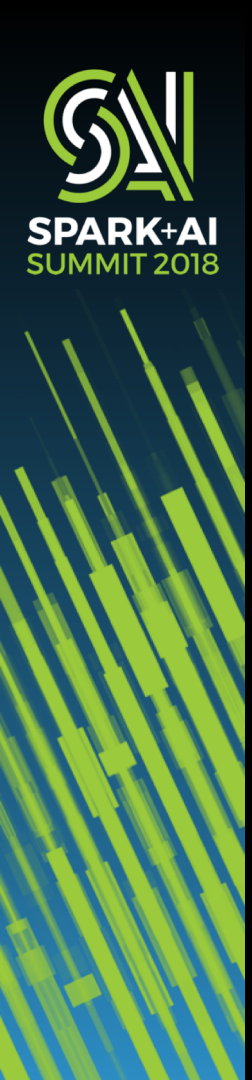


Using Spark to Tune Spark

Adrian Popescu, Shivnath Babu

#AI7SAIS

Meet the speakers

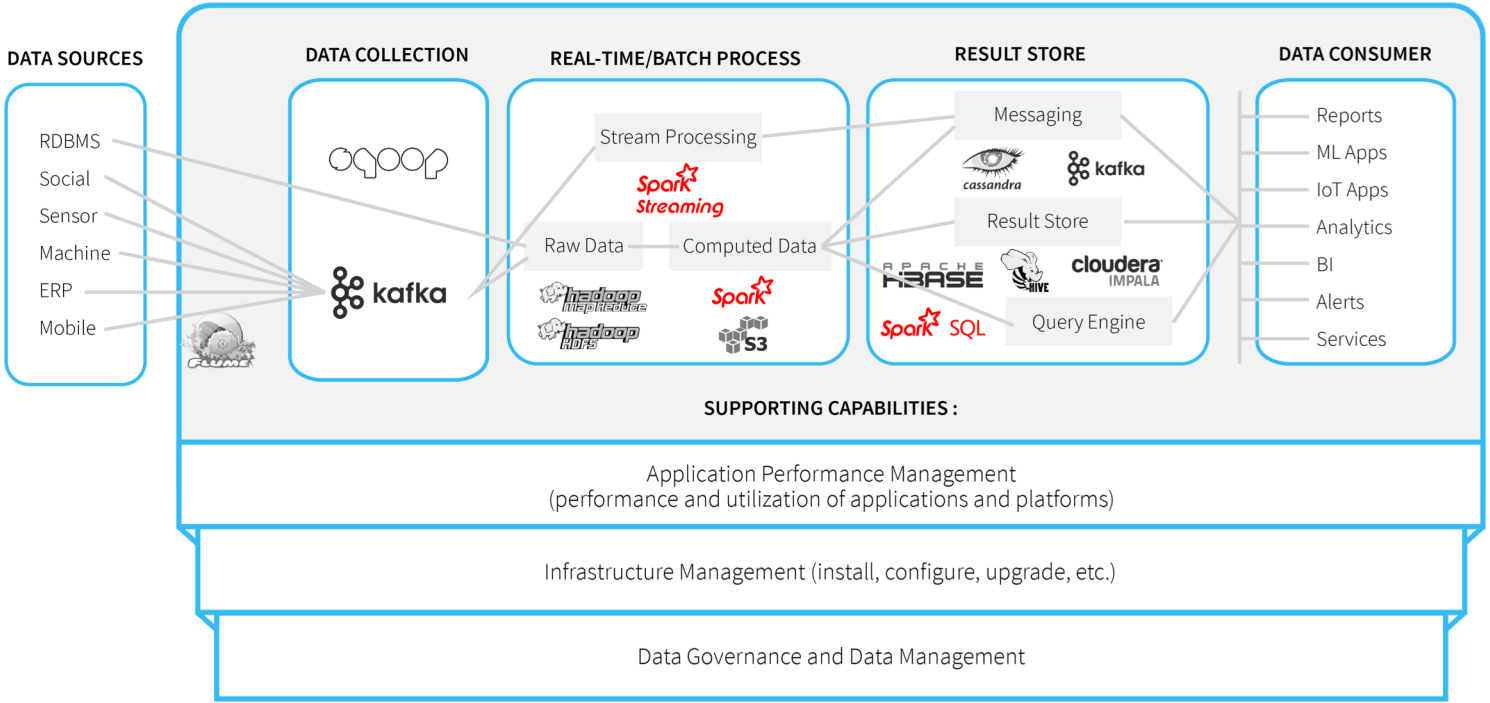
Adrian Popescu

- 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

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

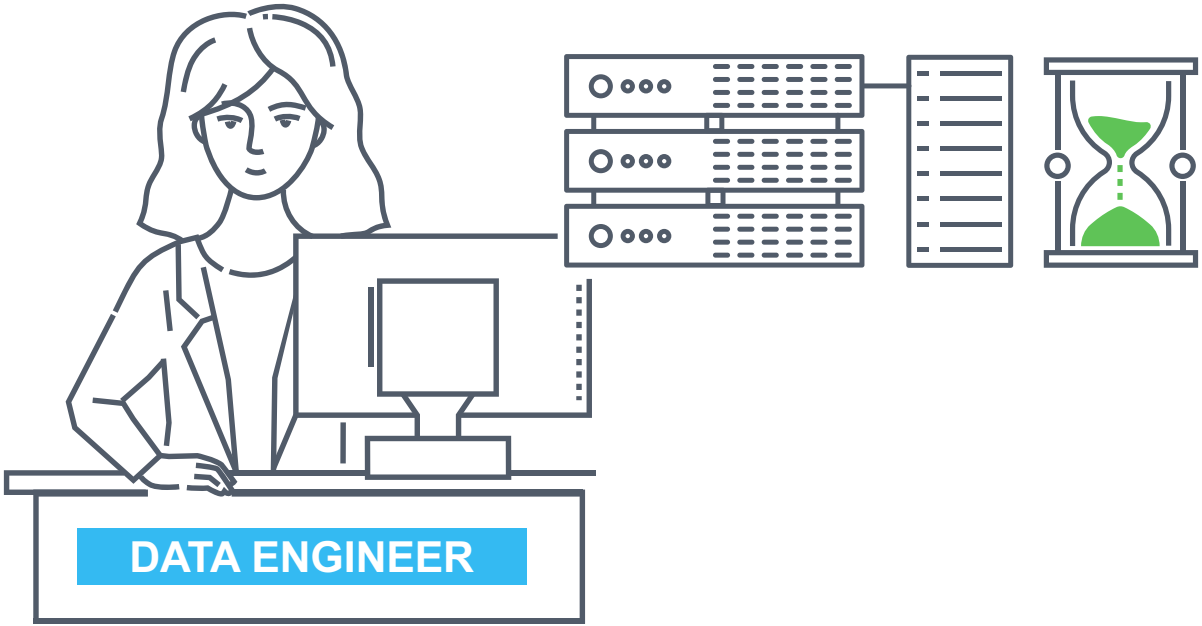
Many apps are being built in Spark



But, let us face it: Running Spark apps in production is hard

My app often fails with Out of Memory…

My app is too slow…

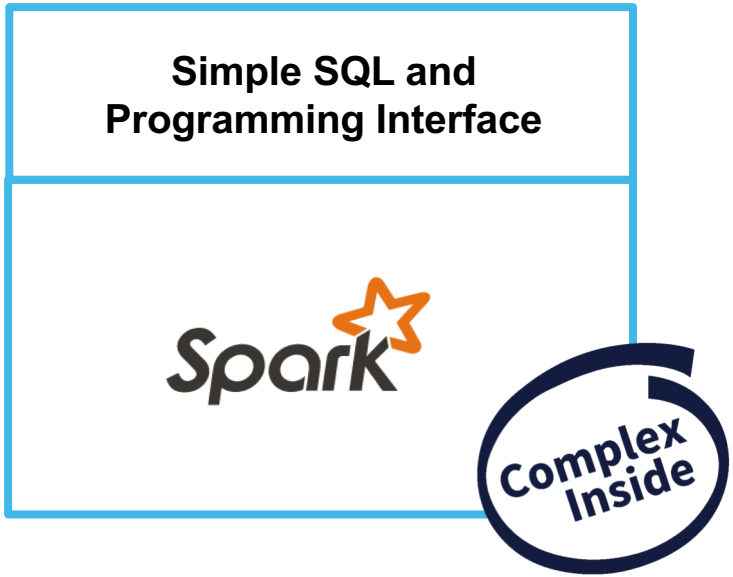


My app is missing SLA…

This rogue app is wasting resources and reducing cluster throughput

Many factors affect app performance

To add to Spark's complexity

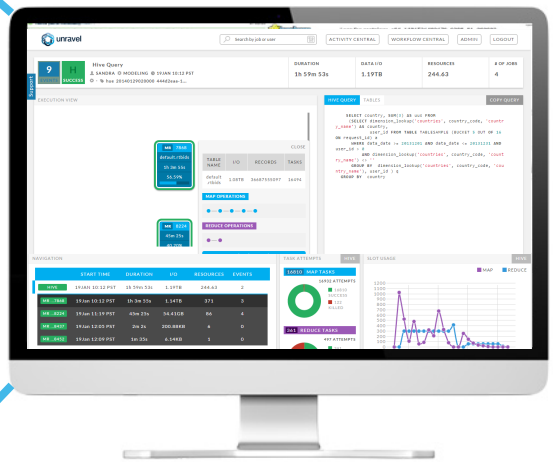


- **Many types of Spark apps**
  - SQL
  - Streaming
  - AI/ML
  - Graph
  - Scala/Python/R
- **Many app submission methods in Spark**
  - CLI
  - Thrift Server
  - Notebooks like Zeppelin, Jupyter, Hue
