Orbitz Worldwide

When You Can't Start From Scratch

Building a Data Pipeline with the Tools You Have

Oct 1, 2014

bigdataeverywhere Chicago

ORBITZ

Agenda

- Introduction
- Motivation
- Consumption
- Storage at Rest
- Transport
- Dead Simple Data Collection
- Key Takeaways

About Us

- Steve Hoffman
  - Senior Principal Engineer - Operations
  - @bacoboy
  - Author of Apache Flume Book (bitly.com/flumebook)
    - Conference Discount until Oct 8
      - Print books: qLyVgEb5d
      - eBook - gON73ZL77
- Ken Dallmeyer
  - Lead Engineer – Machine Learning
- We work at Orbitz – www.orbitz.com
  - Leading Online Travel Agency for 14 years
  - Always Hiring! @OrbitzTalent

Motivation

- What we have:
  - Big Website -> Mountains of Logs
- What we want:
  - Finding customer insight in logs
- What we do:
  - Spending disproportionate amount of time scrubbing logs into parsable data
  - Multiple 1-off transports into Hadoop

Logs != Data

- Logs
  - Humans read them
  - Short lifespan – what's broken now?
- Data
  - Programs read them
  - Long lifespan – Find trends over a period of time
- Developer changes logs 'cause its useful to them -> breaks your MR job.

Look Familiar?

```
20140922-100028.260|I|loyalty-003|loyalty-1.23-
0|ORB|3F3BA823C747FF17~wl00000000000000422c3b14201cdd0|3a
3f3b95||com.orbitz.loyalty.LoyaltyDataServiceImpl:533|Loya
lty+Member+ABC123ZZ+has+already+earned+USD18.55+and+is+eli
gible+for+USD31.45
```



```
{"timestamp":"1411398028260","server":"loyalty-003",
"pos":"ORB","sessionID":"3F3BA823C747FF17",
"requestID":"wl00000000000000422c3b14201cdd0",
"loyalityID":"ABC123ZZ”,"loyalityEarnedAmount":"USD18.55",
"loyalityEligibleAmount":"USD31.45"}
```

Are we asking the correct question?

- Not “How should I store data?”

- But, How do people consume the data?
  - Through Queries? Hive
  - Through Key-Value lookups? Hbase
  - Custom code? MapReduce
  - Existing data warehouse?
  - Web UI/Command Line/Excel(gasp)?

  Start with consumption and work backwards

Consumption

- Hive Tables turned out to be the Orbitz common denominator
- We like Hive because
  - SQL ~= HQL – people understand tables/columns
  - Its a lightweight queryable datasource
  - Something easy to change without a lot of overhead
  - Can join with other people's hive tables
- BUT…
  - Each table was its own MapReduce job
- Too much time spent hunting/scrubbing data than actually using it

Storage at Rest

- How can we generalize to our data feed to be readable by Hive?
- Options:
  - Character delimited text files
    - But brittle to change
    - Cannot remove fields
    - Order matters
  - Avro records
    - Schema defines table
    - Tight coupling with transport to HDFS handoff or verbose passing schema
    - Changes mean re-encoding old data to match new schema
  - HBase
    - Good for flexibility
    - Key selection is very important and hard to change
    - Bad for ad-hoc non-key querying

Storage at Rest

- Our solution:
  - Use Avro with a Map<String, String> schema for storage
  - A custom Hive SerDe to map Hive columns to keys in the map.

- Storage is independent from Consumption
- New keys just sit until Hive mapping updated
- Deleted keys return NULL if not there

- Only Requirements:
  - Bucket Name (aka table name)
  - Timestamp (UTC)

Storage at Rest

- Stored in
  - hdfs://server/root/#{bucket}/#{YYYY}/#{MM}/#{DD}/#{HH}/log.XXXXX.avro
  - Avro Schema: Map<String,String>
- Create external hive table:

```
CREATE EXTERNAL TABLE foo (
  col1 STRING,
  col2 INT
)
PARTITIONED BY (
  dt STRING,
  hour STRING
)
ROW FORMAT SERDE 'com.orbitz.hadoop.hive.avro.AvroSerDe'
STORED AS
INPUTFORMAT
    'com.orbitz.hadoop.hive.avro.AvroContainerInputFormat'
OUTPUTFORMAT
    'com.orbitz.hadoop.hive.avro.AvroContainerOutputFormat'
LOCATION '/root/foo'
TBLPROPERTIES(
'column.mapping' = 'col1,col2'
);
```

Care and Feeding

- Issues
  - Hive partitions aren't automatically added. (Vote for HIVE-6589).
    - A cron job to add a new partition every hour.
- Nice to Have
  - Would be nice to extend schema rather than set properties
    - `col1 STRING KEYED BY 'some_other_key'`

