Big Data Primer For IT Professional

Sujee Maniyam
Founder / Principal @ Elephant Scale
SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.
Abstract

Big Data Primer For IT Professionals

This session will highlight some Big Data technologies that an aspiring Big Data developers should learn. This talk will appeal to developers / engineers who want to learn Big Data technologies.
A look at Big data eco system

The Datafloq Open Source Landscape 2.0

---

Source: datafloq.com
Which One?
Let’s take ‘design driven approach’
Internet of Things – A reality
Data infrastructure
Data Volume?
A Napkin calculation

- Say we have
  - Million sensors
  - Each sensor reports every minute
  - data size 1KB

- This will result in:
  - 1.44 Billions events / day!
  - 1.44 TB / day!!
### Texas Smart Meter Projections

<table>
<thead>
<tr>
<th>variables</th>
<th>description</th>
<th>sensors</th>
<th>signal frequency</th>
<th>event size</th>
<th>events per day per sensor</th>
<th>total events per day (millions)</th>
<th>total events / sec</th>
<th>total data size per day (GB)</th>
<th>total data size per day (TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensors</td>
<td>10 million customers</td>
<td>1.00E+07</td>
<td>10 million</td>
<td></td>
<td>96</td>
<td>9.60E+08 960 millions</td>
<td>1.11E+04</td>
<td>1.34E+12 1344 GB</td>
<td>1.344 TB</td>
</tr>
<tr>
<td>signal frequency</td>
<td>every 15 mins</td>
<td>900 secs</td>
<td></td>
<td>1.4 K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>event size</td>
<td>1.4 K</td>
<td></td>
<td></td>
<td>1400 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sensor Data: Texas utilities
smart meter data
Data Velocity

Say we have

- Million sensors
- Each sensor reports every minute
- data size 1KB

• Millions events / minute
• ~17,000 events / sec
Data processing speed

- Need (near) real time processing most of the time
  - E.g. Need to alert if temperature suddenly spikes
Challenge = big data + real time

- Don’t loose events!
  - Any event could be important
  - Most events are mundane (e.g. temperature stays between 68’F – 72’ F)

- Process them in near real time

- Store the events for a long time
  - Audit
  - Diagnose

- Support various queries
  - Real time (what is the latest temperature for sensor id 123?)
  - Aggregate (what is the avg. temp in zipcode 12345)
High Level Architecture

- Capture
- Process
- Visualization
- Store
- Analytics
Capture
(1) Capture
Requirements

- Requirements:
  - Capture events coming at high speed
    - Tens of thousands events / sec (some times millions / sec)
  - Don’t lose events
    - Tolerate hardware / software failure
    - Tolerate intermittent connectivity issues
  - Scale ‘easily’
(1) Capture Choices

- **MQ (RabbitMQ ..etc)**
  - Good adoption in enterprises / durable
- **FluentD**
  - Data collector for various sources
- **Flume**
  - Part of Hadoop eco system
  - Good for collecting logs from many sources
- **AWS Kinesis**
  - Queue system in Amazon Cloud
- **Kafka**
  - Distributed queue
Meet kafka

- Apache Kafka is a distributed messaging system
- Came out of LinkedIn... open sourced in 2011
- Built to tolerate hardware / software / network failures
- Built for high throughput and scale
  - LinkedIn: 220 Billion messages / day
  - At peak: 3+ million messages / sec
(1) Capture
Kafka architecture

- Publisher - subscriber / producer – consumer model
(1) Capture
Kafka architecture

- Producers write data to brokers
- Consumers read data from brokers
- All of this is distributed / parallel
- Failure tolerant
- Data is stored as topics
  - “sensor_data”
  - “alerts”
  - “emails”
Capture

Capture (Kafka)  Process  Visualization  Store  Analytics
Next : (2) Processing
(2) Processing requirements

- Process events in real time or near real time
- High velocity
  - Tens of thousands → millions of events / sec
- Guaranteed processing
  - Process an event at-least-once
  - Exactly-once (harder to achieve)
- Failure tolerant
- Scale ‘easily’
(2) Processing Choices

- **Storm**
  - ’Original’ stream processing
- **Apache Samza**
  - Stream processing framework based on Kafka + Hadoop YARN
- **Apache NiFi**
  - Data flow
- **Flink**
  - New framework
- **Spark Streaming**
  - Cool framework
# Streaming Systems Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Storm</th>
<th>Spark Streaming</th>
<th>Flink</th>
<th>NiFi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Model</td>
<td>Event based by default (micro batch using Trident)</td>
<td>Micro Batch</td>
<td>Event based + Micro Batch based</td>
<td>Event Based (?)</td>
</tr>
<tr>
<td>Windowing operations</td>
<td>Supported by Trident</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>Latency</td>
<td>Milliseconds</td>
<td>Seconds</td>
<td>Milliseconds</td>
<td>Milliseconds</td>
</tr>
<tr>
<td>At-least-once</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>At-most-once</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>?</td>
</tr>
<tr>
<td>Exactly-once</td>
<td>YES with Trident</td>
<td>YES</td>
<td>YES</td>
<td>?</td>
</tr>
</tbody>
</table>
Spark Streaming Architecture