  - ETL tools like Informatica, Pentaho, and Talend
  - Schedulers like Airflow, Autosys, Control M, Oozie, Tidal, TWS
- **Many infrastructure choices for Spark**
  - On-premises multi-tenant clusters
  - Transient cloud clusters
  - Auto-scaling clusters
  - Containerized deployments

Can we convert this problem into a data problem?

First: Bring all monitoring data to a single platform

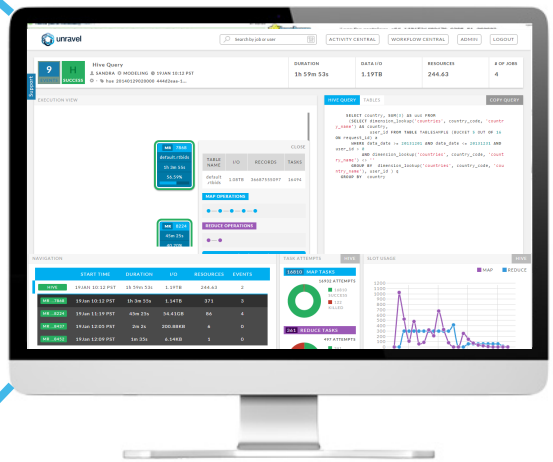
- Resource Manager API
- History Server API
- Container Metrics
- Data Statistics
- SQL Query Plans
- Logs
- Metadata
- Configuration



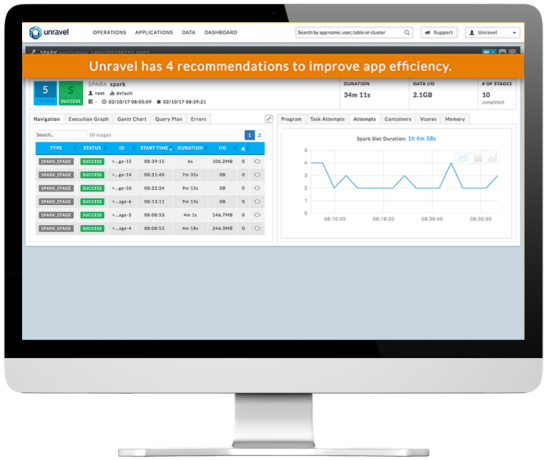
One complete correlated view.

Then: Apply intelligent algorithms to analyze the data automatically

- Resource Manager API
- History Server API
- Container Metrics
- Data Statistics
- SQL Query Plans
- Logs
- Metadata
- Configuration



One complete correlated view.

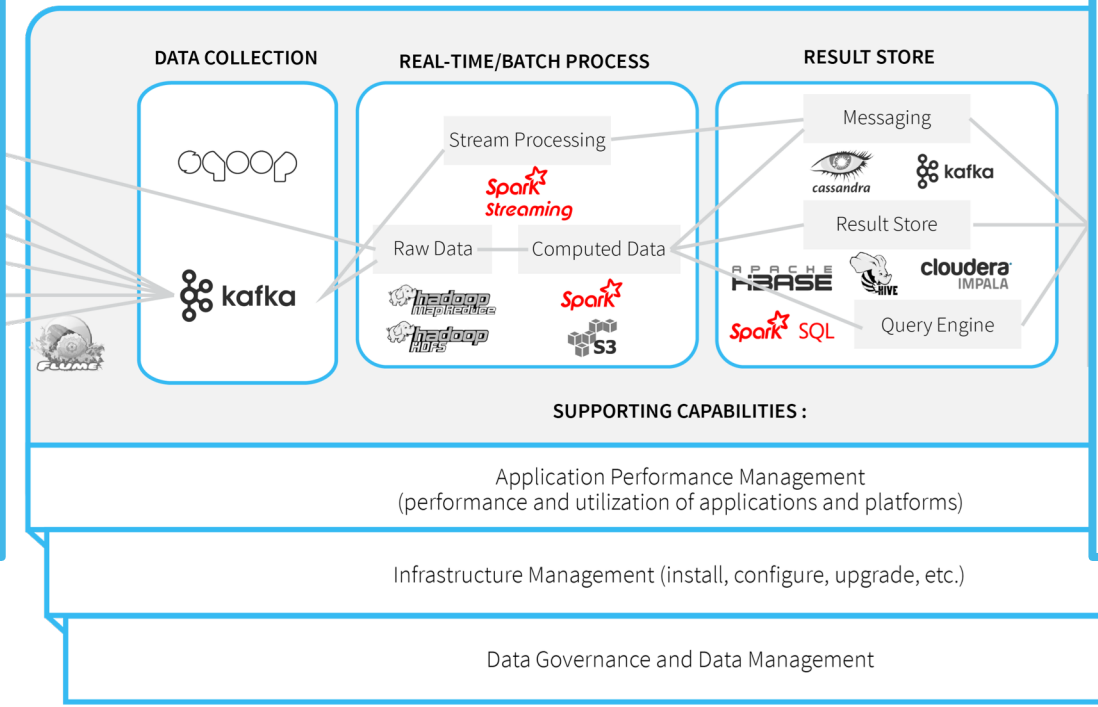


Built-in intelligence.


