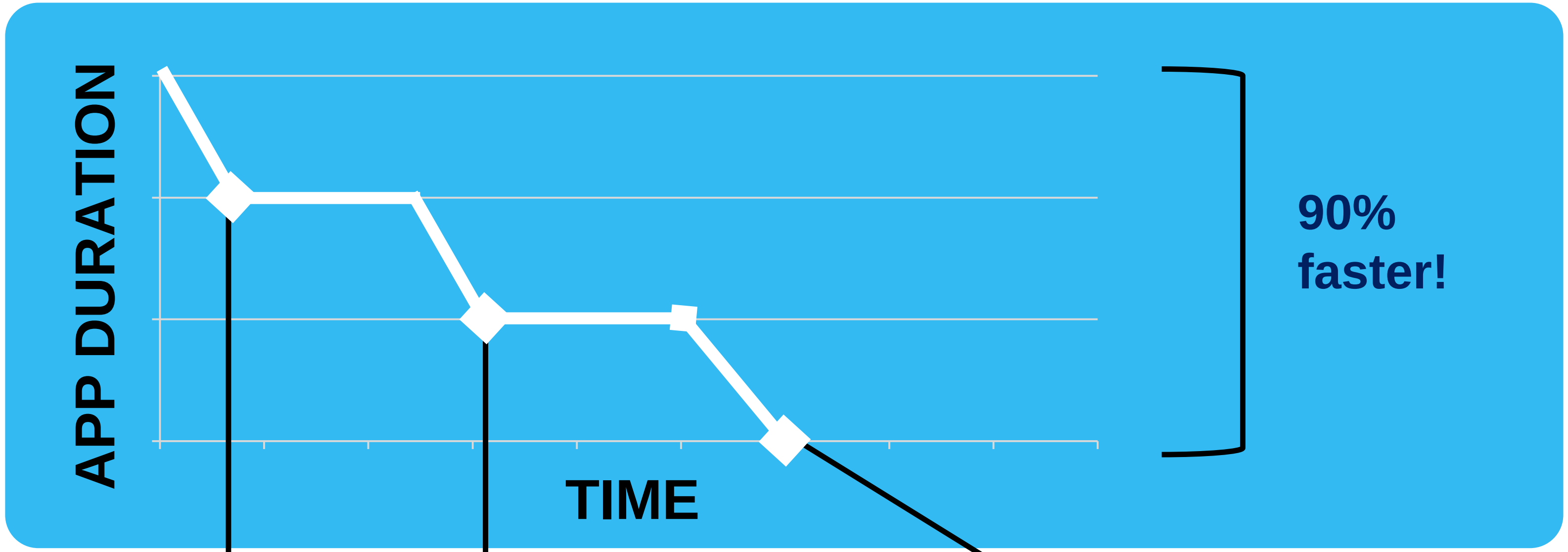
Today, tuning is often by trial-and-error

A New World

INPUTS
1. **App = Spark Query**
2. **Goal = Speedup**

"I need to make this app faster"

