About the book

- @hadooparchbook
- hadooparchitecturebook.com
- github.com/hadooparchitecturebook
- slideshare.com/hadooparchbook
About the presenters

Ted Malaska

• Principal Solutions Architect at Cloudera
• Previously, lead architect at FINRA
• Contributor to Apache Hadoop, HBase, Flume, Avro, Pig and Spark

Jonathan Seidman

• Senior Solutions Architect/Partner Enablement at Cloudera
• Previously, Technical Lead on the big data team at Orbitz Worldwide
• Co-founder of the Chicago Hadoop User Group and Chicago Big
About the presenters

Gwen Shapira
• Solutions Architect turned Software Engineer at Cloudera
• Committer on Apache Sqoop
• Contributor to Apache Flume and Apache Kafka

Mark Grover
• Software Engineer at Cloudera
• Committer on Apache Bigtop, PMC member on Apache Sentry (incubating)
• Contributor to Apache Hadoop, Spark, Hive, Sqoop, Pig and Flume
Logistics

• Break at 10:30-11:00 PM
• Questions at the end of each section
Case Study

Clickstream Analysis
Analytics
Analytics
Analytics
### Demographics

#### Language

<table>
<thead>
<tr>
<th>Language</th>
<th>Sessions</th>
<th>% Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>pl-br</td>
<td>22</td>
<td>51.16%</td>
</tr>
<tr>
<td>en-us</td>
<td>3</td>
<td>6.98%</td>
</tr>
<tr>
<td>pt-pt</td>
<td>3</td>
<td>6.98%</td>
</tr>
<tr>
<td>it-it</td>
<td>2</td>
<td>4.65%</td>
</tr>
<tr>
<td>zh-cn</td>
<td>2</td>
<td>4.65%</td>
</tr>
<tr>
<td>ar-ag</td>
<td>1</td>
<td>2.33%</td>
</tr>
<tr>
<td>de-de</td>
<td>1</td>
<td>2.33%</td>
</tr>
<tr>
<td>en-za</td>
<td>1</td>
<td>2.33%</td>
</tr>
<tr>
<td>es-ar</td>
<td>1</td>
<td>2.33%</td>
</tr>
<tr>
<td>es-cl</td>
<td>1</td>
<td>2.33%</td>
</tr>
</tbody>
</table>
244.157.45.12 - - [17/Oct/2014:21:08:30] "GET /seatposts HTTP/1.0" 200 4463 "http://bestcyclingreviews.com/top_online_shops" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36"

244.157.45.12 - - [17/Oct/2014:21:59:59] "GET /Store/cart.jsp?productID=1023 HTTP/1.0" 200 3757 "http://www.casualcyclist.com" "Mozilla/5.0 (Linux; U; Android 2.3.5; en-us; HTC Vision Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"
Clickstream Analytics

244.157.45.12 - - [17/Oct/2014:21:08:30] "GET /seatposts HTTP/1.0" 200 4463 "http://bestcyclingreviews.com/top_online_shops" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36"
Similar use-cases

• Sensors – heart, agriculture, etc.
• Casinos – session of a person at a table
Pre-Hadoop Architecture

Clickstream Analysis
Click Stream Analysis (Before Hadoop)

Web logs (full fidelity) (2 weeks) → Transform/Aggregate → Data Warehouse → Business Intelligence

Tape Archive
Problems with Pre-Hadoop Architecture

• Full fidelity data is stored for small amount of time (~weeks).
• Older data is sent to tape, or even worse, deleted!
• Inflexible workflow - think of all aggregations beforehand
Effects of Pre-Hadoop Architecture

- Regenerating aggregates is expensive or worse, impossible
- Can’t correct bugs in the workflow/aggregation logic
- Can’t do experiments on existing data
Why is Hadoop A Great Fit?

Clickstream Analysis
Why is Hadoop a great fit?

