Lambda Architecture with Apache Spark

Michael Hausenblas, Chief Data Engineer MapR

First Galway Data Meetup, 2015-02-03
Polyglot Processing
Polyglot Processing

- Combination of different processing engines over DFS/NoSQL stores
- Lambda and Kappa architectures are two prominent examples

http://datadventures.ghost.io/2014/07/06/polyglot-processing/
Fault tolerance

developer

software

hardware
Let’s talk about developers…
Hi, this is your son's school. We're having some computer trouble.

Oh, dear - did he break something?
In a way.

Did you really name your son Robert?; DROP TABLE Students; -- ?

Oh, yes. Little Bobby Tables, we call him.

Well, we've lost this year's student records. I hope you're happy.

And I hope you've learned to sanitize your database inputs.

http://xkcd.com/327/
WORKED FINE IN DEV

OPS PROBLEM NOW
Let's talk about developers...
Human fault tolerance

Handed the Most Dangerous Weapon in the Galaxy

Immediately Point at Face
When things go wrong ...

The Real Reason Facebook Went Down Yesterday: It's Complicated

NICK O'NEILL ON SEPTEMBER 24, 2010 10:20 AM

Yesterday afternoon Facebook experienced the worst outage that the company has had "in over four years", causing the site to go down for most users for "approximately 2.5 hours". One of the company's engineers followed up with a blog post, explaining exactly what went wrong. The cause of the issue sounds relatively complicated, however the conclusion was that the company had to restart the entire site.

According to Robert Johnson:

The key flaw that caused this outage to be so severe was an unfortunate handling of an error condition. An automated system for verifying configuration values ended up causing much more damage than it fixed.

The intent of the automated system is to check for configuration values that are invalid in the cache and replace them with updated values from the persistent store. This works well for a transient problem with the cache, but it doesn't work when the persistent store is invalid.

2010 unfortunate handling of error condition
When things go wrong …

Twitter crashes hard, Internet freaks out

By Julianne Pepitone @CNNMoneyTech June 21, 2012, 3:34 PM ET

2012
cascaded bug

http://money.cnn.com/2012/06/21/technology/twitter-down/index.htm
When things go wrong …

The RBS/Natwest technical outage that blighted the bank and its customers in June has cost the firm £125m according to its latest financial reports.

The banking group, which posted overall losses of £1.5bn for the second quarter of its financial year, was hit by a huge outage that affecting millions of customers from receiving or making payments and lasted for almost an entire week.

"The immediate software issue was promptly identified and rectified. Despite this, significant manual intervention in a highly automated and complex batch processing environment was required. This resulted in a significant backlog of daily data and information processing," the firm said in its filing.

2012 upgrade of batch processing
When things go wrong ...

Google explains reasons behind today's 30-minute service outage

Proverbial 'software bug' sent a spiral of bad configurations to other systems

2014 bug/bad config
Lambda Architecture to the rescue!
Let’s step back a bit …

• Nathan Marz (Backtype, Twitter, stealth startup)

• Creator of …
  – Storm
  – Cascalog
  – ElephantDB

http://manning.com/marz/
Lambda Architecture—Requirements

• Fault-tolerant against both hardware failures and human errors

• Support variety of use cases that include low latency querying as well as updates

• Linear scale-out capabilities

• Extensible, so that the system is manageable and can accommodate newer features easily
Lambda Architecture—Concept

query = function(all data)

- **Latency**—the time it takes to run a query
- **Timeliness**—how up to date the query results are
  (→ consistency)
- **Accuracy**—tradeoff between performance and scalability
  (→ approximations)
Lambda Architecture

**IMMUTABLE MASTER DATA** → **PRECOMPUTE VIEWS** → **REAL-TIME VIEWS**

NEW DATA STREAM → **PROCESS STREAM** → **INCREMENT VIEWS**

**BATCH LAYER**

**SERVING LAYER**

**SPEED LAYER**

VIEW 1 → VIEW 2 → VIEW N

**QUERY**
Lambda Architecture—Layers

• **Batch layer**
  – managing the master dataset, an immutable, append-only set of raw data
  – pre-computing arbitrary query functions, called batch views

