ApacheCon North America

Portland, Oregon 26th - 28th February 2013

Planning and Deploying Apache Flume

Arvind Prabhakar

About me

- Committer and PMC Member of
  - Apache Sqoop
  - Apache Flume
- Member of The Apache Software Foundation
- Engineering Manager at Cloudera

Agenda

- Overview of Flume
- Planning and Sizing for Deployment

Agenda

- Overview of Flume
- Planning and Sizing for Deployment

What is Flume

- Collection, Aggregation of streaming Event Data
  - Typically used for log data
- Advantages over ad-hoc solutions
  - Reliable, Scalable, Manageable, Customizable, High Performance
  - Declarative, Dynamic Configuration
  - Contextual Routing
  - Feature Rich and Fully Extensible

Core Concepts

- Event
- Client
- Agent
  - Source, Channel, Sink
    - Interceptor
    - Channel Selector
    - Sink Processor

Core Concept: 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`)
- Decouples Flume from the system where event data is generated
- 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, Monitoring Support for hosted components.
- Embedded Agent support for light-weight configurations

Core Concepts: Source

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

- Different Source Types:
  - Specialized sources for integrating with well known systems. For example – Syslog, Netcat
  - Auto-Generating Sources: Exec, SEQ
  - IPC Sources for Agent-to-Agent communication: Avro, Thrift
- Requires 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 durability (Memory, File, Database)
- Channels are fully transactional
- Provides 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. Example: HDFS, HBase
  - Auto-Consuming sinks. Example: Null Sink
  - IPC sink for Agent-to-Agent communication: Avro, Thrift
- Require exactly one channel to function

Core Concepts: Interceptor

Interceptors are applied to a source in predetermined order to enable decorating and filtering of events where necessary.

- Built-in interceptors allow adding headers such as timestamps, hostname, static markers
- Custom interceptors can inspect event payload to create specific headers where necessary

Core Concept: Channel Selector

Channel Selector facilitates the selection of one or more Channels from all the configured channels, based on preset criteria.

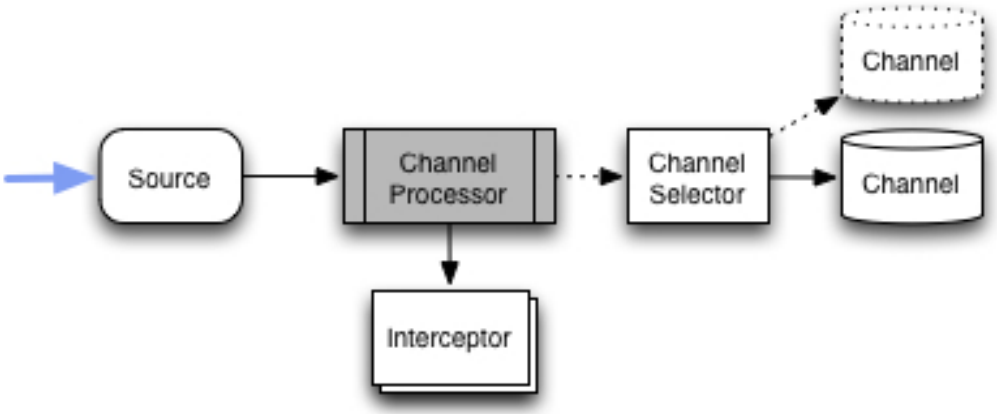
- Built-in Channel Selectors:
  - Replicating: for duplicating the events
  - Multiplexing: for routing based on headers
- Custom Channel Selectors can use dynamic criteria where needed.

Core Concept: Sink Processor

Sink Processor is responsible for invoking one sink from a specified group of Sinks.

