The Real Time Boom..

Facebook Real Time Social Analytics

SaaS Real Time User Tracking

Google Real Time Web Analytics

Twitter paid tweet analytics

New Real Time Analytics Startups..

Google Real Time Search
Not All analytics are real time
(from Analytics @ Twitter)

• Counting
  – How many request?
  – What’s the average latency?
  – How many signups, sms, tweets?

• Correlating
  – Desktop vs Mobile user?
  – What devices fail at the same time?
  – What features get user hooked?

• Researching
  – What features get re-tweeted
  – Duplicate detection
  – Sentiment analysis

Real time (msec/sec)
Near real time (Min/Hours)
Batch (Days..)
volume, variety, velocity, and veracity

- Veracity refers to the quality or trustworthiness of the data.
- A common complication is that the data is saturated with both useful signals and lots of noise (data that can’t be trusted).

LHC ATLAS detector generates about 1 Petabyte raw data per second, during the collision time (about 1 ms).
Big Data platform must include the key imperatives

<table>
<thead>
<tr>
<th>Big Data Platform Imperatives</th>
<th>Technology Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Discover, explore, and navigate Big Data sources</td>
<td>Federated Discovery, Search, and Navigation</td>
</tr>
<tr>
<td>2. Extreme performance-run analytics closer to data</td>
<td>Massively Parallel Processing Analytic appliances</td>
</tr>
<tr>
<td>3. Manage and analyze unstructured data</td>
<td>Hadoop File System/MapReduce Text Analytics</td>
</tr>
<tr>
<td>4. Analyze data in motion</td>
<td>Stream Computing</td>
</tr>
<tr>
<td>5. Rich library of analytical functions and tools</td>
<td>In-Database Analytics Libraries Big Data Visualization</td>
</tr>
<tr>
<td>6. Integrate and govern all data sources</td>
<td>Integration, Data Quality, Security, Lifecycle Management, MDM, etc</td>
</tr>
</tbody>
</table>

The Big Data platform manifesto: imperatives and underlying technologies
content

• General Introduction
• Definitions
• Data Analytics
• Solutions for Big Data Analytics
• The Network (Internet)
• When to consider BigData solution
• Scientific e-infrastructure – some challenges to overcome
Data Analytics

Analytics Characteristics are not new

- **Value**: produced when the analytics output is put into action
- **Veracity**: measure of accuracy and timeliness
- **Quality**:
  - well-formed data
  - Missing values
  - cleanliness

Data types have differing pre-analytics needs
Skills required for Big Data Analytics (A.K.A Data Science)

- **Store and process**
  - Large scale databases
  - Software Engineering
  - System/network Engineering

- **Analyse and model**
  - Reasoning
  - Knowledge Representation
  - Multimedia Retrieval
  - Modelling and Simulation
  - Machine Learning
  - Information Retrieval

- **Understand and design**
  - Decision theory
  - Visual analytics
  - Perception Cognition

http://edison-project.eu/university-programs-list


Nancy Grady, PhD, SAIC Co-Chair Definitions and Taxonomy Subgroup NIST Big Data Working Group
content

• General Introduction
• Definitions
• Data Analytics
• Solutions for Big Data Analytics
• The Network (Internet)
• When to consider BigData solution
• Scientific e-infrastructure – some challenges to overcome
Traditional analytics applications

• **Scale-up Database**
  – Use traditional SQL database
  – Use stored procedure for event driven reports
  – Use flash-based disks **to reduce disk I/O**
  – Use read only replica to **scale-out read queries**

• **Limitations**
  – Doesn't scale on write
  – Extremely expensive (HW + SW)
**NoSQL**

“Work with scientists to find the most common "20 queries" and make them fast.” How to deal with Big Data Advice From Jim Gray (advice number 3)

- **Use distributed database**
  - Hbase, Cassandra, MongoDB

- **Pros**
  - Scale on write/read
  - Elastic

- **Cons**
  - Read latency
  - Consistency tradeoffs are hard
  - Maturity – fairly young technology
## NoSQL

