AWS re:Invent

BDM301

Best Practices for Apache Spark on Amazon EMR

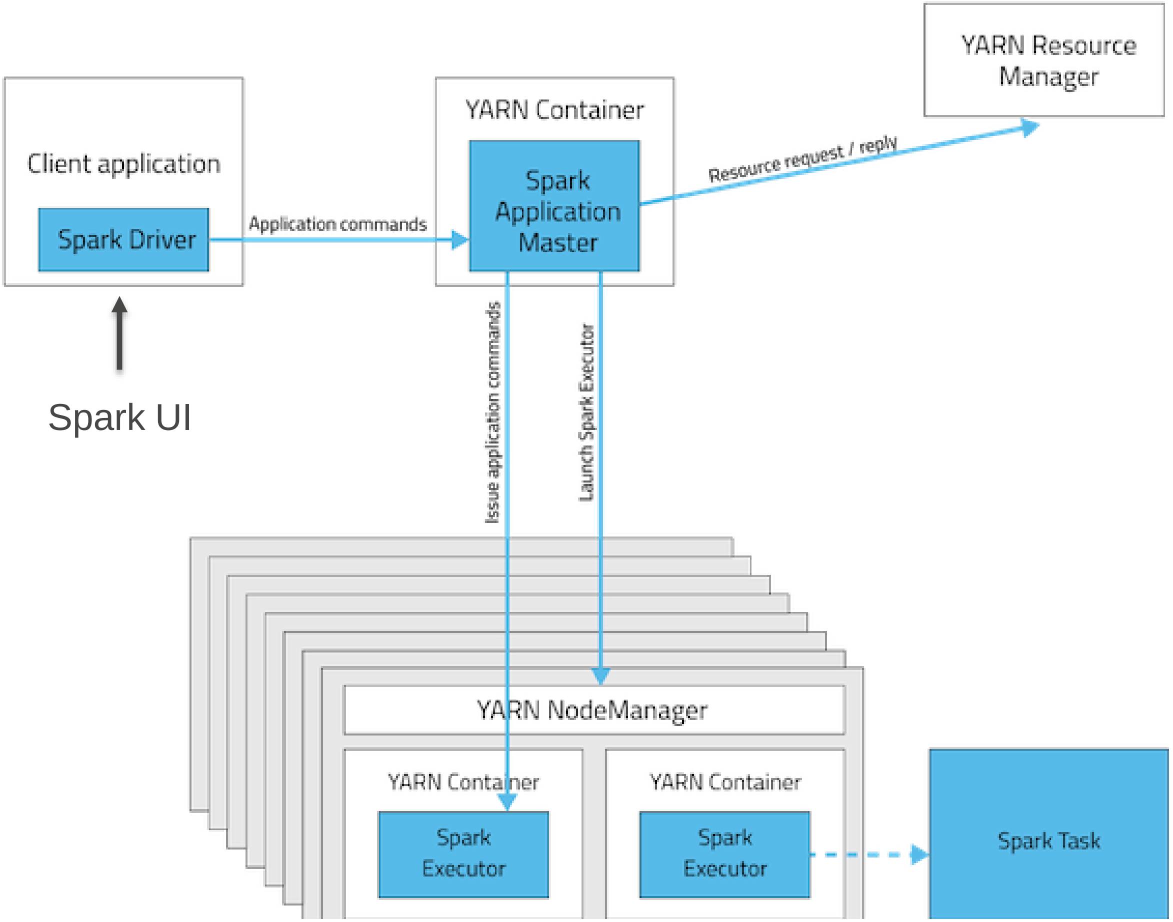
Jonathan Fritz, Sr. Product Manager, Amazon EMR
Yekesa Kosuru, VP of Engineering, DataXu
Dong Jiang, Sr. Principal Software Engineer, DataXu
Saket Mengle, Sr. Principal Data Scientist, DataXu

AWS re:Invent 2016

What to Expect from the Session

- Spark on EMR architecture
- Spark performance and using S3
- Spark security
- Spark for ETL and data science at DataXu
- Demo of DataXu’s Spark workflow

Spark on YARN

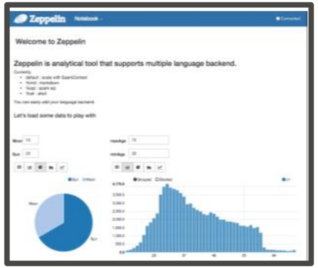


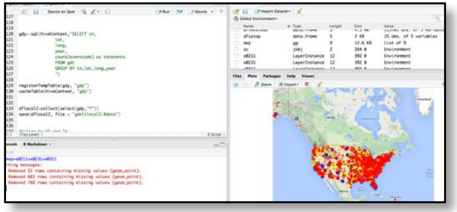
- Run Spark Driver in Client or Cluster mode
- Spark application runs as a YARN application
- SparkContext runs as a library in your program, one instance per Spark application.
- Spark Executors run in YARN Containers on NodeManagers in your cluster
- Access Spark UI through the Resource Manager or Spark History Server

Configuring Executors – Dynamic Allocation

- Optimal resource utilization
- YARN dynamically creates and shuts down executors based on the resource needs of the Spark application
- Spark uses the executor memory and executor cores settings in the configuration for each executor
- Amazon EMR uses dynamic allocation by default, and calculates the default executor size to use based on the instance family of your Core Group
- Or, use maximizeResourceAllocation for single-tenancy

Options to submit jobs – on cluster





Notebooks and IDEs: Apache Zeppelin, Jupyter, RStudio, and more!

Use Spark Actions in your Apache Oozie workflow to create DAGs of jobs.

(start using start-thriftserver.sh)

Connect with ODBC / JDBC to Spark Thriftserver

Or, use Spark Submit or the Spark Shell

Develop fast using notebooks and IDEs

Included in EMR releases

Install using Bootstrap Actions

Thrift Server

(start using start-thriftserver.sh)

Connect with ODBC / JDBC

Options to submit jobs – off cluster

Submit a Spark application

Amazon EMR Step API

Use AWS Lambda to submit applications to EMR Step API or directly to Spark on your cluster