- Volume of clickstream data is huge
- Velocity at which it comes in is high
- Variety of data is diverse - semi-structured data
- Hadoop enables
  - active archival of data
  - Aggregation jobs
  - Querying the above aggregates or raw fidelity data
Click Stream Analysis (with Hadoop)

Web logs → Hadoop → Business Intelligence

Active archive (no tape)
Aggregation engine
Querying engine
Challenges of Hadoop Implementation
Challenges of Hadoop Implementation
Other challenges - Architectural Considerations

• Storage managers?
  – HDFS? HBase?

• Data storage and modeling:
  – File formats? Compression? Schema design?

• Data movement
  – How do we actually get the data into Hadoop? How do we get it out?

• Metadata
  – How do we manage data about the data?

• Data access and processing
  – How will the data be accessed once in Hadoop? How can we transform it? How do we query it?

• Orchestration
  – How do we manage the workflow for all of this?
Case Study
Requirements
Overview of Requirements
Overview of Requirements

Data Sources → Ingestion → Raw Data Storage (Formats, Schema) → Processing → Processed Data Storage (Formats, Schema) → Data Consumption

Orchestration (Scheduling, Managing, Monitoring)
Case Study

Requirements

Data Ingestion
Data Ingestion Requirements

• So we need to be able to support:
  – Reliable ingestion of large volumes of semi-structured event data arriving with high velocity (e.g. logs).
  – Timeliness of data availability – data needs to be available for processing to meet business service level agreements.
  – Periodic ingestion of data from relational data stores.
Case Study
Requirements
Data Storage
Data Storage Requirements

- Store all the data
- Make the data accessible for processing
- Compress the data
Case Study
Requirements
Data Processing
Processing requirements

Be able to answer questions like:

• What is my website’s bounce rate?
  – i.e. how many % of visitors don’t go past the landing page?

• Which marketing channels are leading to most sessions?

• Do attribution analysis
  – Which channels are responsible for most conversions?
Case Study
Requirements
Orchestration
Orchestration is simple
We just need to execute actions
One after another
Actually,
we also need to handle errors
And user notifications
....
And…

• Re-start workflows after errors
• Reuse of actions in multiple workflows
• Complex workflows with decision points
• Trigger actions based on events
• Tracking metadata
• Integration with enterprise software
• Data lifecycle
• Data quality control
• Reports
OK, maybe we need a product
To help us do all that
Data Modeling Considerations

• We need to consider the following in our architecture:
  – File system schemas – how will we lay out the data?
  – File formats – what storage formats to use for our data, both raw and processed data?
  – Data compression formats?
Architectural Considerations

Data Modeling – Storage Layer
Data Storage Layer Choices

• Two likely choices for raw data:
Data Storage Layer Choices

- Stores data directly as files
- Fast scans
- Poor random reads/writes

- Stores data as Hfiles on HDFS
- Slow scans
- Fast random reads/writes
Data Storage – Storage Manager
Considerations

• Incoming raw data:
  – Processing requirements call for batch transformations across multiple records – for example sessionization.

• Processed data:
  – Access to processed data will be via things like analytical queries – again requiring access to multiple records.

• We choose HDFS
  – Processing needs in this case served better by fast scans.
Architectural Considerations

Data Modeling – Raw Data Storage
Storage Formats – Raw Data and Processed Data
Data Storage – Format Considerations

Logs (plain text)
Data Storage – Format Considerations
Data Storage – Compression

Well, maybe. But not splittable.

Splittable. Getting better…

Splittable, but no.

Hmmm….
Raw Data Storage – More About Snappy

- Designed at Google to provide high compression speeds with reasonable compression.
- Not the highest compression, but provides very good performance for processing on Hadoop.
- Snappy is not splittable though, which brings us to…
Hadoop File Types

• Formats designed specifically to store and process data on Hadoop:
  – File based – SequenceFile
  – Serialization formats – Thrift, Protocol Buffers, Avro
  – Columnar formats – RCFile, ORC, Parquet
SequenceFile

- Stores records as binary key/value pairs.
- SequenceFile “blocks” can be compressed.
- This enables splittability with non-splittable compression.
Avro

- Kinda SequenceFile on Steroids.
- Self-documenting – stores schema in header.
- Provides very efficient storage.
- Supports splittable compression.
Our Format Recommendations for Raw Data…

- **Avro with Snappy**
  - Snappy provides optimized compression.
  - Avro provides compact storage, self-documenting files, and supports schema evolution.
  - Avro also provides better failure handling than other choices.

