CS 294-110: Project Suggestions

September 14, 2015

Ion Stoica and Ali Ghodsi

(http://www.cs.berkeley.edu/~istoica/classes/cs294/15/)

Projects

- This is a project-oriented class
- Reading papers should be a means to a great project, not a goal in itself!
- Strongly prefer groups of two students
- Today, I’ll present some suggestions
  - But, you are free to come up with your own proposal

- Main goal: just do a great project

Projects

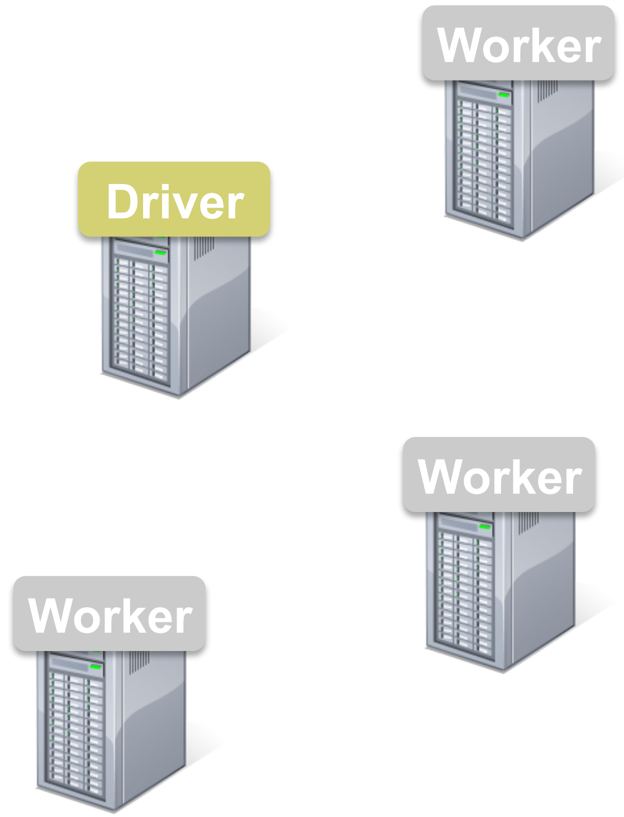
- Many projects around Spark
  - Local expertise
  - Great platform to disseminate your work
  - Short review based on log mining example to provide context

Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

Spark Example: Log Mining

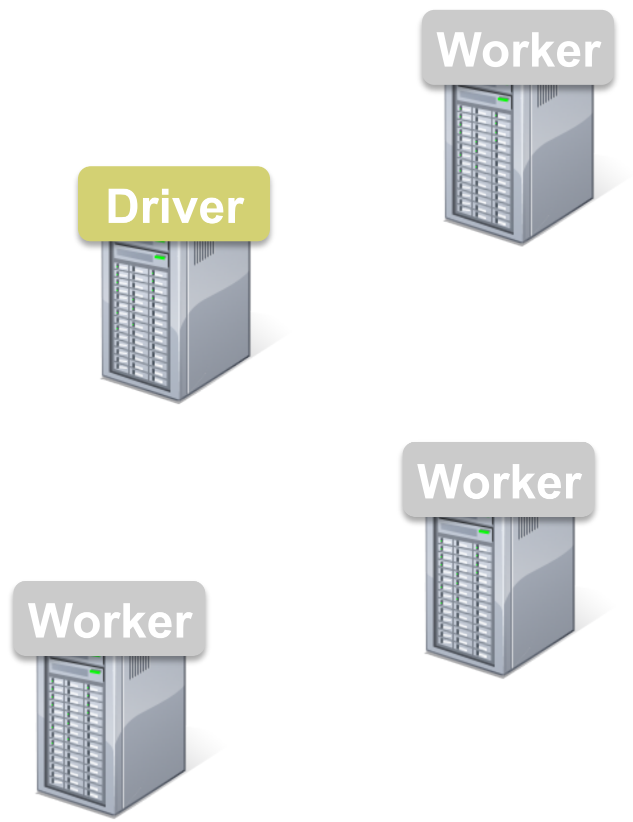
Load error messages from a log into memory, then interactively search for various patterns