AWS Lambda

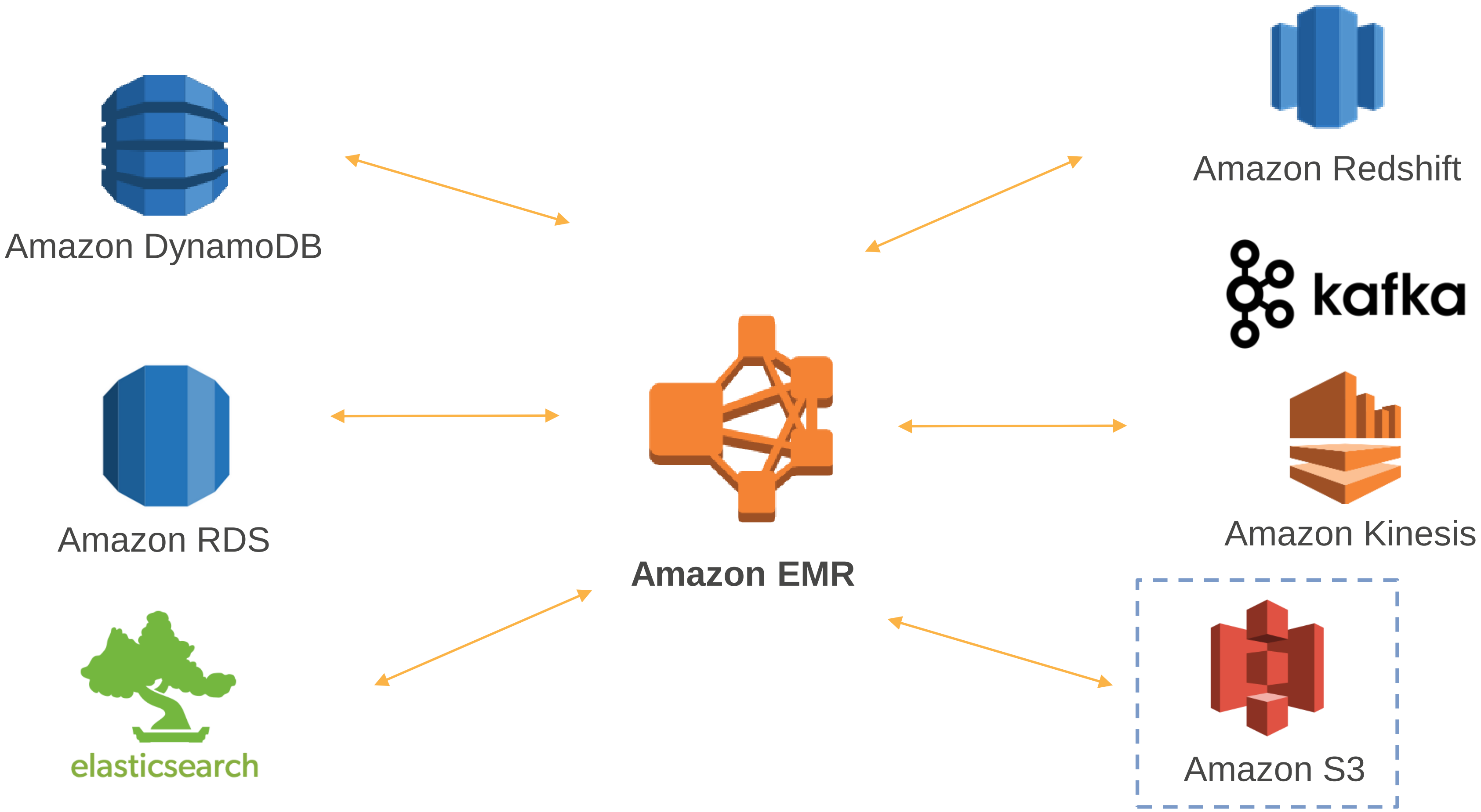
Create a pipeline to schedule job submission or create complex workflows

AWS Data Pipeline

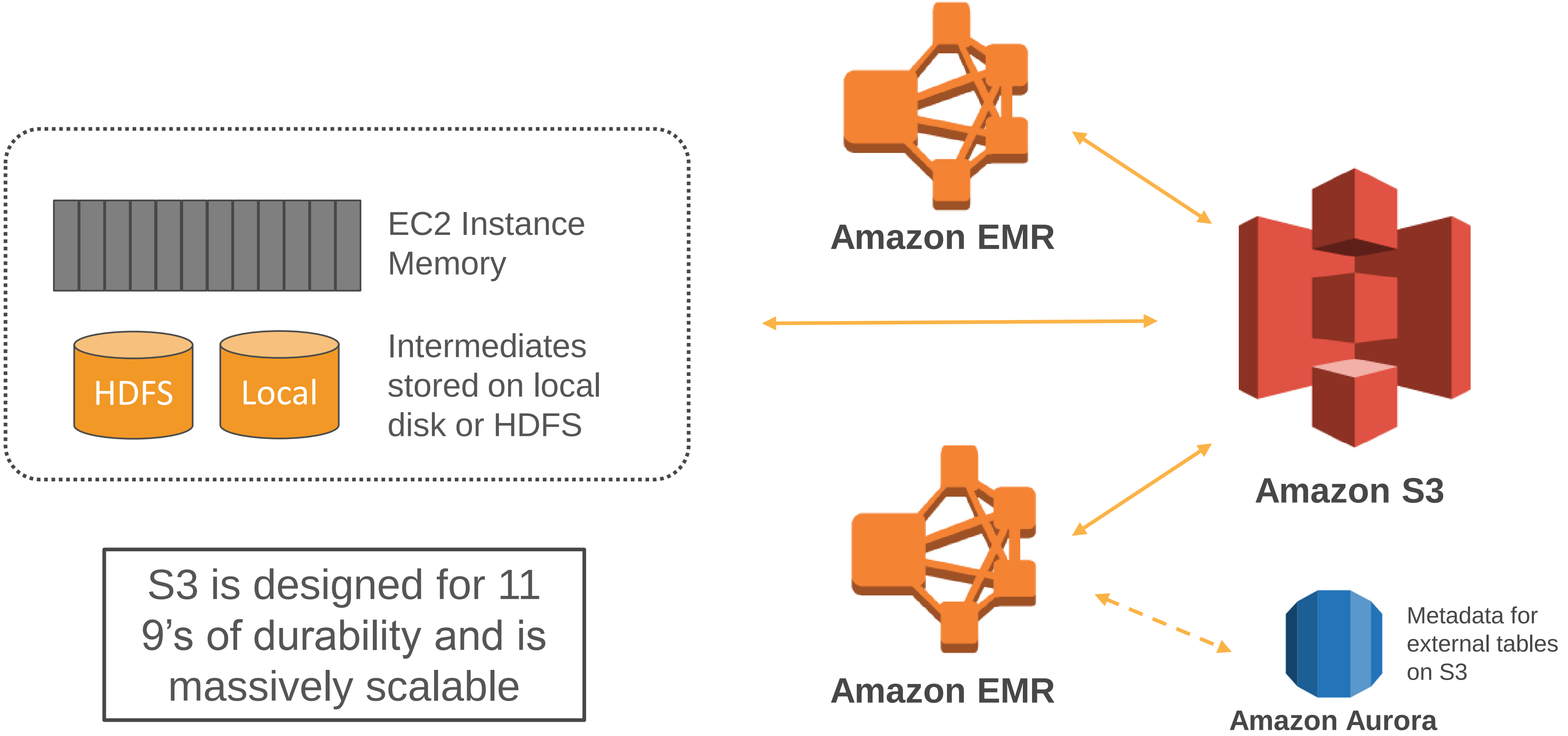
Airflow, Luigi, or other schedulers on EC2