- **SequenceFiles** would also be a good choice, and are directly supported by ingestion tools in the ecosystem.
  - But only supports Java.
But Note…

• For simplicity, we’ll use plain text for raw data in our example.
Architectural Considerations

Data Modeling – Processed Data Storage
Storage Formats – Raw Data and Processed Data
## Access to Processed Data

### Analytical Queries

<table>
<thead>
<tr>
<th>Column A</th>
<th>Column B</th>
<th>Column C</th>
<th>Column D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
</tr>
<tr>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
</tr>
<tr>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
</tr>
<tr>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
</tr>
</tbody>
</table>
Columnar Formats

- Eliminates I/O for columns that are not part of a query.
- Works well for queries that access a subset of columns.
- Often provide better compression.
- These add up to dramatically improved performance for many queries.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th align="right"></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td align="right">2014-10-1</td>
<td>abc</td>
</tr>
<tr>
<td>2</td>
<td align="right">2014-10-1</td>
<td>def</td>
</tr>
<tr>
<td>3</td>
<td align="right">2014-10-1</td>
<td>ghi</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th align="right"></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td align="right">2014-10-1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td align="right">2014-10-1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td align="right">2014-10-1</td>
<td>def</td>
</tr>
<tr>
<td></td>
<td align="right">2014-10-1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td align="right">2014-10-1</td>
<td>ghi</td>
</tr>
</tbody>
</table>
Columnar Choices – RCFile

- Designed to provide efficient processing for Hive queries.
- Only supports Java.
- No Avro support.
- Limited compression support.
- Sub-optimal performance compared to newer columnar formats.
Columnar Choices – ORC

- A better RCFile.
- Also designed to provide efficient processing of Hive queries.
- Only supports Java.
Columnar Choices – Parquet

• Designed to provide efficient processing across Hadoop programming interfaces – MapReduce, Hive, Impala, Pig.
• Multiple language support – Java, C++
• Good object model support, including Avro.
• Broad vendor support.
• These features make Parquet a good choice for our processed data.
Architectural Considerations

Data Modeling – Schema Design
HDFS Schema Design – One Recommendation

/etl – Data in various stages of ETL workflow
/data – processed data to be shared data with the entire organization
/tmp – temp data from tools or shared between users
/user/<username> - User specific data, jars, conf files
/app – Everything but data: UDF jars, HQL files, Oozie workflows
Partitioning

• Split the dataset into smaller consumable chunks.
• Rudimentary form of “indexing”. Reduces I/O needed to process queries.
Partitioning

Un-partitioned HDFS directory structure

dataset
  file1.txt
  file2.txt
  ...
  filen.txt

Partitioned HDFS directory structure

dataset
  col=val1/file.txt
  col=val2/file.txt
  ...
  col=valn/file.txt
Partitioning considerations

- What column to partition by?
  - Don’t have too many partitions (<10,000)
  - Don’t have too many small files in the partitions
  - Good to have partition sizes at least ~1 GB, generally a multiple of block size.

- We’ll partition by timestamp. This applies to both our raw and processed data.
Partitioning For Our Case Study

• Raw dataset:
  – /etl/BI/casualcyclist/clicks/rawlogs/year=2014/month=10/day=10

• Processed dataset:
  – /data/bikeshop/clickstream/year=2014/month=10/day=10
Architectural Considerations

Data Ingestion
Typical Clickstream data sources

- Omniture data on FTP
- Apps
- App Logs
- RDBMS
Getting Files from FTP
Don’t over-complicate things

curl ftp://myftpsite.com/sitecatalyst/
myreport_2014-10-05.tar.gz
--user name:password | hdfs -put - /etl/clickstream/raw/
sitecatalyst/myreport_2014-10-05.tar.gz
Apache NiFi
Event Streaming – Flume and Kafka

Reliable, distributed and highly available systems
That allow streaming events to Hadoop
Flume:

- Many available data collection sources
- Well integrated into Hadoop
- Supports file transformations
- Can implement complex topologies
- Very low latency
- No programming required
We use Flume when:

“We just want to grab data from this directory and write it to HDFS”
Kafka is:

- Very high-throughput publish-subscribe messaging
- Highly available
- Stores data and can replay
- Can support many consumers with no extra latency
Use Kafka When:

“Kafka is awesome. We heard it cures cancer”
Actually, why choose?