Spark Streaming: discretized stream processing

Records processed in batches with short tasks. Each batch is a RDD (partitioned dataset).
Flink Streaming Architecture

Stream platform architecture

- Gather and backup streams
- Offer streams for consumption
- Provide stream recovery
- Analyze and correlate streams
- Create derived streams and state
- Provide these to downstream systems
Stream Processing

Capture (Kafka)

Process (Spark / Flink / Samza)

Visualization

Store

Analytics
(3) storage

Requirements

- Handle ‘Big Data’ (1 TB / day!)
- Traditional storages are not effective (or too expensive)
- Need two types of storage
  1. ‘forever’ storage
     - Store multi terabytes of data for a long periods
     - Support Batch queries
  2. ‘fast / real-time lookup’ storage
     - Query in real time (milliseconds)
       “what is the latest reading for sensor-123?”
     - Store latest / new data (e.g. last 3 months)
     - Flexible schema for semi-structured data
- Both need to scale
(3) Storage Requirements

Data Spectrum

Batch

Real time

Access Time

MySQL  mongoDB  Hbase, Cassandra, Vertica  Google's Spanner

Giga bytes  Tera bytes  Peta bytes

Scale

adapted from: http://www.slideshare.net/medriscool/driscoll-strata-building-data-startups-25may2011clean
Choices

• ‘forever’ storage
  - Scalable distributed file systems
  - Hadoop! (HDFS actually)

• ‘real time store’
  - Traditional RDBMS won’t work
    - Don’t scale well (or too expensive)
    - Rigid schema layout
  - NoSQL!
(3) Storage
HDFS (in 20 secs)

• Distributed file system
• Runs on commodity servers
  . → high ROI
• Can keep ticking even when nodes go down
  . → fault tolerant
• Replicates data to prevent data loss in case of node failures
  . → built in backup 😊
• Scales to Peta bytes (horizontal scalability)
• Proven in the field
(3) Storage
HDFS Architecture

Data Node 1
Data Node 2
Data Node 3

multiple copies
(3) Storage
Cost of Big data

Hadoop: Lower Cost of Storage

Cloud Storage

HADOOP

NAS

Engineered System

Fully-loaded Cost Per Raw TB of Data
(Min–Max Cost)

MPP

SAN

$0 $20,000 $40,000 $60,000 $80,000 $180,000

Source: hortonworks
HDFS

- Can handle big data
- Scales easily
- Cost effective
- "Source of Truth"
  - Files are immutable within HDFS (new data is ‘appended’)
  - Audit friendly
(3) Storage (real time)
Choices for NOSQL

• Too many! 😃
• HBase
  • Part of Hadoop eco system
  • Uses HDFS for storage
  • Provides consistent view of data
• Cassandra
  • Popular NoSQL store
  • No Single Point of Failure (SPOF) – ring architecture
  • No dependency on Hadoop
• Druid
  • Sub second OLAP queries / fast aggregations
Next: Analytics

Capture (Kafka)

Process (Spark / Flink / Samza)

Store (HDFS + NoSQL)

Visualization

Analytics
Next: Analytics

- Must scale to peta bytes of data size
- Large queries
  - Popular #hashtags in 2015
- ETL
  - Shape / clean data
- Data warehousing
  - Batch queries
- Machine Learning
  - Model building (credit scoring ..etc)
Analytics Tools

- **ETL**
  - Pig
  - Spark
  - Flink

- **SQL queries**
  - Hive / Impala
  - Drill
  - Spark SQL
  - Flink SQL

- **Machine Learning**
  - Spark ML
  - Flink ML
Analytics

Capture (Kafka)

Process (Spark / Flink / Samza)

Store (HDFS + NoSQL)

Visualization

Analytics (ETL, SQL, ML)
Next : Visualization
Visualization Tools

- Ready made for enterprises
  - Tableau
  - SiSense
  - Pentaho

- Roll your own
  - Notebooks
  - D3.js
  - R

- And don’t forget…
  - Excel !!
Notebooks

- Zeppelin
- Spark Notebook
- iPython
Notebook Example
Visualization

- Capture (Kafka)
- Process (Spark / Flink / Samza)
- Visualization (Notebooks, D3js, Tableau..)
- Store (HDFS + NoSQL)
- Analytics (ETL, SQL, ML)
Final Stack

- Capture (Kafka)
- Process (Spark / Flink / Samza)
- Visualization (Notebooks, D3js, Tableau..)
- Store (HDFS + NoSQL)
- Analytics (ETL, SQL, ML)
Final Words

❖ No one can know every thing !
  ❖ At least get a basic understanding

❖ Levels of knowledge
  ❖ I haven’t heard of it
  ❖ I have heard of it
  ❖ I have played around with it on my laptop
  ❖ I have working knowledge
  ❖ I am an expert / I wrote the damn thing !
Also keep in mind...

At scale nothing works as advertised!
The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

Authorship History
Sujee Maniyam - May 2016

Additional Contributors

Please send any questions or comments regarding this SNIA Tutorial to tracktutorials@snia.org