USE THE
RIGHT TOOL
FOR THE JOB

Why not use Spark itself?

- Resource Manager API
- History Server API
- Container Metrics
- Data Statistics
- SQL Query Plans
- Logs
- Metadata
- Configuration



What application & cluster management tasks can we automate with intelligent algorithms in Spark?

Let us take three (hard) tasks

- **Failures in Spark**
- SLA management for real-time data pipelines
- Application autotuning

Manual Root Cause Analysis of Spark Failures

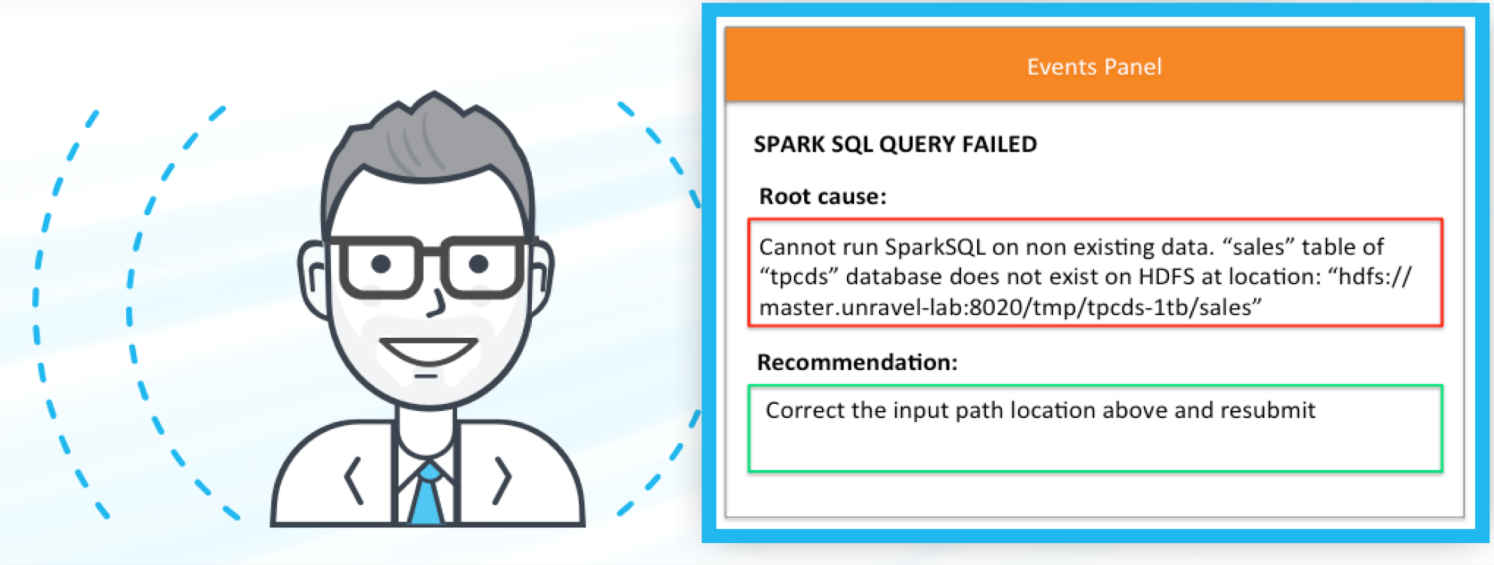
Typical Failure in Spark

```
e.spark.sql.execution.ConvertToSafe.doExecute(rowFormatConverters.scala:56)
e.spark.sql.execution.SparkPlan$$anonfun$execute$5.apply(SparkPlan.scala:132)
e.spark.sql.execution.SparkPlan$$anonfun$execute$5.apply(SparkPlan.scala:130)
e.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:150)
e.spark.sql.execution.SparkPlan.execute(SparkPlan.scala:130)
e.spark.sql.execution.Exchange.prepareShuffleDependency(Exchange.scala:164)
e.spark.sql.execution.Exchange$$anonfun$doExecute$1.apply(Exchange.scala:254)
e.spark.sql.execution.Exchange$$anonfun$doExecute$1.apply(Exchange.scala:248)
e.spark.sql.catalyst.errors.package$.attachTree(package.scala:48)

e.spark.sql.catalyst.errors.package$TreeNodeException: execute, tree:
Partitioning(nation#71,o_year#72,200), None
(key=[nation#71,o_year#72], functions=[(sum(amount#73),mode=Partial,isDistinct=false)], output=[nation#71,o_year#72,sum#78])
e#50 AS nation#71,year(cast(o_orderdate#20 as date)) AS o_year#72,(((l_extendedprice#5 * (1.0 - l_discount#6)) - (ps_supplycost#28 * l_quantity#4)) AS

in [o_orderkey#16L], [l_orderkey#0L]
rderkey#16L ASC], false, 0
enExchange hashpartitioning(o_orderkey#16L,200), None
ject [o_orderkey#16L,o_orderdate#20]
Scan ExistingRDD[o_orderkey#16L,o_custkey#17L,o_orderstatus#18,o_totalprice#19,o_orderdate#20,o_orderpriority#21,o_clerk#22,o_shippriority#23L,o_comm
rderkey#0L ASC], false, 0
enExchange hashpartitioning(l_orderkey#0L,200), None
ject [l_extendedprice#5,l_discount#6,l_quantity#4,l_orderkey#0L,n_name#50,ps_supplycost#28]
SortMergeJoin [p_partkey#30L], [l_partkey#1L]
:- Sort [p_partkey#30L ASC], false, 0
:  +- TungstenExchange hashpartitioning(p_partkey#30L,200), None
:     +- Project [p_partkey#30L]
:        +- Filter Contains(p_name#31, ghost)
:           +- Scan ExistingRDD[p_partkey#30L,p_name#31,p_mfgr#32,p_brand#33,p_type#34,p_size#35L,p_container#36,p_retailprice#37,p_comment#38]
+- Sort [l_partkey#1L ASC], false, 0
   +- TungstenExchange hashpartitioning(l_partkey#1L,200), None
      +- Project [l_extendedprice#5,l_discount#6,l_quantity#4,l_partkey#1L,l_orderkey#0L,n_name#50,ps_supplycost#28]
         +- SortMergeJoin [ps_suppkey#26L,ps_partkey#25L], [l_suppkey#2L,l_partkey#1L]
            :- Sort [ps_suppkey#26L ASC,ps_partkey#25L ASC], false, 0
            :  +- TungstenExchange hashpartitioning(ps_suppkey#26L,ps_partkey#25L,200), None
```