Amazon EMR

Many storage layers to choose from



Decouple compute and storage by using S3 as your data layer

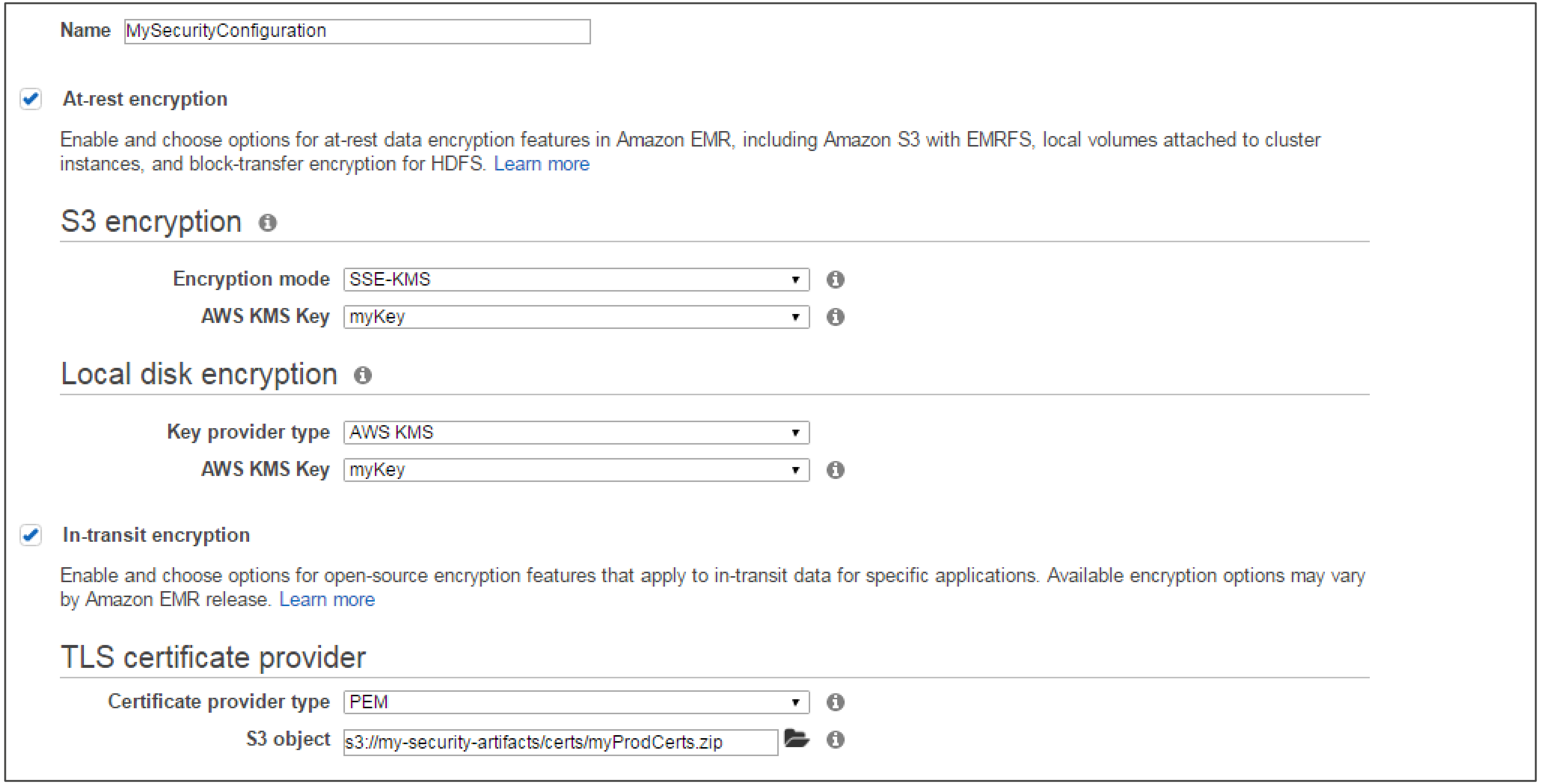


S3 is designed for 11 9's of durability and is massively scalable

S3 tips: Partitions, compression, and file formats

- Avoid key names in lexicographical order
- Improve throughput and S3 list performance
- Use hashing/random prefixes or reverse the date-time
- Compress data set to minimize bandwidth from S3 to EC2
    - Make sure you use splittable compression or have each file be the optimal size for parallelization on your cluster
- Columnar file formats like Parquet can give increased performance on reads

Encryption at-rest and in-flight



Name: MySecurityConfiguration

At-rest encryption

Enable and choose options for at-rest data encryption features in Amazon EMR, including Amazon S3 with EMRFS, local volumes attached to cluster instances, and block-transfer encryption for HDFS. Learn more

S3 encryption

Encryption mode: SSE-KMS

AWS KMS Key: myKey

Local disk encryption

Key provider type: AWS KMS

AWS KMS Key: myKey

In-transit encryption

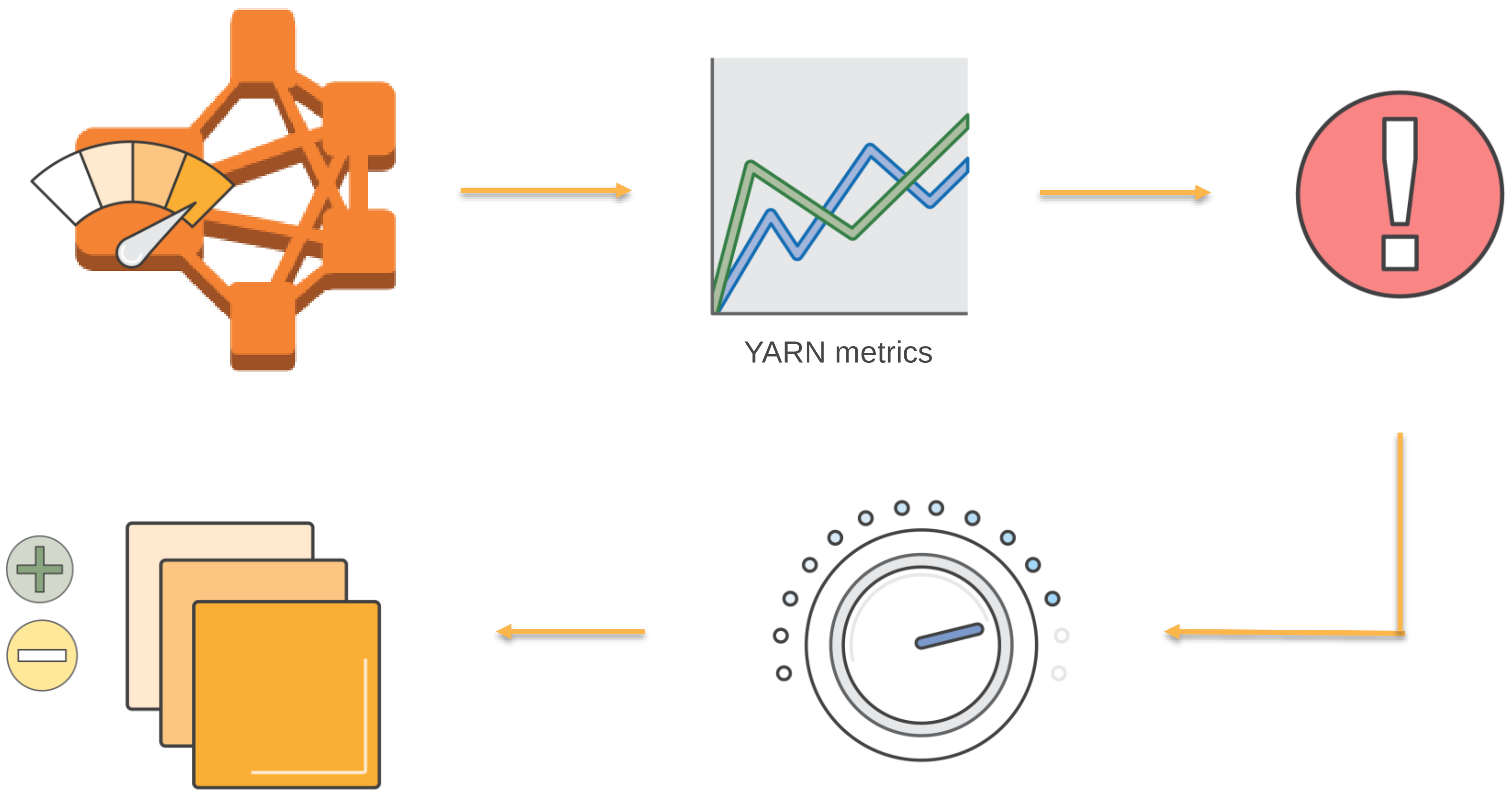
Enable and choose options for open-source encryption features that apply to in-transit data for specific applications. Available encryption options may vary by Amazon EMR release. Learn more