• Use Flume with a Kafka Source
• Allows to get data from Kafka, run some transformations, write to HDFS, HBase or Solr
In Our Example…

• We want to ingest events from log files
• Flume’s Spooling Directory source fits
• With HDFS Sink

• We would have used Kafka if…
  – We wanted the data in non-Hadoop systems too
Short Intro to Flume

Sources
- Twitter, logs, JMS, webserver, Kafka

Interceptors
- Mask, re-format, validate...

Selectors
- DR, critical

Channels
- Memory, file, Kafka

Sinks
- HDFS, HBase, Solr

Flume Agent
Configuration

• Declarative
  – No coding required.
  – Configuration specifies how components are wired together.
Interceptors

- Mask fields
- Validate information against external source
- Extract fields
- Modify data format
- Filter or split events
Any sufficiently complex configuration is indistinguishable from code
A Brief Discussion of Flume Patterns – Fan-in

- Flume agent runs on each of our servers.
- These client agents send data to multiple agents to provide reliability.
- Flume provides support for load balancing.
A Brief Discussion of Flume Patterns – Splitting

- Common need is to split data on ingest.
- For example:
  - Sending data to multiple clusters for DR.
  - To multiple destinations.
- Flume also supports partitioning, which is key to our implementation.
Flume Architecture – Client Tier
Flume Architecture – Collector Tier

- Flume Agent
- Avro Source
- File Channel
- HDFS Sink
- HDFS
What if…. We were to use Kafka?

- Add Kafka producer to our webapp
- Send clicks and searches as messages
- Flume can ingest events from Kafka
- We can add a second consumer for real-time processing in SparkStreaming
- Another consumer for alerting…
- And maybe a batch consumer too
The Kafka Channel
The Kafka Channel

- **Sources**: Twitter, logs, JMS, webserver
- **Interceptors**: Mask, re-format, validate...
- **Channels**: Kafka
- **Kafka Consumers**: Consumer A, Consumer B, Consumer C
- **Flume Agent**
The Kafka Channel

Twitter, logs, JMS, webserver → Flume Agent → Interceptors → Selectors → Channels → Kafka → Sinks → HDFS, HBase, Solr

- Sources: Twitter, logs, JMS, webserver
- Interceptors: Mask, re-format, validate...
- Selectors: DR, critical
- Channels: Kafka
- Sinks: HDFS, HBase, Solr
Architectural Considerations

Data Processing – Engines

tiny.cloudera.com/app-arch-slides
Processing Engines

- MapReduce
- Abstractions
- Spark
- Spark Streaming
- Impala
MapReduce

- Oldie but goody
- Restrictive Framework / Innovated Work Around
- Extreme Batch
MapReduce Basic High Level

Mapper

Reducer

Block of Data

Temp Spill Data

Partitioned Sorted Data

Reducer Local Copy

Output File

HDFS (Replicated)

Native File System

Remote read for all but 1 node
MapReduce Innovation

- Mapper Memory Joins
- Reducer Memory Joins
- Buckets Sorted Joins
- Cross Task Communication
- Windowing
- And Much More
Abstractions

• SQL
  – Hive

• Script/Code
  – Pig: Pig Latin
  – Crunch: Java/Scala
  – Cascading: Java/Scala
Spark

- The New Kid that isn’t that New Anymore
- Easily 10x less code
- Extremely Easy and Powerful API
- Very good for machine learning
- Scala, Java, and Python
- RDDs
- DAG Engine
Spark - DAG
Spark - DAG

