Using AI to Build a Self-Driving Query Optimizer

Shivnath Babu, Adrian Popescu

#AI7SAIS

Meet the speakers

Shivnath Babu

- Cofounder and CTO at Unravel, Adjunct Professor at Duke University
- Focusing on ease-of-use and manageability of data-intensive systems
- Recipient of US National Science Foundation CAREER Award, three IBM Faculty Awards, HP Labs Innovation Research Award

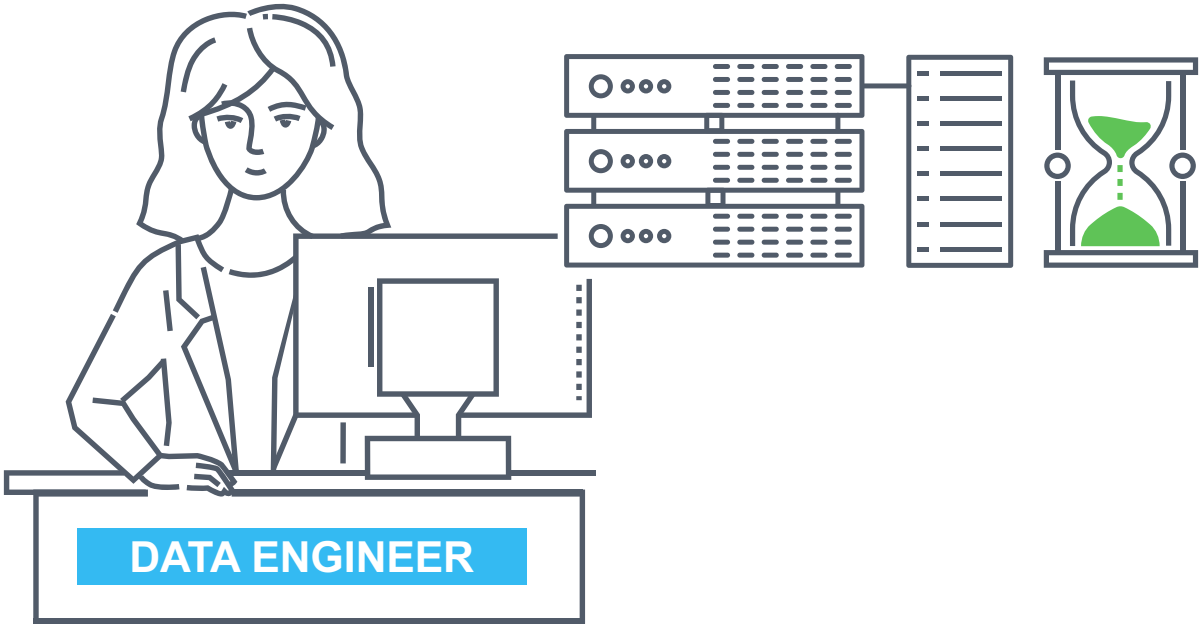
Adrian Popescu

- Data engineer at Unravel
- PhD from EPFL, Switzerland
- 8+ years of experience in performance monitoring & modeling of data management systems
- Focusing on tuning and optimization of Big Data apps

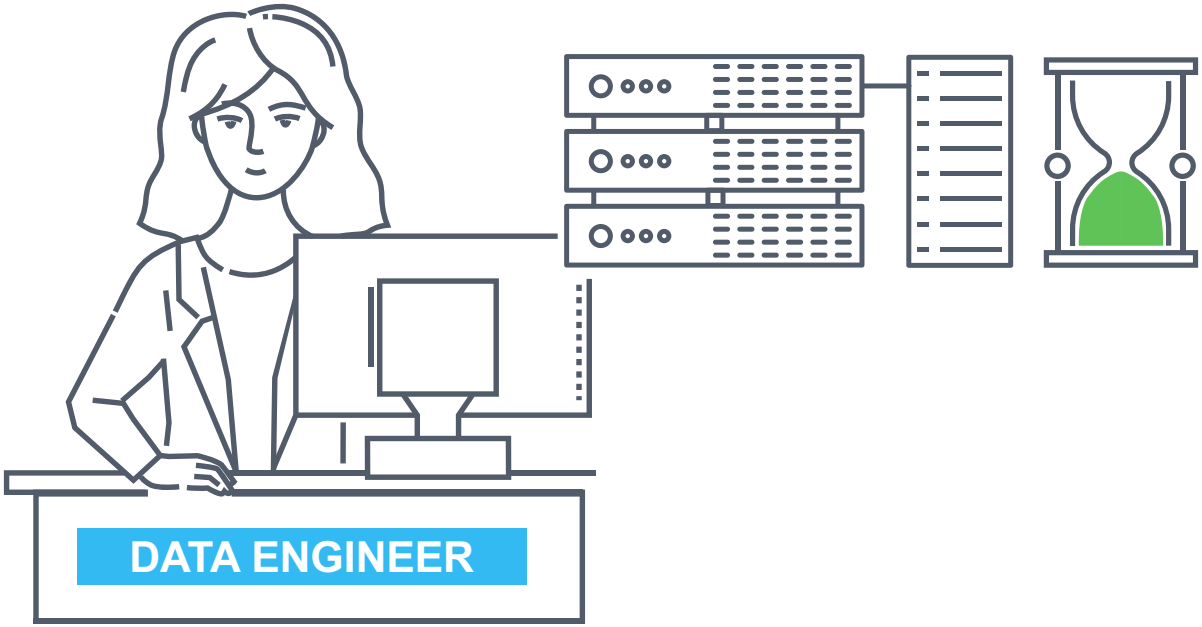
My app often fails with Out of Memory…

How can I make it more reliable?