TLS certificate provider

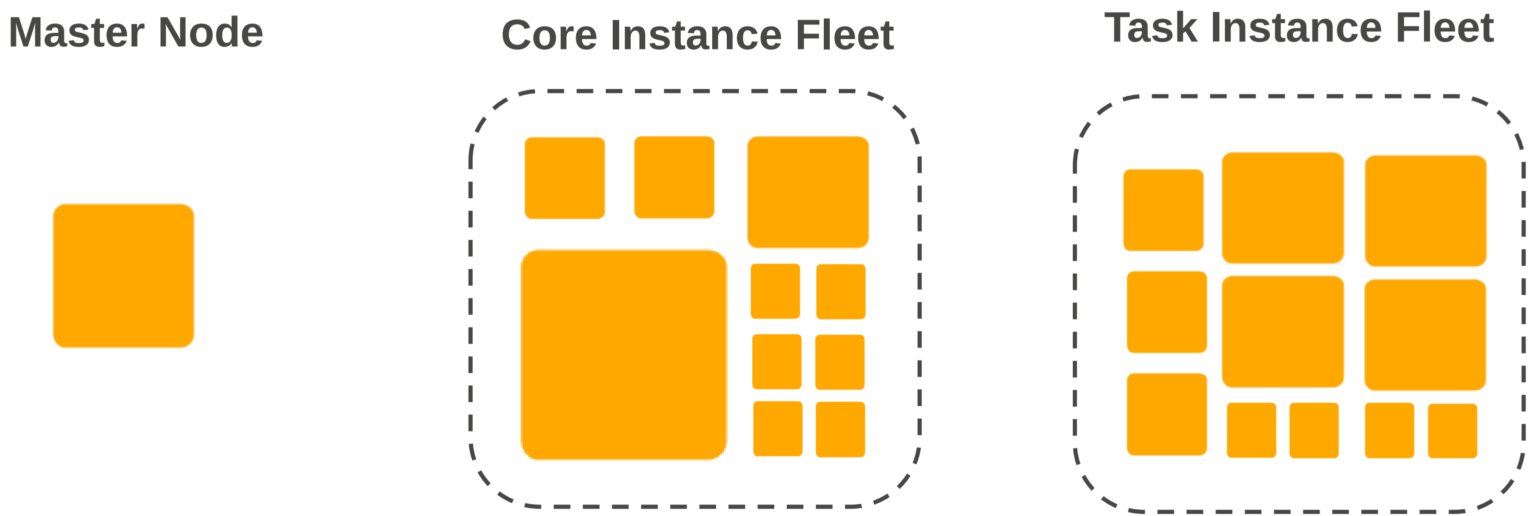
Certificate provider type: PEM

S3 object: s3://my-security-artifacts/certs/myProdCerts.zip

Auto Scaling for data science on-demand



Coming Soon: Advanced Spot provisioning



- Provision from a list of instance types with Spot and On-Demand
- Launch in the most optimal AZ based on capacity/price
- Spot Block support

Using Spark at DataXu - Overview

- Who is DataXu
- Why Spark
- DataXu Processing Flows
    - Benchmarks
    - Tips & Lessons Learned
    - Demo
- DataXu Science
    - Benchmarks
    - Tips and Lessons Learned
    - Demo

