A Retrospective on AMPLab and the Berkeley Data Analytics Stack

Michael Franklin

Sept 24, 2016
Symposium on Frontiers in Big Data
UIUC

A Data Management Inflection Point

- Massively scalable processing and storage
- Pay-as-you-go processing and storage
- Flexible schema on read vs. schema on write
- Easier integration of search, query and analysis
- Variety of languages for interface/interaction
- Open source ecosystem driving

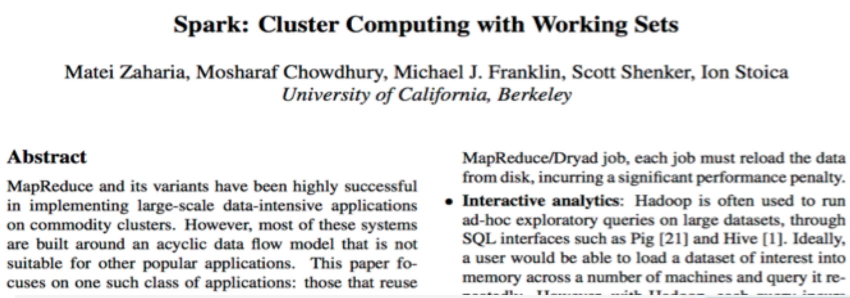
i.e., “Not your grandfather's Relational Database Management Sy

AMPLab in Context

2006-2010
Autonomic Computing & Cloud

2011-2016
Big Data Analytics

Usenix HotCloud Workshop 2010



Spark: Cluster Computing with Working Sets

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley

Abstract

MapReduce and its variants have been highly successful in implementing large-scale data-intensive applications on commodity clusters. However, most of these systems are built around an acyclic data flow model that is not suitable for other popular applications. This paper focuses on one such class of applications: those that reuse

MapReduce/Dryad job, each job must reload the data from disk, incurring a significant performance penalty.

- **Interactive analytics**: Hadoop is often used to run ad-hoc exploratory queries on large datasets, through SQL interfaces such as Pig [21] and Hive [1]. Ideally, a user would be able to load a dataset of interest into memory across a number of machines and query it re-

…

Spark Meetups (Feb 2013)

Group	Members	Interested	City	Country
1	538	170	1	1

spark.meetup.com

datanami
• BIG DATA • BIG ANALYTICS • BIG INSIGHTS •

November 4, 2015

Skip the Ph.D and Learn Spark, Data Science Salary Survey Says

Alex Woodie

Prospective data scientists can boost their salary more by learning Apache Spark and its tied-at-the-hip language Scala than obtaining a Ph.D., a recent data science survey by O'Reilly suggests.

CXO

Apache Spark rises to become most active open source project in big data

Adoption interest in Spark has topped MapReduce, says a new survey. What's supporting interest is the need for speed, boosting agility, and revenues.

By Brian Taylor | February 8, 2016, 12:11 PM PST

Apache Spark Meetups (Sept 2016)

526 groups with 245,287 **members**

spark.meetup.com

AMPLab: A Public/Private Partnership

Launched 2011; ~90 Students, Postdocs, and Faculty

from: Systems, ML, Database, Networks, Security, Apps

Wrapping up this year (transition to new lab)

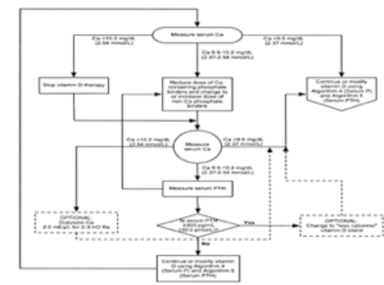
National Science Foundation Expedition Award

Darpa XData; DoE/Lawrence Berkeley National Lab

40 Industry Sponsors including:

AMP: 3 Key Resources

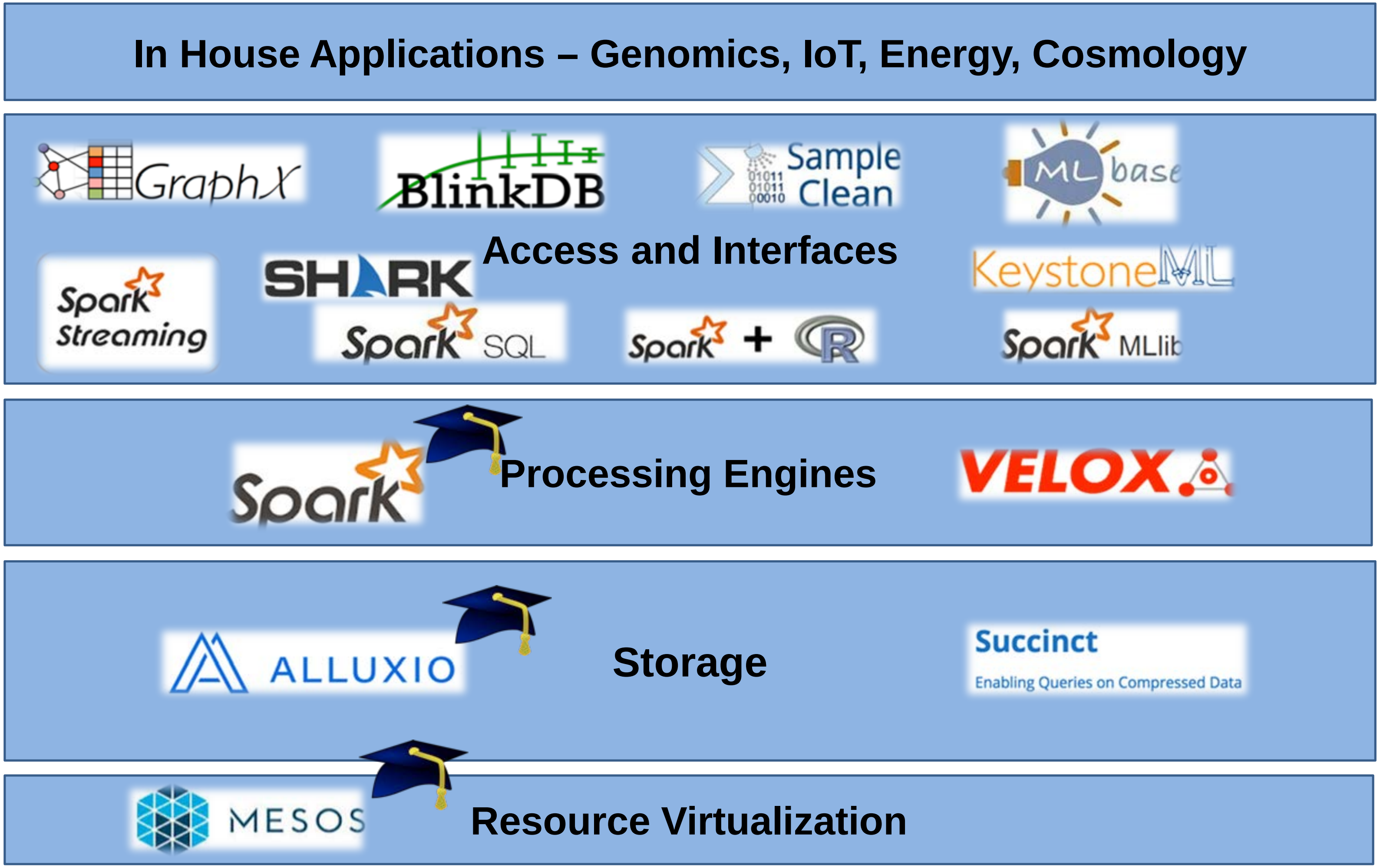
Algorithms
- Machine Learning, Statistical Methods
- Prediction, Business Intelligence