My app is too slow…



I need to make it faster…



My app is missing SLA…

How can I tune my app to guarantee SLAs?

This rogue app is wasting resources and reducing cluster throughput

OPERATIONS TEAMS

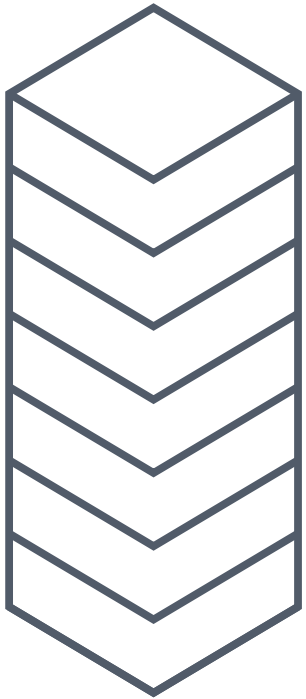
Can this app use less resources while finishing on time?

Current approach: Find the needle in the hay STACKS

1 System

1 User

1 App



1. Find the app. Review Spark/YARN UI to find the app
2. Review metrics on web UI
3. Review stages associated with the app
4. Identify executors associated with the stages and outliers
5. Deep dive into “outlier” stage
6. Identify problematic “executors”
7. Review and debug container logs
8. Rinse & repeat across other executor/container logs to identify the problem

Now imagine the problem at scale

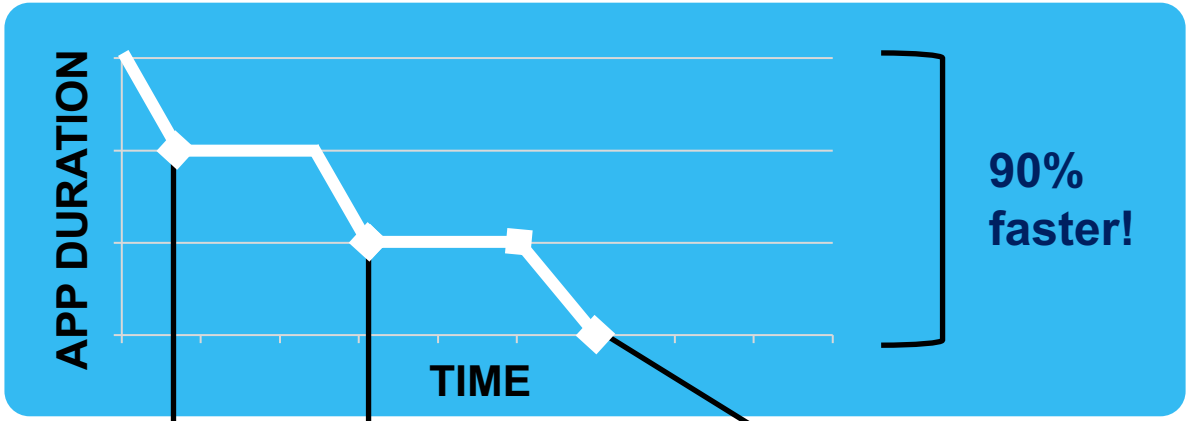
10x Systems
100x Users
1000x Apps

Imagine a new world

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

"I need to make this app faster"

Imagine 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**

At Unravel, we want to bring
the **new world** to you today

Introducing Unravel Sessions

Demo

Unravel Sessions

App

Goal

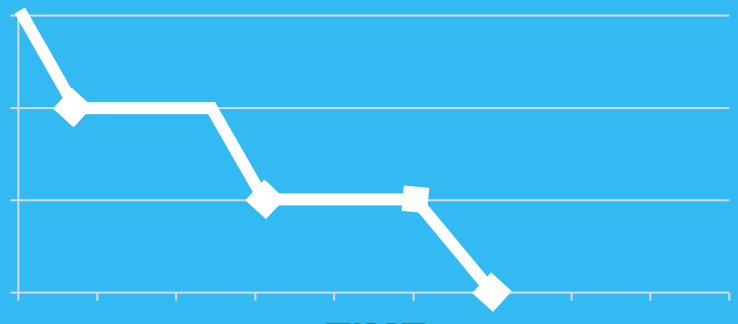
SQL Program DAG Workload

Efficiency

Reliability

Speedup

SLA



Unravel Sessions will quickly get you there!