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
```

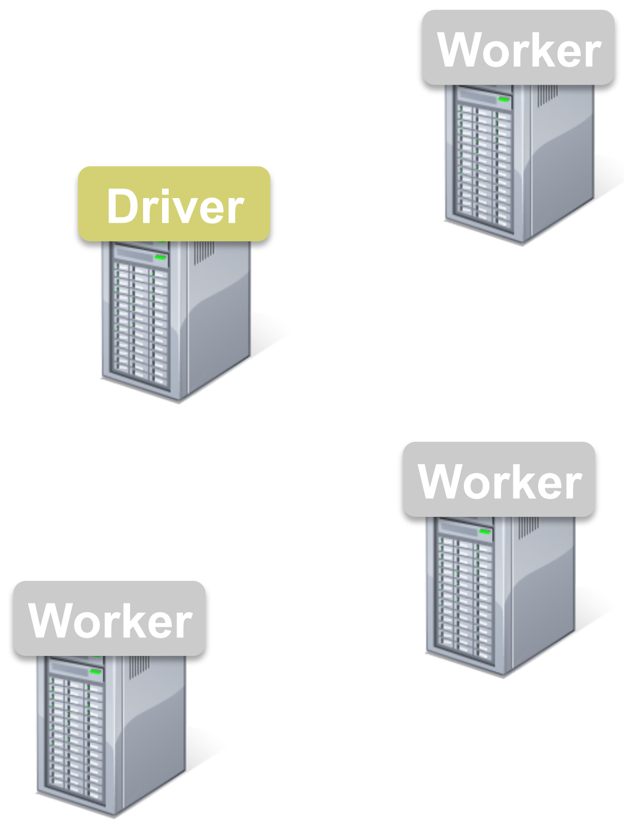


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

Base RDD

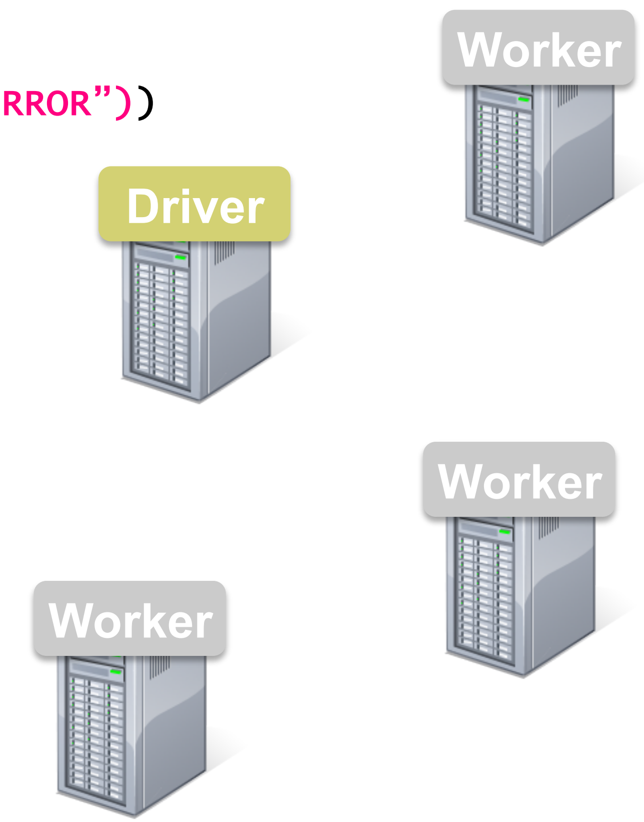
```
lines = spark.textFile("hdfs://...")
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
```

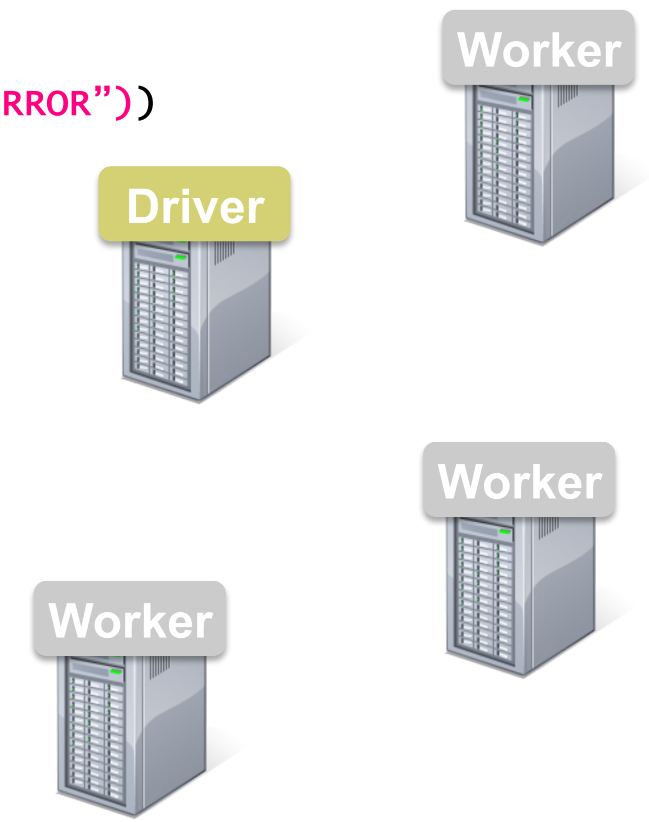


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

Transformed RDD

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
```