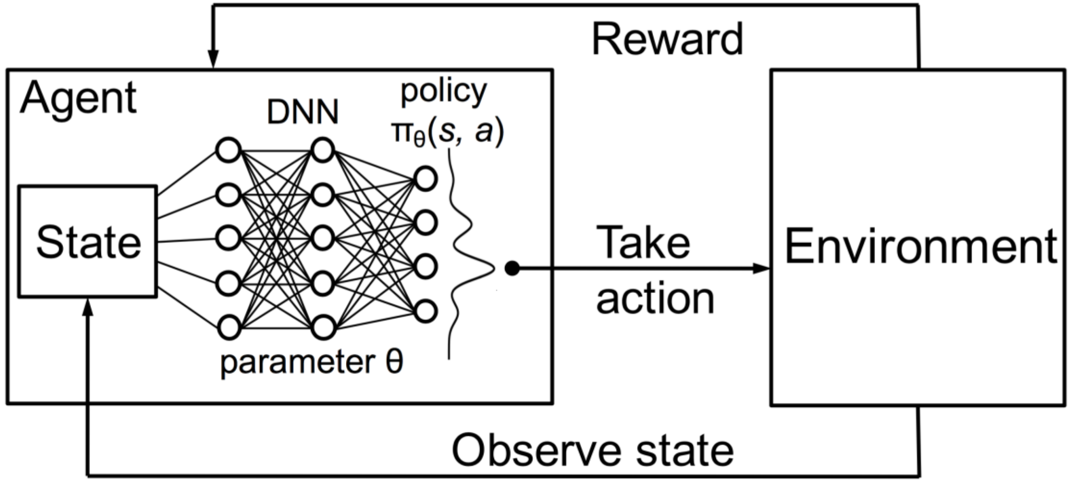
A New World



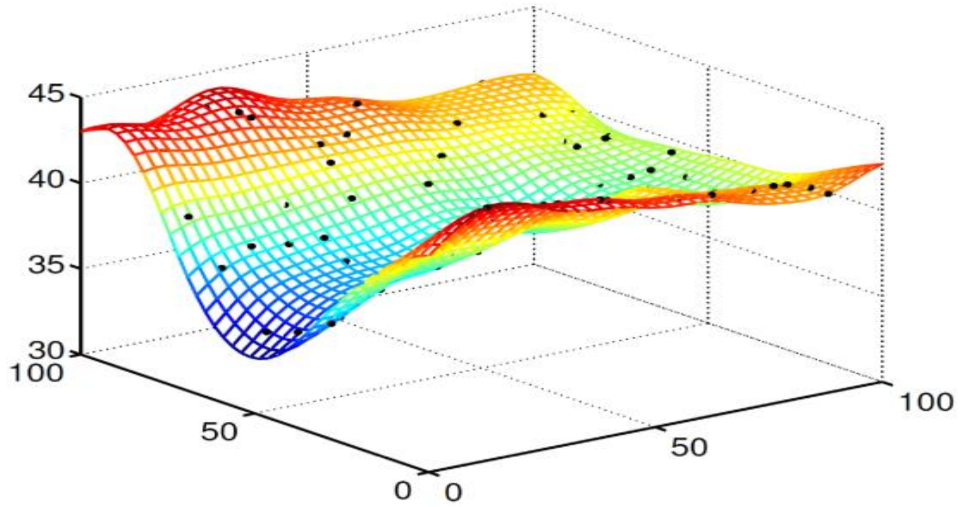
In blink of an eye, user gets recommendations to make the app **30% faster**

As user finishes checking email, she has a verified run that is **60% faster**

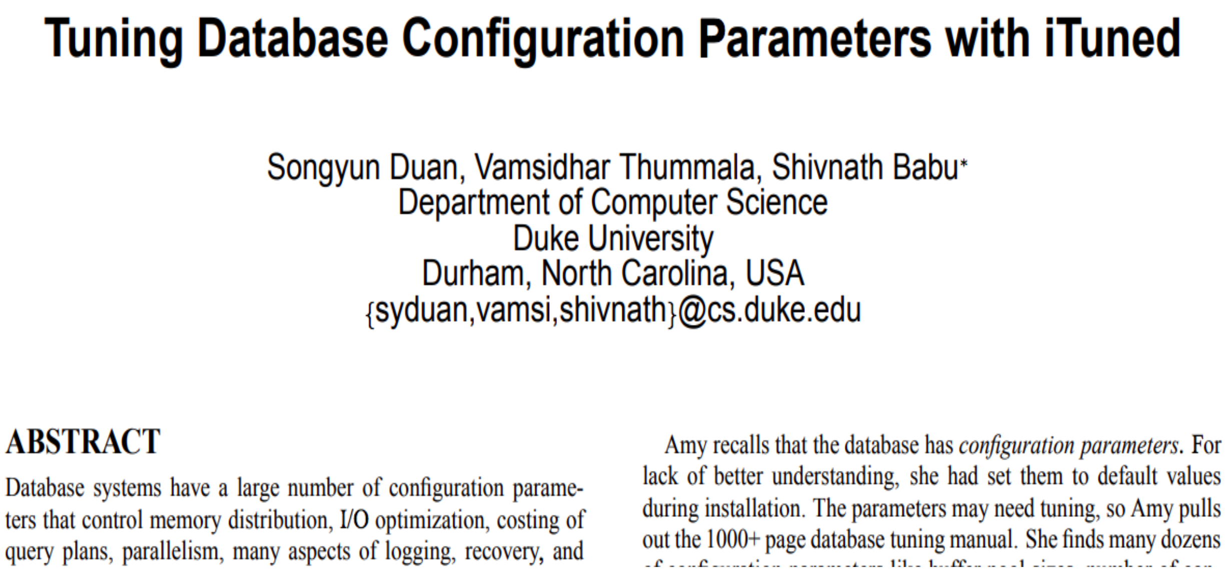
User comes back from lunch. A verified run that is **90% faster**