Machines
- Clusters and Clouds
- Warehouse Scale Computing

People
- Crowdsourcing, Human Computation
- Data Scientists, Analysts

Berkeley Data Analytics Stack

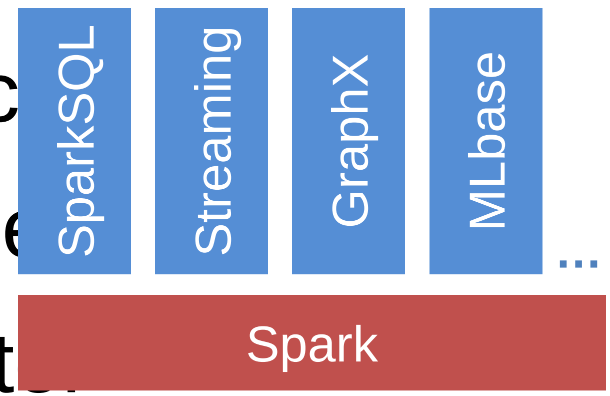


AMPLab Unification Strategy

Specializing MapReduce leads to stovepiped systems

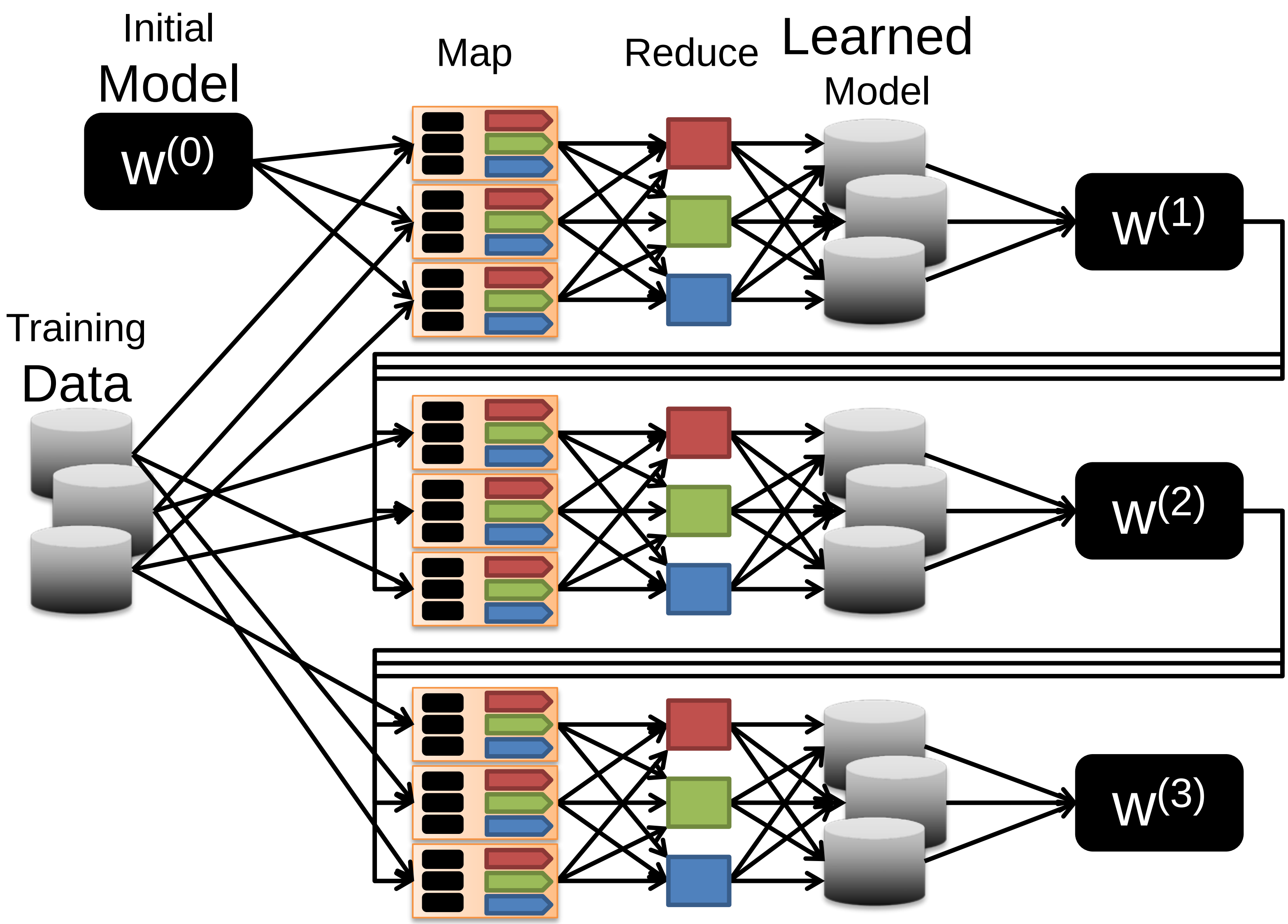
Instead, **generalize** MapReduc

1. Richer Programming Mode
   - ➔Fewer Systems to Mast
2. Data Sharing
   - ➔Less Data Movement

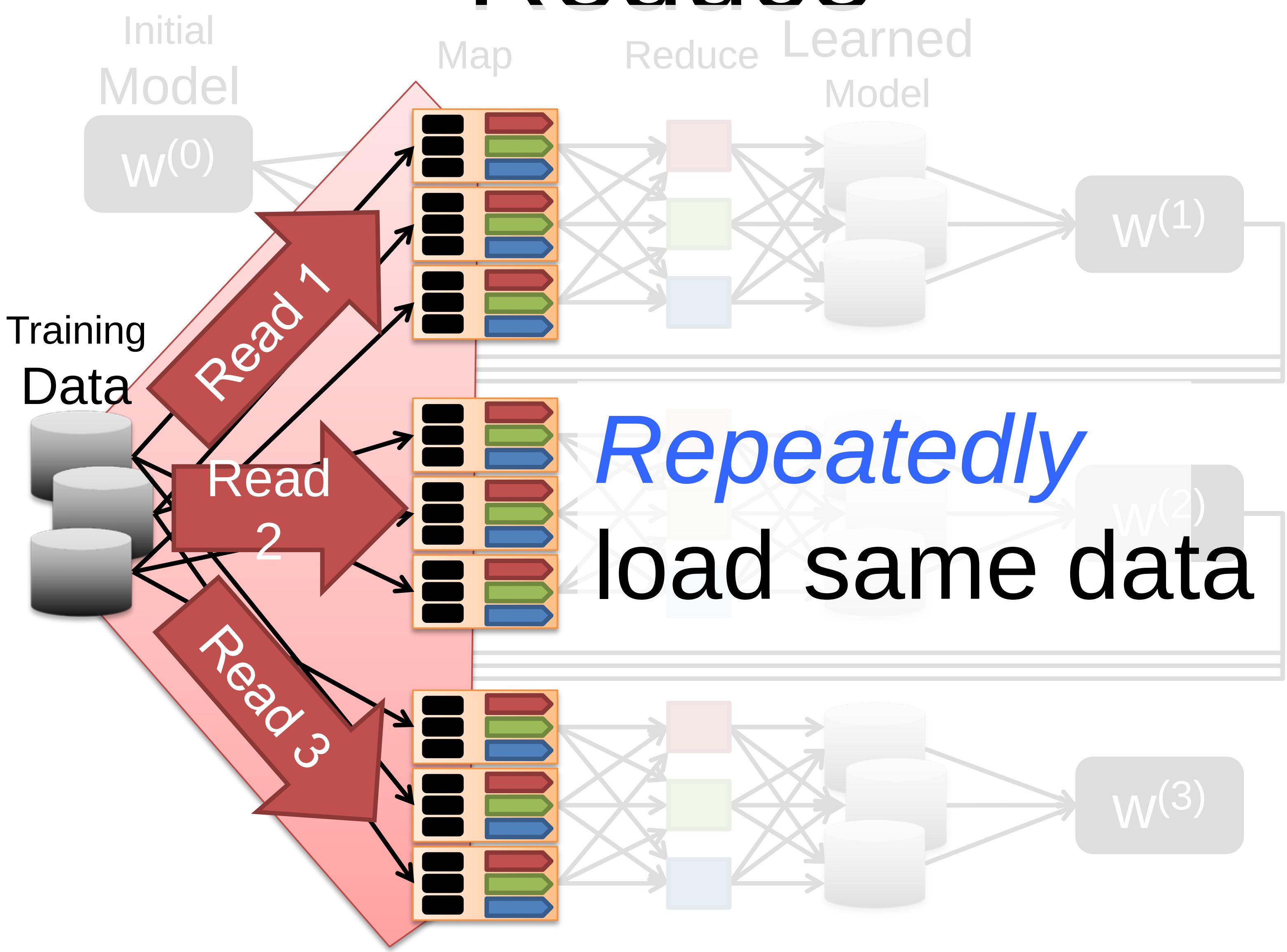


For improved productivity and performance

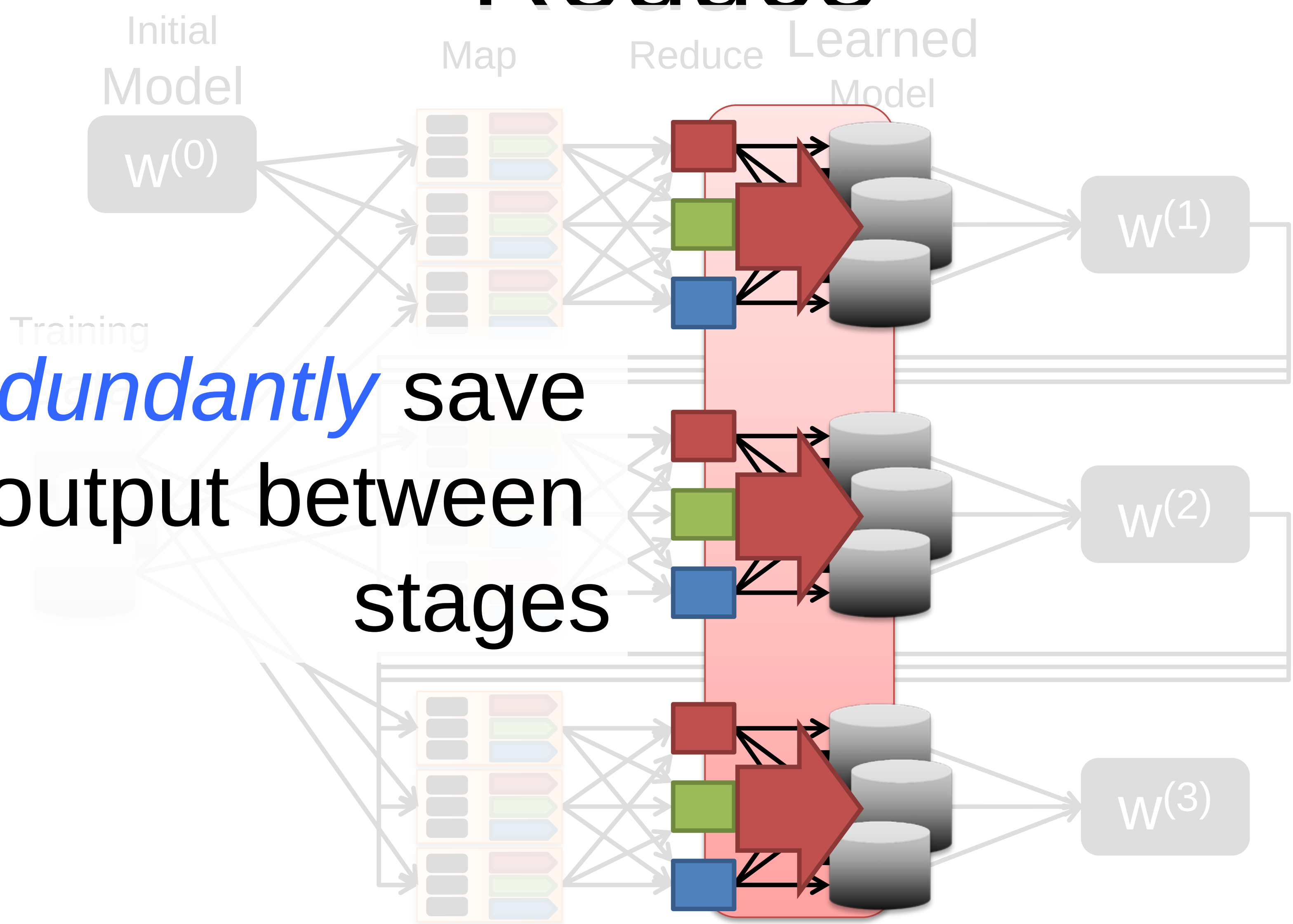
Iteration in Map-Reduce



Cost of Iteration in Map-Reduce