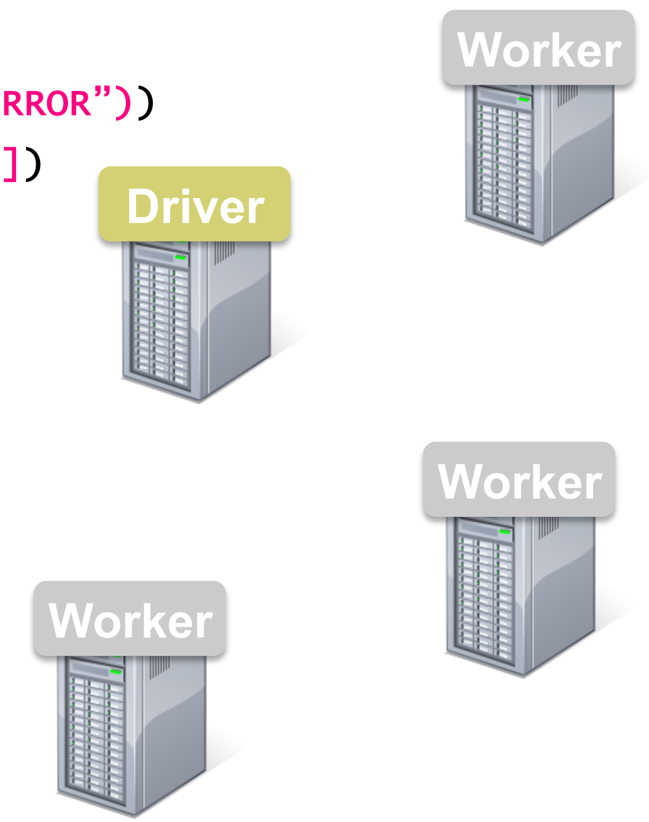


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```

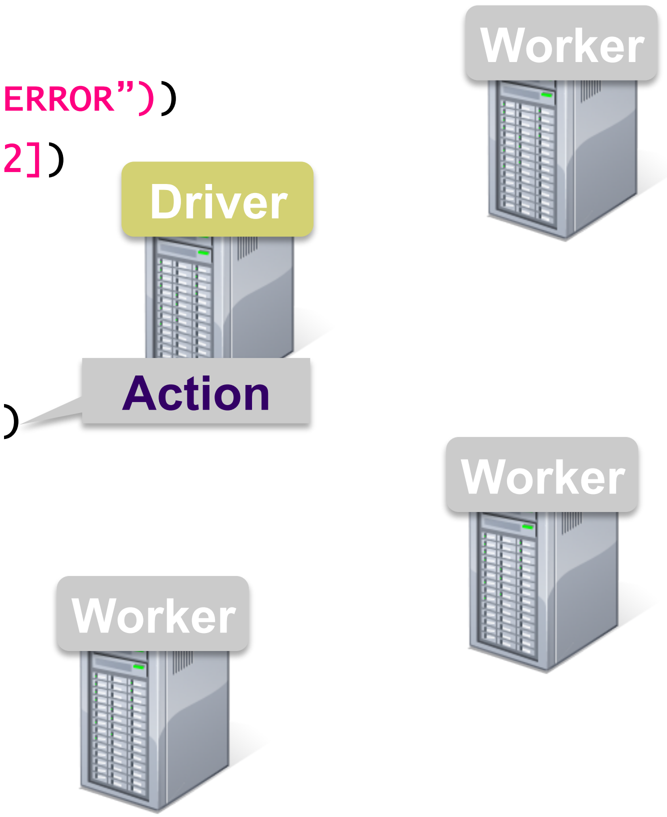


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```

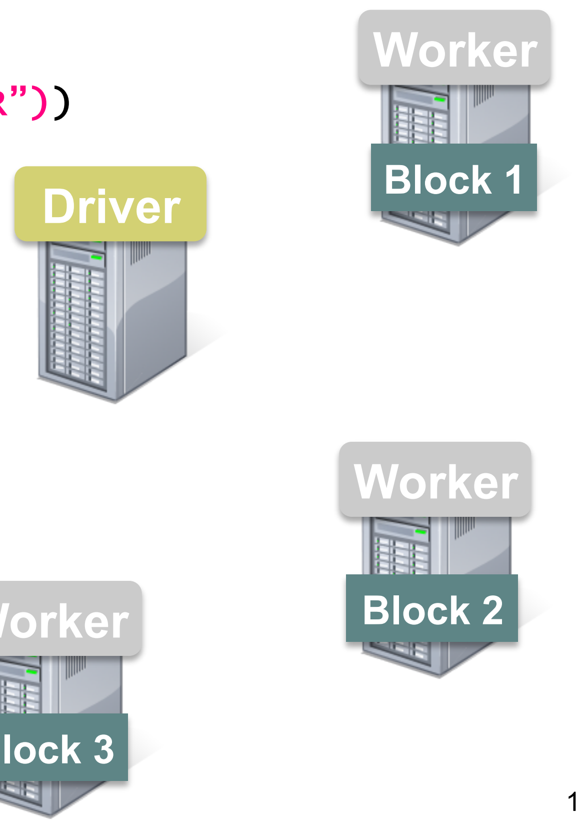


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```

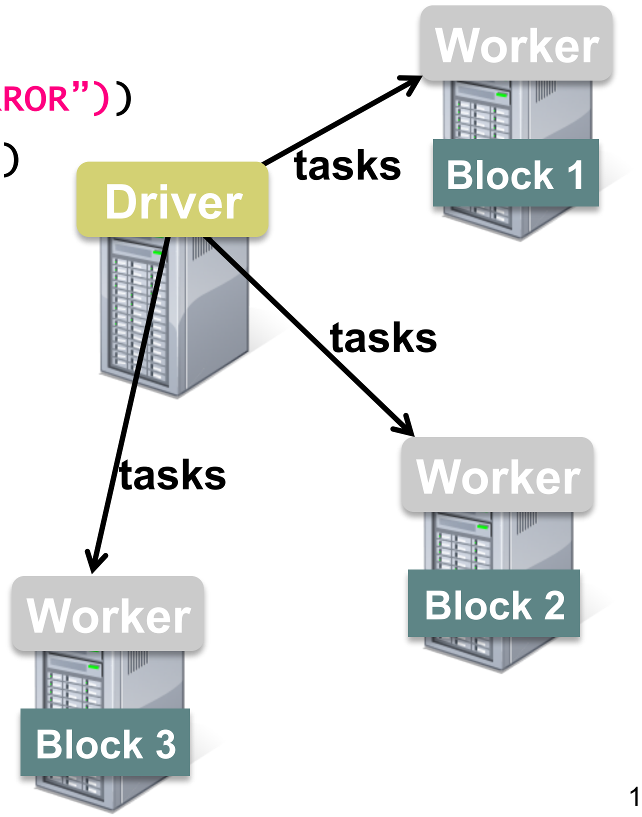


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```