Reinforcement Learning



Response Surface Methodology



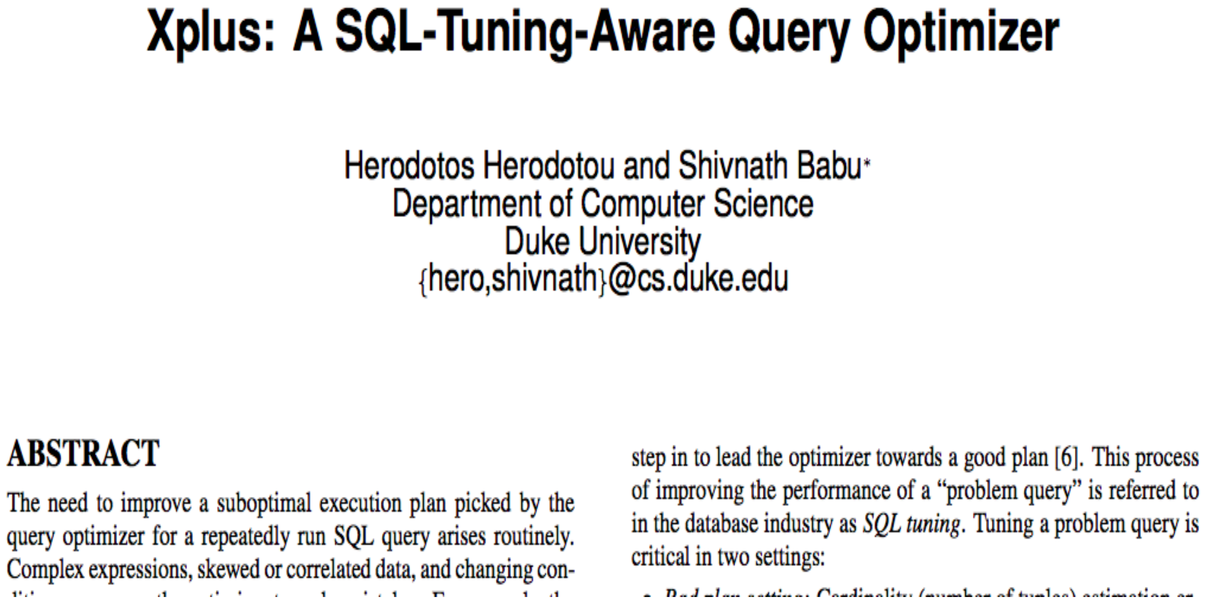
Tuning Database Configuration Parameters with iTuned

Songyun Duan, Vamsidhar Thummala, Shivnath Babu*
Department of Computer Science
Duke University
Durham, North Carolina, USA
{syduan,vamsi,shivnath}@cs.duke.edu