Demo

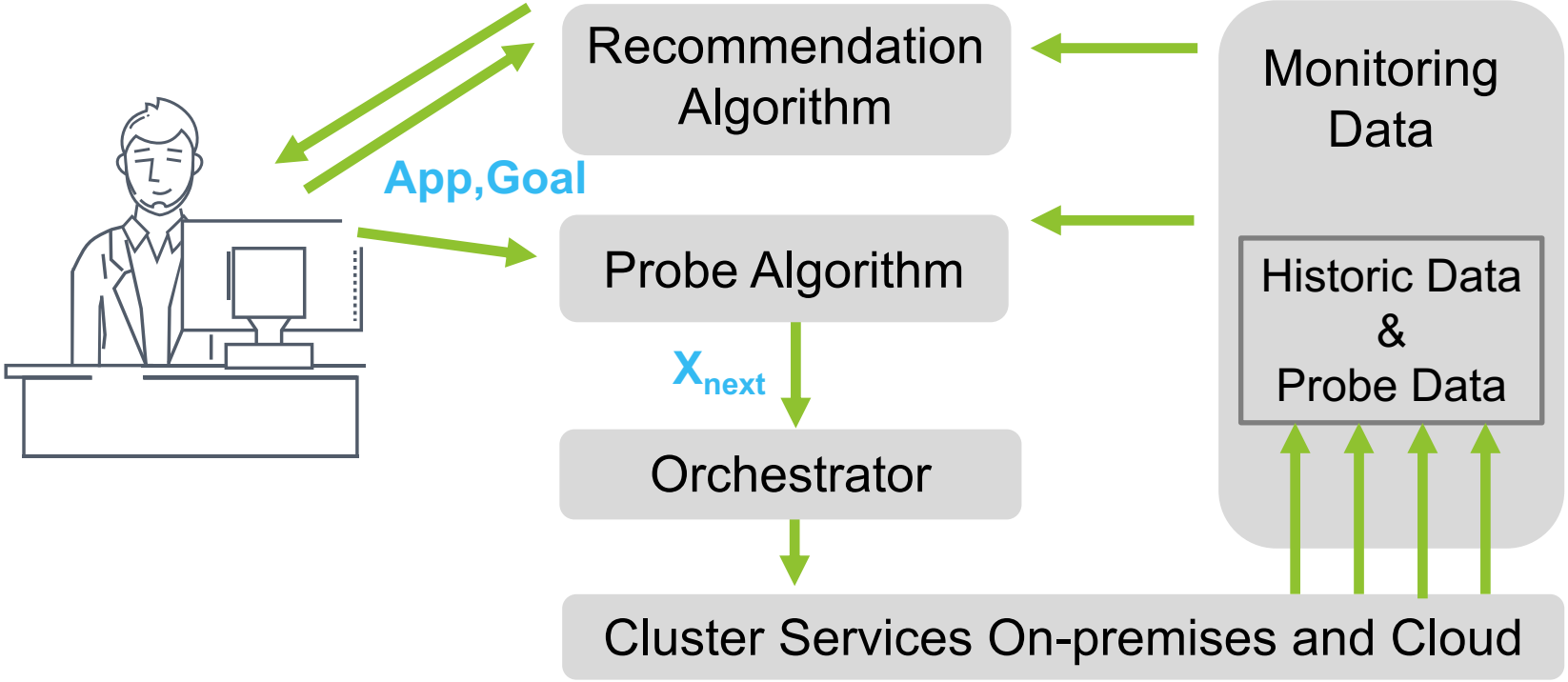
How Unravel Sessions work

AI

for

Application Performance Management

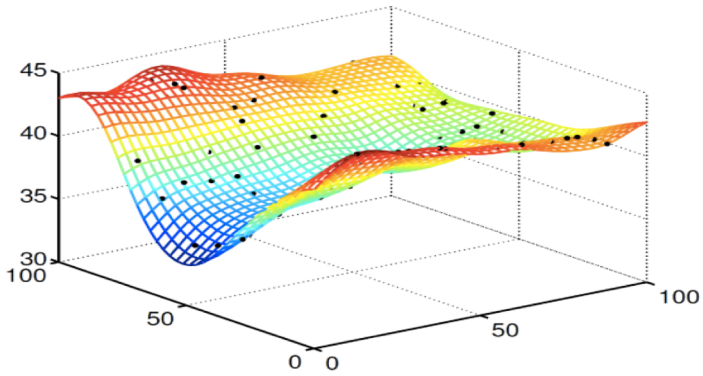
Unravel Sessions Architecture



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
…	

We represent the setting as vector X

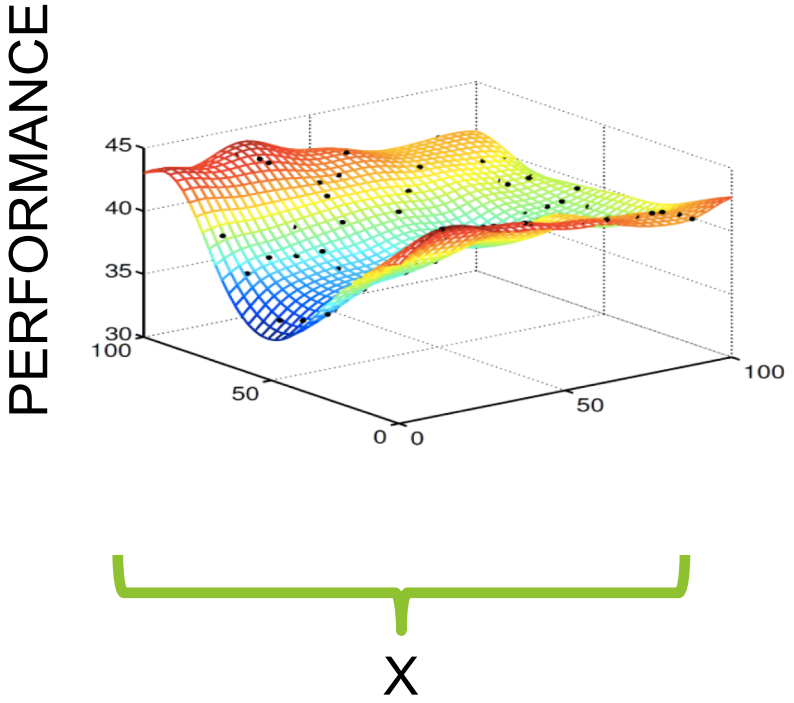
PERFORMANCE

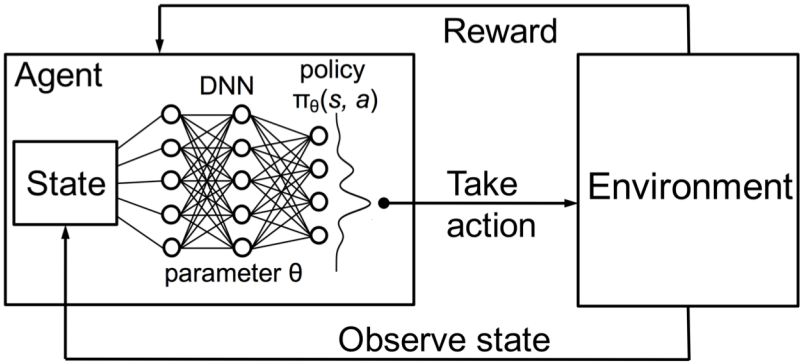


X

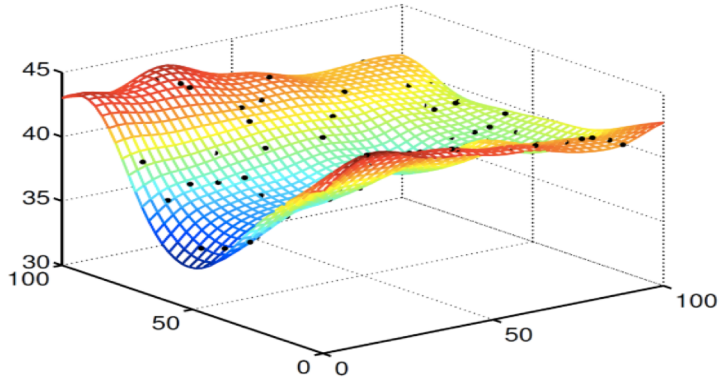
Given: App + Goal

- **Goal:** Find the setting of X that best meets the goal
- **Challenge:** Response surface y = $f(X)$ is unknown





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:

Credit: http://people.csail.mit.edu/hongzi/

Challenge: Response surface y = f(X) is unknown