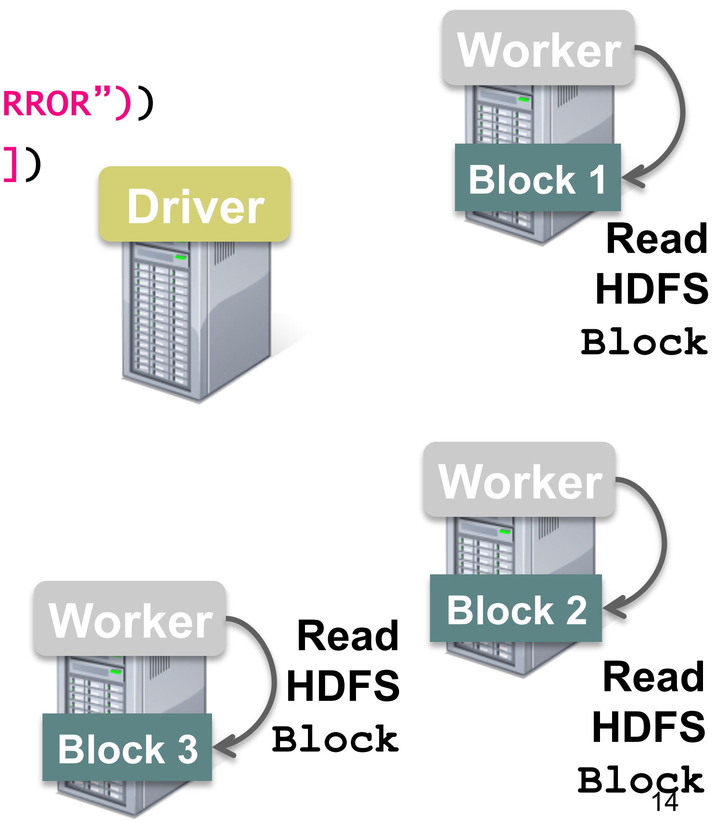


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```

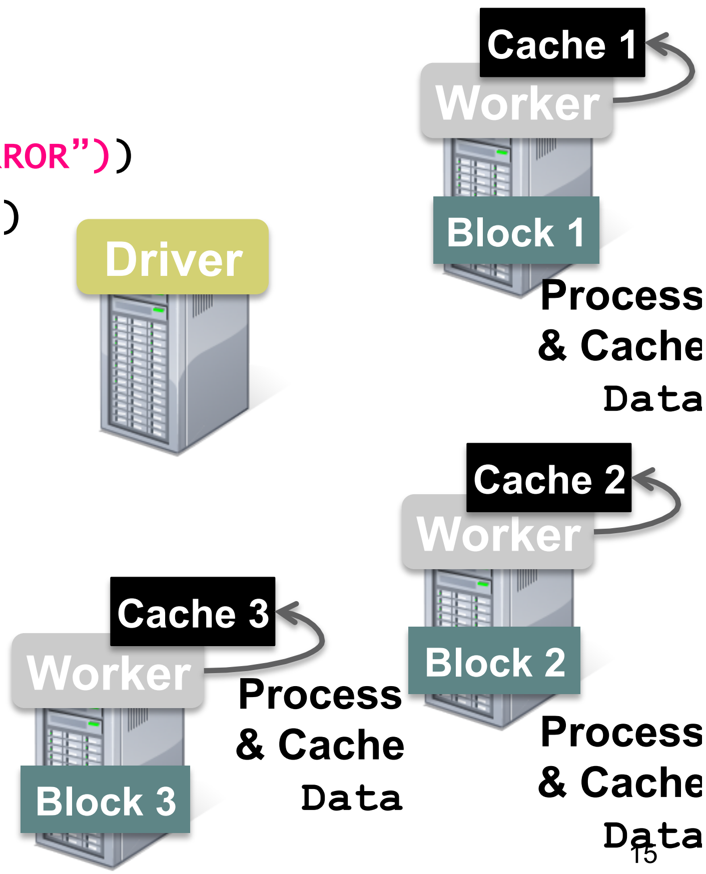


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```

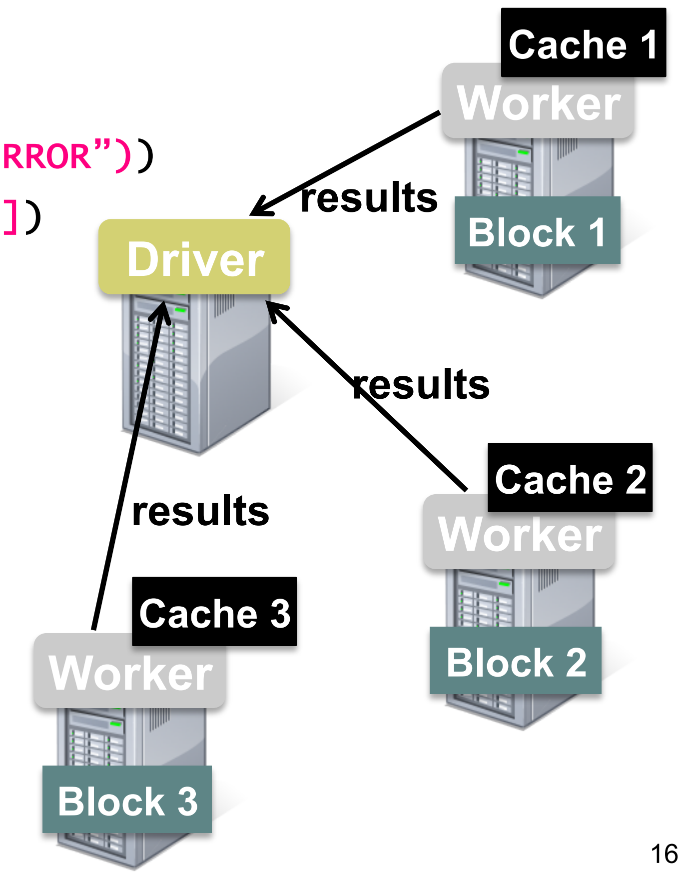


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```

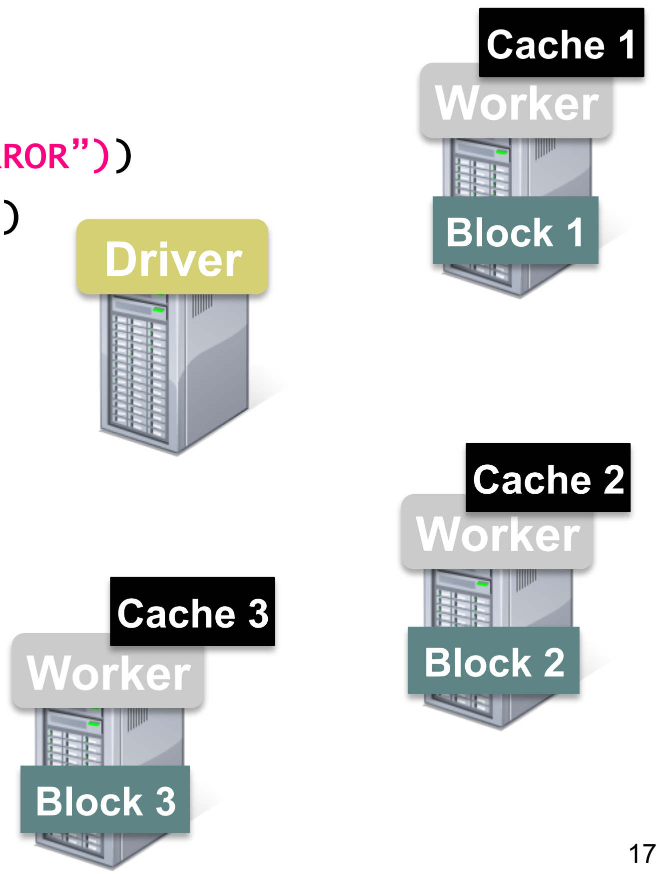


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```