ABSTRACT

Database systems have a large number of configuration parameters that control memory distribution, I/O optimization, costing of query plans, parallelism, many aspects of logging, recovery, and

Amy recalls that the database has *configuration parameters*. For lack of better understanding, she had set them to default values during installation. The parameters may need tuning, so Amy pulls out the 1000+ page database tuning manual. She finds many dozens



Xplus: A SQL-Tuning-Aware Query Optimizer

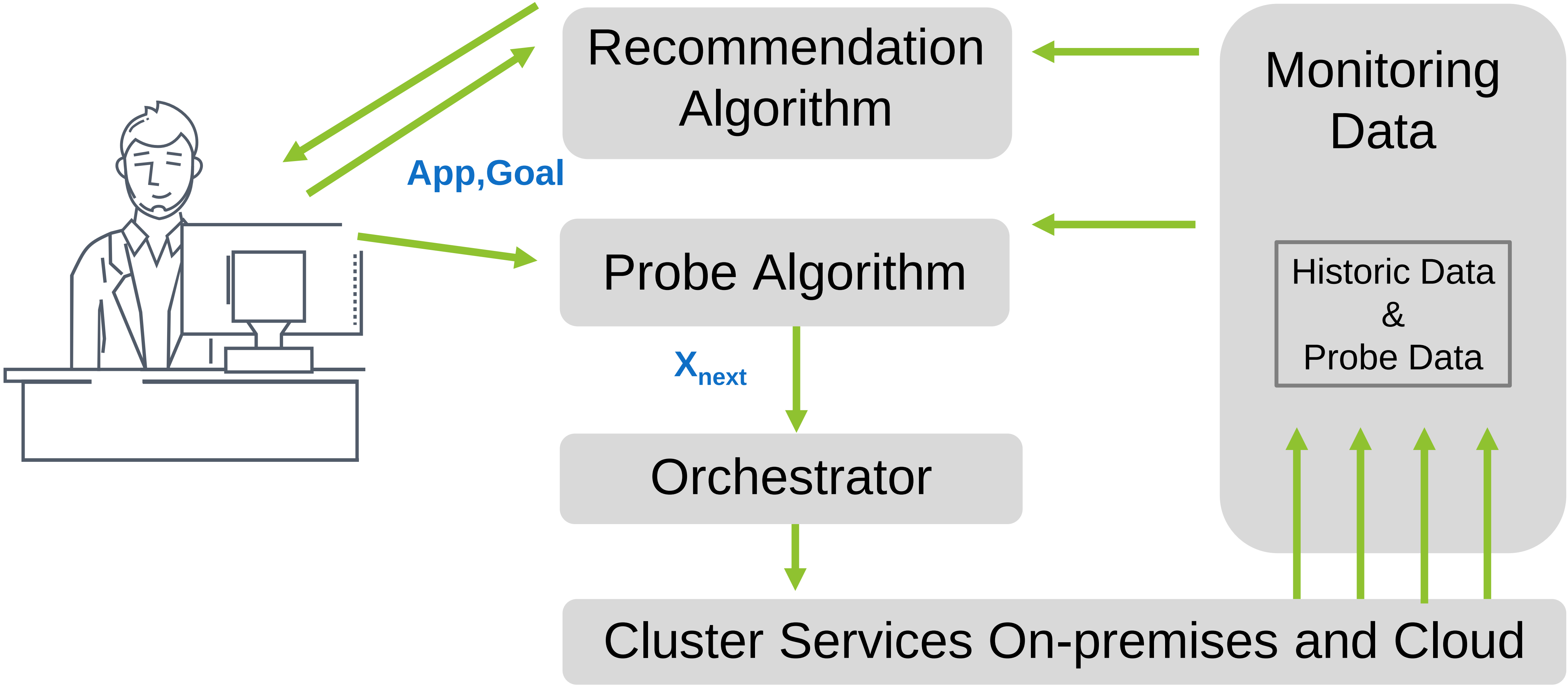
Herodotos Herodotou and Shivnath Babu*
Department of Computer Science
Duke University
{hero,shivnath}@cs.duke.edu