TextFile
  Good
  Good
  Good

Filter
  Good
  Good
  Good

KeyBy
  Good-Replay
  Good-Replay
  Good-Replay

Join
  Lost Block
  Good

Filter
  Future
  Future

Take
  Future
  Future

KeyBy
  Good-Replay

Lost Block Replay
  Good-Replay
  Good-Replay
Spark Streaming

• Calling Spark in a Loop
• Extends RDDs with DStream
• Very Little Code Changes from ETL to Streaming
Spark Streaming

Pre-first Batch

Source → Receiver → RDD

First Batch

Source → Receiver → RDD → Single Pass
  |          | Filter → Count → Print

Second Batch

Source → Receiver → RDD → Single Pass
  |          | Filter → Count → Print

RDD
**Impala**

- MPP Style SQL Engine on top of Hadoop
- Very Fast
- High Concurrency
- Analytical windowing functions (C5.2).
Impala – Broadcast Join

Impala Daemon

100% Cached Smaller Table

Smaller Table Data Block

Bigger Table Data Block

Impala Daemon

100% Cached Smaller Table

Smaller Table Data Block

Impala Daemon

100% Cached Smaller Table

Smaller Table Data Block

Output

Impala Daemon

Hash Join Function

100% Cached Smaller Table

Bigger Table Data Block

Output

Impala Daemon

Hash Join Function

100% Cached Smaller Table

Bigger Table Data Block

Output

Impala Daemon

Hash Join Function

100% Cached Smaller Table

Bigger Table Data Block
Impala – Partitioned Hash Join

Impala Daemon
Hash Partitioner
~33% Cached Smaller Table
Smaller Table Data Block
Output

Impala Daemon
Hash Partitioner
~33% Cached Smaller Table
Smaller Table Data Block
Output

Impala Daemon
Hash Partitioner
~33% Cached Smaller Table
Smaller Table Data Block
Output

Impala Daemon
Hash Partitioner
33% Cached Smaller Table
BiggerTable Data Block
Hash Join Function
Output

Impala Daemon
Hash Partitioner
33% Cached Smaller Table
BiggerTable Data Block
Hash Join Function
Output

Impala Daemon
Hash Partitioner
33% Cached Smaller Table
BiggerTable Data Block
Hash Join Function
Output

Confidentiality Information Goes Here
Impala vs Hive

• Very different approaches and
• We may see convergence at some point
• But for now
  – Impala for speed
  – Hive for batch
Architectural Considerations

Data Processing – Patterns and Recommendations
What processing needs to happen?

• Sessionization
• Filtering
• Deduplication
• BI / Discovery
Sessionization

Visitor 1
Session 1

Visitor 1
Session 2

Visitor 2
Session 1

> 30 minutes
Why sessionize?

Helps answers questions like:

• What is my website’s bounce rate?
  – i.e. how many % of visitors don’t go past the landing page?

• Which marketing channels (e.g. organic search, display ad, etc.) are leading to most sessions?
  – Which ones of those lead to most conversions (e.g. people buying things, signing up, etc.)

• Do attribution analysis – which channels are responsible for most conversions?
Sessionization

244.157.45.12 - - [17/Oct/2014:21:08:30] "GET /seatposts HTTP/1.0" 200 4463 "http://bestcyclingreviews.com/top_online_shops" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36"

244.157.45.12+1413580110

244.157.45.12 - - [17/Oct/2014:21:59:59] "GET /Store/cart.jsp?productID=1023 HTTP/1.0" 200 3757 "http://www.casualcyclist.com" "Mozilla/5.0 (Linux; U; Android 2.3.5; en-us; HTC Vision Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1" 244.157.45.12+1413583199
How to Sessionize?

1. Given a list of clicks, determine which clicks came from the same user (Partitioning, ordering)
2. Given a particular user's clicks, determine if a given click is a part of a new session or a continuation of the previous session (Identifying session boundaries)
#1 – Which clicks are from same user?

- We can use:
  - IP address (244.157.45.12)
  - Cookies (A9A3BECE0563982D)
  - IP address (244.157.45.12) and user agent string (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36)
#1 – Which clicks are from same user?

- We can use:
  - IP address (244.157.45.12)
  - Cookies (A9A3BECE0563982D)
  - IP address (244.157.45.12) and user agent string (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36)
#1 – Which clicks are from same user?