<table>
<thead>
<tr>
<th>Year</th>
<th>System/Paper</th>
<th>Scale to 1000s</th>
<th>Primary Index</th>
<th>Secondary Indexes</th>
<th>Transactions</th>
<th>Joins/Analytics</th>
<th>Integrity Constraints</th>
<th>Views</th>
<th>Language/Algebra</th>
<th>Data model</th>
<th>my label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1971</td>
<td>RDBMS</td>
<td>O</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>memcached</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>key-val</td>
<td>nosql</td>
</tr>
<tr>
<td>2004</td>
<td>MapReduce</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>key-val</td>
<td>batch</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>CouchDB</td>
<td>✓</td>
<td>✓</td>
<td>record</td>
<td>MR</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>document</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>BigTable (Hbase)</td>
<td>✓</td>
<td>✓</td>
<td>record</td>
<td>compat. w/MR</td>
<td>/</td>
<td>0</td>
<td>0</td>
<td>ext. record</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>MongoDB</td>
<td>✓</td>
<td>✓</td>
<td>EC, record</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>document</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Dynamo</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ext. record</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Pig</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>HIVE</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Cassandra</td>
<td>✓</td>
<td>✓</td>
<td>EC, record</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>key-val</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Voldemort</td>
<td>✓</td>
<td>✓</td>
<td>EC, record</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>key-val</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Riak</td>
<td>✓</td>
<td>✓</td>
<td>EC, record</td>
<td>MR</td>
<td>0</td>
<td></td>
<td></td>
<td>key-val</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Dremel</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Megastore</td>
<td>✓</td>
<td>✓</td>
<td>entity groups</td>
<td>0</td>
<td>/</td>
<td>0</td>
<td>/</td>
<td>tables</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Tenzing</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Spark/Shark</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Spanner</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Accumulo</td>
<td>✓</td>
<td>✓</td>
<td>record</td>
<td>compat. w/MR</td>
<td>/</td>
<td>0</td>
<td>0</td>
<td>ext. record</td>
<td>nosql</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Impala</td>
<td>✓</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>tables</td>
<td>sql-like</td>
<td></td>
</tr>
</tbody>
</table>

*Scale was the primary motivation!*

Bill Howe, UW
CEP – Complex Event Processing

- Process the data as it comes
- Maintain a window of the data in-memory

Pros:
- Extremely low-latency
- Relatively low-cost

Cons:
- Hard to scale (Mostly limited to scale-up)
- Not agile - Queries must be pre-generated
- Fairly complex
In Memory Data Grid

- **Distributed** in-memory database
  - **Scale out** (Horizontal scaling)

- **Pros**
  - Scale on write/read
  - Fits to event driven (CEP style), ad-hoc query model

- **Cons**
  - **Cost** of memory vs disk
  - Memory **capacity** is limited
In Memory Data Grid products

- Hazelcast [hazelcast.org](http://hazelcast.org)
- JBOSS Infinispan [www.infinispan.org](http://www.infinispan.org)
- Pivotal Gemfire [pivotal.io/big-data/pivotal-gemfire](http://pivotal.io/big-data/pivotal-gemfire)
Hadoop MapReduce

- Distributed **batch** processing
- Pros
  - Designed to process massive amount of data
  - Mature
  - Low cost
- Cons
  - **Not** real-time
Sorting 1 TB of DATA

• Estimate:
  – read 100MB/s, write 100MB/s
  – no disk seeks, **instant sort**
  – 341 minutes → 5.6 hours

• The terabyte benchmark winner (2008):
  – 209 seconds (3.48 minutes)
  – 910 nodes x (4 dual-core processors, 4 disks, 8 GB memory)

• October 2012
  – ? see
    [http://www.youtube.com/watch?v=XbUPIbYxT8g&feature=youtu.be](http://www.youtube.com/watch?v=XbUPIbYxT8g&feature=youtu.be)
MapReduce vs. Databases


• Conclusions: ... at the scale of the experiments we conducted, both parallel database systems displayed a significant performance advantage over Hadoop MR in executing a variety of data intensive analysis benchmarks.
“With the paths that go through Hadoop [at Yahoo!], the latency is about fifteen minutes. ... [I]t will never be true real-time..” (Yahoo CTO Raymie Stata)

Hadoop/Hive..Not realtime. Many dependencies. Lots of points of failure. Complicated system. Not dependable enough to hit realtime goals (Alex Himel, Engineering Manager at Facebook.)

