Real Time Data Ingest into Hadoop using Flume

Hari Shreedharan
Committer and PMC Member, Apache Flume
Software Engineer, Cloudera
What is Flume

- Collection, Aggregation of streaming Event Data
  - Typically used for log data
- Significant advantages over ad-hoc solutions
  - Reliable, Scalable, Manageable, Customizable and High Performance
  - Declarative, Dynamic Configuration
  - Contextual Routing
  - Feature rich
  - Fully extensible
Core Concepts: Event

*An Event is the fundamental unit of data transported by Flume from its point of origination to its final destination. Event is a byte array payload accompanied by optional headers.*

- Payload is opaque to Flume
- Headers are specified as an unordered collection of string key-value pairs, with keys being unique across the collection
- Headers can be used for contextual routing
Core Concepts: Client

*An entity that generates events and sends them to one or more Agents.*

- **Example**
  - Flume log4j Appender
  - Custom Client using Client SDK ([org.apache.flume.api](http://org.apache.flume.api))
  - Embedded Agent – An agent embedded within your application
- Decouples Flume from the system where event data is consumed from
- Not needed in all cases
Core Concepts: Agent

A container for hosting Sources, Channels, Sinks and other components that enable the transportation of events from one place to another.

- Fundamental part of a Flume flow
- Provides Configuration, Life-Cycle Management, and Monitoring Support for hosted components
Typical Aggregation Flow

\[ [\text{Client}]^+ \rightarrow \text{Agent} \rightarrow [\text{Agent}]^* \rightarrow \text{Destination} \]
Core Concepts: Source

An active component that receives events from a specialized location or mechanism and places it on one or Channels.

- Different Source types:
  - Specialized sources for integrating with well-known systems. Example: Syslog, Netcat
  - Auto-Generating Sources: Exec, SEQ
  - IPC sources for Agent-to-Agent communication: Avro
- Require at least one channel to function
Sources

- Different Source types:
  - Specialized sources for integrating with well-known systems. Example: Spooling Files, Syslog, Netcat, JMS
  - Auto-Generating Sources: Exec, SEQ
  - IPC sources for Agent-to-Agent communication: Avro, Thrift
- Require at least one channel to function
Core Concepts: Channel

A passive component that buffers the incoming events until they are drained by Sinks.

- Different Channels offer different levels of persistence:
  - Memory Channel: volatile
    - Data lost if JVM or machine restarts
  - File Channel: backed by WAL implementation
    - Data not lost unless the disk dies.
    - Eventually, when the agent comes back data can be accessed.
- Channels are fully transactional
- Provide weak ordering guarantees
- Can work with any number of Sources and Sinks.
Core Concepts: Sink

An active component that removes events from a Channel and transmits them to their next hop destination.

- Different types of Sinks:
  - Terminal sinks that deposit events to their final destination. For example: HDFS, HBase, Morphline-Solr, Elastic Search
  - Sinks support serialization to user’s preferred formats.
  - HDFS sink supports time-based and arbitrary bucketing of data while writing to HDFS.
  - IPC sink for Agent-to-Agent communication: Avro, Thrift
- Require exactly one channel to function
Flow Reliability

Reliability based on:
- Transactional Exchange between Agents
- Persistence Characteristics of Channels in the Flow

Also Available:
- Built-in Load balancing Support
- Built-in Failover Support
Flow Reliability

Normal Flow

Communication Failure between Agents

Communication Restored, Flow back to Normal
Flow Handling

Channels decouple impedance of upstream and downstream

- Upstream burstiness is damped by channels
- Downstream failures are transparently absorbed by channels

→ Sizing of channel capacity is key in realizing these benefits
Configuration

• Java Properties File Format
  
  # Comment line
  key1 = value
  key2 = multi-line \\
  value

• Hierarchical, Name Based Configuration
  
  agent1.channels.myChannel.type = FILE
  agent1.channels.myChannel.capacity = 1000

• Uses soft references for establishing associations
  
  agent1.sources.mySource.type = HTTP
  agent1.sources.mySource.channels = myChannel
Configuration

Typical Deployment
• All agents in a specific tier could be given the same name
• One configuration file with entries for three agents can be used throughout
Contextual Routing

Achieved using Interceptors and Channel Selectors
Interceptors

Interceptor

*An Interceptor is a component applied to a source in pre-specified order to enable decorating and filtering of events where necessary.*

- Built-in Interceptors allow adding headers such as timestamps, hostname, static markers etc.
- Custom interceptors can introspect event payload to create specific headers where necessary
Contextual Routing

Channel Selector

A Channel Selector allows a Source to select one or more Channels from all the Channels that the Source is configured with based on preset criteria.

- Built-in Channel Selectors:
  - Replicating: for duplicating the events
  - Multiplexing: for routing based on headers
Contextual Routing

- Terminal Sinks can directly use Headers to make destination selections
  - HDFS Sink can use headers values to create dynamic path for files that the event will be added to.
  - Some headers such as timestamps can be used in a more sophisticated manner

- Custom Channel Selector can be used for doing specialized routing where necessary
Load Balancing and Failover

Sink Processor

A Sink Processor is responsible for invoking one sink from an assigned group of sinks.

• Built-in Sink Processors:
  • Load Balancing Sink Processor – using RANDOM, ROUND_ROBIN or Custom selection algorithm
  • Failover Sink Processor
  • Default Sink Processor
Sink Processor

- Invoked by Sink Runner
- Acts as a proxy for a Sink
Sink Processor Configuration

• A Sink can exist in at most one group
• A Sink that is not in any group is handled via Default Sink Processor

• Caution:
  Removing a Sink Group does not make the sinks inactive!
Client API

• Simple API that can be used to send data to Flume agents.
• Simplest form – send a batch of events to one agent.
• Can be used to send data to multiple agents in a round-robin, random or failover fashion (send data to one till it fails).
• Java only.
• `flume.thrift` can be used to generate code for other languages.
  • Use with Thrift source.
Embedded Agent

- Client API throws exceptions if data could not be sent.
- Applications may not be able to tolerate this.
- Embedded Agent – A (limited) Flume agent within your application
  - Has a channel – so buffers data in-memory or on-disk till the data is sent or the channel is full.
  - Throws exceptions only if the channel is full (or error writing to channel).
  - Cushion for application if something causes data to be stuck within the application
  - Supports sending data to other Flume agents only, no HDFS, HBase etc.
Summary

• Clients send Events to Agents
• Agents hosts number Flume components – Source, Interceptors, Channel Selectors, Channels, Sink Processors, and Sinks.
• Sources and Sinks are active components, where as Channels are passive
• Source accepts Events, passes them through Interceptor(s), and if not filtered, puts them on channel(s) selected by the configured Channel Selector
• Sink Processor identifies a sink to invoke, that can take Events from a Channel and send it to its next hop destination
• Channel operations are transactional to guarantee one-hop delivery semantics
• Channel persistence allows for ensuring end-to-end reliability
Questions?