5 levels of stack traces of this form

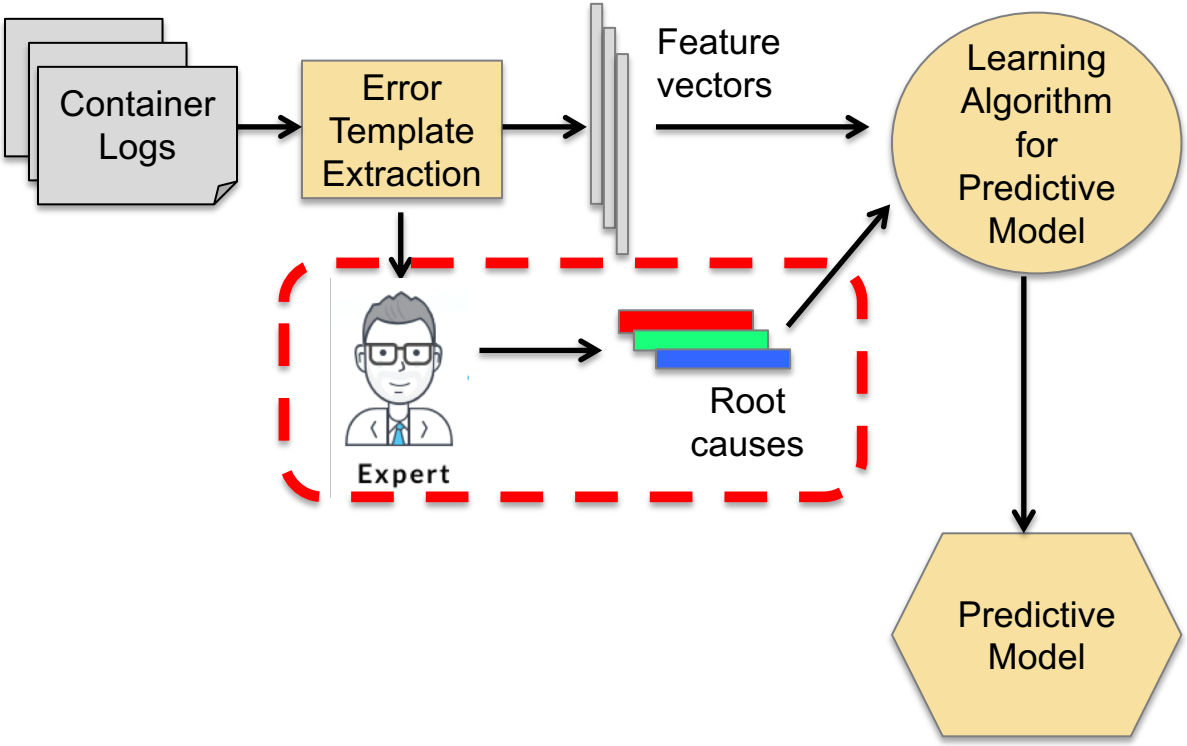
- Many levels of correlated stack traces
- Identifying the root cause is hard and time consuming

Automated Root Cause Analysis of Spark Failures

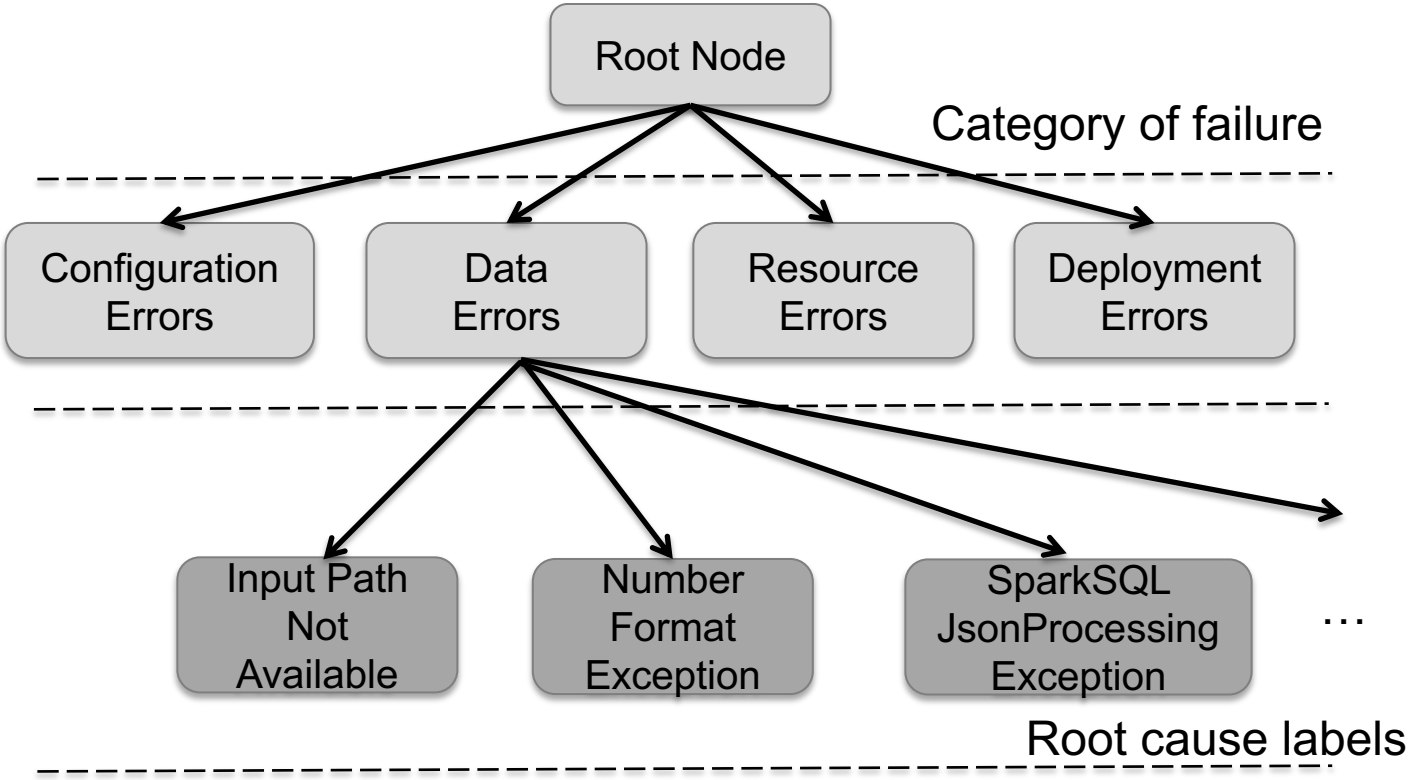


- **Reduce troubleshooting time from days to seconds**
- Improve productivity of data scientists and analysts