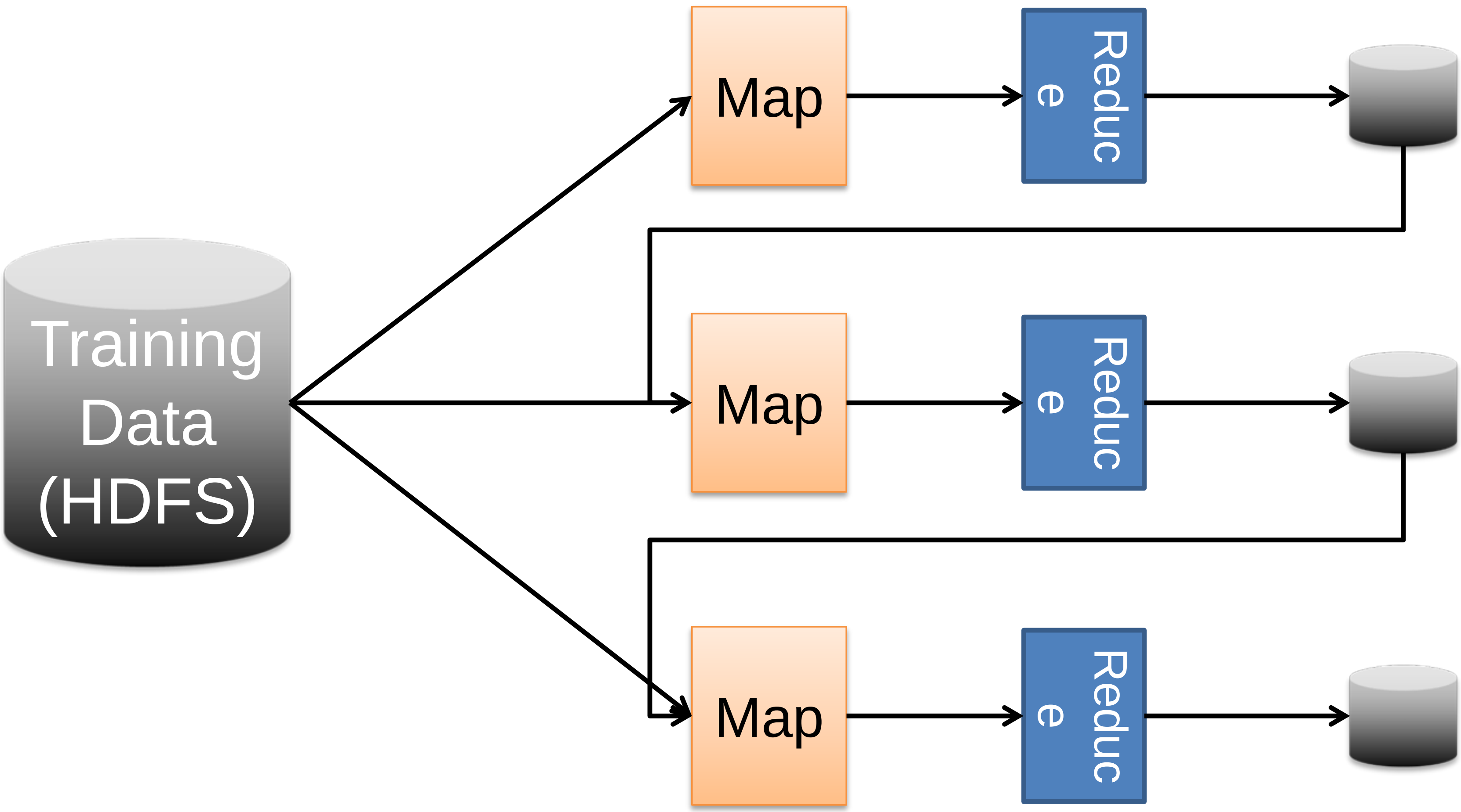


Cost of Iteration in Map-Reduce

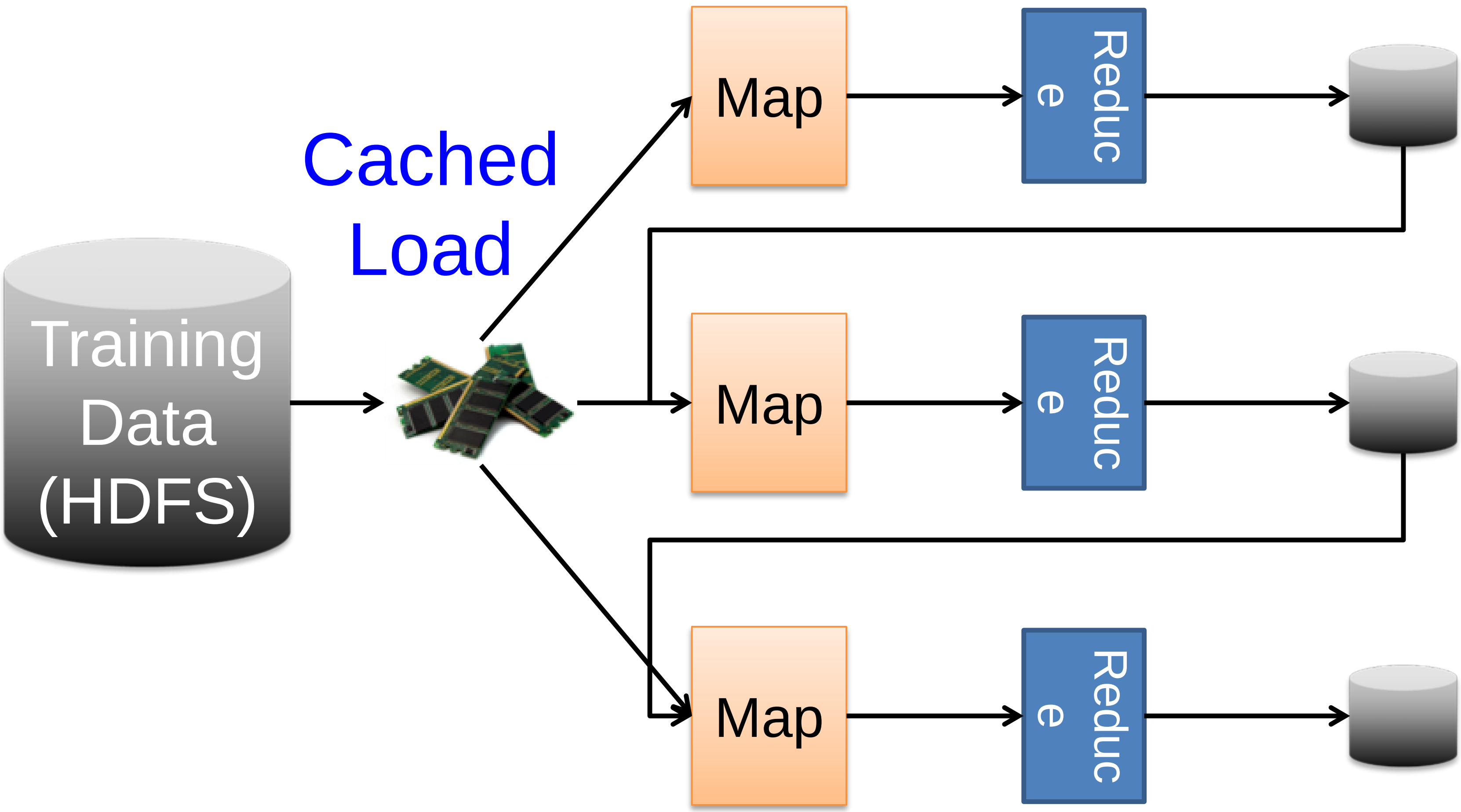


Redundantly save output between stages

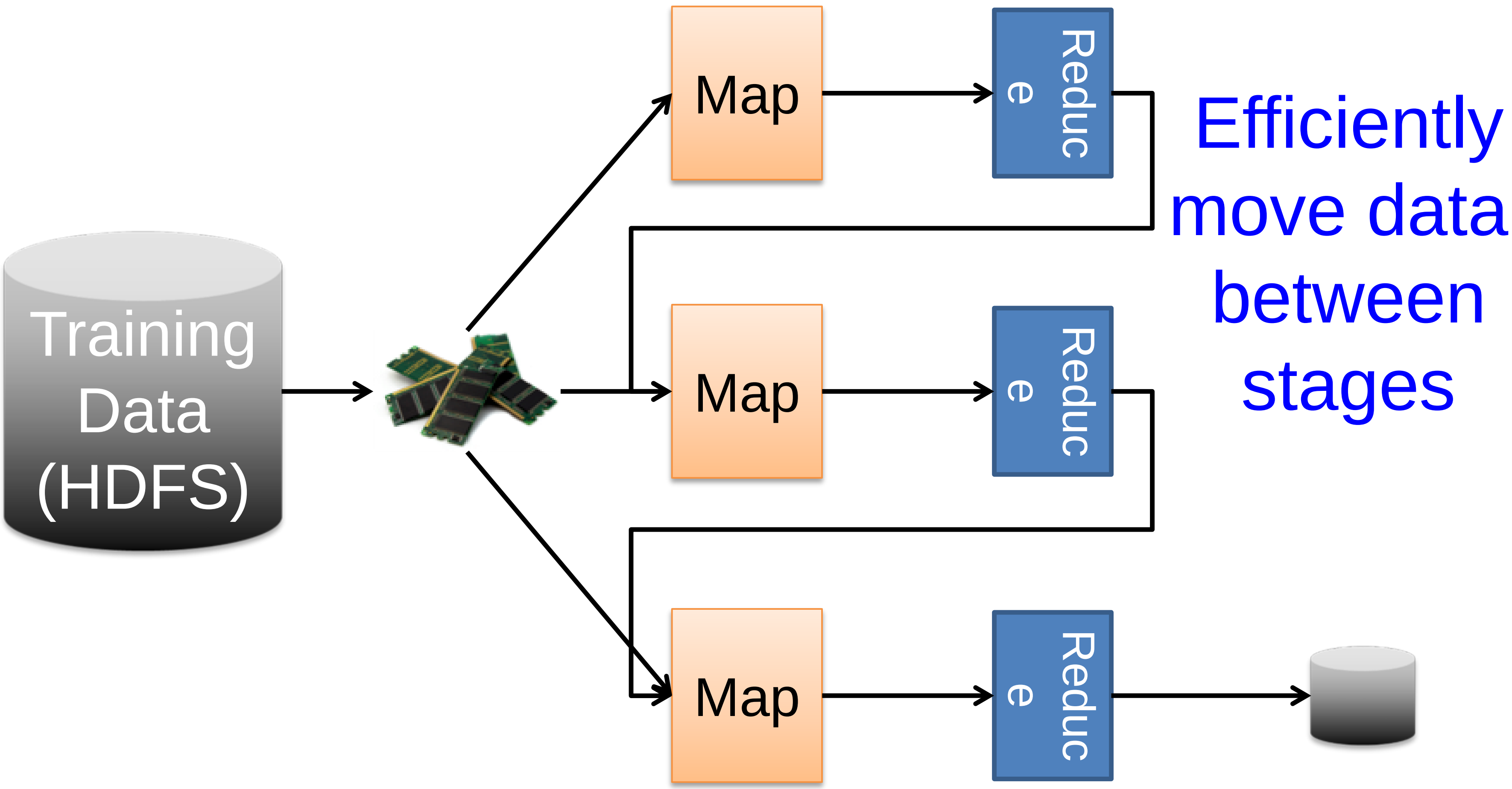
Dataflow View



Memory Opt. Dataflow



Memory Opt. Dataflow View



Spark:**10-100×** faster than Hadoop MapReduce

Resilient Distributed Datasets (RDDs)

API: coarse-grained *transformations* (map, group-by, join, sort, filter, sample,…) on immutable collections

Resilient Distributed Datasets (RDDs)

» Collections of objects that can be stored in memory or disk across a cluster
» Built via parallel transformations (map, filter, …)
» Automatically rebuilt on failure