ABSTRACT

The need to improve a suboptimal execution plan picked by the query optimizer for a repeatedly run SQL query arises routinely. Complex expressions, skewed or correlated data, and changing con-

step in to lead the optimizer towards a good plan [6]. This process of improving the performance of a "problem query" is referred to in the database industry as *SQL tuning*. Tuning a problem query is critical in two settings:

Autotuning Workflow



Outline

Motivation and Background

History and Classification

Parameter Tuning on Databases

Parameter Tuning on Big Data Systems

Applications of Automatic Parameter Tuning

Open Challenges and Discussion

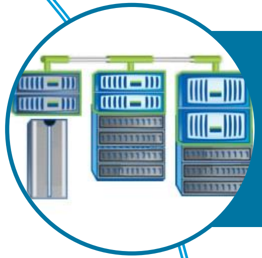
Putting it all Together

Approach	Pros	Cons
Cost Modeling	Very efficient for predicting performance Good accuracy in many (not complex) scenarios Very efficient for predicting performance Good accuracy in many (not complex) scenarios	Hard to capture complexity of system internals & pluggable components (e.g., schedulers) Models often based on simplified assumptions Not effective on heterogeneous clusters
Simulation-based	High accuracy in simulating dynamic system behaviors Efficient for predicting fine-grai[illegible]	
	[illegible]ware [illegible] observations of system performance	[illegible] large training sets, which are expensive to collect Training from history logs leads to data under-fitting Typically low accuracy for unseen analytics applications
Adaptive	Finds good settings based on actual task runs Able to adjust to dynamic runtime status Works well for ad-hoc analytics applications	Only applies to long-running analytics applications Inappropriate configuration can cause issues (e.g., stragglers) Neglects efficient resource utilization in the whole system

No single approach is able to provide **good prediction accuracy** with **low overhead** in most scenarios

Open Challenges

Ensuring good and robust system performance at scale poses new challenges



Clusters are becoming heterogeneous in nature, both for compute and storage

The proliferation of Cloud leads to multi-tenancy, overheads, perf interaction issues

Real-time analytics pushes boundaries on latency requirements and combination of systems

Thank you!

References (1/6)

- Shivnath Babu. Towards Automatic Optimization of MapReduce Programs. In Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC), pages 137–142. ACM, 2010.
- Shivnath Babu, Herodotos Herodotou, et al. Massively Parallel Databases and MapReduce Systems. Foundations and Trends® in Databases, 5(1):1–104, 2013.
- Liang Bao, Xin Liu, and Weizhao Chen. Learning-based Automatic Parameter Tuning for Big Data Analytics Frameworks. arXiv preprint arXiv:1808.06008, 2018.
- Zhendong Bei, Zhibin Yu, Huiling Zhang, Wen Xiong, Chengzhong Xu, Lieven Eeckhout, and Shengzhong Feng. RFHOC: A Random-Forest Approach to Auto-Tuning Hadoop's Configuration. IEEE Transactions on Parallel and Distributed Systems (TPDS), 27(5):1470–1483, 2016.
- Chi-Ou Chen, Ye-Qi Zhuo, Chao-Chun Yeh, Che-Min Lin, and Shih-Wei Liao. Machine Learning-Based Configuration Parameter Tuning on Hadoop System. In Proceedings of the 2015 IEEE International Congress on Big Data (BigData Congress), pages 386–392. IEEE, 2015.
- Keke Chen, James Powers, Shumin Guo, and Fengguang Tian. CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds. IEEE Transactions on Parallel and Distributed Systems (TPDS), 25(6):1403–1412, 2014.
- Dazhao Cheng, Jia Rao, Yanfei Guo, and Xiaobo Zhou. Improving MapReduce Performance in Heterogeneous Environments with Adaptive Task Tuning. In Proceedings of the 15th International Middleware Conference, pages 97–108. ACM, 2014.
- Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 51(1):107–113, 2008.