244.157.45.12 - - [17/Oct/2014:21:08:30 ] "GET /seatposts HTTP/1.0" 200 4463 "http://bestcyclingreviews.com/top_online_shops" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36"

244.157.45.12 - - [17/Oct/2014:21:59:59 ] "GET /Store/cart.jsp?productID=1023 HTTP/1.0" 200 3757 "http://www.casualcyclist.com" "Mozilla/5.0 (Linux; U; Android 2.3.5; en-us; HTC Vision Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"
#2 – Which clicks part of the same session?

244.157.45.12 - - [17/Oct/2014:21:08:30] "GET /seatposts HTTP/1.0" 200 4463 "http://bestcyclingreviews.com/top_online_shops" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36"

244.157.45.12 - - [17/Oct/2014:21:59:59] "GET /Store/cart.jsp?productID=1023 HTTP/1.0" 200 3757 "http://www.casualcyclist.com" "Mozilla/5.0 (Linux; U; Android 2.3.5; en-us; HTC Vision Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"

> 30 mins apart = different sessions
Sessionization engine recommendation

• We have sessionization code in MR and Spark on github. The complexity of the code varies, depends on the expertise in the organization.

• We choose MR
  – MR API is stable and widely known
  – No Spark + Oozie (orchestration engine) integration currently
Filtering – filter out incomplete records

244.157.45.12 -- [17/Oct/2014:21:08:30] "GET /seatposts HTTP/1.0" 200 4463 "http://bestcyclingreviews.com/top_online_shops" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36"

Filtering – filter out records from bots/spiders

244.157.45.12 - - [17/Oct/2014:21:08:30 ] "GET /seatposts HTTP/1.0" 200 4463 "http://bestcyclingreviews.com/top_online_shops" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1944.0 Safari/537.36"

209.85.238.11 - - [17/Oct/2014:21:59:59 ] "GET /Store/cart.jsp?productID=1023 HTTP/1.0" 200 3757 "http://www.casualcyclist.com" "Mozilla/5.0 (Linux; U; Android 2.3.5; en-us; HTC Vision Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"

Google spider IP address
Filtering recommendation

- Bot/Spider filtering can be done easily in any of the engines
- Incomplete records are harder to filter in schema systems like Hive, Impala, Pig, etc.
- Flume interceptors can also be used
- Pretty close choice between MR, Hive and Spark
- Can be done in Spark using rdd.filter()
- We can simply embed this in our MR sessionization job
Deduplication – remove duplicate records
Deduplication recommendation

• Can be done in all engines.
• We already have a Hive table with all the columns, a simple DISTINCT query will perform deduplication
• reduce() in spark
• We use Pig
BI/Discovery engine recommendation

• Main requirements for this are:
  – Low latency
  – SQL interface (e.g. JDBC/ODBC)
  – Users don’t know how to code

• We chose Impala
  – It’s a SQL engine
  – Much faster than other engines
  – Provides standard JDBC/ODBC interfaces
End-to-end processing

- Deduplication
- Filtering
- Sessionization

BI tools
Architectural Considerations

Orchestration
Orchestrating Clickstream

• Data arrives through Flume
• Triggers a processing event:
  – Sessionize
  – Enrich – Location, marketing channel…
  – Store as Parquet
• Each day we process events from the previous day
Choosing Right

• Workflow is fairly simple
• Need to trigger workflow based on data
• Be able to recover from errors
• Perhaps notify on the status
• And collect metrics for reporting
Oozie or Azkaban?
Oozie Architecture
Oozie features

- Part of all major Hadoop distributions
- Hue integration
- Built-in actions – Hive, Sqoop, MapReduce, SSH
- Complex workflows with decisions
- Event and time based scheduling
- Notifications
- SLA Monitoring
- REST API
Oozie Drawbacks

- Overhead in launching jobs
- Steep learning curve
- XML Workflows
Azkaban features

• Simplicity
• Great UI – including pluggable visualizers
• Lots of plugins – Hive, Pig…
• Reporting plugin
Azkaban Limitations

- Doesn’t support workflow decisions
- Can’t represent data dependency
Choosing…

- Workflow is fairly simple
- Need to trigger workflow based on data
- Be able to recover from errors
- Perhaps notify on the status
- And collect metrics for reporting

Easier in Oozie
Choosing the right Orchestration Tool

- Workflow is fairly simple
- Need to trigger workflow based on data
- Be able to recover from errors
- Perhaps notify on the status
- And collect metrics for reporting

Better in Azkaban
Important Decision Consideration!