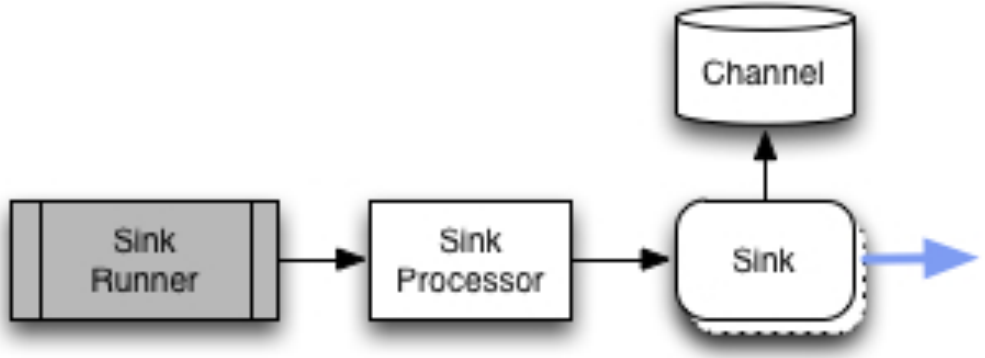
- Built-in Sink Processors:
  - Load Balancing Sink Processor – using RANDOM, ROUND_ROBIN or Custom Selection Algorithm
  - Failover Sink Processor
  - Default Sink Processor

Agent Data Ingest



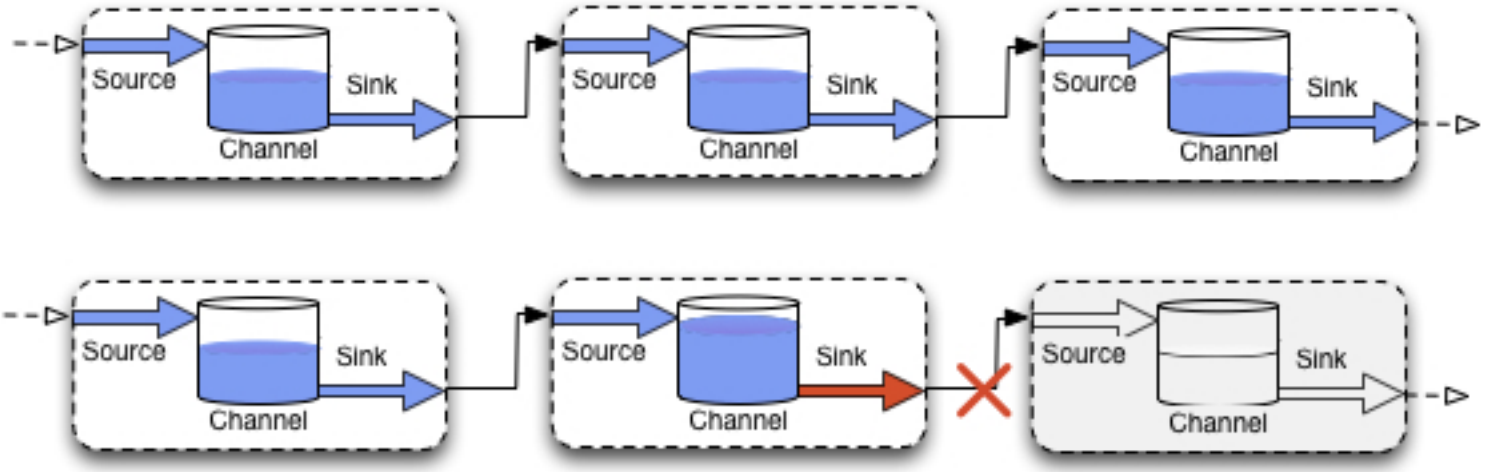
- Clients send Events to Agents
- Source(s) operating with the Agent receive the Events
- Source(s) passes received Events through Interceptor(s) and if not filtered, puts them on Channel(s) identified by the configured Channel Selector.
  - Event population in Channel is transactional.