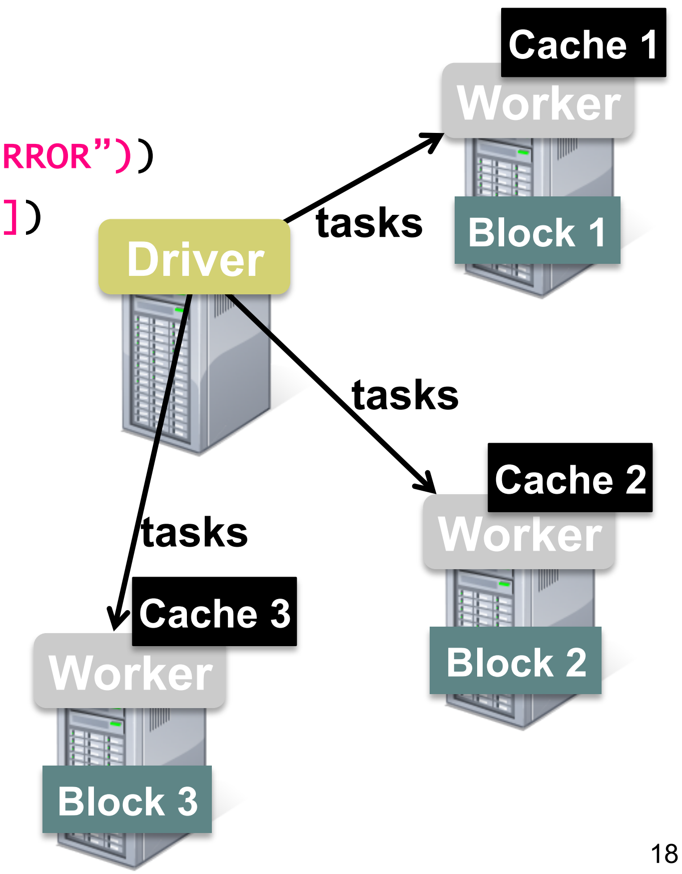


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```

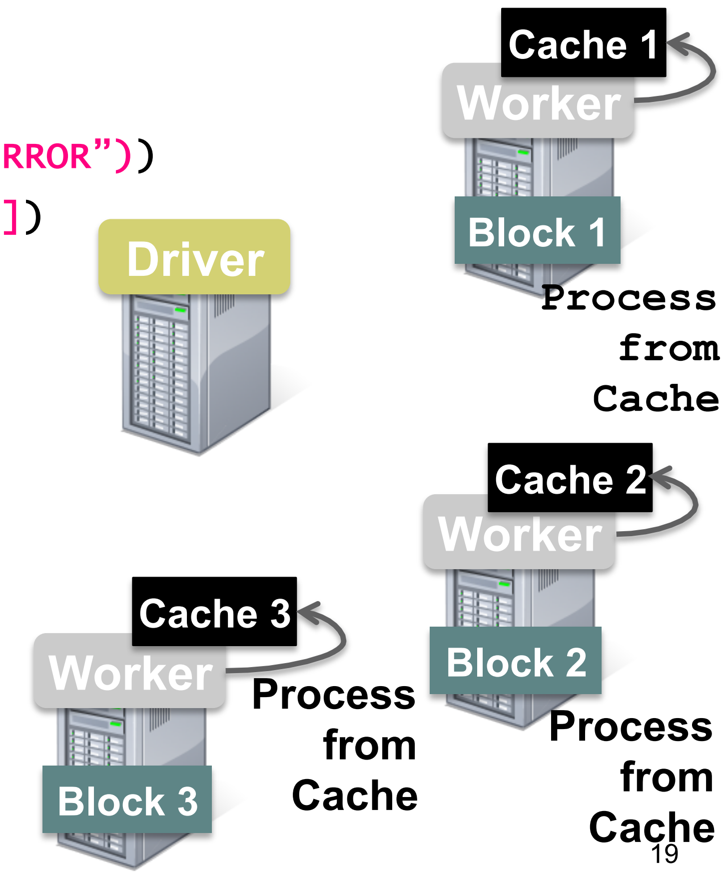


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```

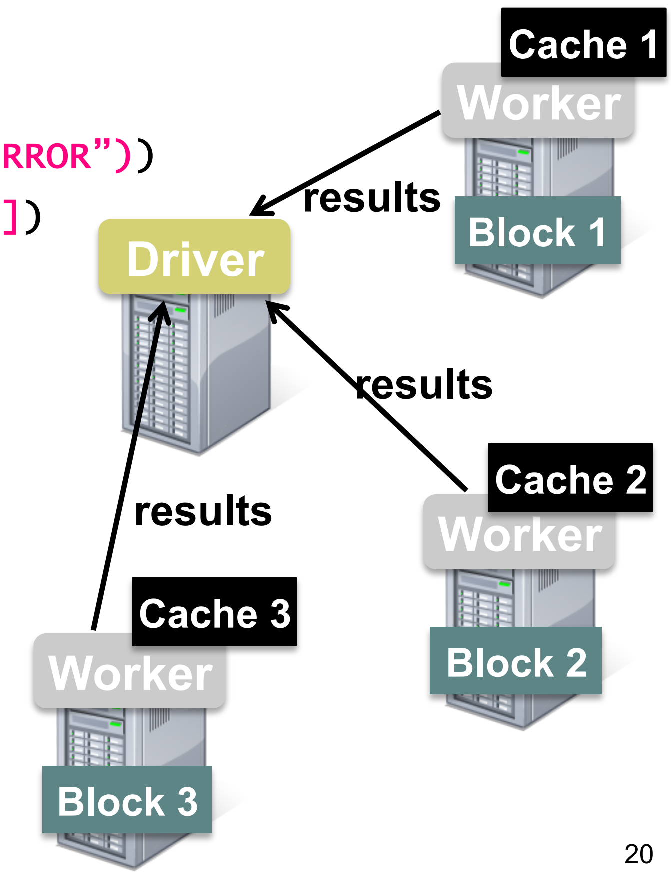


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

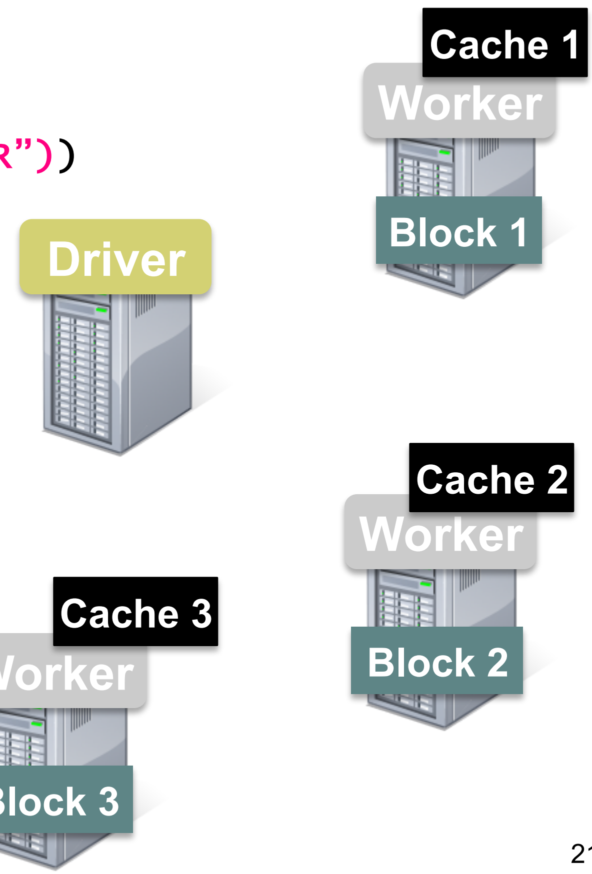
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```

