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

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

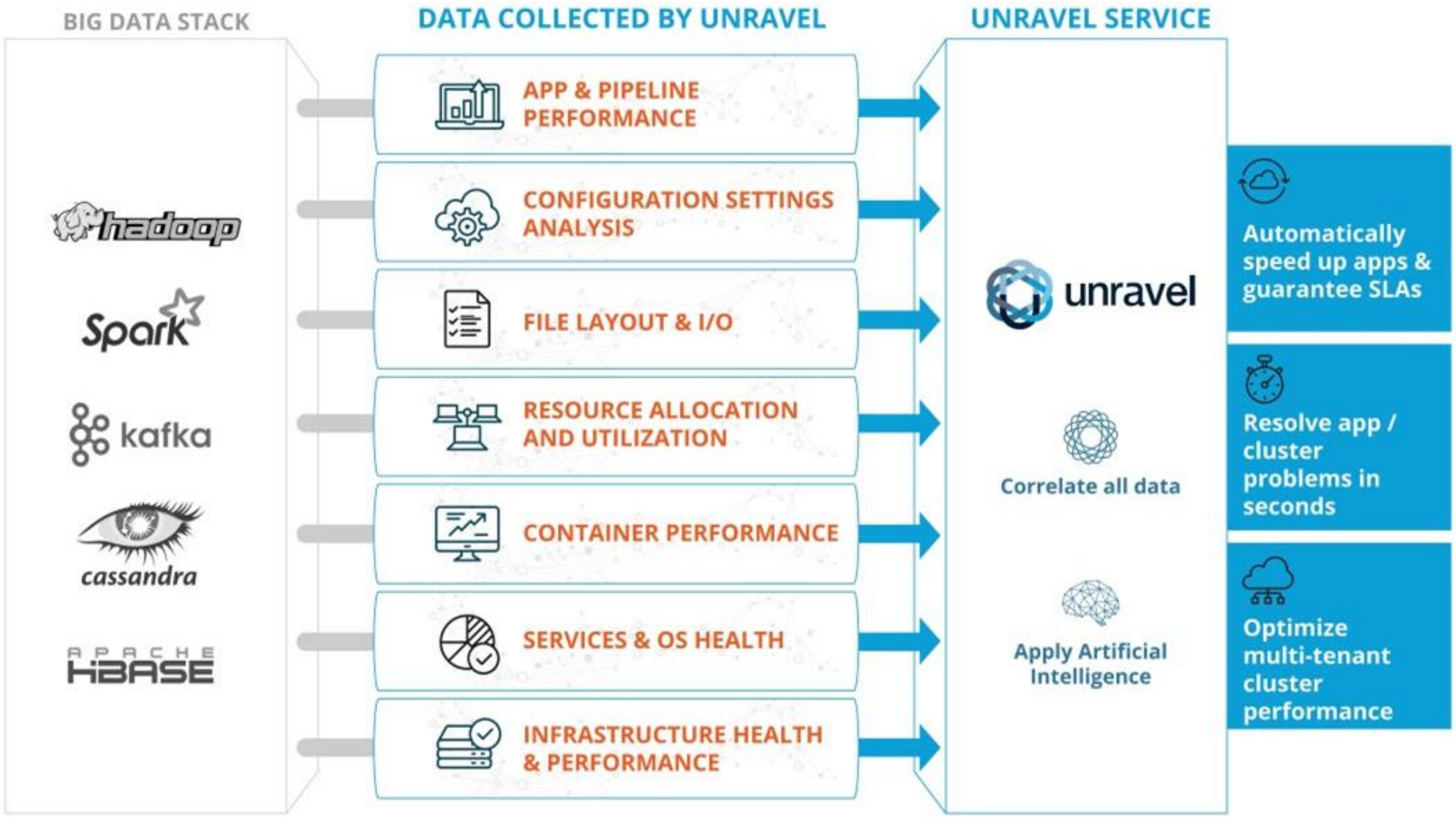
Auto Parameter Tuning in Big Data Systems

- **Databricks Optimized Autoscaling**
  - Automatically scale number of executors in Spark up and down