Automatic Root Cause Analysis



We have created a Failure Taxonomy



Two Ways to get Root-Cause Labels

- Manual diagnosis by a domain expert
- Automatic injection of the root cause

Unravel's Large-scale Lab Framework for Automatic Root Cause Analysis

Environment:

- Lab created on demand on cloud or on-premises
- Workloads are run and failures are injected

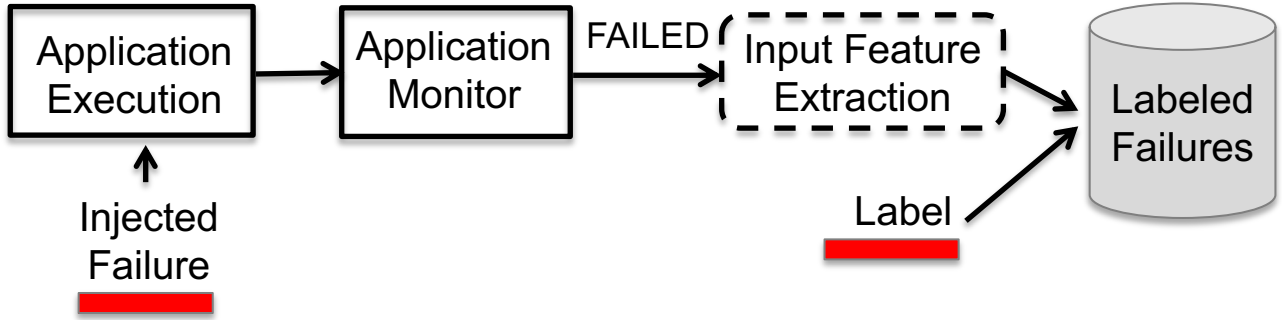
Spark and multi-tenant Workloads:

- Variety of workloads: Batch, ML, SQL, Streaming, etc.

Failures:

- Large set of root causes learned from customers & partners. Constantly updated
- Continuously inject these root causes to train & test models for root-cause prediction

Injecting Failures



Injected failure examples:

- Invalid input
- Invalid memory configuration
- OOME: Java heap space
- OOME: GC overhead limit
- Container killed by YARN
- Runtime incompatibility
- No space left on device
- Transformations inside other transformations
- Runtime error
- Arithmetic error
- Invalid configuration settings

Extracting Input Features from Logs

```
java.lang.OutOfMemoryError: Java heap space
        at
scala.reflect.ManifestFactory$$anon$9.newArray(Manifest.scala:114)
        at
scala.reflect.ManifestFactory$$anon$9.newArray(Manifest.scala:112)
        at …
```

- Extracting stack traces and error messages
- Tokenize by class names and words
- Create a vocabulary of words from all words collected

Tokens example:

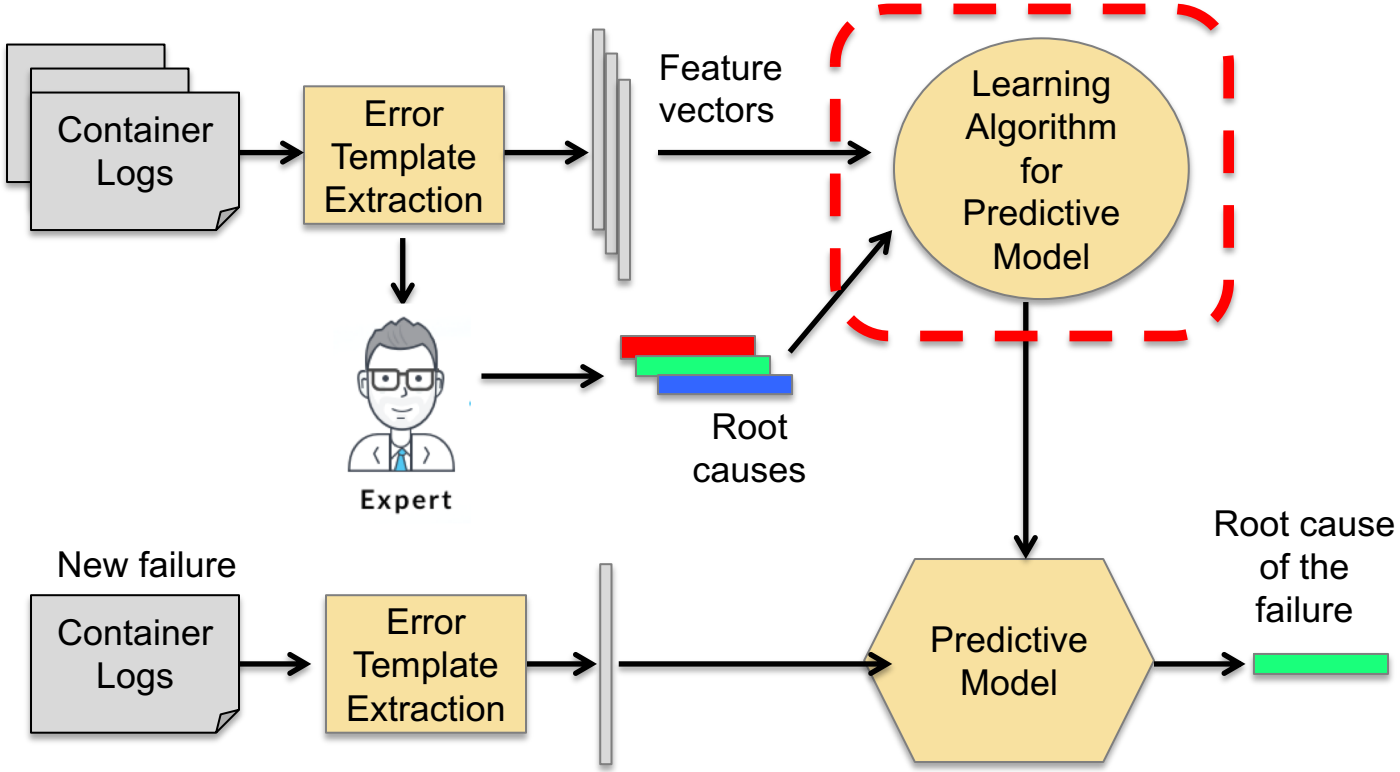
```
java.lang.OutOfmemoryError Java heap space at
scala.reflect.ManifestFactory$$anon$9.newArray(Manife
st.scala:114)
```

Input Feature extraction

- **Bag of Words with TF-IDF**
  - Computes a vocabulary of words
  - Uses TF-IDF to reflect importance of words in a document

- **Doc2Vec**
  - Maps words, paragraphs, or documents to multi-dimensional vectors
  - Evaluates the placement of words wrt neighboring words
  - Uses a 3-layer neural network