Rich enough to capture many models:

» **Data flow models**: MapReduce, Dryad, SQL, …
» **Specialized models**: Pregel, Hama, …

M. Zaharia, et al, Resilient Distributed Datasets: A fault-tolerant abstraction for in-memory cluster computing, NSDI 2012.

Abstraction: *Dataflow Operators*

`map`	`reduce`	`sample`
`filter`	`count`	`take`
`groupBy`	`fold`	`first`
`sort`	`reduceByKey`	`partitionBy`
`union`	`groupByKey`	`mapWith`
`join`	`cogroup`	`pipe`
`leftOuterJoin`	`cross`	`save`
`rightOuterJoin`	`zip`	`...`

Fault Tolerance with RDDs

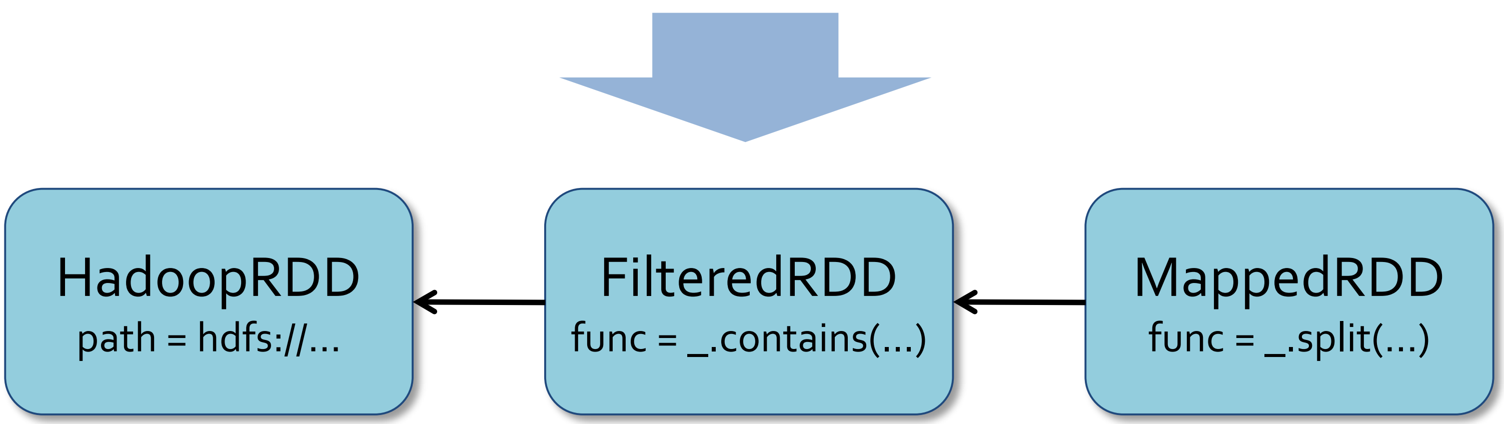
RDDs track the series of transformations used to build them (their *lineage*)