"MapReduce and other batch-processing systems cannot process small updates individually as they rely on creating large batches for efficiency,“ (Google senior director of engineering Eisar Lipkovitz)
Map Reduce

• **Map:**
  – Accepts
    • *input* key/value pair
  – Emits
    • *intermediate* key/value pair

• **Reduce:**
  – Accepts
    • *intermediate* key/value* pair
  – Emits
    • *output* key/value pair

Very big data

WING Group Meeting, 13 Oct 2006 Hendra Setiawan
Apache Spark

Lightning-fast cluster computing (in-memory)

• Generality
  – Combine SQL, streaming, complex analytics.

• Runs Everywhere
  – Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources (HDFS, Cassandra, HBase, and S3).

• Ease of Use
  – Write applications quickly in Java, Scala, Python, R.

Developer(s)  Apache Software Foundation, UC Berkeley AMPLab, Databricks
Initial release  May 30, 2014; 18 months ago
Stable release  v1.5.2 / November 9, 2015; 51 days ago
Development status  Active
Written in  Scala, Java, Python, R
Operating system  Linux, Mac OS, Windows
Type  data analytics, machine learning algorithms
License  Apache License 2.0
Website  spark.apache.org
Apache Spark

Lightning-fast cluster computing

Resilient Distributed Datasets (RDD)
  – **Immutable**, partitioned **collections** of records
  – can only be built through **coarse-grained** deterministic transformations (map, filter, join...)

Efficient fault-tolerance using lineage
  – Log coarse-grained operations instead of fine-grained data updates
  – An RDD has enough information about how it’s derived from other dataset
  – Recompute lost partitions on failure

https://dzone.com/refcardz/apache-spark
Apache Spark

Lightning-fast cluster computing

• 10 iterations on 100GB data using 25-100 machines

• 10 iterations on 54GB data with approximately 4M articles

• 10 iterations of k-means on 75 nodes, each iteration contains 400 tasks on 100GB data

Matei Zaharia, Mosharaf Chowdhury, Resilient Distributed Datasets A Fault-Tolerant Abstraction for In-Memory Cluster Computing NSDI’12 presentation
Apache Storm

By Nathan Marz

• **Storm** is a distributed real-time computation system that solves typical
  – downsides of queues & workers systems.
  – Built with Big Data in mind (the “Hadoop of realtime”).

• **Storm Trident** (high level abstraction over Storm core)
  – Micro-batching (~ streaming)
Apache Storm

Core concepts
- Topologies
- Spouts and bolts
- Data model
- Groupings

What storm does
- Distributes code and configurations
- Manage processes (robust)
- Monitors topologies & reassigns failed tasks
- Provides reliability by tracking tuples
- Routing and partitioning of Streams
- Serialization
- Fine-Grained performance stats of topologies

Tuple = datum containing 1+ fields

(1.1.1.1, “foo.com”)

Values can be of any type such as Java primitive types, String, byte[]. Custom objects should provide their own Kryo serializer though.

Stream = unbounded sequence of tuples

... (1.1.1.1, “foo.com”) (2.2.2.2, “bar.net”) (3.3.3.3, “foo.com”) ...

Performance

OpenSOC
The Open Security Operations Center
for
Analyzing 1.2 Million Network Packets per Second in Real Time

James Sirota,
Big Data Architect
Cisco Security Solutions Practice
jasirota@cisco.com
June 3, 2014

Sheetal Dolas
Principal Architect
Hortonworks
sheetal@hortonworks.com

http://www.slideshare.net/JamesSirotajames/sirota/cisco-opensoc

https://twitter.com/nathanmarz/status/207989068519317505

Nathan Marz
@nathanmarz

Just clocked Storm 0.8.0 at 1.64 million tuples processed per second per node on an internal Twitter cluster
8:17 PM - 30 May 2012
39 RETWEETS 19 FAVORITES
Apache Kafka
A high-throughput distributed messaging system

- Apache Kafka is publish-subscribe messaging rethought as a distributed commit log.

- Kafka maintains feeds of messages in categories called topics.
  - Processes can publish messages to a Kafka (topic producers).
  - Processes can subscribe to topics and process the feed of published messages consumers.

- Kafka is run as a cluster comprised of one or more servers each of which is called a broker.
Apache Kafka

A high-throughput distributed messaging system

Credit: http://kafka.apache.org/design.html

Big data Analytics in Microsoft Azure

- HDInsight
- Map reduce type job
- Other types of data analytics