References (2/6)

- Xiaoan Ding, Yi Liu, and Depei Qian. Jellyfish: Online Performance Tuning with Adaptive Configuration and Elastic Container in Hadoop YARN. In Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), pages 831–836. IEEE, 2015.
- Mostafa Ead, Herodotos Herodotou, Ashraf Aboulnaga, and Shivnath Babu. PStorM: Profile Storage and Matching for Feedback-Based Tuning of MapReduce Jobs. In Proceedings of the 17th International Conference on Extending Database Technology (EDBT), pages 1–12. OpenProceedings, March 2014.
- Mikhail Genkin, Frank Dehne, Maria Pospelova, Yabing Chen, and Pablo Navarro. Automatic, On-Line Tuning of YARN Container Memory and CPU Parameters. In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications (HPCC), pages 317–324. IEEE, 2016.
- Anastasios Gounaris, Georgia Kougka, Ruben Tous, Carlos Tripiana Montes, and Jordi Torres. Dynamic Configuration of Partitioning in Spark Applications. IEEE Transactions on Parallel and Distributed Systems (TPDS), 28(7):1891–1904, 2017.
- Anastasios Gounaris and Jordi Torres. A Methodology for Spark Parameter Tuning. Big Data Research, 11:22–32, March 2017.
- Suhel Hammoud, Maozhen Li, Yang Liu, Nasullah Khalid Alham, and Zelong Liu. MRSim: A Discrete Event Based MapReduce Simulator. In Proceedings of the Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), volume 6, pages 2993–2997. IEEE, 2010.
- Álvaro Brandón Hernández, María S Perez, Smrati Gupta, and Victor Muntés-Mulero. Using Machine Learning to Optimize Parallelism in Big Data Applications. Future Generation Computer Systems, pages 1–12, 2017.

References (3/6)

- Herodotos Herodotou and Shivnath Babu. Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs. Proceedings of the VLDB Endowment, 4(11):1111–1122, 2011.
- Herodotos Herodotou and Shivnath Babu. A What-if Engine for Cost-based MapReduce Optimization. IEEE Data Engineering Bulletin, 36(1):5–14, 2013.
- Herodotos Herodotou, Fei Dong, and Shivnath Babu. No One (Cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics. In Proceedings of the 2nd ACM Symposium on Cloud Computing (SoCC). ACM, 2011.
- Herodotos Herodotou, Harold Lim, Gang Luo, Nedyalko Borisov, Liang Dong, Fatma Bilgen Cetin, and Shivnath Babu. Starfish: A Self-tuning System for Big Data Analytics. In Proceedings of the Fifth Biennial Conference on Innovative Data Systems Research (CIDR), volume 11, pages 261–272, 2011.
- Soila Kavulya, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan. An Analysis of Traces from a Production MapReduce Cluster. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 94–103. IEEE Computer Society, 2010.
- Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang, and Changjun Jiang. Hadoop Performance Modeling for Job Estimation and Resource Provisioning. IEEE Transactions on Parallel and Distributed Systems (TPDS), 27(2):441–454, 2016.
- Palden Lama and Xiaobo Zhou. AROMA: Automated Resource Allocation and Configuration of MapReduce Environment in the Cloud. In Proceedings of the 9th international conference on Autonomic computing (ICAC), pages 63–72. ACM, 2012.