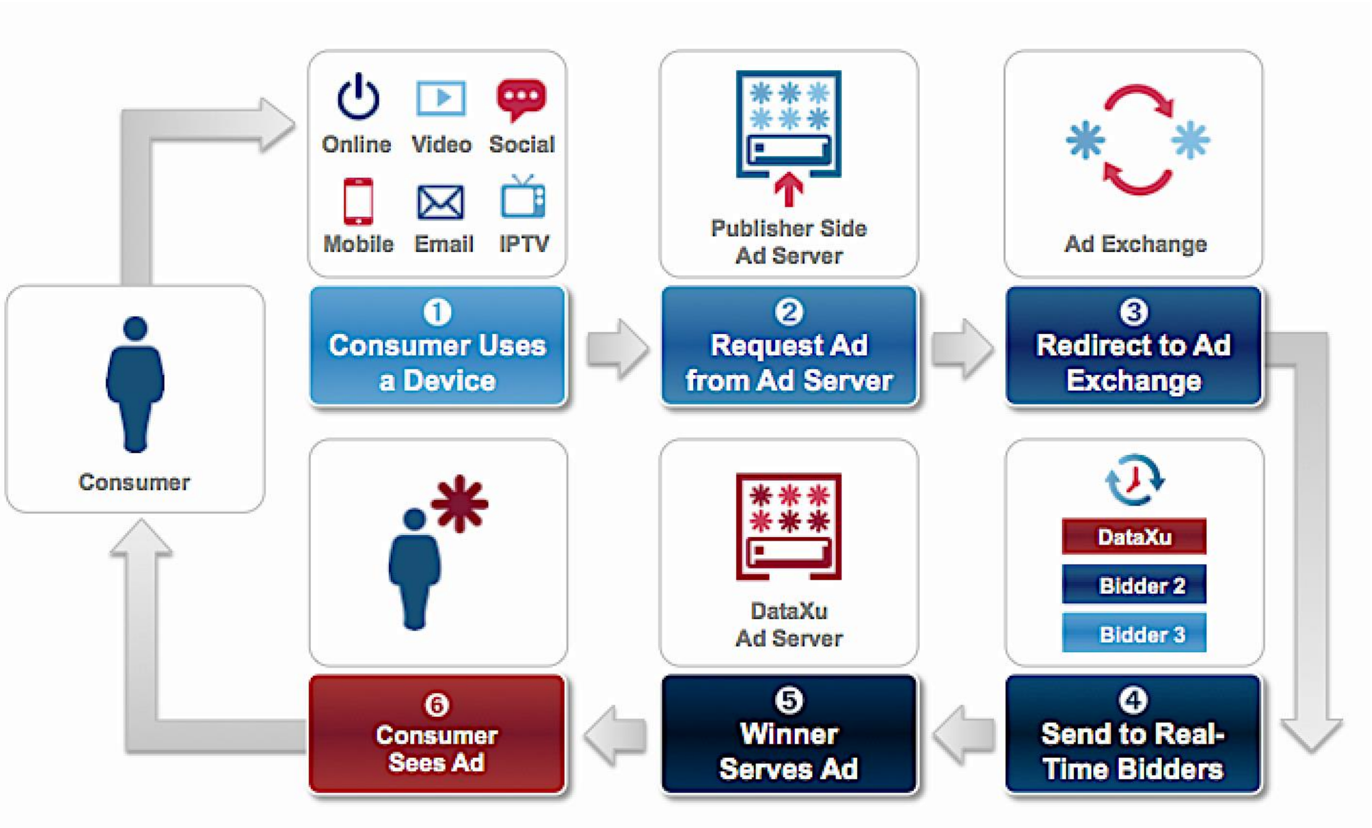
DataXu

- **Who**
  - Spun out of MIT Labs
  - A petabyte scale digital marketing platform
  - One of the fastest growing companies in Inc. 5000
- **What**
  - Help world's most valuable brands understand and engage with their consumer
  - Maximize ROI

Quick Statistics

- 2M+ bid requests per second
- Billions of impressions, Petabytes of data
- ~10ms round trip response time
- 180+TB logs per day
- 2PB data analyzed
- 3000+ servers powering the platform
- 13 regions, 24x7

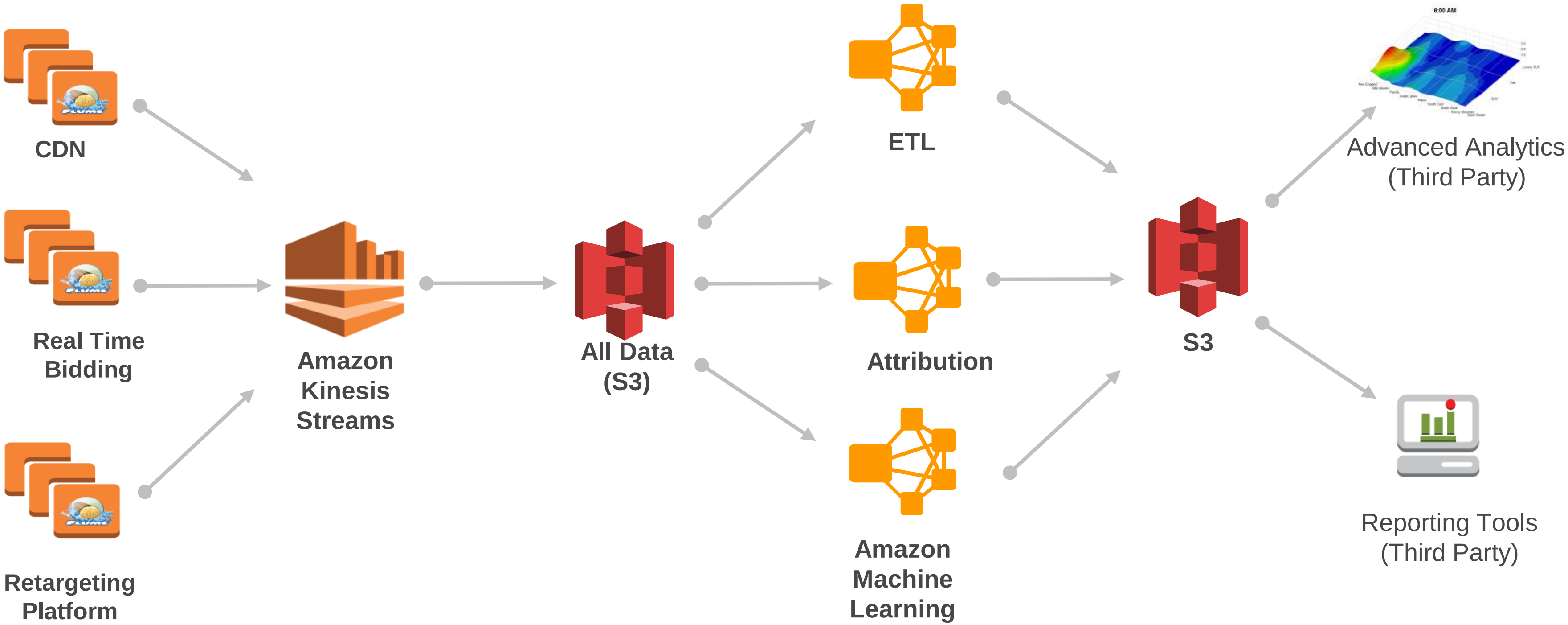
Real Time Bidding



Why Spark?

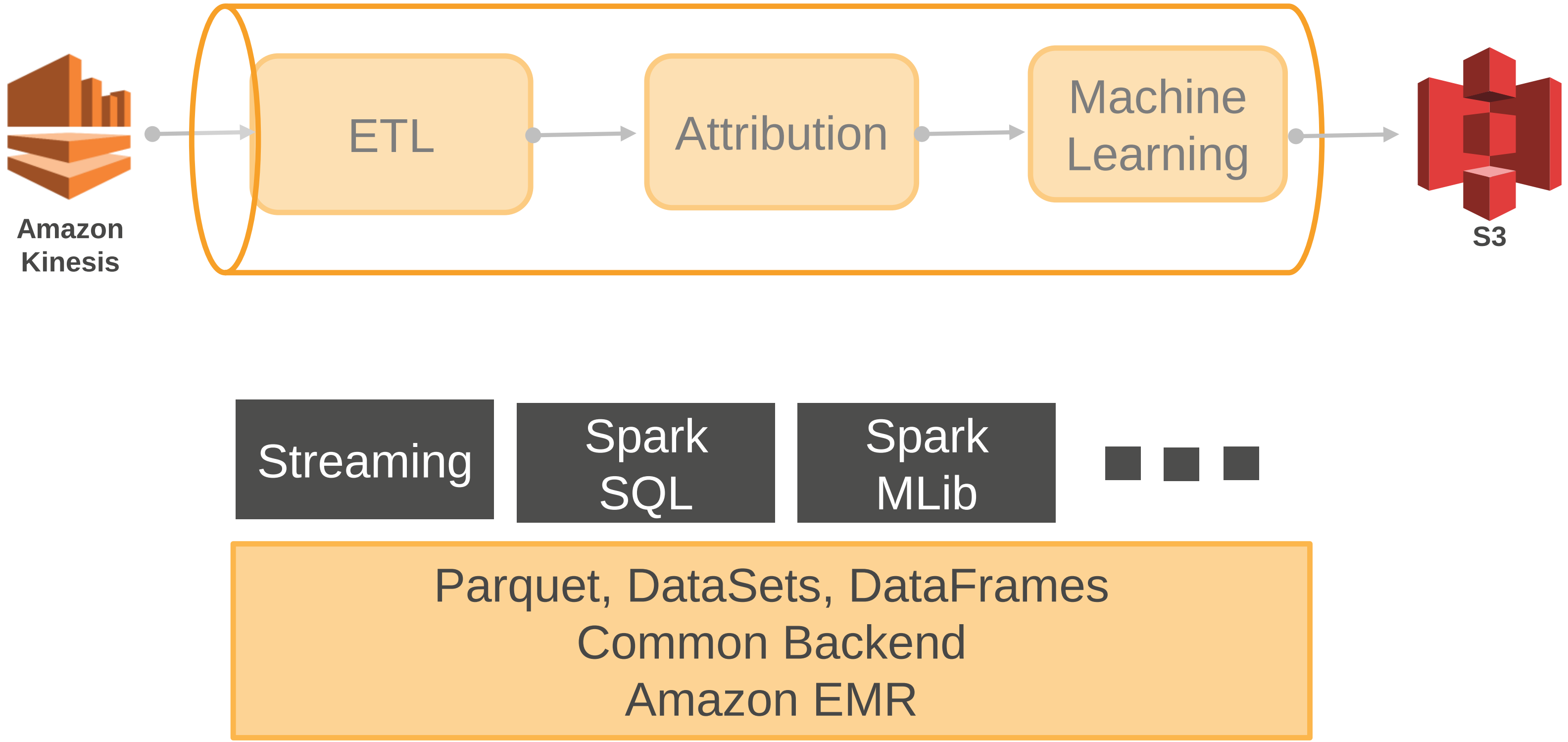
- Performance
  - Speed of execution: Spark sorted 100 TB of data (1 trillion records) 3X faster using 10X fewer machines than traditional MapReduce
  - Massively parallel
  - In-memory vs disk IO
  - Partition aware shuffles
- Speaks your language – Java, Scala, Python, R
- Streaming Use Cases – Streaming API
- Spark on EMR
  - EMR as Common Foundation – mature platform
    - Elasticity, Security, Spot Instances, Decoupled Storage and Compute
  - Low TCO: Spot Instances, Low OPEX