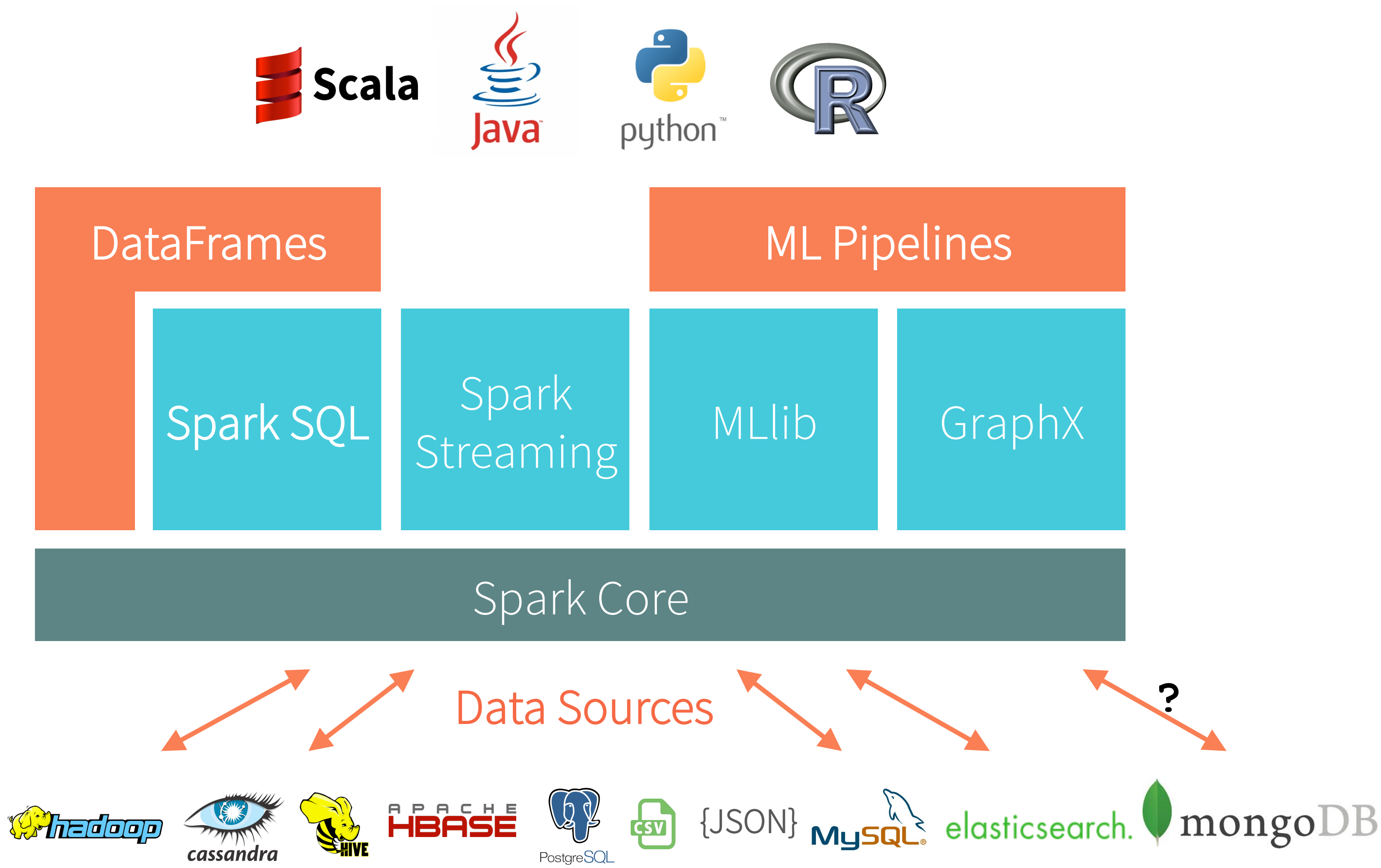
Cache your data ➔ Faster Results

Full-text search of Wikipedia

- 60GB on 20 EC2 machines
- 0.5 sec from mem vs. 20s for on-disk



Spark


Scala
Java
python
DataFrames
ML Pipelines
Spark SQL
Spark Streaming
MLlib
GraphX
Spark Core
Data Sources
?
hadoop
cassandra
HIVE
APACHE HBASE
PostgreSQL
CSV
{JSON}
MySQL
elasticsearch.
mongoDB

Pipeline Shuffle

- Problem
  - Right now shuffle senders write data on storage after which the data is shuffled to receivers
  - Shuffle often most expensive communication pattern, sometimes dominates job comp. time
- Project
  - Start sending shuffle data as it is being produced
- Challenge
  - How do you do recovery and speculation?
  - Could store data as being sent, but still not easy….

Fault Tolerance & Perf. Tradeoffs

- Problem:
  - Maintaining lineage in Spark provides fault recovery, but comes at performance cost
    - E.g., hard to support super small tasks due to lineage overhead
- Project:
  - Evaluate how much you can speed up Spark by ignoring fault tolerance
    - Can generalize to other cluster computing engines
- Challenge
  - What do you do for large jobs, how do you treat stragglers?
    - Maybe a hybrid method, i.e., just don't do lineage for small jobs? Need to figure out when a job is small…

(Eliminating) Scheduling Overhead

- Problem: with Spark, driver schedules every task
  - Latency 100s ms or higher; cannot run ms queries
  - Driver can become a bottleneck
- Project:
  - Have workers perform scheduling
- Challenge:
  - How do you handle faults?
    - Maybe some hybrid solution across driver and workers?

Cost-based Optimization in SparkSQL

- Problem:
  - Spark employs a rule-based Query Planner (Catalyst)
  - Limited optimization opportunities especially when operator performance varies widely based on input data
    - E.g., join and selection on skewed data
- Project: cost-based optimizer
  - Estimate operators' costs, and use these costs to compute the query plan

Streaming Graph Processing

- Problem:
  - With GraphX, queries can be fast but updates are typically in batches (slow)
- Project:
  - Incrementally update graphs
  - Support window based graph queries
- Note:
  - Discuss with Anand Iyer and Ankur Dave if interested

Streaming ML

- Problem:
  - Today ML algorithms typically performed on static data
  - Cannot update model in real-time
- Project:
  - Develop on-line ML algorithms that update the model continuously as new data is streamed
- Notes:
  - Also contact Joey Gonzalez if interested

Beyond JVM: Using Non-Java Libraries

- Problem:
  - Spark tasks are executed within JVMs
  - Limits performance and use of non-Java popular libraries
- Project:
  - General way to add support for non-Java libraries
  - Example: use JNI to call arbitrary libraries
- Challenges:
  - Define interface, shared data formats, etc
- Notes
  - Contact Guanhua and Shivaram, if needed

Beyond JVM: Dynamic Code Generation

- Problem:
  - Spark tasks are executed within JVMs
  - Limits performance and use of non-Java popular libraries
- Project:
  - Generate non-Java code, e.g., C++, CUDA for GPUs
- Challenges:
  - API and shared data format
- Notes
  - Contact Guanhua and Shivaram, if needed

Beyond JVM: Resource Management and Scheduling

- Problem
  - Need to schedule processes hosting non-Java code
  - GPU cannot be invoked by more than one process
- Project:
  - Develop scheduling, and resource management algorithms
- Challenge:
  - Preserve fault tolerance, straggler mitigation
- Notes
  - Contact Guanhua and Shivaram, if needed