References (4/6)

- Min Li, Liangzhao Zeng, Shicong Meng, Jian Tan, Li Zhang, Ali R Butt, and Nicholas Fuller. MROnline: MapReduce Online Performance Tuning. In Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing (HPDC), pages 165–176. ACM, 2014.
- Guangdeng Liao, Kushal Datta, and Theodore L Willke. Gunther: Search-based Auto-tuning of MapReduce. In Proceedings of the European Conference on Parallel Processing (Euro-Par), pages 406–419. Springer, 2013.
- Chao Liu, Deze Zeng, Hong Yao, Chengyu Hu, Xuesong Yan, and Yuanyuan Fan. MR-COF: A Genetic MapReduce Configuration Optimization Framework. In Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), pages 344–357. Springer, 2015.
- Jun Liu, Nishkam Ravi, Srimat Chakradhar, and Mahmut Kandemir. Panacea: Towards Holistic Optimization of MapReduce Applications. In Proceedings of the Tenth International Symposium on Code Generation and Optimization (CGO), pages 33–43. ACM, 2012.
- Yang Liu, Maozhen Li, Nasullah Khalid Alham, and Suhel Hammoud. HSim: A MapReduce Simulator in Enabling Cloud Computing. Future Generation Computer Systems, 29(1):300–308, 2013.
- Mumak: Map-Reduce Simulator, 2010. https://issues.apache.org/jira/browse/MAPREDUCE-728.
- Panagiotis Petridis, Anastasios Gounaris, and Jordi Torres. Spark Parameter Tuning via Trial-and-Error. In Proceedings of the INNS Conference on Big Data, pages 226–237. Springer, 2016.
- Juwei Shi, Jia Zou, Jiaheng Lu, Zhao Cao, Shiqiang Li, and Chen Wang. MRTuner: a Toolkit to Enable Holistic Optimization for MapReduce Jobs. Proceedings of the VLDB Endowment, 7(13):1319–1330, 2014.

References (5/6)

- Rekha Singhal and Praveen Singh. Performance Assurance Model for Applications on SPARK Platform. In Proceedings of the Technology Conference on Performance Evaluation and Benchmarking (TPCTC), pages 131–146. Springer, 2017.
- Fei Teng, Lei Yu, and Frederic Magoulès. SimMapReduce: A Simulator for Modeling MapReduce Framework. In Proceedings of the 5th FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE), pages 277–282. IEEE, 2011.
- Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache Hadoop YARN: Yet Another Resource Negotiator. In Proceedings of the 4th annual Symposium on Cloud Computing (SoCC), page 5. ACM, 2013.
- Shivaram Venkataraman, Zongheng Yang, Michael J Franklin, Benjamin Recht, and Ion Stoica. Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics. In Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI), pages 363–378. USENIX Association, 2016.
- Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell. ARIA: Automatic Resource Inference and Allocation for Mapreduce Environments. In Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC), ICAC ’11, pages 235–244, New York, NY, USA, 2011. ACM.
- Abhishek Verma, Ludmila Cherkasova, and Roy H Campbell. Play it Again, SimMR! In Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER), pages 253–261. IEEE, 2011.

References (6/6)

- Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell. Resource Provisioning Framework for MapReduce Jobs with Performance Goals. In Proceedings of the ACM/IFIP/USENIX 12th International Middleware Conference, pages 165–186. Springer, 2011.
- Guanying Wang, Ali R Butt, Prashant Pandey, and Karan Gupta. A Simulation Approach to Evaluating Design Decisions in MapReduce Setups. In Proceedings of the 2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pages 1–11. IEEE, 2009.
- Guolu Wang, Jungang Xu, and Ben He. A Novel Method for Tuning Configuration Parameters of Spark based on Machine Learning. In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications (HPCC), pages 586–593. IEEE, 2016.
- Kewen Wang, Xuelian Lin, and Wenzhong Tang. Predator - An Experience Guided Configuration Optimizer for Hadoop MapReduce. In Proceedings of the IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), pages 419–426. IEEE, 2012.
- Dili Wu and Aniruddha Gokhale. A Self-tuning System based on Application Profiling and Performance Analysis for Optimizing Hadoop MapReduce Cluster Configuration. In Proceedings of the 20th International Conference on High Performance Computing (HiPC), pages 89–98. IEEE, 2013.