Model the response surface as

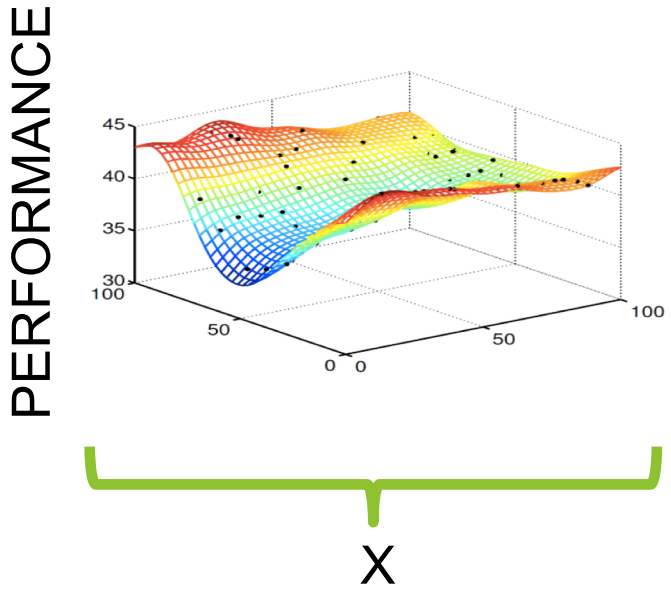
$$\hat{y}(X) = \vec{f}^{\,t}(X)\vec{\beta} + Z(X)$$

Here:

$\vec{f}^{\,t}(X)\vec{\beta}$ is a regression model

$Z(X)$ is the residual captured as a **Gaussian Process**

The Gaussian Process model captures the **uncertainty** in our current knowledge of the response surface



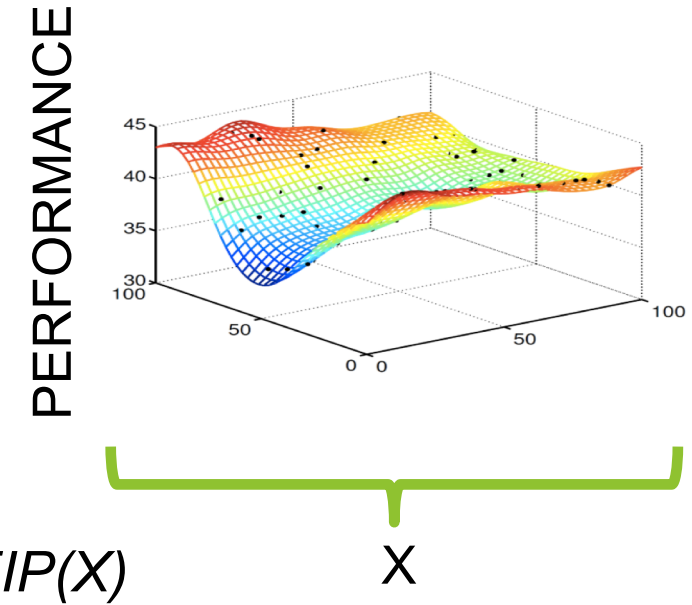
Opportunity

We can now estimate the **expected improvement *EIP(X)*** from doing **a probe** at any setting X

$$EIP(X) = \int_{p=-\infty}^{p=y(X^*)} (y(X^*) - p)\, pdf_{\hat{y}(X)}(p)\, dp$$

Improvement at any setting X over the best performance seen so far

Probability density function (uncertainty estimate)