• **Serving layer** indexes batch views so that they can be queried in ad hoc with low latency

• **Speed layer** accommodates all requests that are subject to low latency requirements. Using fast and incremental algorithms, deals with recent data only
Lambda Architecture—Compensate Batch

Data absorbed into batch view...not absorbed...now
Lambda Architecture—Immutable Data + Views

http://openflights.org
Lambda Architecture—Immutable Data + Views

<table>
<thead>
<tr>
<th>timestamp</th>
<th>airport</th>
<th>flight</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-01-01T10:00:00</td>
<td>DUB</td>
<td>EI123</td>
<td>take-off</td>
</tr>
<tr>
<td>2014-01-01T10:05:00</td>
<td>HEL</td>
<td>SAS45</td>
<td>take-off</td>
</tr>
<tr>
<td>2014-01-01T10:07:00</td>
<td>AMS</td>
<td>BA99</td>
<td>take-off</td>
</tr>
<tr>
<td>2014-01-01T10:09:00</td>
<td>LHR</td>
<td>LH17</td>
<td>landing</td>
</tr>
<tr>
<td>2014-01-01T10:10:00</td>
<td>CDG</td>
<td>AF03</td>
<td>landing</td>
</tr>
<tr>
<td>2014-01-01T10:10:00</td>
<td>FCO</td>
<td>AZ501</td>
<td>take-off</td>
</tr>
</tbody>
</table>

immutable master dataset
### Lambda Architecture—Immutable Data + Views

**immutable master dataset**

<table>
<thead>
<tr>
<th>timestamp</th>
<th>airport</th>
<th>flight</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-01-01T10:00:00</td>
<td>DUB</td>
<td>EI123</td>
<td>take-off</td>
</tr>
<tr>
<td>2014-01-01T10:05:00</td>
<td>HEL</td>
<td>SAS45</td>
<td>take-off</td>
</tr>
<tr>
<td>2014-01-01T10:07:00</td>
<td>AMS</td>
<td>BA99</td>
<td>take-off</td>
</tr>
<tr>
<td>2014-01-01T10:09:00</td>
<td>LHR</td>
<td>LH17</td>
<td>landing</td>
</tr>
<tr>
<td>2014-01-01T10:10:00</td>
<td>CDG</td>
<td>AF03</td>
<td>landing</td>
</tr>
<tr>
<td>2014-01-01T10:10:00</td>
<td>FCO</td>
<td>AZ501</td>
<td>take-off</td>
</tr>
</tbody>
</table>

**airport load:**

<table>
<thead>
<tr>
<th>airport</th>
<th>planes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMS</td>
<td>69</td>
</tr>
<tr>
<td>CDG</td>
<td>44</td>
</tr>
<tr>
<td>DUB</td>
<td>31</td>
</tr>
<tr>
<td>FCO</td>
<td>10</td>
</tr>
<tr>
<td>HEL</td>
<td>17</td>
</tr>
<tr>
<td>LHR</td>
<td>101</td>
</tr>
</tbody>
</table>

**air-borne:** 2307

**air-borne per airline:**

<table>
<thead>
<tr>
<th>airline</th>
<th>planes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>59</td>
</tr>
<tr>
<td>AZ</td>
<td>23</td>
</tr>
<tr>
<td>BA</td>
<td>167</td>
</tr>
<tr>
<td>EI</td>
<td>19</td>
</tr>
<tr>
<td>LH</td>
<td>201</td>
</tr>
<tr>
<td>SAS</td>
<td>28</td>
</tr>
</tbody>
</table>