The best orchestration tool is the one you are an expert on
Orchestration Patterns – Fan Out
Putting It All Together

Final Architecture
Final Architecture – High Level Overview

Data Sources → Ingestion → Raw Data Storage (Formats, Schema) → Processing → Processed Data Storage (Formats, Schema) → Data Consumption

Orchestration (Scheduling, Managing, Monitoring)
Final Architecture – High Level Overview

Data Sources

Ingestion

Raw Data Storage (Formats, Schema)

Processing

Processed Data Storage (Formats, Schema)

Data Consumption

Orchestration (Scheduling, Managing, Monitoring)
Final Architecture – Ingestion/Storage

Fan-in Pattern

Web Server → Flume Agent
Web Server → Flume Agent
Web Server → Flume Agent
Web Server → Flume Agent
Web Server → Flume Agent
Web Server → Flume Agent
Web Server → Flume Agent
Web Server → Flume Agent

Flume Agent
Flume Agent
Flume Agent
Flume Agent

Multi Agents for Failover and rolling restarts

HDFS

Path: /etl/BI/casualcyclist/clicks/rawlogs/year=2014/month=10/day=10
Final Architecture – High Level Overview

Data Sources → Ingestion → Raw Data Storage (Formats, Schema) → Processing → Processed Data Storage (Formats, Schema) → Data Consumption

Orchestration (Scheduling, Managing, Monitoring)
Final Architecture – Processing and Storage

/etl/BI/casualcyclist/clicks/rawlogs/year=2014/month=10/day=10...

dedup->filtering->sessionization

parquetize

/data/bikeshop/clickstream/year=2014/month=10/day=10...
Final Architecture – High Level Overview

Data Sources → Ingestion → Raw Data Storage (Formats, Schema) → Processing → Orchestration (Scheduling, Managing, Monitoring) → Processed Data Storage (Formats, Schema) → Data Consumption
Final Architecture – Data Access

- Hive/Impala
- BI/Analytics Tools
- DWH
- Local Disk
- R, etc.

Tools:
- JDBC/ODBC
- Sqoop
- File export
- DB import tool

©2014 Cloudera, Inc. All Rights Reserved.
Demo
Join the Discussion

Hello, Cloudera Customers and Users!

These community forums are intended for developers and admins using Cloudera’s Apache Hadoop-based platform to welcome your suggestions and feedback here.

Join this community to get a 40% discount for O’Reilly Media print books, and 50% for e-books and videos (bundles not included) — as well as to participate in upstream open source projects, use their respective upstream mailing lists.

Ask a Question

Type your question here...

Continue

Community

News (2 Items)

Title Post

Community Guidelines & News
Latest Post - This community is now mobile-friendly

Release Announcements
Latest Post - Announcing: New Cloudera ODBC drivers for Impala a...

Get community help or provide feedback

cloudera.com/community
Try Cloudera Live today—cloudera.com/live
Visit us at Booth #809

HIGHLIGHTS:

Apache Kafka is now fully supported with Cloudera

Learn why Cloudera is the leader for security in Hadoop
Free books and Lots of Q&A!

• Ask Us Anything!
  – Feb 19\textsuperscript{th}, 1:30-2:10PM in 211 B

• Book signings
  – Feb 19\textsuperscript{th}, 11:15PM in Expo Hall – Cloudera Booth (#809)
  – Feb 19\textsuperscript{th}, 3:00PM in Expo Hall - O'Reilly Booth

• Stay in touch!
  – @hadooparchbook
  – hadooparchitecturebook.com
  – slideshare.com/hadooparchbook