Gaussian Process model helps estimate *EIP(X)*

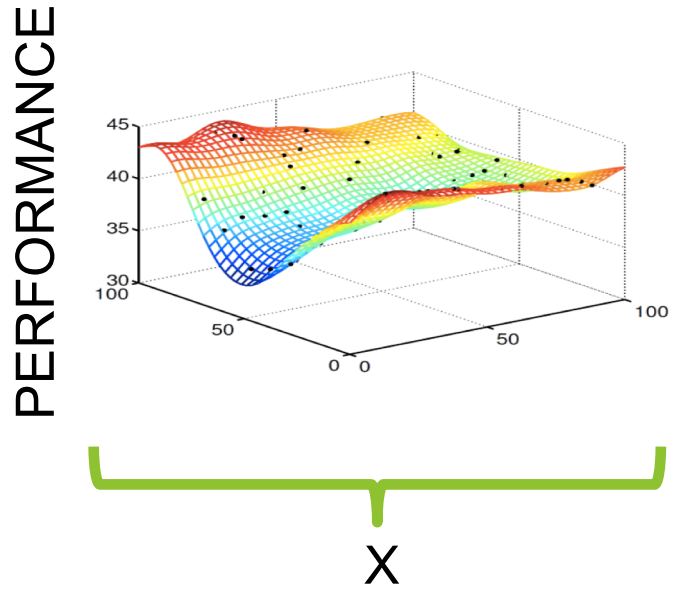
1 Bootstrap

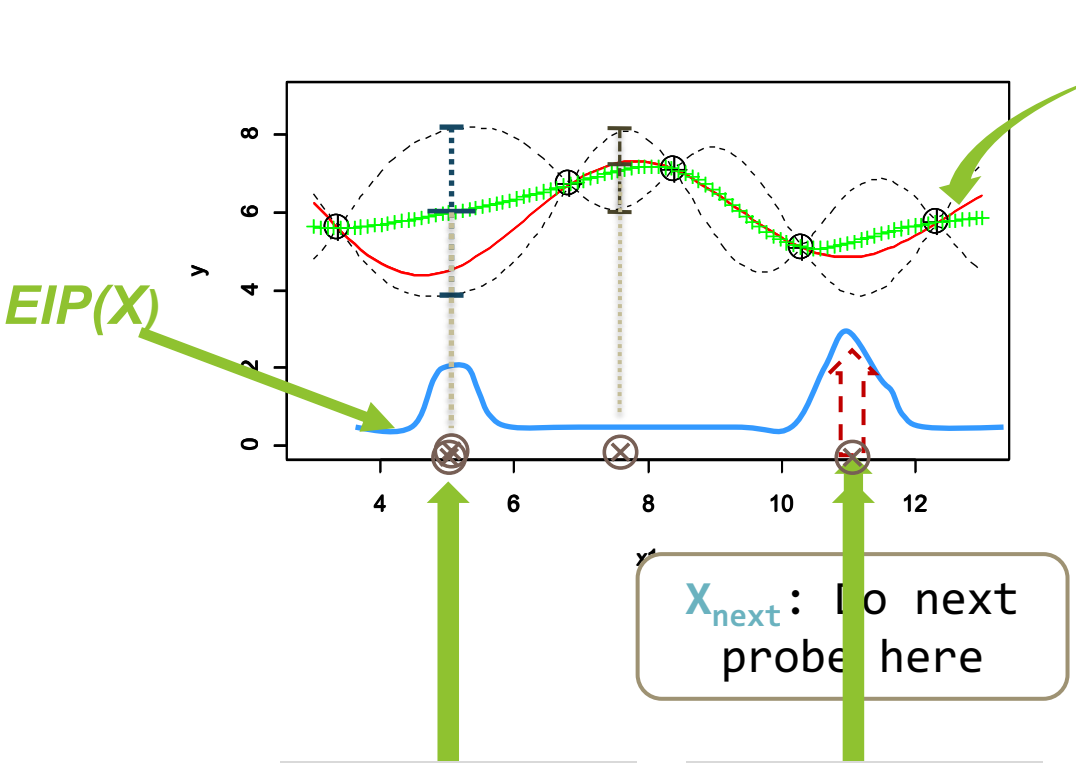
Get initial set of monitoring data from history or via probes: $<X_1,y_1>$, $<X_2,y_2>$, …, $<X_n,y_n>$

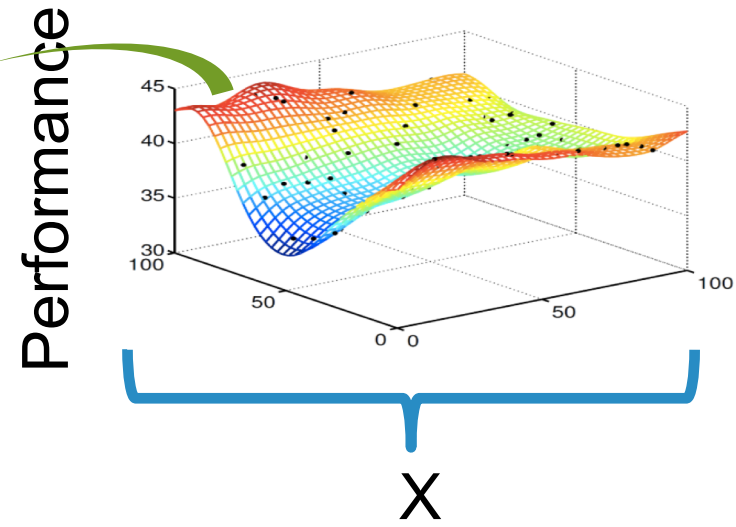
2 Probe Algorithm

Select *next probe* X_{next} based on all history and probe data available so far to calculate the setting with *maximum expected improvement EIP(X)*