**views**

```
Implementing the Lambda Architecture
How about an integrated approach?

- Twitter Summingbird
- Lambdoop
- Apache Spark
Apache Spark
Apache Spark

- Originally developed in 2009 in UC Berkeley’s AMP Lab
- As part of BDAS stack open sourced in 2010
- Top-level Apache Project as of 2014

http://spark.apache.org/
Apache Spark—a unified platform …

Continued innovation bringing new functionality, such as:

• **Tachyon** (Shared RDDs, off-heap solution)
• **BlinkDB** (approximate queries)
• **SparkR** (R wrapper for Spark)
Easy and fast Big Data

• Easy to Develop
  – Rich APIs available through Java, Scala, Python
  – Interactive shell

• Fast to Run
  – Advanced data storage model (automated optimization between memory and disk)
  – General execution graphs

2-5× less code

up to 10× faster on disk,
100× in memory

https://amplab.cs.berkeley.edu/benchmark/
… for complex workloads …

• Iterative Algorithms
  – machine learning
  – graph processing beyond DAG

• Interactive Data Mining

• Streaming Applications
... across multiple datasources

- Local Files
  - file:///opt/httpd/logs/access_log

- Object Stores (e.g. Amazon S3)

- HDFS
  - text files, sequence files, any other Hadoop InputFormat

- Key-Value datastores (e.g. Apache HBase)
Easy: expressive API

map          reduce
Easy: expressive API

map  reduce  sample
filter  count  take
groupBy  fold  first
sort  reduceByKey  partitionBy
groupByKey  cogroup  mapWith
groupBy  cross  pipe
gleam  zip  save  ...
Easy: get started immediately

Python
lines = sc.textFile(...) 
lines.filter(lambda s: "ERROR" in s).count()

Scala
val lines = sc.textFile(...) 
lines.filter(x => x.contains("ERROR")).count()

Java
JavaRDD<String> lines = sc.textFile(...) 
lines.filter(new Function<String, Boolean>() {
    Boolean call(String s) {
        return s.contains("error");
    }
}).count();
... and scale as you go (mentally and physically)
Resilient Distributed Datasets (RDD)

- RDDs are the core of the Spark execution engine.
- Collections of elements that can be operated on in parallel.
- Persistent in memory between operations.

 RDD Operations

• Lazy evaluation is key to Spark

• Transformations
  – Creation of a new dataset from an existing:
    map, filter, distinct, union, sample, groupByKey, join, etc.

• Actions
  – Return a value after running a computation:
    collect, count, first, takeSample, foreach, etc.
## RDD persistence

<table>
<thead>
<tr>
<th>Level</th>
<th>Space Used</th>
<th>CPU time</th>
<th>In memory</th>
<th>On Disk</th>
<th>Nodes with data</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMORY_ONLY</td>
<td>High</td>
<td>Low</td>
<td>Y</td>
<td>N</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MEMORY_ONLY_2</td>
<td>High</td>
<td>Low</td>
<td>Y</td>
<td>N</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEMORY_ONLY_SER</td>
<td>Low</td>
<td>High</td>
<td>Y</td>
<td>N</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MEMORY_ONLY_SER_2</td>
<td>Low</td>
<td>High</td>
<td>Y</td>
<td>N</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEMORY_AND_DISK</td>
<td>High</td>
<td>Medium</td>
<td>Some</td>
<td>Some</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MEMORY_AND_DISK_2</td>
<td>High</td>
<td>Medium</td>
<td>Some</td>
<td>Some</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>MEMORY_AND_DISK_SER</td>
<td>Low</td>
<td>High</td>
<td>Some</td>
<td>Some</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MEMORY_AND_DISK_SER_2</td>
<td>Low</td>
<td>High</td>
<td>Some</td>
<td>Some</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DISK_ONLY</td>
<td>Low</td>
<td>High</td>
<td>N</td>
<td>Y</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DISK_ONLY_2</td>
<td>Low</td>
<td>High</td>
<td>N</td>
<td>Y</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

[Source](http://spark.apache.org/docs/latest/scala-programming-guide.html#rdd-persistence)
Spark Streaming

- High-level language operators for streaming data
- Fault-tolerant semantics
- Support for merging streaming data with historical data

Spark Streaming

