Real-time Streaming Analysis for Hadoop and Flume

Aaron Kimball
odiago, inc.
OSCON Data 2011
The plan

- Background: Flume introduction
- The need for online analytics
- Introducing FlumeBase
- Demo!
- FlumeBase architecture
- Wrap up
Flume is...

- A distributed data transport and aggregation system for event- or log-structured data
- Principally designed for continuous data ingestion into Hadoop... But more flexible than that
Flume terminology

- Every machine in Flume is a “node”
- Each node has a “source” and a “sink”
  - Example source: tail("/var/log/httpd/access_log")
  - Example sink: dfs("hdfs://namenode/logs/%{host}/%Y%M%D")
- Some sinks send data to “collector” nodes, which aggregate data from many agents before writing to HDFS
Flume control plane

- All Flume nodes heartbeat to/receive config from master
- Operator tools interact with the master via a Thrift API – e.g., the Flume shell
- Nodes can be reconfigured to use different sources, sinks
Real-time data moves through Flume

- Events enter Flume within seconds of generation
- Hadoop MapReduce analysis runs at best once/10 minutes
- Desirable behavior: analyze this data on-the-fly
  - Ad campaign cut-off
  - Real-time personalization, recommendations
  - Load and performance monitoring
  - Error alerting
But Flume isn’t an analytic system

- No ability to inspect message bodies
- No notion of aggregates, rolling counters, etc
  - ... or even filters
But Flume isn’t an analytic system

- No ability to inspect message bodies
- No notion of aggregates, rolling counters, etc
  - ... or even filters
- This leads to fascinating hacks (see right)
Flume and Flexibility

- New sources, sinks can be added from plugins
- Flume topology can be dynamically reconfigured by sending commands to master over Thrift API
- Contents of Flume events (messages) are uninterpreted
Flume and Flexibility

- New sources, sinks can be added from plugins
- Flume topology can be dynamically reconfigured by sending commands to master over Thrift API
- Contents of Flume events (messages) are uninterpreted

- \dots Meaning we can define *new endpoints* for Flume data, store arbitrary data in events, and *control Flume programmatically*. 
FlumeBase: Online Analytics for Flume

Agents

Clickstream

Collector

HDFS

Query output is another Flume data stream

FlumeBase server

Client shell for submitting queries, viewing output
FlumeBase server

• Runs <em>persistent queries</em> analyzing data streams
  – Events interpreted relative to a user-specified schema, parser
• Transparently reconfigures source Flume nodes to tee data
• Acts as a Flume node
  – Output events are just another Flume data stream
rtsql: FlumeBase’s query language

- SQL-like language for defining event schemas, queries

```
CREATE STREAM foo(status INT, msg STRING, priority INT)
    FROM NODE 'backend-server-5';

SELECT * FROM foo WHERE priority > 10;
```
rtsql language features

• Lots of standard SQL features available
  – SELECT, WHERE, GROUP BY, HAVING, JOIN...

• Streams are infinite: GROUP BY and JOIN both use *windowing* to operate over rolling time windows of events
  – Standard aggregate functions: COUNT, MIN, MAX, SUM, AVG
Demo time

• (Buckle your seatbelts)
Under the hood...

Client process

- Submits queries

- Output data printed to client console

FlumeBase Server

- rtsql compiler
- Stream dictionary
- Flume controller

Flow operator DAG

- EventParser

Flume "in" node

Generates

Flow

Incoming data from Flume network

Flume "out" node

Emitted records return to Flume network

Output data printed to client console
Life of a query

• Clients submit rtsql queries as simple strings to server

• Compiler parses query to an AST, generates a logical plan (DAG), and maps that to a DAG of physical operators (“HashJoin”, “Filter”, etc)
Life of a query

• Physical operators form a “flow”, which is injected into the execution thread; continuously reads from input and processes events.

• Many flows (queries) may run in parallel.

• Flows must be explicitly dropped when they’re no longer useful.
Schemas, types and serialization

- Event data can enter FlumeBase in any format
- Each stream has:
  - A *schema*, specifying which fields it has, and their type
  - An *EventParser*, which can extract fields from the input event
- Data is internally represented in Avro generic records
- Output events have Avro binary-encoded records for bodies
Interacting with Flume

• CREATE STREAM defines a schema that could be applied to the output of a Flume node or source
• Submitting a query against that stream requires reading from Flume
  – The Flume controller reconfigures the upstream node to send data to FlumeBase, or hosts a new source locally
• Dropping a query restores the upstream node’s original configuration
FlumeBase components and processes

- FlumeBase abstracts the “server” concept into an ExecEnvironment
- Everything can run in a single process: client shell, ExecEnvironment, even Flume nodes and master
- Better is to leave a long-lived FlumeBase server running and connect clients as needed to examine output, submit or modify queries
Conclusions

• Real-time analytics require a different system than Hadoop MapReduce
• Flume provides a suitable basis for an online analytic system
• SQL-like language allows sophisticated queries with a low learning curve
Check it out!

- Web site: flumebase.org (docs, blog, etc.)
  - Binary release is “batteries included” with a data set + walkthrough
- Get the source: github.com/flumebase/flumebase
- 100% Apache 2.0 licensed – contributors welcome!

Thanks for listening!
aaron@odiago.com