Until the stopping condition is reached







This approach balances Exploration Vs. Exploitation

App,Goal

Probe Algorithm

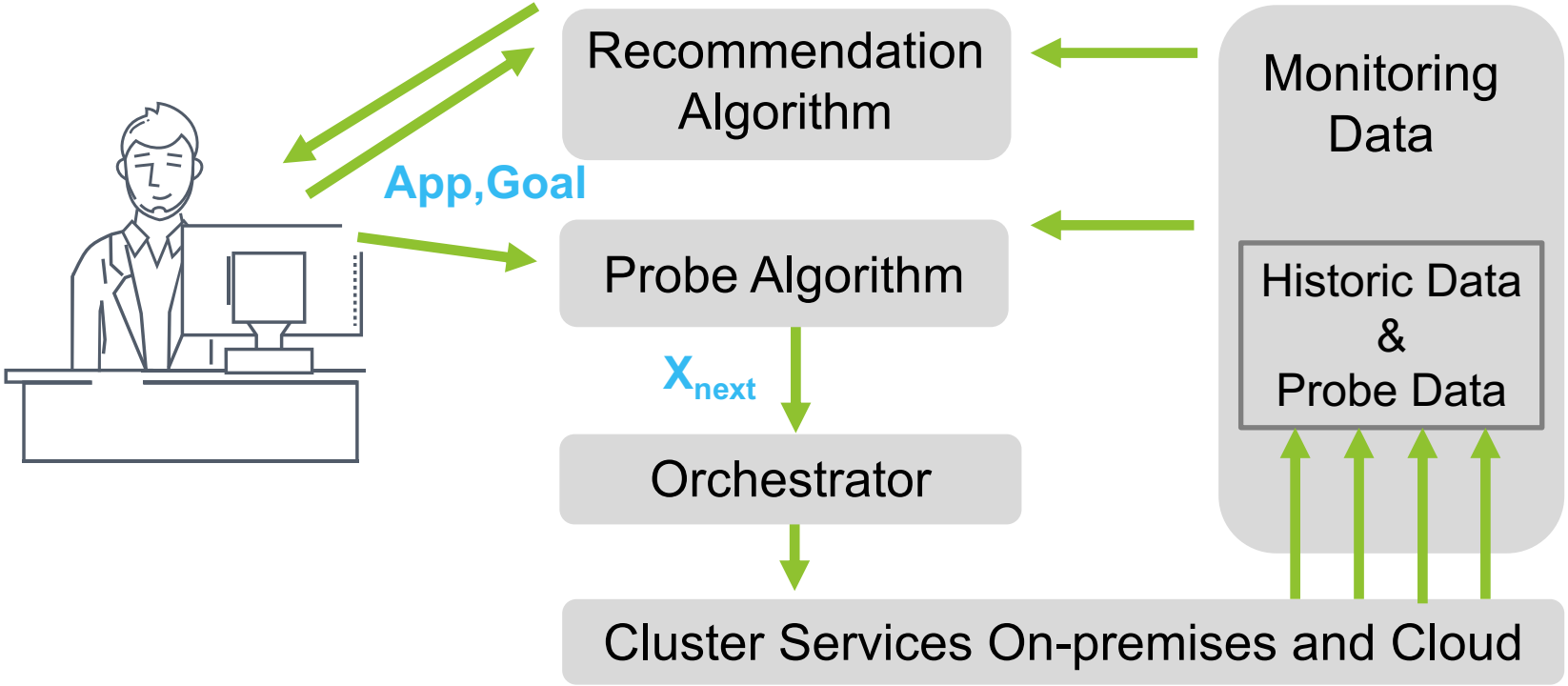
X_{next}

Data Starved
& High Uncertainty

Data Rich
& Low
Uncertainty

Credit: https://discovery.rsm.nl/articles/detail/130-how-to-balance-exploration-and-exploitation-in-multinational-enterprises