Transport

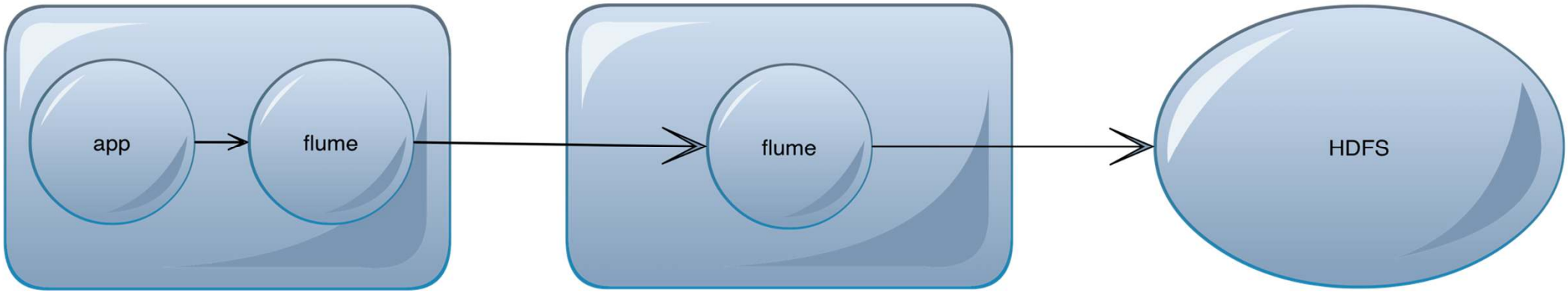
- Lots of Options at the time
  - Flume
  - Syslog variants
  - Logstash
- Newer options
  - Storm
  - Spark
  - Kite SDK
- And probably so many more

Transport

- We chose Flume, since at the time it was best option
  - HDFS aware
  - Did time-bucketing
  - Provided a buffering tier
    - Inevitable Hadoop maintenance
    - Isolates us from Hadoop versions and incompatibilities (less of an issue today)
- Have ‘localhost’ agent to simplify configuration
  - Use provisioning tool to externalize configuration of where to send for the environments

Transport Plan

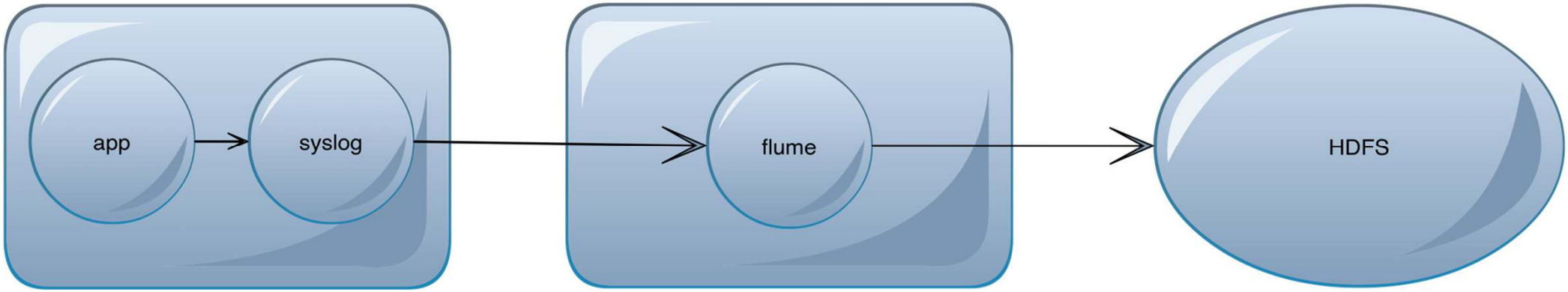
- Application writes generic JSON payload using Avro client to local Flume agent
- Local Flume agent forwards to collector tier
- Collector Tier to HDFS



- However, an additional Java agent on every application server = big memory footprint

Transport Updated

- Application write JSON to local syslog agent
  - (already there doing log work /var/log/*)
- Local syslog agent to flume collector tier
- Flume collector to HDFS