DataXu Flows

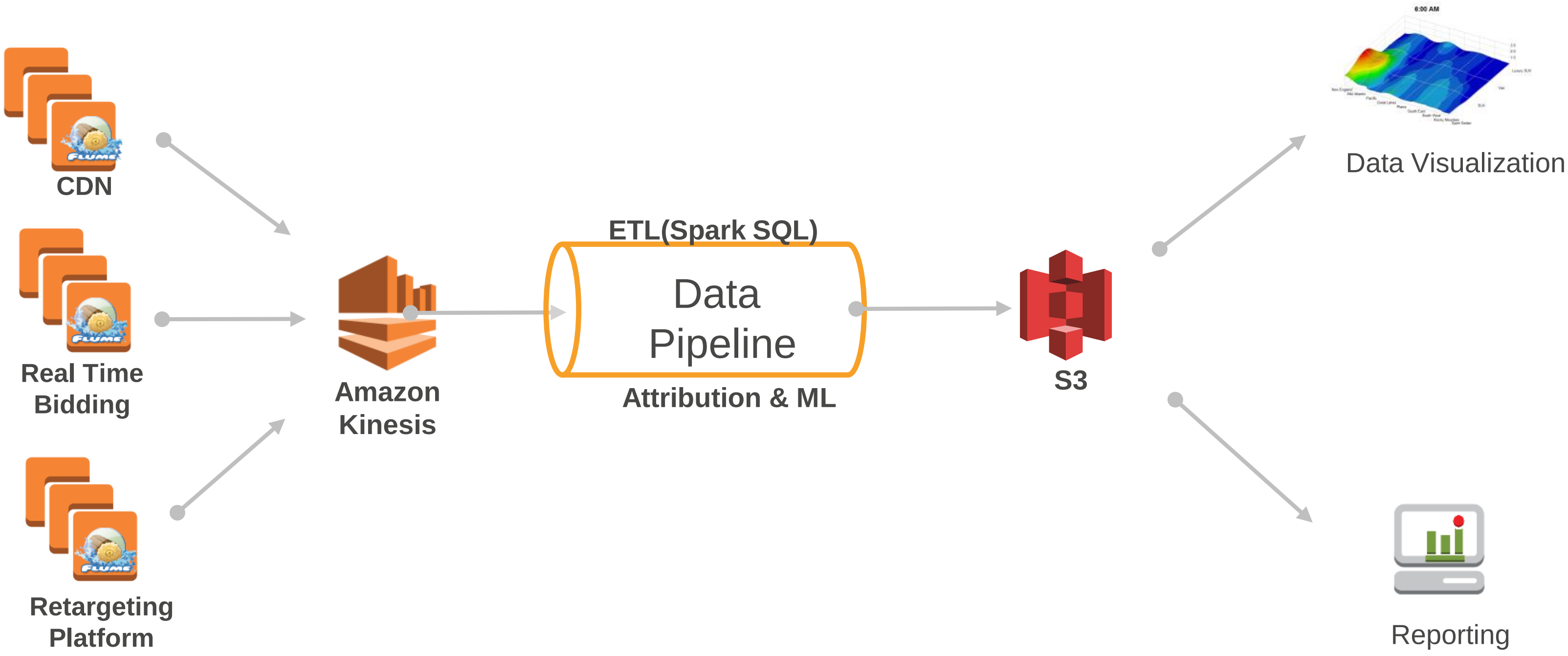


AWS Ecosystem of tools and services

DataXu Spark Pipeline

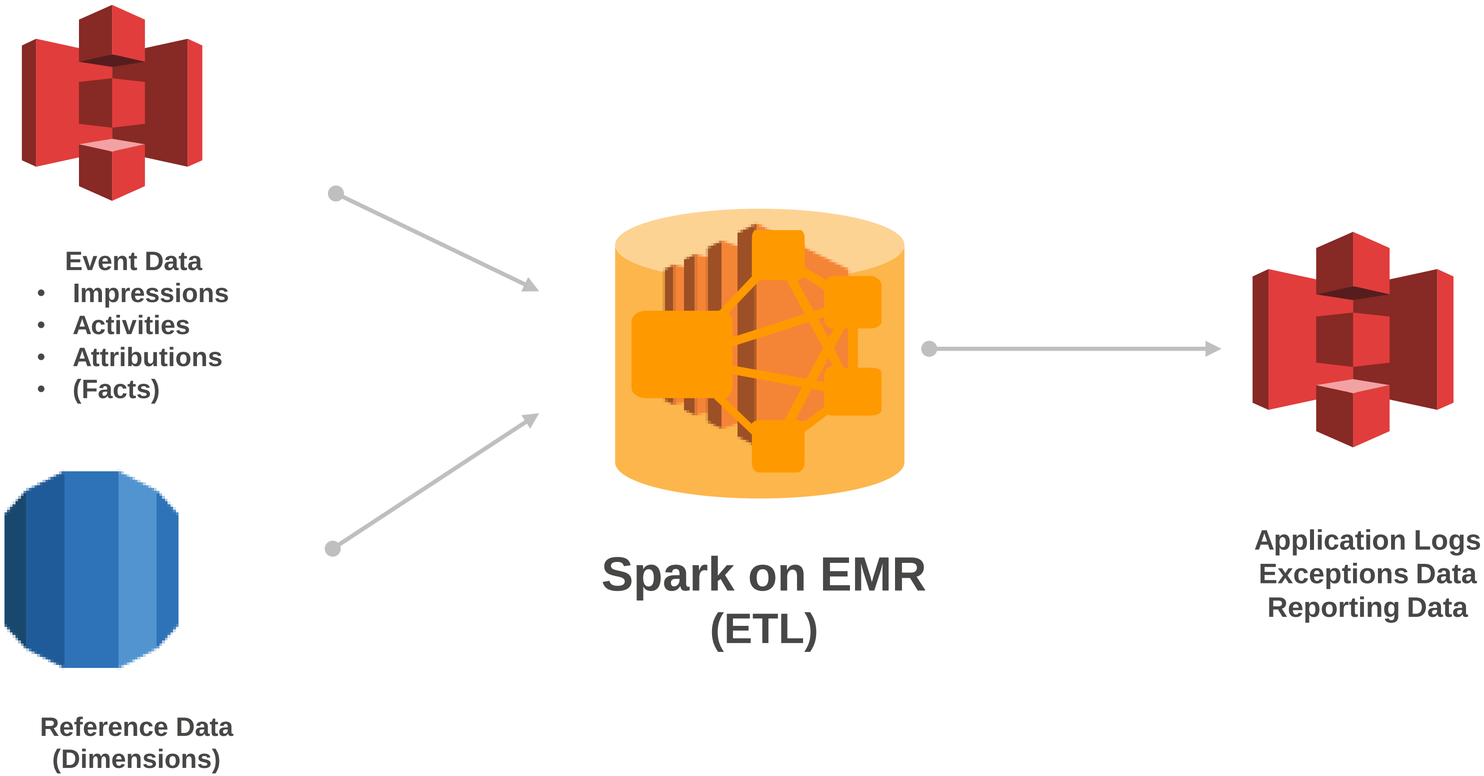


DataXu Flows – New Generation



AWS Ecosystem of tools and services

ETL Pipeline

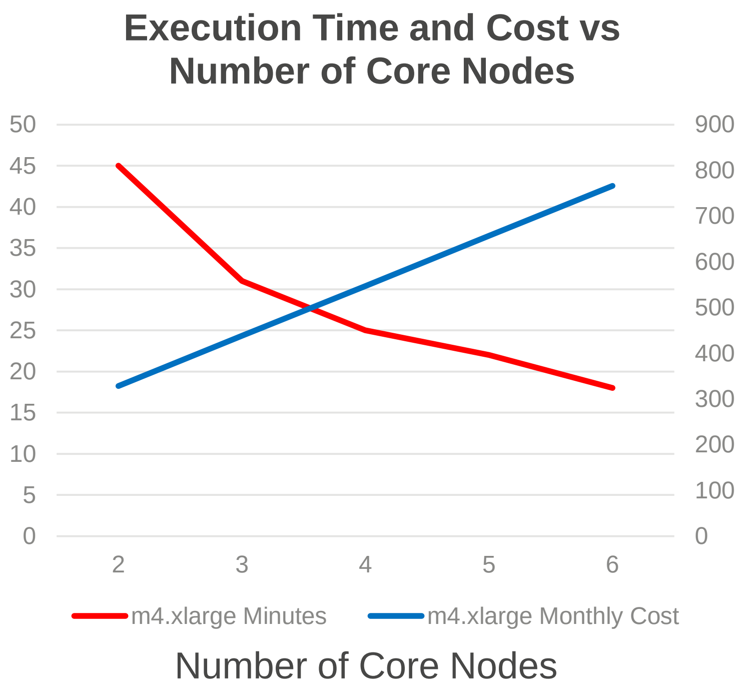


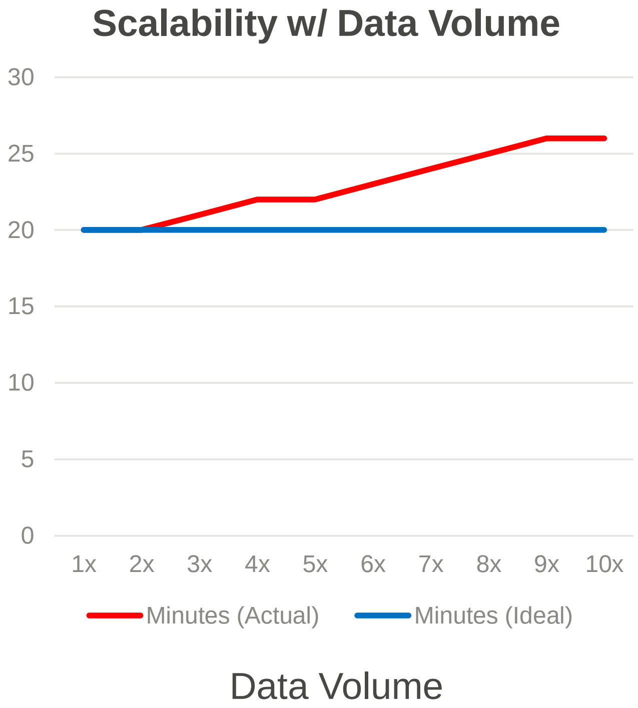
Cost/Performance Benchmarks

ETL Workload: Spark on EMR vs MPP Database

Cluster	Instance Type, Count	Execution Time (mins)	Monthly Costs
Shared MPP* (COLO)	48 x nodes (24 cores, 64GB)	20, 30, 54	$20,000 (Excluding SW license)