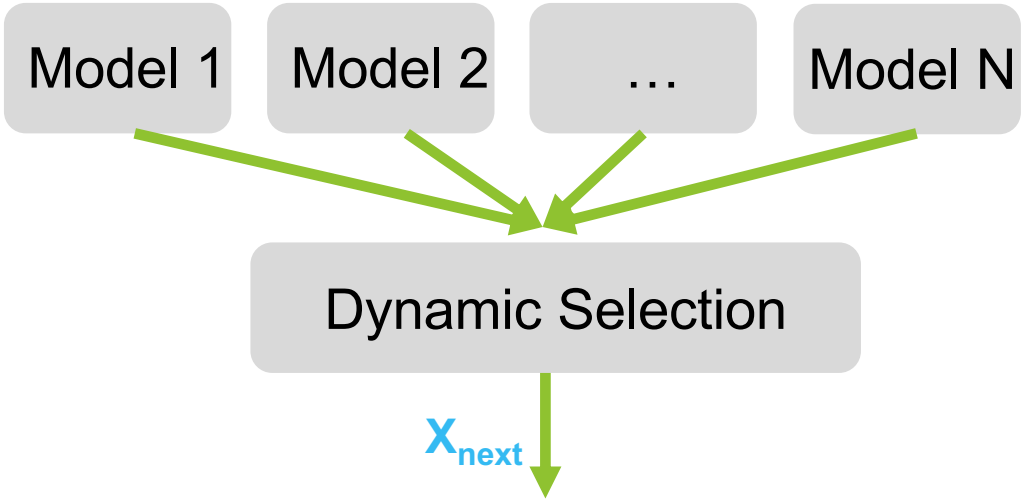
Unravel Sessions Architecture



Probe Algorithm: Has to deal with a diverse space of data & uncertainty

What is the goal?	What is the app?	Are trained models available?	Is similar historic data available?	How soon user needs the answer?
▪ Make the app faster ▪ Make app reliable ▪ Make app meet SLA ▪ Make app use less resources	▪ SQL ▪ Program ▪ DAG ▪ Workload	▪ Yes ▪ No ▪ Partially	▪ No ▪ A little ▪ A lot	▪ Now ▪ Soon ▪ By a deadline

Probe Algorithm: Use Model Ensembles

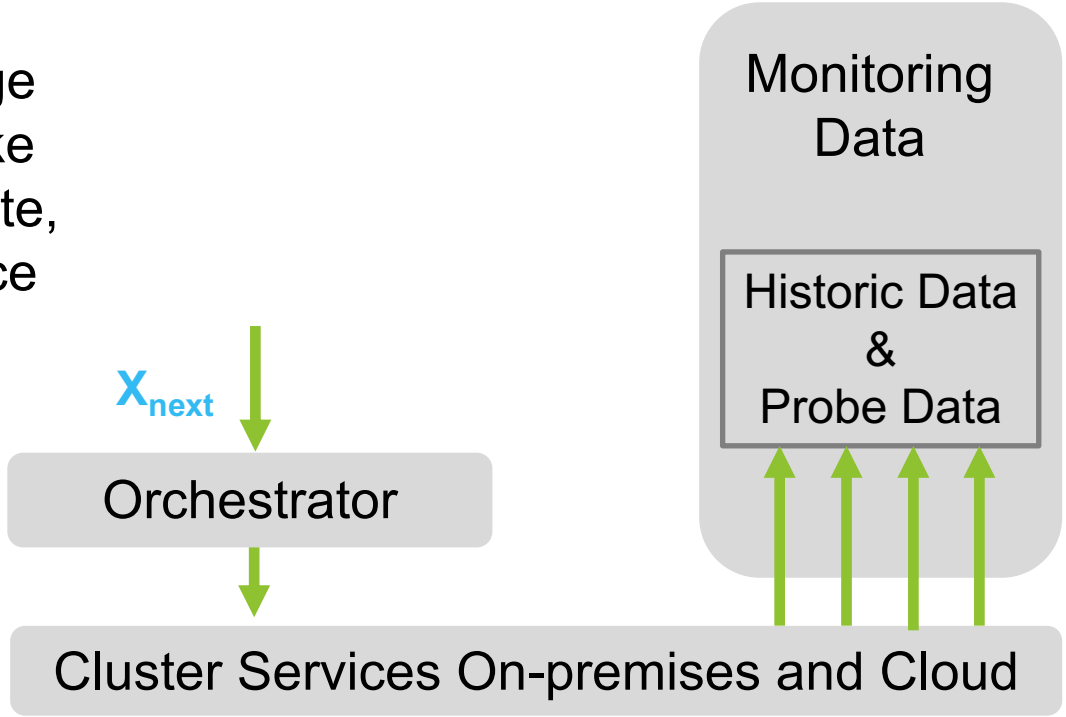


Lessons Learned Building the Unravel Sessions Architecture

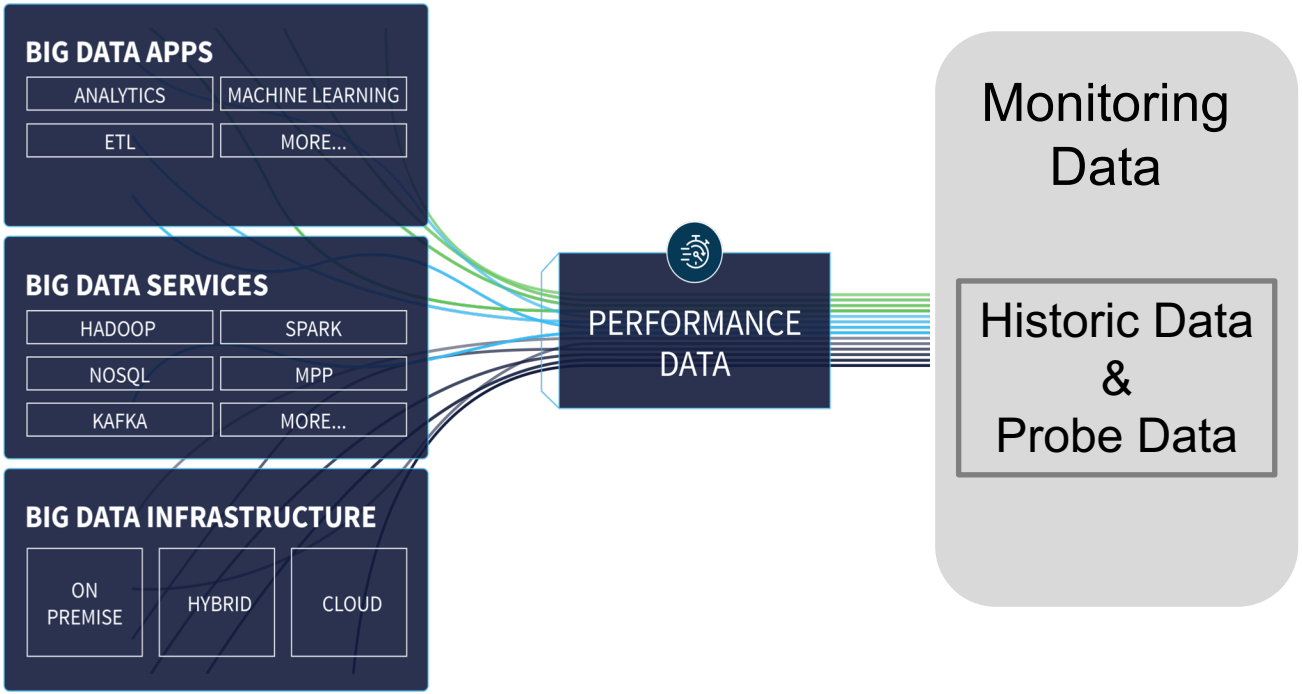
- Probe Algorithm: Use model ensembles
- Orchestrator: Use cheap probes when possible
- Monitoring data: Use full-stack data collection
- Recommendation Algorithm: Keep user in the loop

