Berkeley Data Analytics Stack (BDAS) Overview

Ion Stoica
UC Berkeley
What is Big Data used For?

- Reports, e.g.,
  - Track business processes, transactions

- Diagnosis, e.g.,
  - Why is user engagement dropping?
  - Why is the system slow?
  - Detect spam, worms, viruses, DDoS attacks

- Decisions, e.g.,
  - Decide what feature to add
  - Decide what ad to show
  - Block worms, viruses, …

Data is only as useful as the decisions it enables
Data Processing Goals

- **Low latency (interactive) queries on historical data**: enable faster decisions
  - E.g., identify why a site is slow and fix it

- **Low latency queries on live data (streaming)**: enable decisions on real-time data
  - E.g., detect & block worms in real-time (a worm may infect 1mil hosts in 1.3sec)

- **Sophisticated data processing**: enable “better” decisions
  - E.g., anomaly detection, trend analysis
Today’s Open Analytics Stack…

- mostly focused on large on-disk datasets: great for **batch** but **slow**
Goals

- **Easy** to combine *batch, streaming, and interactive* computations
- **Easy** to develop *sophisticated* algorithms
- **Compatible** with existing open source ecosystem (Hadoop/HDFS)
Our Approach: Support Interactive and Streaming Comp.

- Aggressive use of *memory*
- Why?
  1. Memory transfer rates >> disk or even SSDs
     - Gap is growing especially w.r.t. disk
  2. Many datasets already fit into memory
     - The inputs of over 90% of jobs in Facebook, Yahoo!, and Bing clusters fit into memory
     - E.g., 1TB = 1 billion records @ 1 KB each
  3. Memory density (still) grows with Moore’s law
     - RAM/SSD hybrid memories at horizon
Our Approach: Support Interactive and Streaming Comp.

- Increase *parallelism*

- Why?
  - Reduce work per node → improve latency

- Techniques:
  - Low latency parallel scheduler that achieve high locality
  - Optimized parallel communication patterns (e.g., shuffle, broadcast)
  - Efficient recovery from failures and straggler mitigation

\[ T_{\text{new}} (< T) \]
Our Approach: Support Interactive and Streaming Comp.

- Trade between result **accuracy** and **response times**

- Why?
  - In-memory processing does not guarantee interactive query processing
  - E.g., ~10’s sec just to scan 512 GB RAM!
  - Gap between memory capacity and transfer rate increasing

- Challenges:
  - accurately estimate error and running time for…
  - ... arbitrary computations
Our Approach

- **Easy** to combine *batch*, *streaming*, and *interactive* computations
  - Single execution model that **supports** all computation models
- **Easy** to develop *sophisticated* algorithms
  - Powerful Python and Scala shells
  - High level abstractions for graph based, and ML algorithms
- **Compatible** with existing open source ecosystem (Hadoop/HDFS)
  - Interoperate with existing storage and input formats (e.g., HDFS, Hive, Flume, ..)
  - Support existing execution models (e.g., Hive, GraphLab)
Berkeley Data Analytics Stack (BDAS)

- **Application**
- **Data Processing**
  - in-memory processing
  - trade between time, quality, and cost
- **Data Management**
- **Resource Management**
  - Share infrastructure across frameworks (multi-programming for datacenters)

New apps: AMP-Genomics, Carat, …
The Berkeley AMPLab

- “Launched” January 2011: 6 Year Plan
  - 8 CS Faculty
  - ~40 students
  - 3 software engineers
- Organized for collaboration:
The Berkeley AMPLab

- **Funding:**
  - DARPA XData, NSF CISE Expedition Grant
  - Industrial, founding sponsors
  - 18 other sponsors, including

**Goal:** Next Generation of Analytics Data Stack for Industry & Research:
- Berkeley Data Analytics Stack (BDAS)
- Release as Open Source
Berkeley Data Analytics Stack (BDAS)
Berkeley Data Analytics Stack (BDAS)

- Existing stack components….
Mesos [Released, v0.9]

- Management platform that allows multiple frameworks to share cluster
- Compatible with existing open analytics stack
- Deployed in production at Twitter on 3,500+ servers
Spark [Release, v0.7]

- In-memory framework for **interactive** and **iterative** computations
  - Resilient Distributed Dataset (RDD): fault-tolerance, in-memory storage abstraction
- Scala interface, Java and Python APIs

```
+---+---+---+---+---+
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |
```

HDFS

```
Spark
```

Data Processing

```
HIVE Pig ...
Hadoop
Storm MPI
```

Data Mgmnt.

```
Mesos
```

Resource Mgmnt.
Spark Community

- 3000 people attended online training in August
- 500+ meetup members
- 14 companies contributing
Spark Streaming **[Alpha Release]**

- Large scale streaming computation
- Ensure exactly one semantics
- Integrated with Spark → unifies *batch, interactive, and streaming* computations!
Shark [Release, v0.2]

- HIVE over Spark: SQL-like interface (supports Hive 0.9)
  - up to 100x faster for in-memory data, and 5-10x for disk
- In tests on hundreds node cluster at Yahoo!

Diagram:
- Spark Streaming
- Spark
- HDFS
- Mesos
- HIVE
- Pig
- Storm
- MPI
- Data Processing
- Data Mgmt.
- Resource Mgmt.
Spark & Shark available now on EMR!
Tachyon [Alpha Release, this Spring]

- High-throughput, fault-tolerant in-memory storage
- Interface compatible to HDFS
- Support for Spark and Hadoop
**BlinkDB** [Alpha Release, this Spring]

- Large scale approximate query engine
- Allow users to specify **error** or **time** bounds
- Preliminary prototype starting being tested at Facebook
SparkGraph [Alpha Release, this Spring]

- **GraphLab API** and **Toolkits** on top of Spark
- Fault tolerance by leveraging Spark
**MLbase [In development]**

- Declarative approach to ML
- Develop scalable ML algorithms
- Make ML accessible to non-experts
Compatible with Open Source Ecosystem

- **Support** existing interfaces whenever possible
Compatible with Open Source Ecosystem

- **Use** existing interfaces whenever possible

Accept inputs from **Kafka, Flume, Twitter, TCP Sockets, ...**

Support **Hive API**

Support **HDFS API, S3 API, and Hive metadata**

- Spark Streaming
- Spark Graph
- MLbase
- BlinkDB
- Shark
- Hive
- Pig
- Storm
- MPI

Data Processing

Data Mgmnt.

Resource Mgmnt.

Spark

Tachyon

HDFS

Mesos
Summary

Holistic approach to address next generation of Big Data challenges!

- Support *interactive* and *streaming* computations
  - In-memory, fault-tolerant storage abstraction, low-latency scheduling,...
- **Easy** to combine *batch*, *streaming*, and *interactive* computations
  - Spark execution engine supports all comp. models
- **Easy** to develop *sophisticated* algorithms
  - Scala interface, APIs for Java, Python, Hive QL, ...
  - New frameworks targeted to graph based and ML algorithms
- **Compatible** with existing open source ecosystem
- **Open source** (Apache/BSD) and fully committed to release *high quality* software
  - Three-person software engineering team lead by Matt Massie (creator of Ganglia, 5th Cloudera engineer)
What’s Next?

- This tutorial:
  - Matei Zaharia: Spark
  - Tathagata Das (TD): Spark Streaming
  - Reynold Xin: Shark

- Afternoon tutorial:
  - Hands on with Spark, Spark Streaming, and Shark