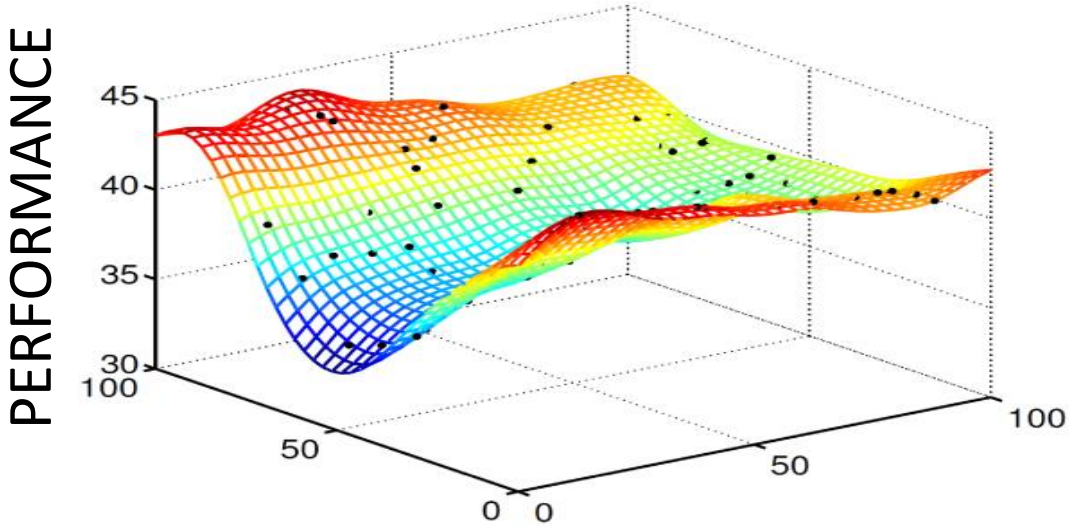
- **Spotfire Data Science Autotuning**
  - Automatically set Spark parameters for number and memory size of executors

- **Sparklens: Qubole's Spark Tuning Tool**
  - Automatically set memory of Spark executors