» Log one operation to apply to many elements

» No cost if nothing fails

Enables per-node recomputation of lost data

```
messages = textFile(...).filter(_.contains("error"))
                        .map(_.split('\t')(2))
```



Spark SQL – Deeper Integration

Replaces "Shark" – Spark's implementation of Hive

- Hive dependencies were cumbersome
- Missed integration opportunities

Spark SQL has two main additions

1) Tighter Spark integration, including Data Frames
2) Catalyst Extensible Query Optimizer

First release May 2014; in production use

- e.g., large Internet co has deployed on 8000 nodes; >100PB with typical queries covering 10's of TB

R. Xin, J. Rosen, M. Zaharia, M. Franklin, S. Shenker, I. Stoica, "Shark: SQL and Rich Analytics at Scale, SIGMOD 2013.

M. Armbrust, R. Xin et al., "Spark SQL: Relational Data Processing in Spark", SIGMOD 2015.

DataFrames

```
employees
.join(dept, employees("deptId") === dept("id"))
.where(employees("gender") === "female")
.groupBy(dept("id"), dept("name"))
.agg(count("name"))
```

Notes:

1) Some people think this is an improvement over SQL ☺
2) Spark 2.0 integrates “Datasets”, which are effectively typed dataframes

Catalyst Optimizer

Extensibility via Optimization Rules written in Scala

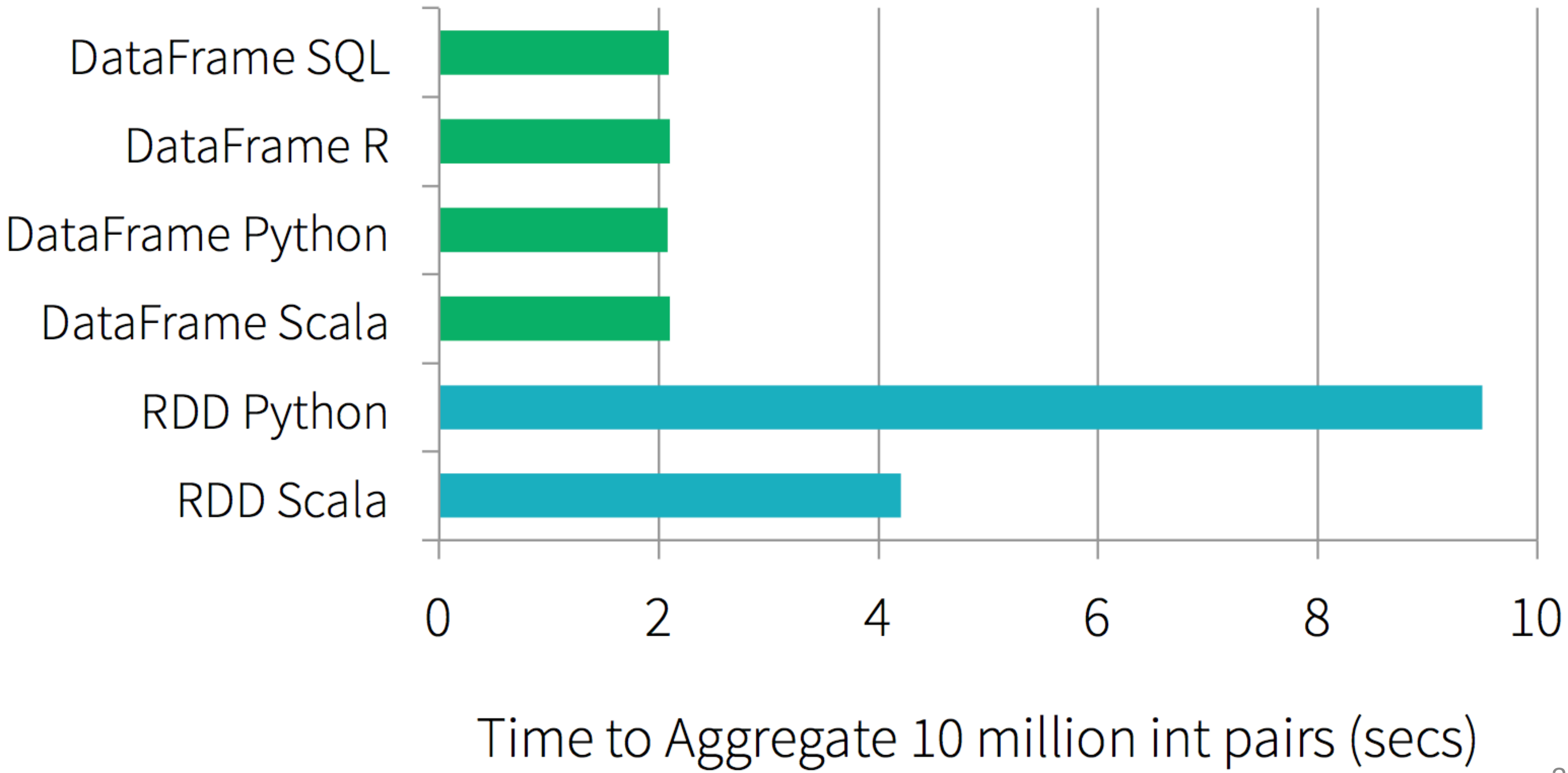
Code generation for inner-loops

Extension Points:

Data Sources: e.g., CSV, Avro, Parquet, JDBC, …

- via TableScan (all cols), PrunedScan (project), FilteredPrunedScan(push advisory selects and projects) CatalystScan (push advisory full Catalyst expression trees)

An interesting thing about SparkSQL Performance



Don’t Forget About Approximation

BDAS Uses Approximation in two main ways:

1) BlinkDB (Agarwal et al. EuroSys 13)
   - Run queries on a sample of the data
   - Returns answer and confidence interval
   - Can adjust time vs confidence

2) Sample Clean (Wang et al. SIGMOD 14)
   - Clean a sample of the data rather than whole data set
   - Run query on sample (get error bars) OR
   - Run query on dirty data and correct the answer

SQL + ML + Streaming

```
// Load historical data as an RDD using Spark SQL
val trainingData = sql(
  "SELECT location, language FROM old_tweets")

// Train a K-means model using MLlib
val model = new KMeans()
  .setFeaturesCol("location")
  .setPredictionCol("language")
  .fit(trainingData)

// Apply the model to new tweets in a stream
TwitterUtils.createStream(...)
  .map(tweet => model.predict(tweet.location))
```

"Apache Spark has made big data processing, machine learning, and advanced analytics accessible to the masses. This is awesome."