System Architecture



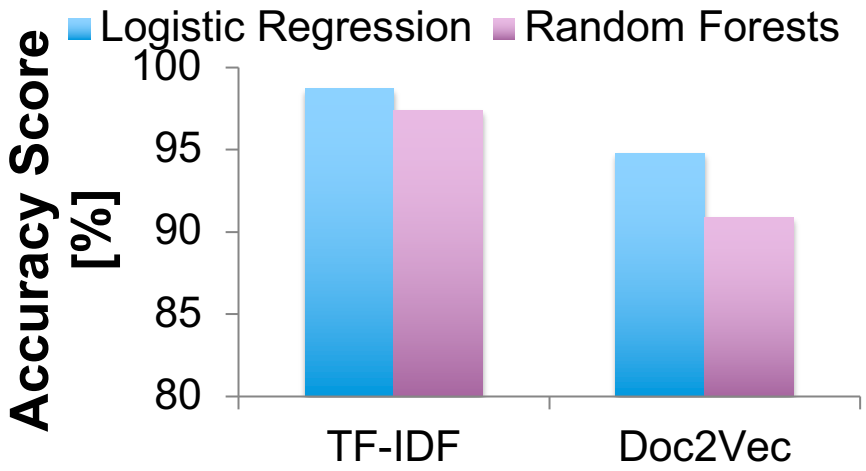
Predictive Models

- **Shallow Learning**
  - Logistic Regression
  - Random forests
- **Deep Learning**
  - Neural networks

Very easy to implement these in Spark

Predicting the Root Cause of Failures

- **Training** and **testing** with injected failures
- Test to train data set ratio **75% to 25%**
- **Models**: logistic regression, random forests



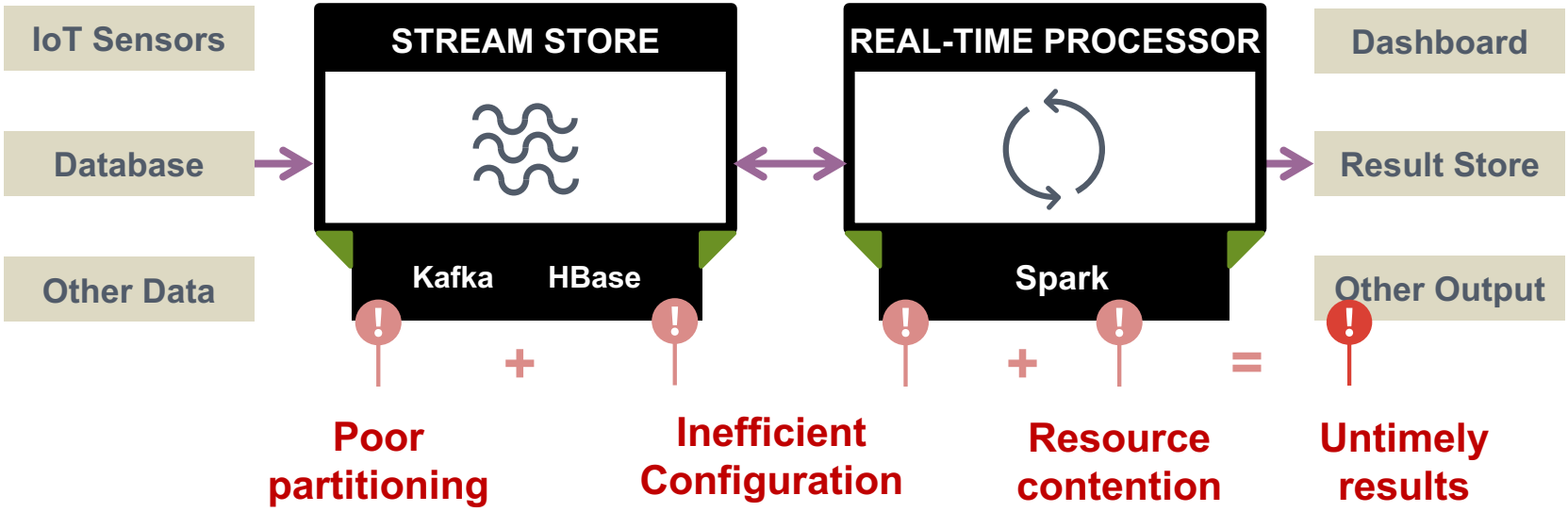
Work with deep learning is in progress

See our talk at Strata, NY 2017 for more details

Let us take three (hard) tasks

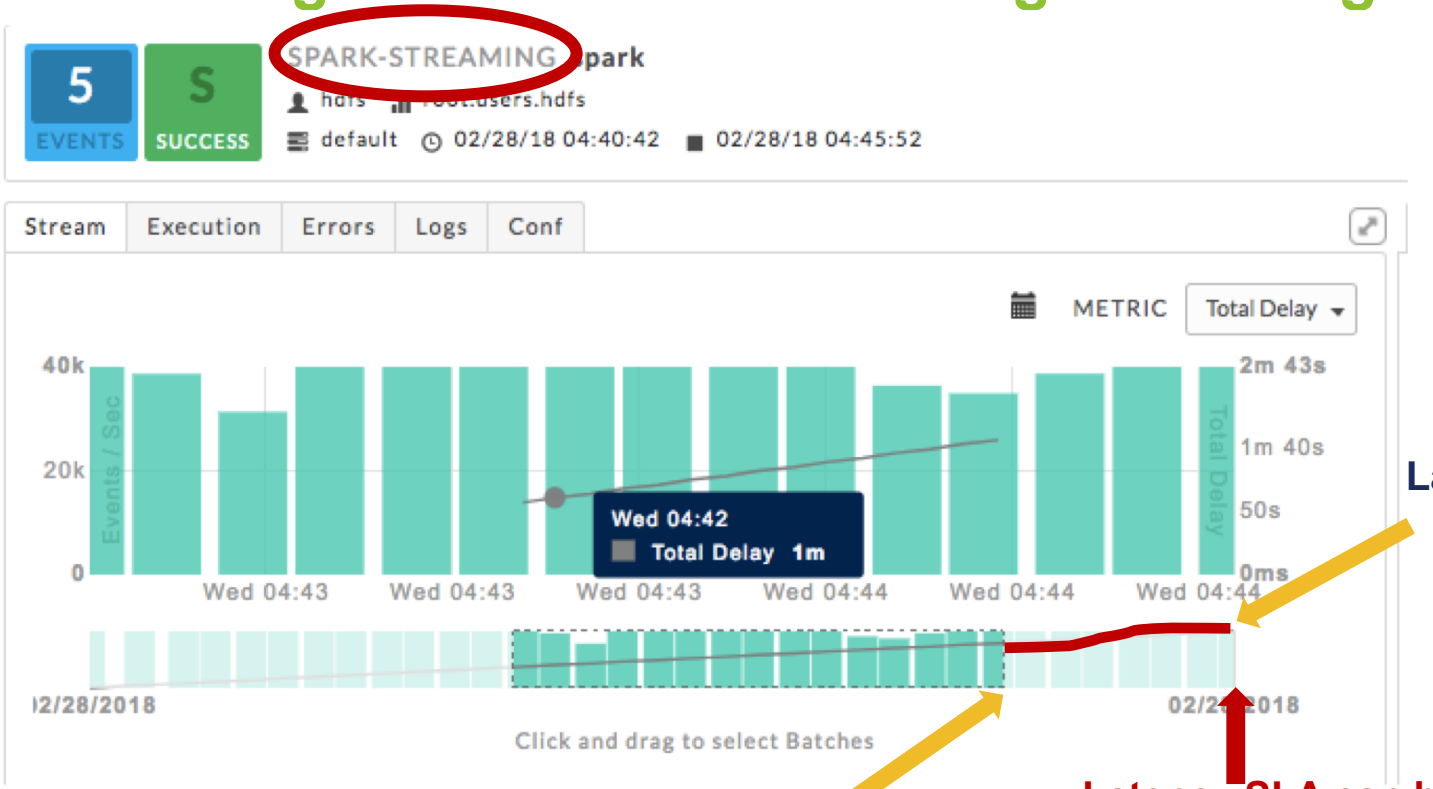
- Failures in Spark
- **SLA management for real-time data pipelines**
- Application autotuning

