SMACK Stack and Beyond
Building Fast Data Pipelines

Jörg Schad & Matt Jarvis
MapReduce is crunching Data... Ancient Times...
Today...

We need to turn faster!
Scope of Jan’s nightmares...

Datacenter Operating System (DC/OS)

Modern App Components

Microservices (in containers)
- Streaming
- Batch
- Machine Learning

Functions & Logic
- Search
- Time Series
- SQL / NoSQL

Big Data + Analytics Engines

Analytics

Databases

Datacenter Operating System (DC/OS)

Distributed Systems Kernel (Apache Mesos)

Any Infrastructure (Physical, Virtual, Cloud)
(Fast) Data Processing

**Days**     **Hours**     **Minutes**     **Seconds**     **Microseconds**

**Batch**     **Micro-Batch**     **Event Processing**

Reports what has happened using descriptive analytics
Solves problems using predictive and prescriptive analytics

- Billing, Chargeback
- Product recommendations
- Real-time Pricing and Routing
- Real-time Advertising
- Predictive User Interface
The SMACK Stack

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

ANALYZE
Real-time and batch process data

STORE
Distributed & highly scalable database

ACT
Visualize data and build data driven applications

<table>
<thead>
<tr>
<th>Sensors</th>
<th>Devices</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Kafka</td>
<td>Apache Spark</td>
<td>Apache Cassandra</td>
</tr>
<tr>
<td>Akka</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Apache Mesos
NAIVE APPROACH

Industry Average
12-15% utilization

Typical Datacenter
siloed, over-provisioned servers,
low utilization

Spark/Hadoop
Kafka
mySQL
microservice
Cassandra

12-15% utilization
MULTIPLEXING OF DATA, SERVICES, USERS, ENVIRONMENTS

Typical Datacenter
- siloed, over-provisioned servers, low utilization

Mesos/ DC/OS
- automated schedulers, workload multiplexing onto the same machines

- Spark/Hadoop
- Kafka
- mySQL
- microservice
- Cassandra
Apache Mesos

- A top-level Apache project
- A cluster resource negotiator
- Scalable to 10,000s of nodes
- Fault-tolerant, battle-tested
- An SDK for distributed apps
- Native Docker support
Two-level Scheduling
1. Agents advertise resources to Master
2. Master offers resources to Framework
3. Framework rejects / uses resources
4. Agent reports task status to Master
DC/OS ENABLES MODERN DISTRIBUTED APPS

Modern App Components

Microservices (in containers)

Big Data + Analytics Engines

Functions & Logic

- Streaming
- Batch
- Machine Learning

- Search
- Time Series
- SQL / NoSQL

Datacenter Operating System (DC/OS)

Distributed Systems Kernel (Mesos)

Any Infrastructure (Physical, Virtual, Cloud)
The SMACK Stack

**EVENTS**
Ubiquitous data streams from connected devices

**INGEST**
Ingest millions of events per second

**ANALYZE**
Real-time and batch process data

**STORE**
Distributed & highly scalable database

**ACT**
Visualize data and build data driven applications

- **Apache Kafka**
- **Apache Spark**
- **Apache Cassandra**
- **Akka**

---

*Sensors*  
*Devices*  
*Clients*
The SMACK Stack

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

ANALYZE
Real-time and batch process data

STORE
Distributed & highly scalable database

ACT
Visualize data and build data driven applications
The SMACK Stack

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

ANALYZE
Real-time and batch process data

STORE
Distributed & highly scalable database

ACT
Visualize data and build data driven applications
DATA PROCESSING AT HYPERSONE

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

STORE
Distributed & highly scalable database

ANALYZE
Real-time and batch process data

ACT
Visualize data and build data driven applications

Apache Kafka
Apache Cassandra
Apache Spark
Akka
DC/OS
STREAM PROCESSING

- Apache Storm
- Apache Spark
- Apache Samza
- Apache Flink
- Apache Apex
- Concord
- cloud-only: AWS Kinesis, Google Cloud Dataflow
APACHE SPARK