Single-node MPP (AWS)	1 i2.8xlarge	~40	$4,992 (Excluding SW license)
Spark on EMR (AWS)	7 m3.xlarge 7 m4.xlarge	20, 20, 20 18, 18, 18	$875 $766
Spark on EMR (AWS)	3 m3.xlarge 3 m4.xlarge	50, 51, 51 44, 45, 46	$375 $328

Scalability





Tips

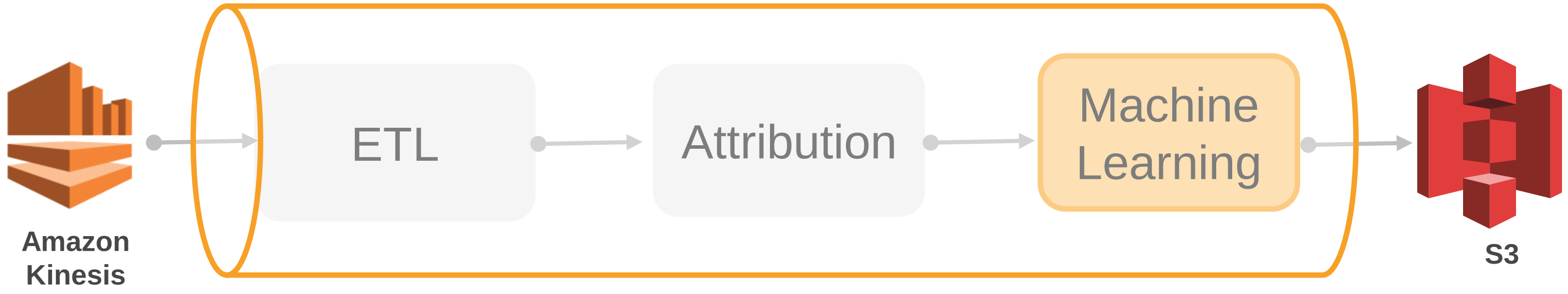
Spark SQL
BroadcastHashJoin, Easy in 2.0
UDF for Readability, Modularity and Testability, no performance penalty
Complex SQL queries may need to be materialized
Watch out for bottlenecks, like Spark driver operations