- Chris Fregly "creator of the "PANCAKE STACK", infoQ 8/29/16

Renewed Excitement Around Streaming

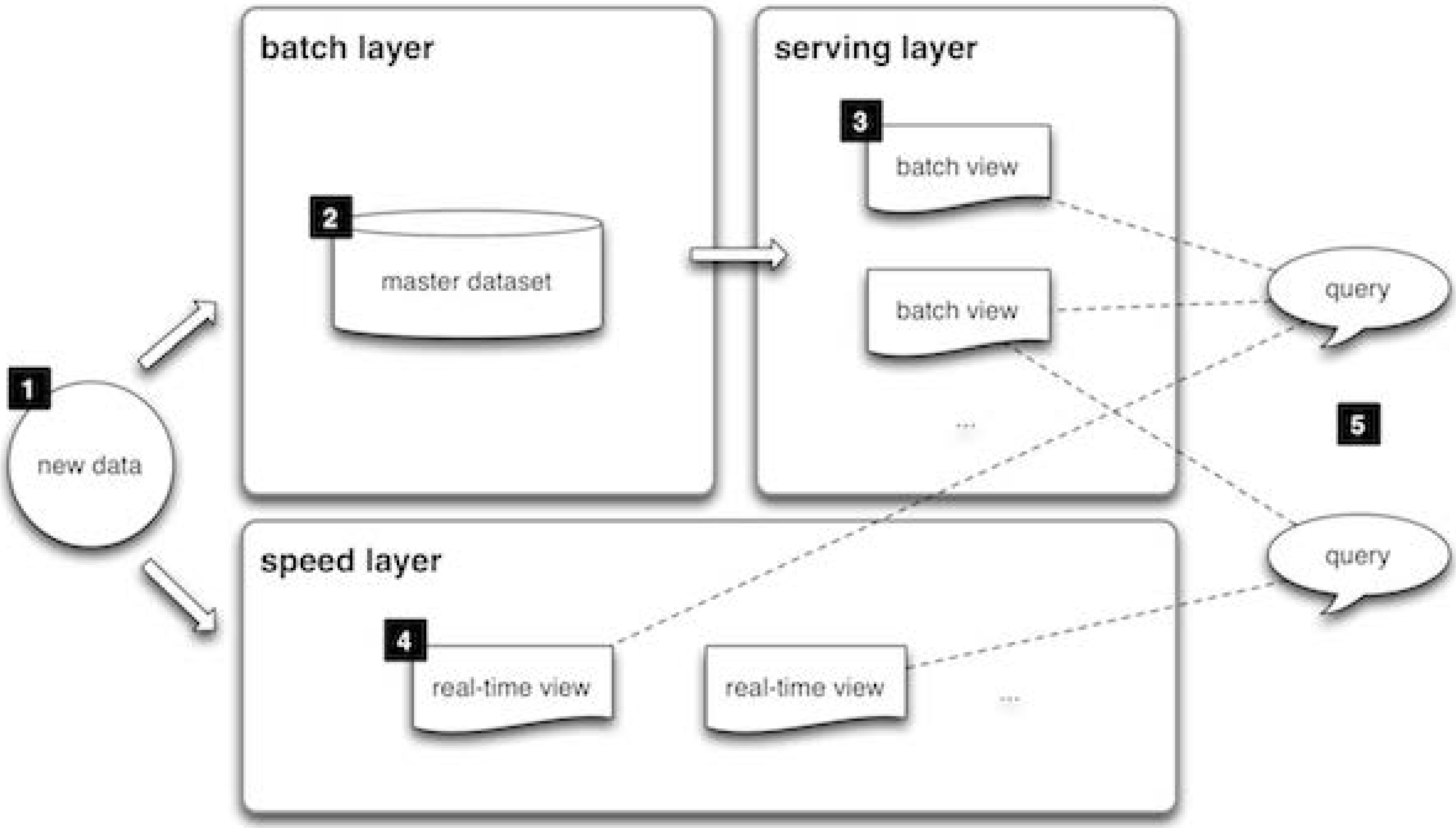
Stream Processing (esp. Open Source)

- » Spark Streaming
- » Samza
- » Storm
- » Flink Streaming
- » Google Millwheel and Cloud Dataflow
- » <YOUR FAVORITE SYSTEM HERE>

Message Transport

- » Kafka
- » Kenesis
- » Flume

Lambda Architecture: Real-Time + Batch



lambda-

Lambda: How Unified Is It?

Have to write everything twice!

Have to fix everything (maybe) twice.

Subtle differences in semantics

how much Duct Tape required?

What about Graphs, ML, SQL, etc.?

see e.g., Jay Kreps: http://radar.oreilly.com/2014/07/questioning-the-lambda-architect
and Franklin et al., CIDR 2009.

Spark Streaming

Scalable, fault-tolerant stream processing system

High-level API

joins, windows, …
often 5x less code

Fault-tolerant

Exactly-once semantics, even for stateful ops

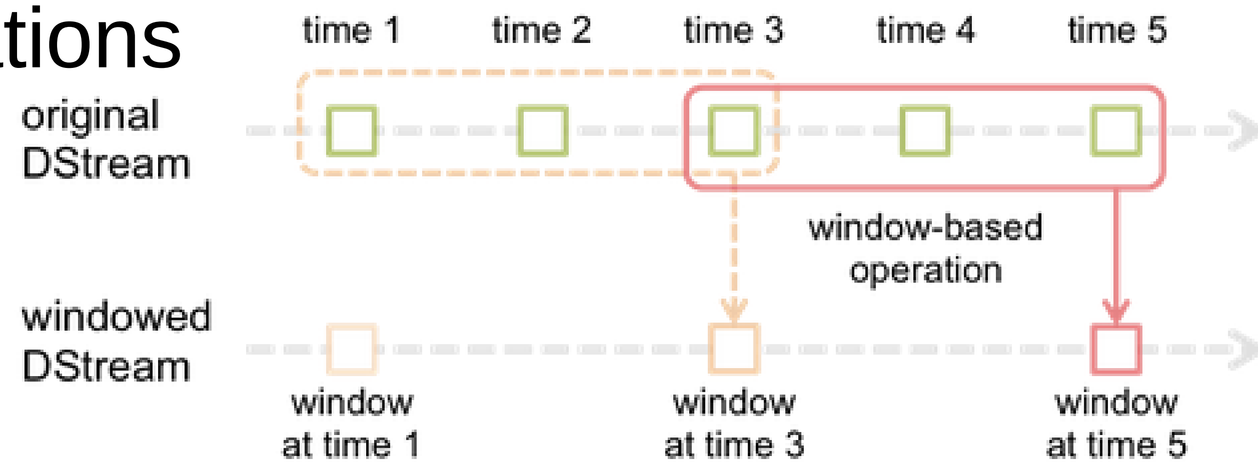
Integration

Integrate with MLlib, SQL, DataFrames, GraphX

Spark Streaming

Microbatch approach provides low latency

Additional operators provide windowed operations



M. Zaharia, et al, Discretized Streams: Fault-Tollerant Streaming Computation at Scale, SOSP 2013.

Structured Streams (Spark 2.0)

Batch Analytics

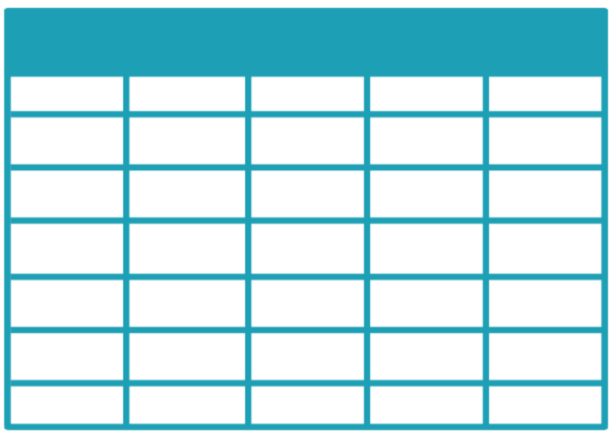
```
// Read data once from an S3 location
val inputDF = spark.read.json("s3://logs")

// Do operations using the standard DataFrame API and write to MySQL
inputDF.groupBy($"action", window($"time", "1 hour")).count()
       .write.format("jdbc")
       .save("jdbc:mysql//...")
```

Streaming Analytics

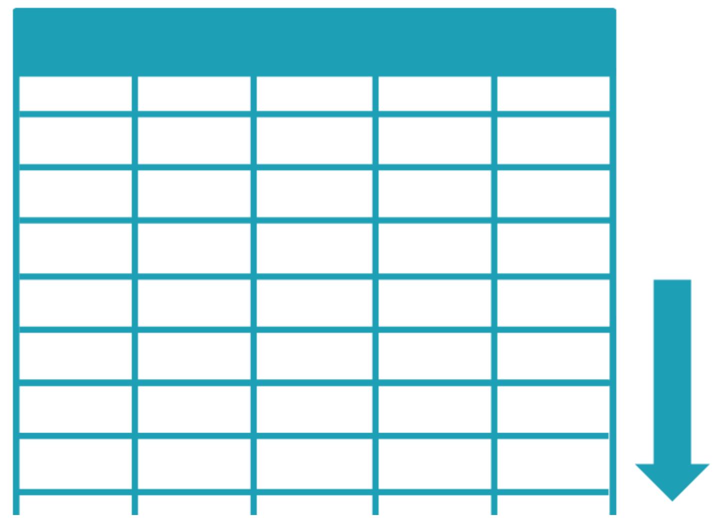
```
// Read data continuously from an S3 location
val inputDF = spark.readStream.json("s3://logs")

// Do operations using the standard DataFrame API and write to MySQL
inputDF.groupBy($"action", window($"time", "1 hour")).count()
       .writeStream.format("jdbc")
       .start("jdbc:mysql//...")
```

Conceptual View

Spark 1.3
Static DataFrames



Spark 2.0
Infinite DataFrames



Note: Spark 2.0 was done by the Apache Spark community after Spark’s “graduation” from the AMPLab

Spark Streaming - Comments

Mini-batch approach appears to be "low latency" enough for many applications.