Time Series for DataFrames

- Insprired by Pandas and R DataFrames, Spark recently introduced DataFrames
- Problem
  - Spark DataFrames don't support time series
- Project:
  - Develop and contribute distributed time series operations for Data Frames
- Challenge:
  - Spark doesn't have indexes
  - http://pandas.pydata.org/pandas-docs/stable/timeseries.html

ACID transactions to Spark SQL

- Problem
  - Spark SQL is used for Analytics and doesn’t support ACID
- Project:
  - Develop and add row-level ACID tx on top of Spark SQL
- Challenge:
  - Challenging to provide transactions and analytics in one system
  - https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions

Typed Data Frames

- Problem
  - DataFrames in Spark, unlike Spark RDDs, do not provide type safety
- Project:
  - Develop a typed DataFrame framework for Spark
- Challenge:
  - SQL-like operations are inherently dynamic (e.g. filter(“col”) and make it hard to have static typing unless fancy reflection mechanisms are used

General pipelines for Spark

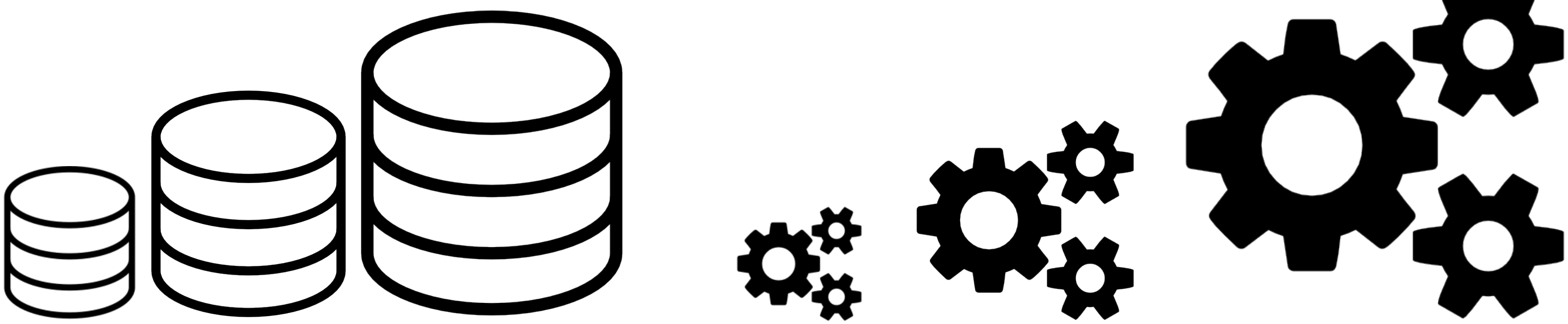
- Problem
  - Spark.ml provides a pipeline abstraction for ML, generalize it to cover all of Spark
- Project:
  - Develop a pipeline abstraction (similar to ML pipelines) that spans all of Spark, allowing users to perform SQL operations, GraphX operations, etc

Beyond BSP

- Problem
  - With BSP each worker executes the same code
- Project
  - Can we extend Spark (or other cluster computing framework) to support non-BSP computation
  - How much better than emulating everything with BSP?
- Challenge
  - Maintain simple APIs
  - More complex scheduling, communication patterns

Project idea: cryptography & big data
(Alessandro Chiesa)

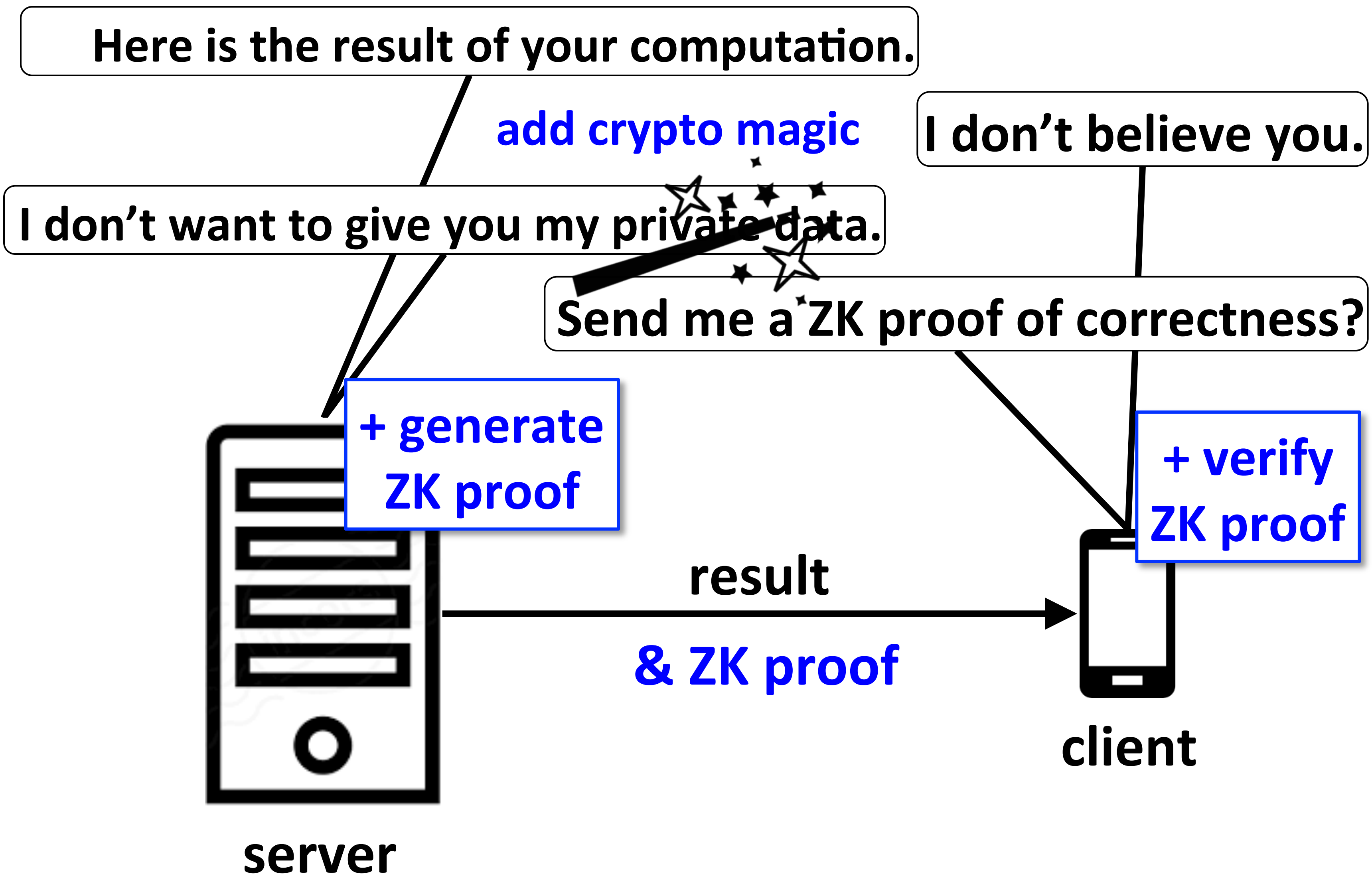
As data and computations scale up to larger sizes…



… can cryptography follow?