Run a streaming computation as a series of very small, deterministic batch jobs

- Chop up live stream into batches of X seconds
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Finally, processed results of the RDD operations are returned in batches
Spark Streaming

Run a streaming computation as a series of very small, deterministic batch jobs

- Batch sizes as low as $\frac{1}{2}$ second, latency of about 1 second
- Potential for combining batch processing and streaming processing in the same system

live data stream

batches of $X$ seconds

processed results

Spark Streaming

Spark
Spark Streaming Comparison

- Spark Streaming: 670k records/sec/node
- Storm: 115k records/sec/node
- Commercial systems: 100-500k records/sec/node
And in the real world?
Spotify: Collaborative Filtering with Spark

- collaborative filter (ALS) for music recommendation
- Hadoop suffers from I/O overhead
- show a progression of code rewrites, converting a Hadoop-based app into efficient use of Spark

www.slideshare.net/MrChrisJohnson/collaborative-filtering-with-spark
Cisco: Security Intelligence Operations

Sensor data lands in MapR
Spark Streaming on MapR for first check on known threats
Data next processed on GraphX and Mahout
Additional SQL querying done via Shark and Impala
Leading Pharma Company: NextGen Genomics

Existing process takes several weeks to align chemical compounds with genes

ADAM on Spark allows realignment in a few hours

Geneticists can minimize engineering dependency
Where to go from here
The book: Learning Spark

http://shop.oreilly.com/product/0636920028512.do
Apache Spark developer certificate program

http://oreilly.com/go/sparkcert
Lambda Architecture

A repository dedicated to the Lambda Architecture (LA). We collect and publish examples and good practices around the LA.

Updates

- 27 Aug 2014 • A RAD Stack: Kafka, Storm, Hadoop, and Druid by Druid Committers
- 24 Jul 2014 • Deploop: A Lambda Architecture Provisioning Tool by Javi Roman
- 01 Jul 2014 • Nathan Marz’s Big Data book by Michael Hausenblas
- 30 Jun 2014 • Speed Components by Michael Hausenblas
- 30 Jun 2014 • Serving Components by Michael Hausenblas
- 30 Jun 2014 • Batch Components by Michael Hausenblas
- 22 Jun 2014 • Buildloop: A Lambda Architecture ecosystem builder by Javi Roman
- 20 Jan 2014 • Lambda Architecture: A state-of-the-art by Pere Ferrera
- 19 Jan 2014 • An example Lambda Architecture for real-time analysis of hashtags using Trident, Hadoop and Splout SQL by Pere Ferrera
- 25 Dec 2013 • Twitter Summingbird by Michael Hausenblas
- 25 Dec 2013 • Lambdaop by Michael Hausenblas
- 25 Dec 2013 • Issues in Combined Static and Dynamic Data Management by Michael Hausenblas
- 24 Dec 2013 • Where Polygeist Persistence meets the Lambda Architecture by Michael Hausenblas
- 11 Dec 2013 • A real-time architecture using Hadoop and Storm by Nathan Bijnens
- 10 Dec 2013 • Why are we doing this and why are we doing this now? by Michael Hausenblas

What is the Lambda Architecture?

Nathan Marz came up with the term Lambda Architecture (LA) for a generic, scalable and fault-tolerant data processing architecture, based on his experience working on distributed data processing systems at Backtype and Twitter.
Spark Packages

The folks from Databricks have launched Spark Packages, a community site hosting modules that are not (directly) part of the Apache Spark project. At time of writing the site contains 16 packages including stuff like launch scripts (GCE, Azure, etc.), integrations (for Kafka, Avro, etc.), utils (testing, RDDs, etc.) and...

Read more

Spark 1.2 released

Today, Apache Spark 1.2 has been released with the following highlights: Improved Spark Core Spark Streaming now more or less fully available via Python as well MLLib has been improved GraphX is now stable Big congrats and thanks to the team!...

Read more

http://spark-stack.org
MapR Sandbox with Spark

mapr.com/blog/getting-started-spark-mapr-sandbox
Conclusion

• Let’s scale systems and humans
• How? Lambda Architecture!
• Apache Spark is an efficient way to implement Lambda Architecture