Integration with the rest of the BDAS/Spark stack is a big deal for users

We're also adding a "**timeseries**" capability to BDAS (see AMPCamp 6 `ampcamp.berkeley.edu`)

- initially batch but streaming integration planned

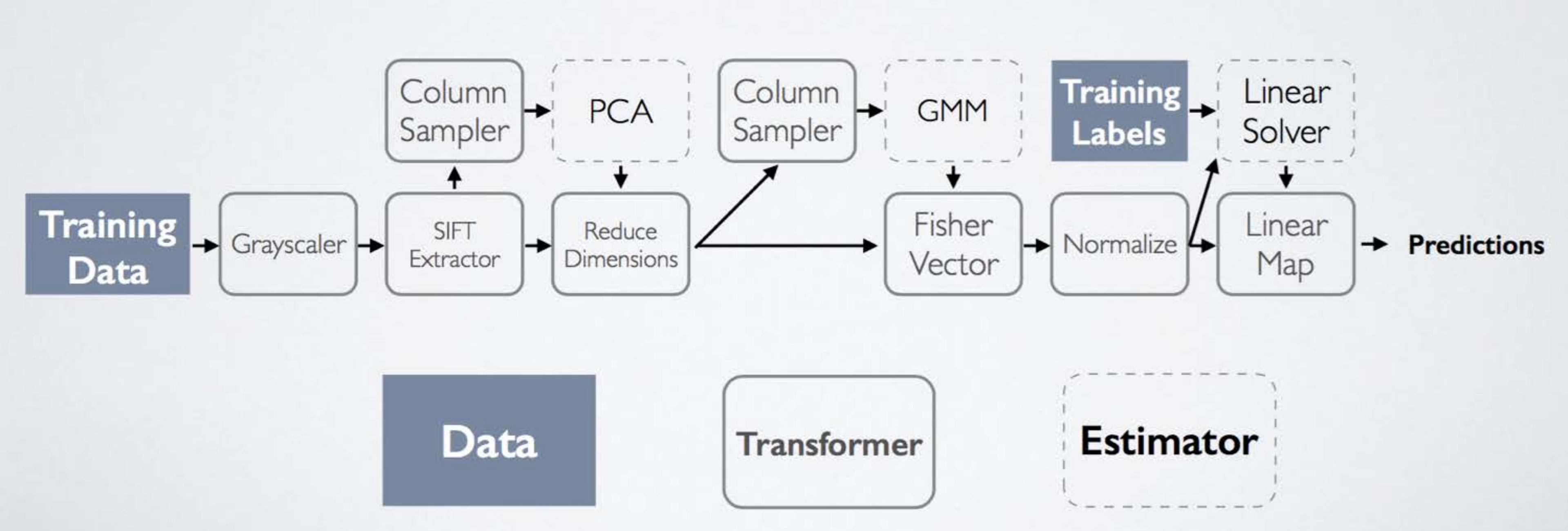
Beyond ML Operators

- Data Analytics is a complex process
- Rare to simply run a single algorithm on an existing data set
- Emerging systems support more complex workflows:
  - Spark MLPipelines
  - Google TensorFlow
  - KeystoneML (BDAS)

KeystoneML

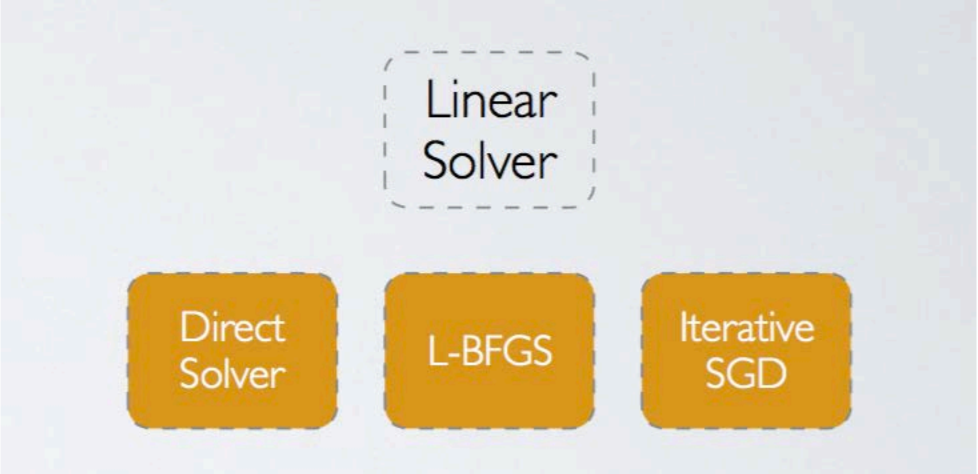
Software framework for describing complex *machine learning pipelines* built on Apache Spark.

Pipelines are specified using domain specific

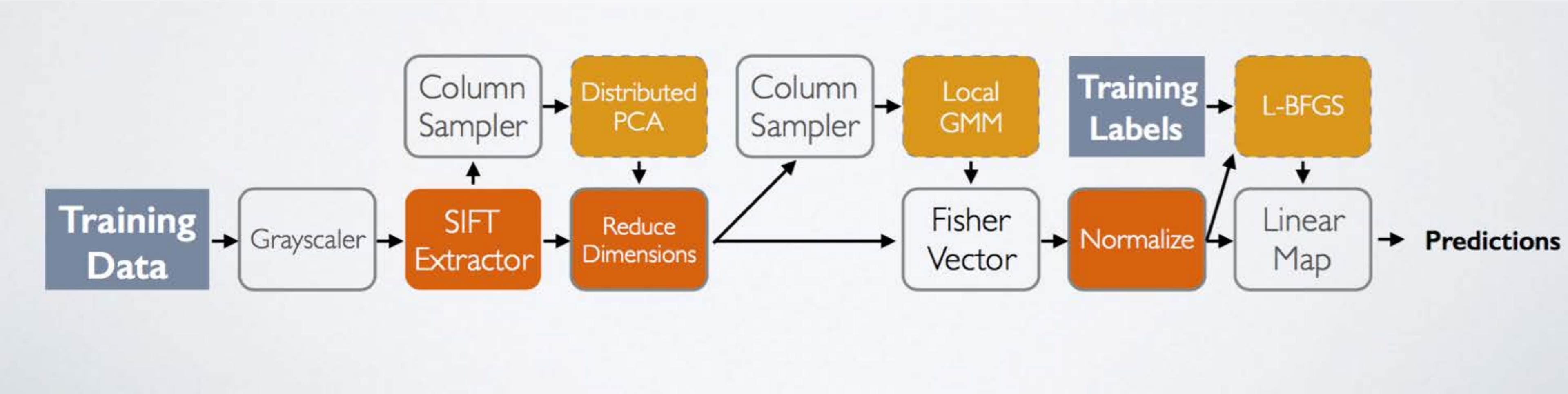


High-level API ➔ Optimizations

Automated ML operator selection



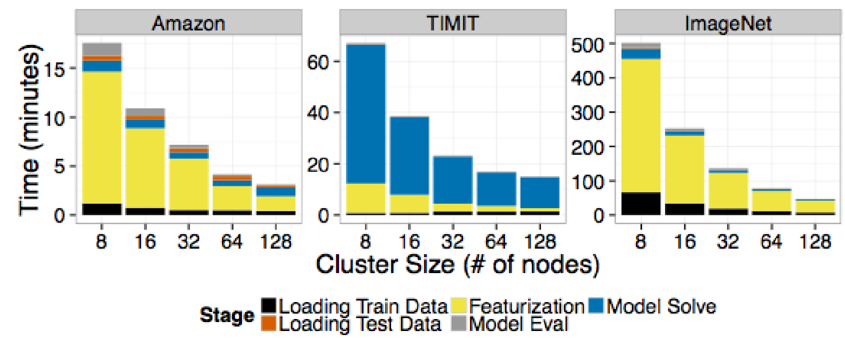
Auto-caching for iterative workloads



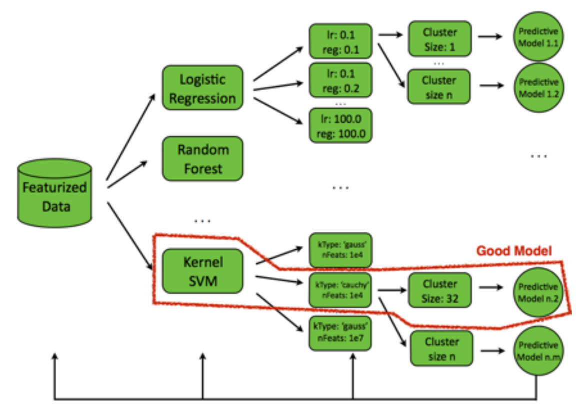
KeystoneML: Status

Current version: v0.3

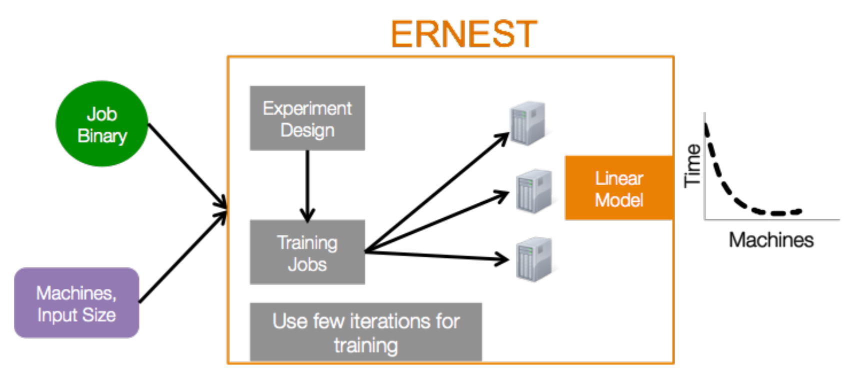
Scale-out performance on 10s of TBs of training features on 100s of machines. apps: Image Classification, Speech, Text.