One direction: <u>zero knowledge proofs for big data</u>

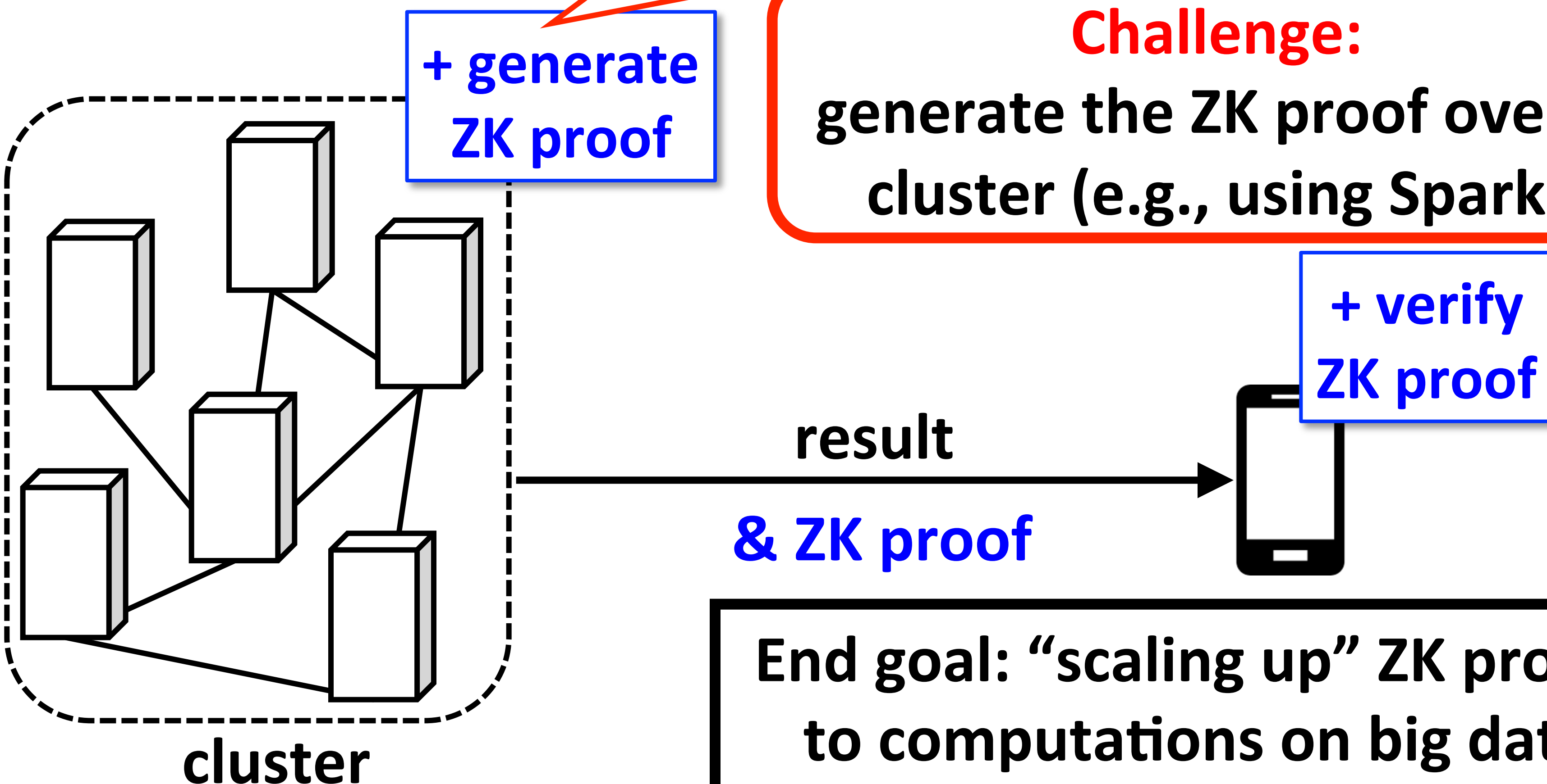
Classical setting:
zero knowledge proofs on 1 machine



New setting for big data: zero knowledge proofs on clusters

Problem: cannot generate ZK proof on 1 machine (as before)

Challenge:
generate the ZK proof over a cluster (e.g., using Spark)



End goal: "scaling up" ZK proofs to computations on big data & explore security applications!

Succinct (quick overview)

- Queries on compressed data
- Basic operations:
  - Search: given a substring “s” return offsets of all occurrences of “s” within the input
  - Extract: given an offset “o” and a length “l” uncompress and return “l” bytes from original file starting at “o”
  - Count: given a substring “s” return the number of occurrences of “s” within the input
- Can implement key-value store on top of it

Succinct: Efficient Point Query Support

- Problem:
  - Spark implementation: expensive, as always queries all workers
- Project:
  - Implement Succinct on top of Tachyon (storage layer)
  - Provide efficient key-value store lookups, i.e., lookup a single worker if key is there
- Note:
  - Contact Anurag and Rachit, if interested

Succinct: External Memory Support

- Problem:
  - Some data increases faster than main memory
  - Need to execute queries on external storage (e.g., SSDs)
- Project:
  - Design & implement compressed data structures for efficient external memory execution
  - A lot of work in theory community, that could be exploited
- Note:
  - Contact Anurag and Rachit, if interested

Succinct: Updates

- Problem:
  - Current systems use a multi-store architecture
  - Expensive to update compressed representation
- Project:
  - Develop a low overhead update solution with minimal impact on memory overhead and query performance
  - Start from multi-store architecture (see NSDI paper)
- Note:
  - Contact Anurag and Rachit, if interested

Succinct: SQL

- Problem:
  - Arbitrary sub-string search powerful but not as many workloads
- Project:
  - Support SQL on top of Succinct
  - Start from SparkSQL and Succinct Spark package?
- Note:
  - Contact Anurag and Rachit, if interested

Succinct: Genomics

- Problem:
  - Genomics pipeline still expensive
- Project:
  - Genome processing on a single machine (using compressed data)
  - Enable queries on compressed genomes
- Challenges:
  - Domain specific query optimizations
- Note:
  - Contact Anurag and Rachit, if interested