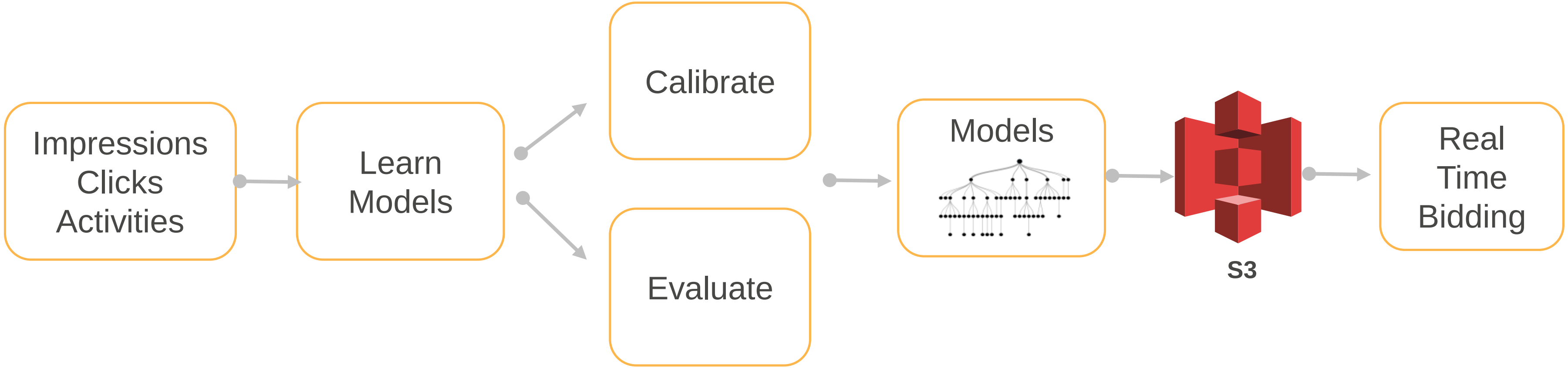
Lessons Learned

Big Picture
Re-think infrastructure (Cloud vs Host Solution)
• Think cloud native instead of fork lifting
Re-think ETL process (Spark vs MPP database vs Hadoop MR)
• Spark Core and SQL are rock solid, MR as fallback • Streamline processing
Re-think in-memory processing
• Minimize incoming data footprints • Control unnecessary joins
EMR is an excellent common backend
• Default setting just works • Consistent performance

Demo

DataXu Machine Learning





Why is this hard?

Huge Scale	• 2 Petabytes Processed Daily • 2 Million Bid Decisions Per Second • Runs 24 X 7 on 5 Continents • Thousands of ML Models Trained per Day
Unattended Operation	• Model training and deployment runs automatically every day
Changing Industry	• Need ability to adapt quickly to new customer requirements

Benchmark – Training Time

- Training times are linear using Spark AWS

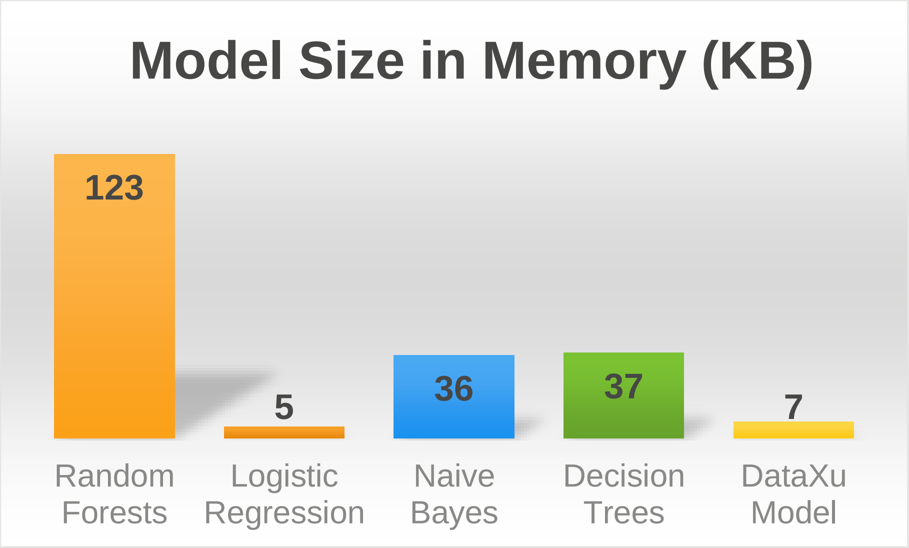
- Three different algorithms tested at scale
  - Decision Tree
  - Random Forest
  - Logistic Regression

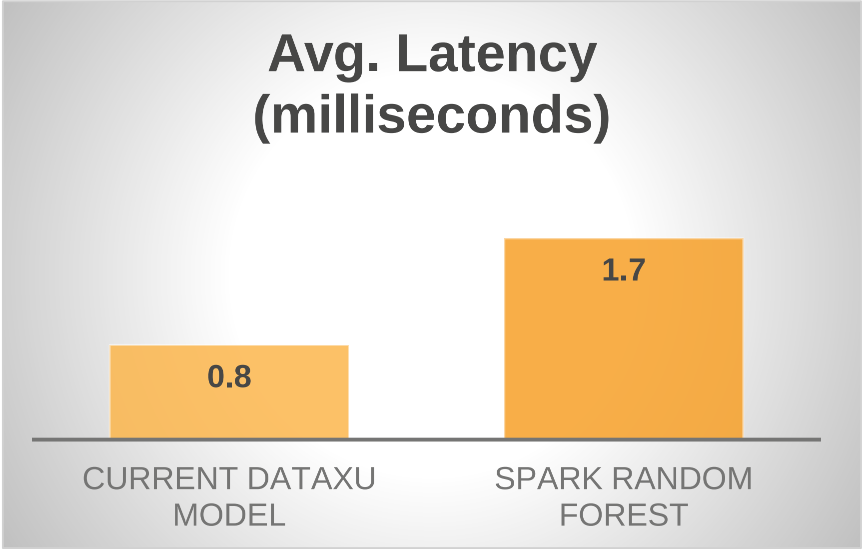
- Linear Scalability is important

For training/evaluating we used a 6 node r3.2xlarge cluster - 150 GB Memory, 20 Cores.

Benchmarks – Bidding time

- Out-of-the-box Spark performance is impressive
  - Latency
  - Model Size
- DataXu models are faster and smaller but they were optimized for years





Takeaways

Big Picture
One common big data platform
Spark on Amazon EMR works well out of the box
Very little code to train and evaluate models
Automated & unattended ML at scale

Thank you!

jonfritz@amazon.com
aws.amazon.com/emr/
aws.amazon.com/blogs/big-data/

ykosuru@dataxu.com
djiang@dataxu.com
smengle@dataxu.com

We are hiring!
www.dataxu.com/careers

Remember to complete your evaluations!