Many problems can cause unreliable performance in streaming pipelines



Forecasting & Anomaly Detection are important for streaming pipelines

Predicting when SLAs are in danger of being missed

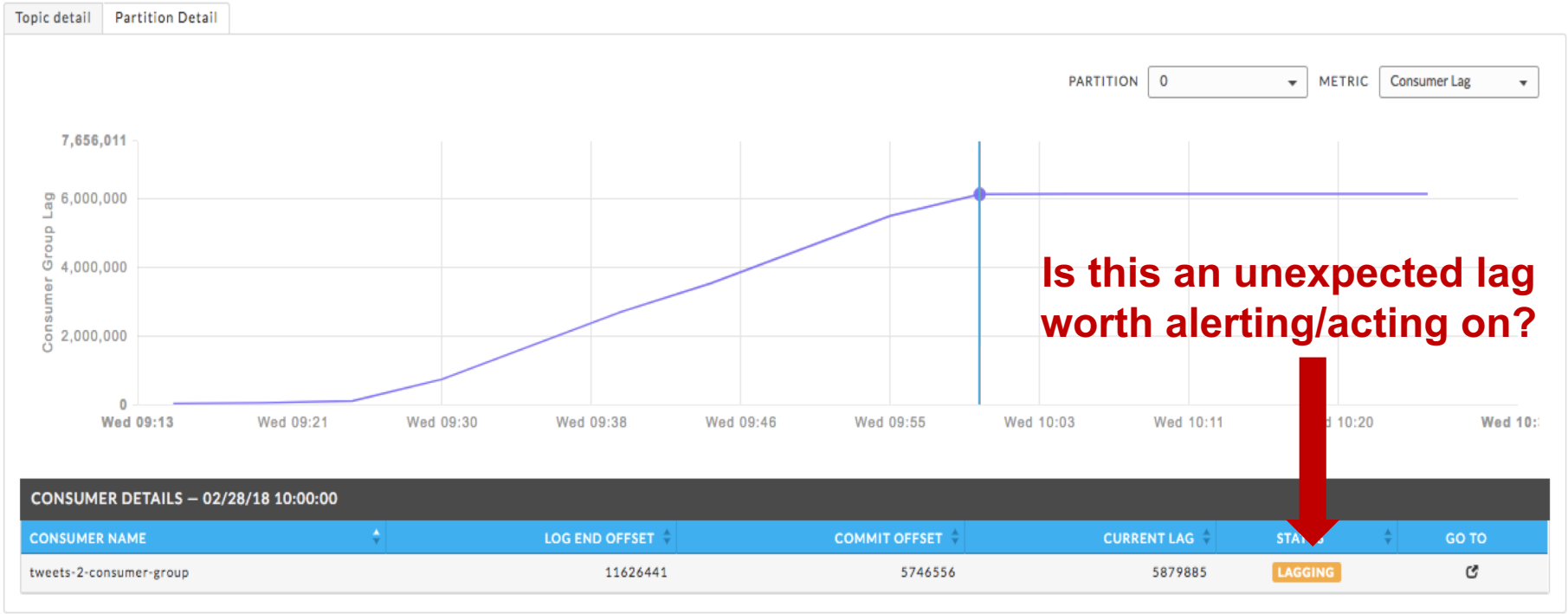


Latency SLA is 3 minutes

Current time is here

Latency SLA can be missed by this time

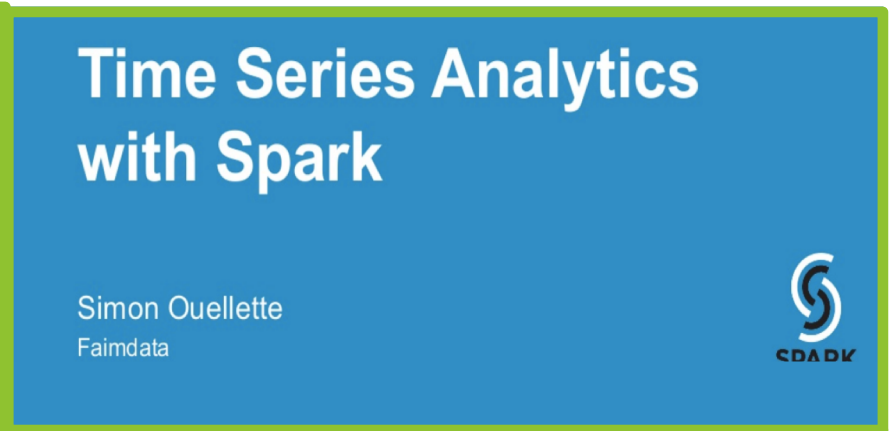
Detecting anomalies is tricky



What is an anomaly?

1. An unexpected change that needs your attention
2. Smart alerts/actions:
    1. False positives should be minimal
    2. False negatives should be minimal

Time Series Analysis in Spark



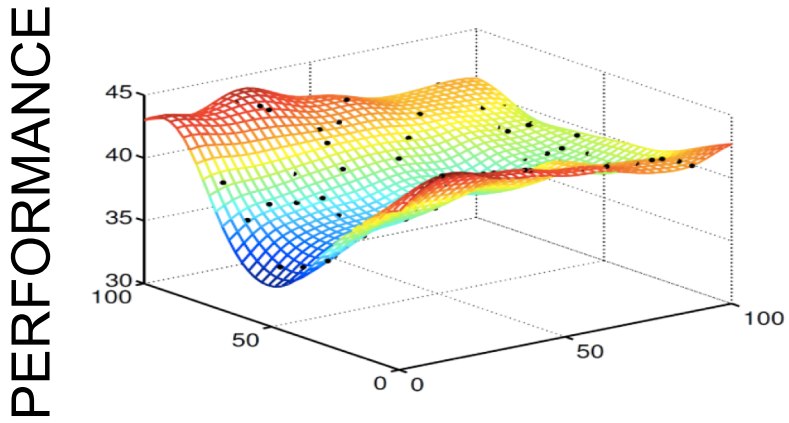
Time Series Analysis in Spark

See our talk at Strata, San Jose 2018 for more insights

Let us take three (hard) tasks

- Failures in Spark
- SLA management for real-time data pipelines
- **Application autotuning**