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

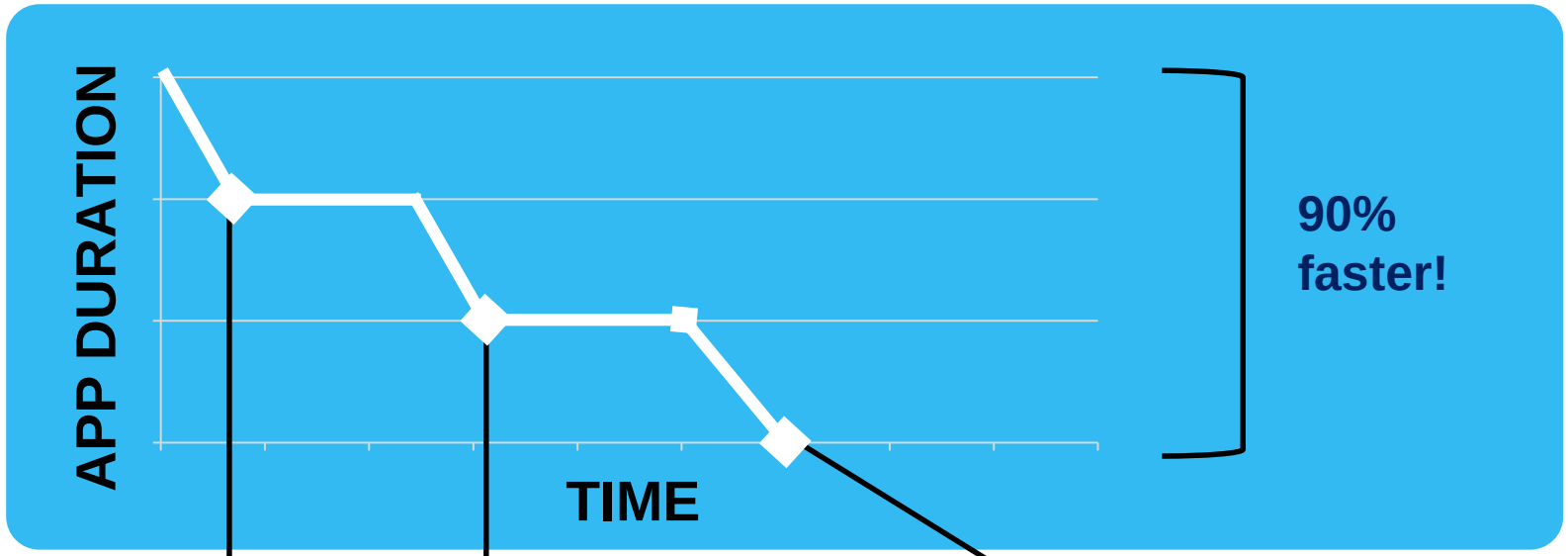
A New World

INPUTS

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

"I need to make this app faster"

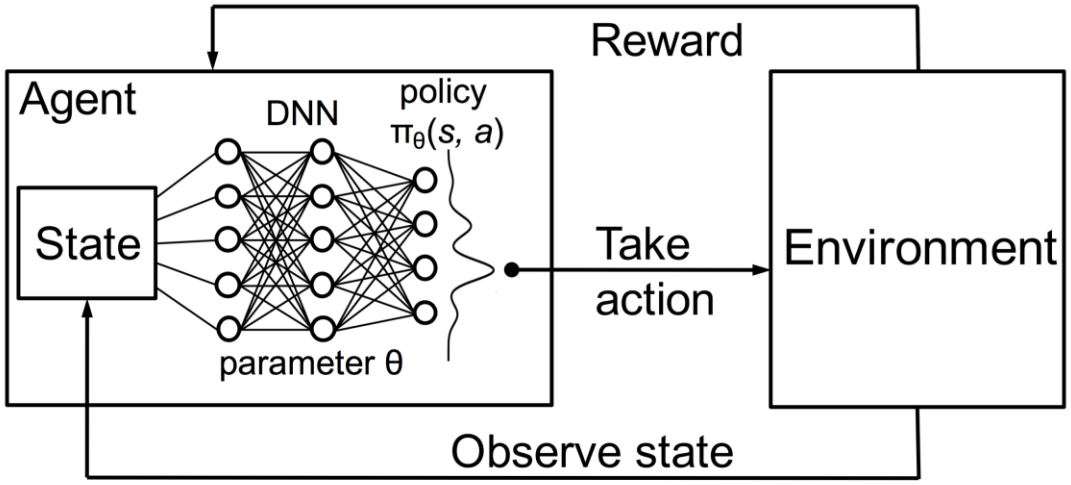
A New World



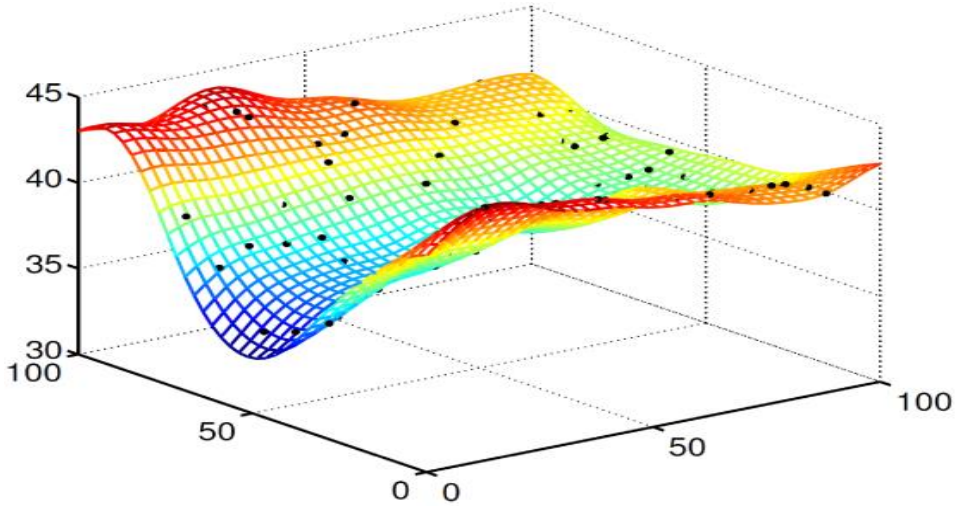
In blink of an eye, user gets recommendations to make the app **30% faster**