Lessons Learned: Cheap Probes

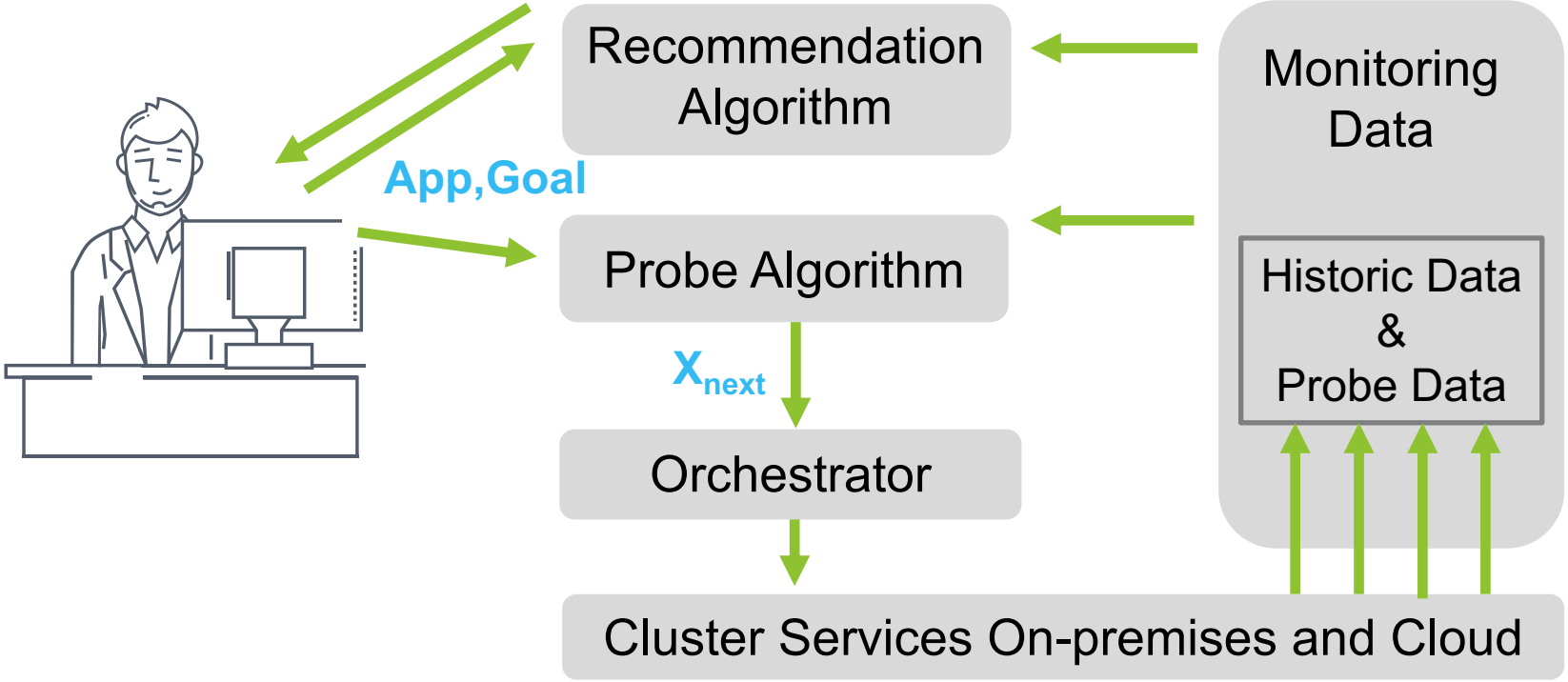
Orchestrator can leverage cost-based optimizers like Catalyst as an approximate, but very fast, performance model in some cases



Lessons Learned: Full Stack



Lessons Learned: User in the Loop



In Summary

AI for **A**pplication **P**erformance **M**anagement

Sign up for a free trial, we value your feedback!
https://unraveldata.com/free-trial

Join the Unravel team!

adrian, shivnath [@unraveldata.com]