Care and Feeding

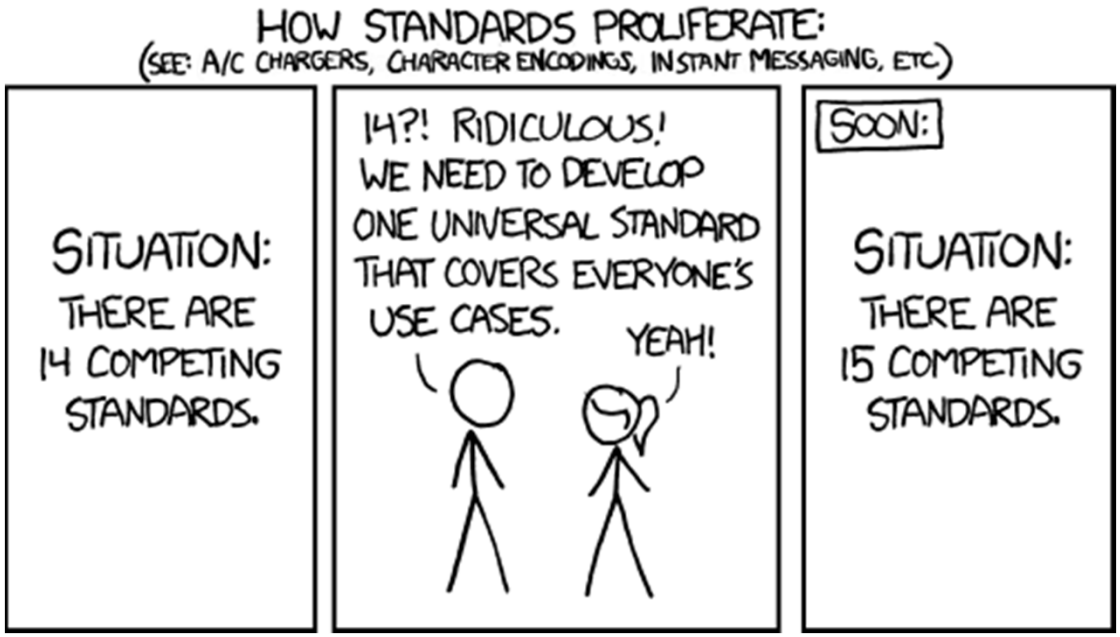
- Issues
  - Hive partitions aren't automatically added. (Vote for HIVE-6589).
    - A cron job to add a new partition every hour.
  - Flume streaming data creates lots of little files (Bad for NameNode)
    - A cron job to combine many tiny poorly compresed files into 1 better compressed avro file once per hour (similar to in functionality to HBase compaction)
    - Create custom serializer to write Map<String,String> instead of default Flume Avro record format.
  - Syslog
    - Need to pass single line of data in syslog format. Multiple lines, non-ascii, etc. would cause problems. Just need to make sure JSON coming in has special characters escaped out.

Dead Simple Data Collection

- Want a low barrier to entry. Think log4j or another simple API

```
public sendData(Map<String,String> myData);
```

But Beware of creating a new standard…



http://xkcd.com/927/

Dead Simple Data Collection

- Thought about using Flume log4j Appender, but would have to wrap JSON payload creation anyway
  - Logs != Data
  - log.warn(DATA) ??
- We already were using ERMA which was close enough to this. May not be right choice for you.
  - Create custom ERMA monitor processor and renderer to create the payload for syslog
    - Make sure it was RFC5424
    - Assemble JSON payload
    - Add “default fields” like timestamp, session id, etc.

Dead Simple Data Collection

- But what about more complicated data structures?
- Flatten Keys with dot notation
- Hash?
  - `{a:{b:6}}  {a.b:6}`
- Arrays?
  - `{ts:4, data:[4,6,7]} {ts:4, data.0:4, data.1:6, data.2:7, data.length=3}`
- It depends… Again think consumption first – Hive tables are flat

ORBITZ

Care and Feeding

- Issues
  - Hive partitions aren't automatically added. (Vote for HIVE-6589).
    - A cron job to add a new partition every hour.
  - Flume streaming data creates lots of little files (Bad for NameNode)
    - A cron job to combine many tiny poorly compresed files into 1 better compressed avro file once per hour (similar to in functionality to HBase compaction)
    - Create custom serializer to write Map<String,String> instead of default Flume Avro record format.
  - Syslog
    - Need to pass single line of data in syslog format. Multiple lines, non-ascii, etc. would cause problems. Just need to make sure JSON coming in has special characters escaped out.
  - Application
    - Whatever data emitter we choose, needs to be async
  - Scaling and Monitoring
    - Be aware that as we add more applications, we will need to scale the Collectors and Hadoop

Key Takeaways

- How you consume the data should drive your solution
- Decouple Storage from API and Transport
- If 100% persistence then use a DATABASE.
- Use in-memory when possible
  - much faster than disk = less hardware you have to buy === value of the data/ what is really lost if you lost an hour/day/all? how soon to recover
- Minimize transforms at source, en-route, and destination
- Minimize hops from Source to Destination
- Data as a Minimal Viable Product, not a data warehouse – grow organically as your applications will.

Thanks

Thanks!

&

Questions?