Spark SQL

Spark Streaming

MLlib (machine learning)

GraphX (graph processing)

Spark core (RDD)

Mesos

Standalone

YARN

Filesystem (local, HDFS, S3) or data store (HBase, Cassandra, Elasticsearch, etc.)
APACHE SPARK (STREAMING)

**Typical Use:** distributed, large-scale data processing; micro-batching

**Why Spark Streaming?**
- Micro-batching creates very low latency, which can be faster
- Well defined role means it fits in well with other pieces of the pipeline
HOW TO CHOOSE?

- Execution Model
  - Native Streaming vs Microbatch
- Fault Tolerance Granularity
  - Per record, per batch
- Delivery Guarantees
- API
  - SQL
  - Spark
- Performance…
  - Realtime ≠ Realtime
- Community
- Mesos Support ;-)

© 2016 Mesosphere, Inc. All Rights Reserved.
EXECUTION MODEL

Micro-Batching

Native Streaming

Source    Operator    Sink

Source    Operator    Sink

Spark

APEX

Flink

STORM
FAULT TOLERANCE

Ack-Per-Record

Checkpoint per "Batch"

Checkpoint per Batch
DELIVERY GUARANTEES

At least Once

“Exactly once”
DATA PROCESSING AT HYPERSCALE

EVENTS
Ubiquitous data streams from connected devices

INGEST
Ingest millions of events per second

STORE
Distributed & highly scalable database

ANALYZE
Real-time and batch process data

ACT
Visualize data and build data driven applications

DC/OS

Apache Kafka

Apache Cassandra

Apache Spark

Akka
Datastores
## Data Model

<table>
<thead>
<tr>
<th>Relational</th>
<th>Key-Value</th>
<th>Graph</th>
<th>Document</th>
<th>Time-Series</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Schema</td>
<td>• Simple</td>
<td>• Complex relations</td>
<td>• Schema-Less</td>
<td>• Schema-</td>
<td>• HDFS</td>
</tr>
<tr>
<td>• SQL</td>
<td>• Scalable</td>
<td>• Social Graph</td>
<td>• Less</td>
<td>• Semi-structured</td>
<td>• Quobyte</td>
</tr>
<tr>
<td>• Foreign Keys/Joins</td>
<td>• Cache</td>
<td>• Recommendation</td>
<td>• queries</td>
<td>• Product</td>
<td>• Ceph</td>
</tr>
<tr>
<td>• OLTP/OLAP</td>
<td></td>
<td>• Fraud detections</td>
<td>• catalogue</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Examples:**
- **Relational**: SQL, HBase, Hadoop, Cassandra, MemSQL, Redis
- **Key-Value**: Redis, Cassandra, MongoDB, Couchbase, ArangoDB
- **Graph**: Neo4j, HBase, Cassandra, MongoDB, Couchbase
- **Document**: MongoDB, Couchbase, ArangoDB
- **Time-Series**: InfluxDB
- **Files**: HDFS
Demo Time

1. Financial data created by generator
2. Written to Kafka topics
3. Kafka Topics consumed by Flink
4. Flink pipeline operates on Kafka data
5. Results written back into Kafka stream (another topic)
6. Results displayed
Keep it running!
SERVICE OPERATIONS

- Configuration **Updates** (ex: Scaling, re-configuration)
- Binary **Upgrades**
- Cluster **Maintenance** (ex: Backup, Restore, Restart)
- **Monitor** progress of operations
- **Debug** any runtime blockages
ANY QUESTIONS?
APACHE SPARK (STREAMING)

Typical Use: distributed, large-scale data processing; micro-batching

Why Spark Streaming?
- Micro-batching creates very low latency, which can be faster
- Well defined role means it fits in well with other pieces of the pipeline