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

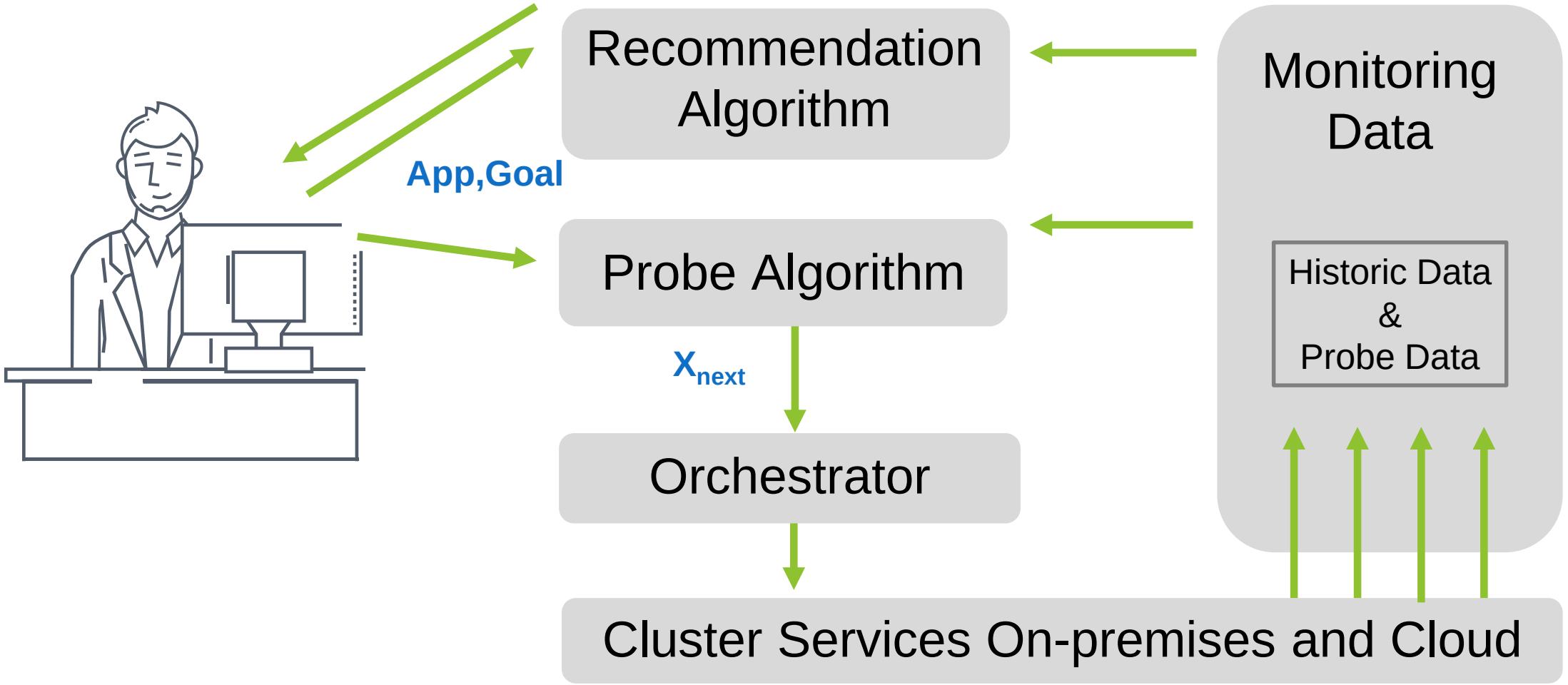
Herodotos Herodotou and Shivnath Babu*
Department of Computer Science
Duke University
{hero,shivnath}@cs.duke.edu

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

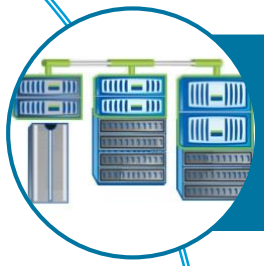
Putting it all Together

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-grain	
	observations of system performance	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!