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
…	



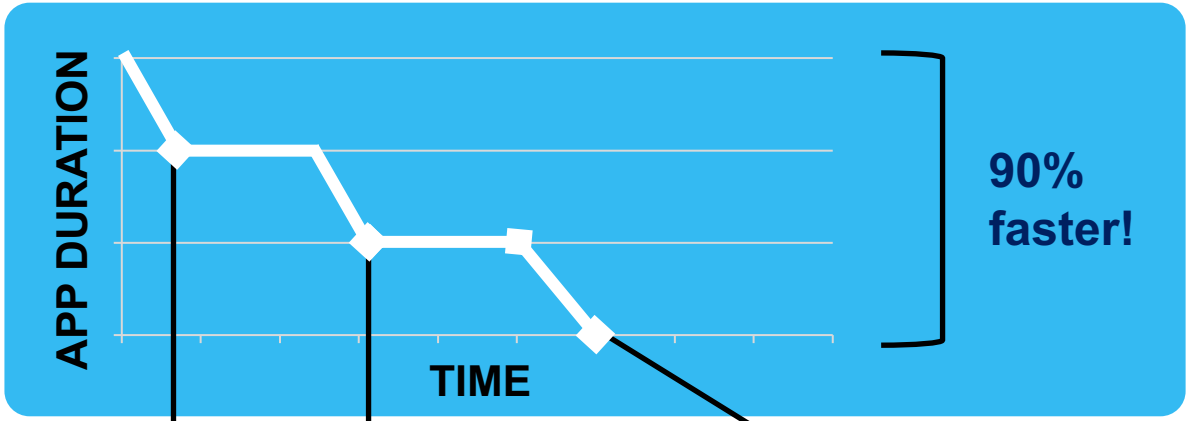
Today, tuning is often by trial-and-error

Imagine a new world

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

"I need to make this app faster"

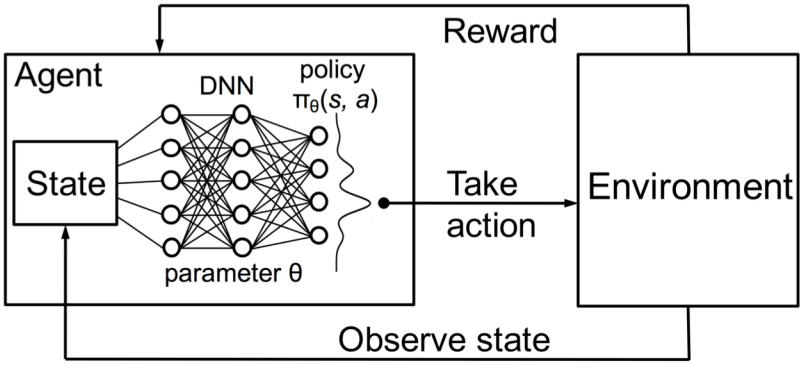
Imagine a new world



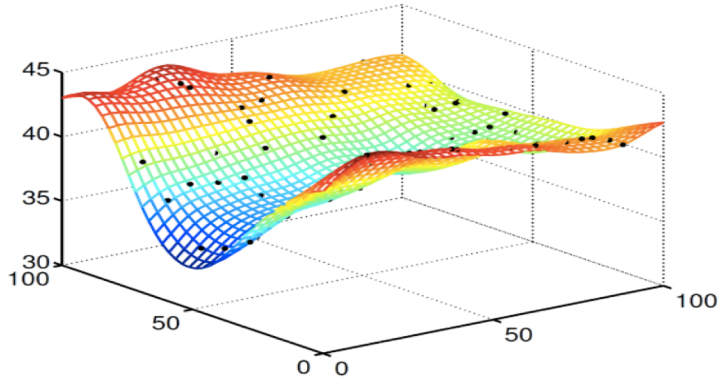
In blink of an eye, user gets recommendations to make the app **30% faster**

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**



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

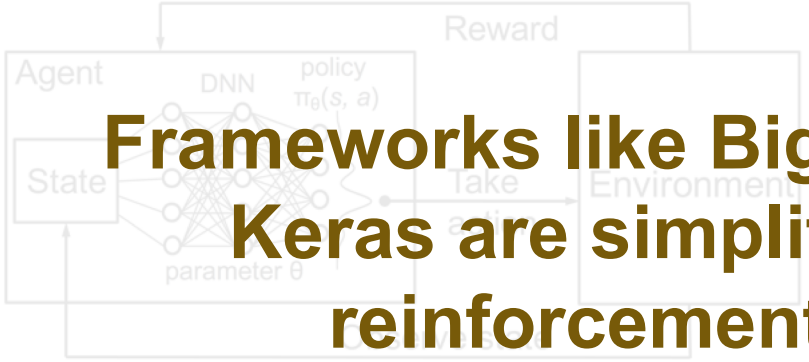
Xplus: A SQL-Tuning-Aware Query Optimizer

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:



Reinforcement Learning

Response Surface Methodology

Frameworks like BigDL, Deeplearning4j, and Keras are simplifying the use of deep reinforcement learning in Spark

Tuning Database Configuration Parameters with iTuned

Songyun Duan, Vamsidhar Thummala, Shivnath Babu
Department of Computer Science
Duke University
Durham, NC 27708, 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, ... 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, 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 cost-

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:

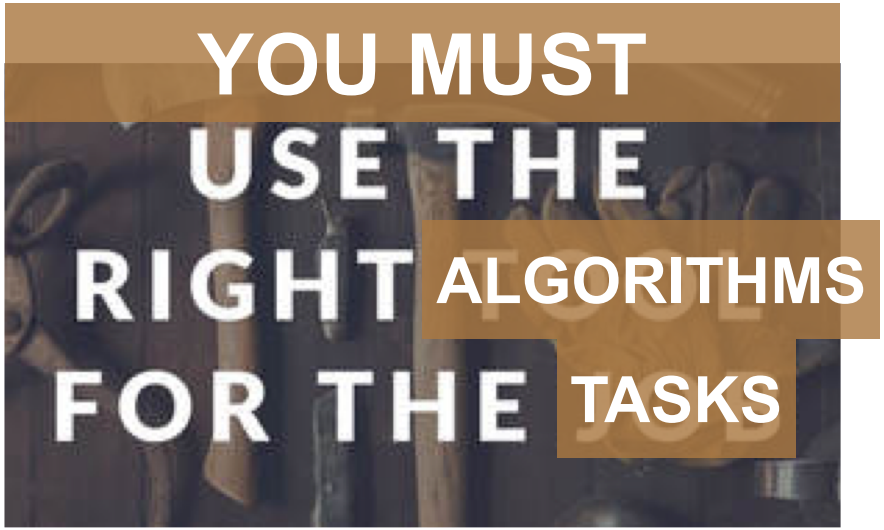
See our talk on Unravel Sessions at Spark+AI Summit 2018 on this problem

In Summary

- Spark Performance Management can be addressed using Spark itself

In Summary

- Spark Performance Management can be addressed using Spark itself



Sign up for a free trial. We value your feedback!

https://unraveldata.com/free-trial

Join the Unravel team!

adrian & shivnath [@unraveldata.com]