First versions of node-level and whole-pipeline optimizations.



Many new high-speed, scalable operators

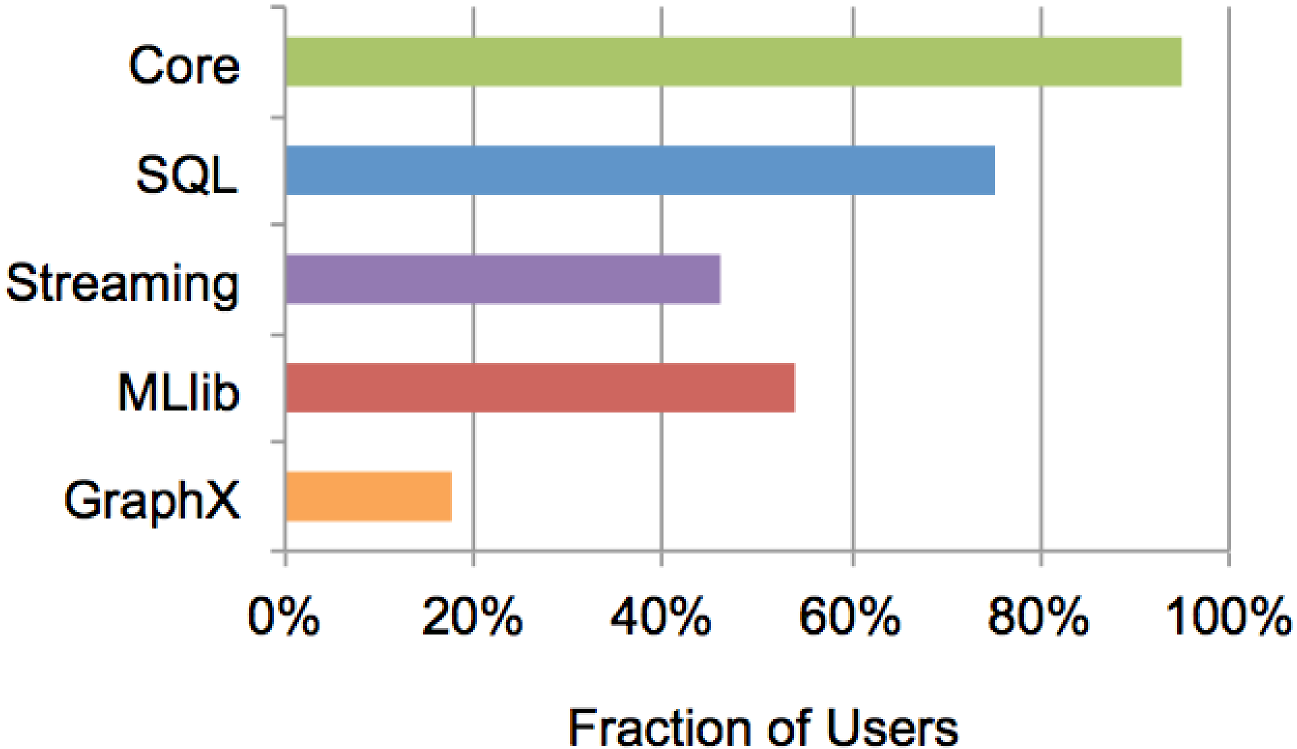


Coming soon:

» Principled, scalable hyperparameter tuning. (TuPAQ - SoCC 2015)

» Advanced cluster sizing/job

Spark User Survey 7/2015 (One Size Fits Many)

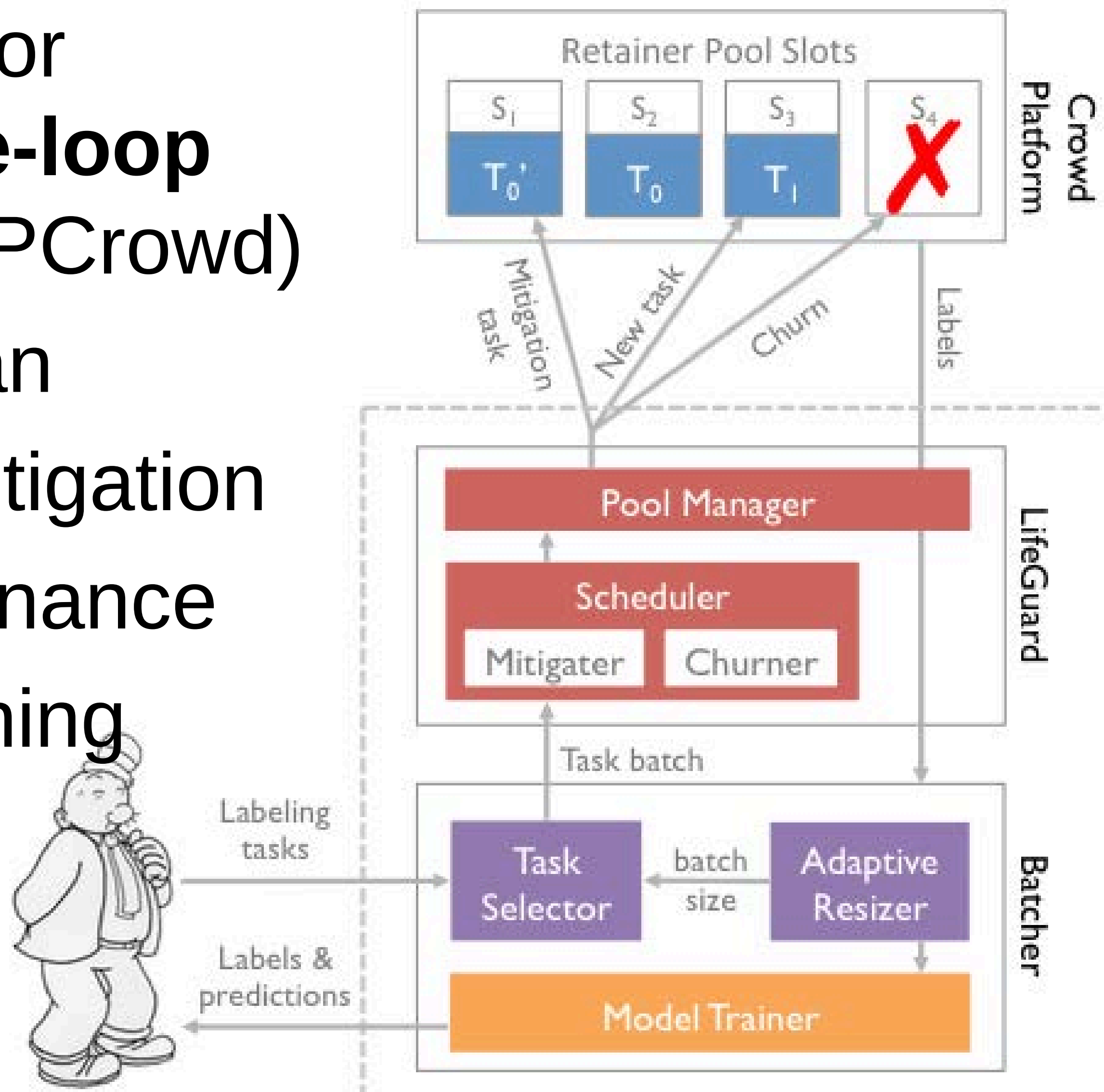


~1400 respondents; 88% Use at least 2 components; 60% at least 3; 27% at least
Source: Databricks

Integrating the "P" in AMP

Optimization for **human-in-the-loop** analtyics (AMPCrowd)

- SampleClean
- Straggler Mitigation
- Pool Maintenance
- Active Learning



Some Early Reflections (tech)

Integration vs Silos

Scala vs ???

Real time for real this time?

Deep learning

Privacy and Security

What did we learn from database technology?

Robust answers, interpretability and

The Patterson Lessons

1) Build a cross-disciplinary team
2) Sit together
3) Engage Industry and Collaborators
4) Build artifacts and get people to use them
5) Start your project with an end date

See Dave Patterson "How to Build a Bad Research Center", CACM March, 2014

Thanks and More Info

Thanks to NSF CISE Expeditions in Computing, DARPA XData,
Founding Sponsors: Amazon Web Services, Google, IBM, and SAP,
the Thomas and Stacy Siebel Foundation,
all our industrial sponsors, partners and collaborators,
and all the amazing students, staff, and faculty of the AMPLab.

amplab
UC BERKELEY

amplab.berkeley.edu