Agent Data Drain



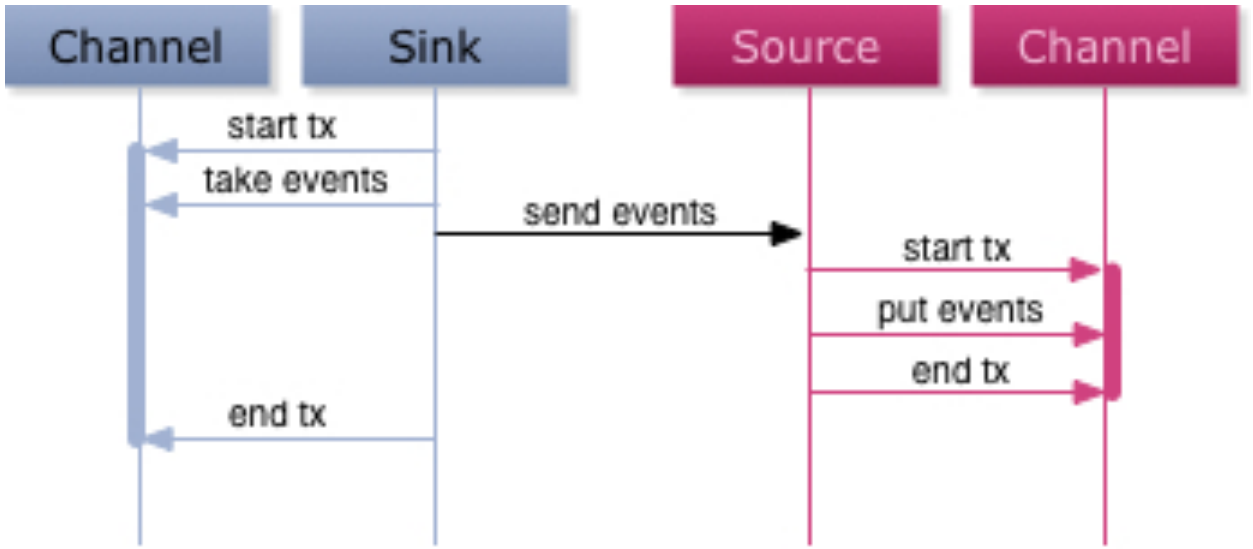
- Sink Processor identifies one of the configured Sinks and invokes it
- Invoked Sink takes Event(s) from its configured Channel and sends it to the next hop destination.
- Event Removal from Channel is transactional.
- If the transmission of Event(s) fail, the Sink Processor can take secondary action.

Agent Pipeline Flow



- Agents transmit events to next hop agents
- In case of transmission failure the second agent retains the data in its channel(s)

Delivery Guarantee



- Agents use transactional exchange to guarantee delivery across hops
- Durable channels ensure that data is never lost in transit

Agenda

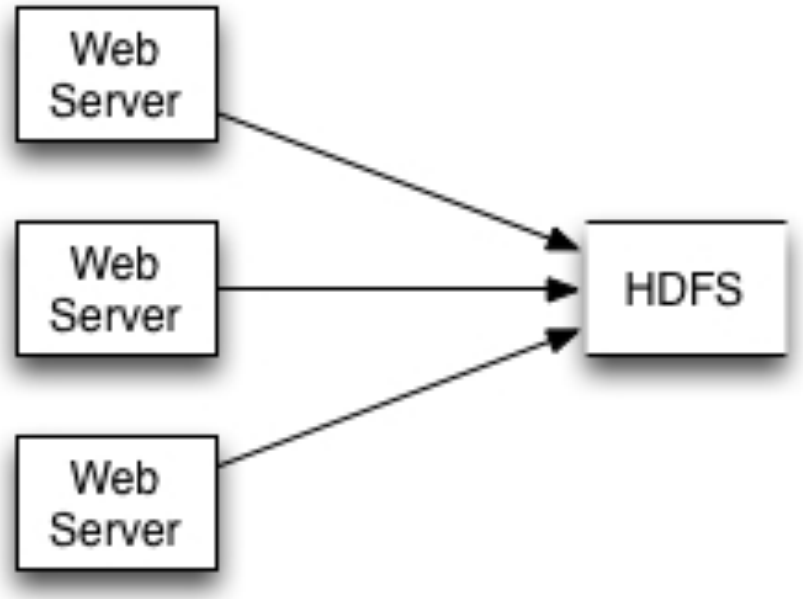
- Overview of Flume
- Planning and Sizing for Deployment

Aggregation Use-Case

Collect Web Server logs

Ad-hoc Solutions

- Configuration Management – Output paths
- Complex Coding and Evolution
- Maintenance and upkeep
- Scalability

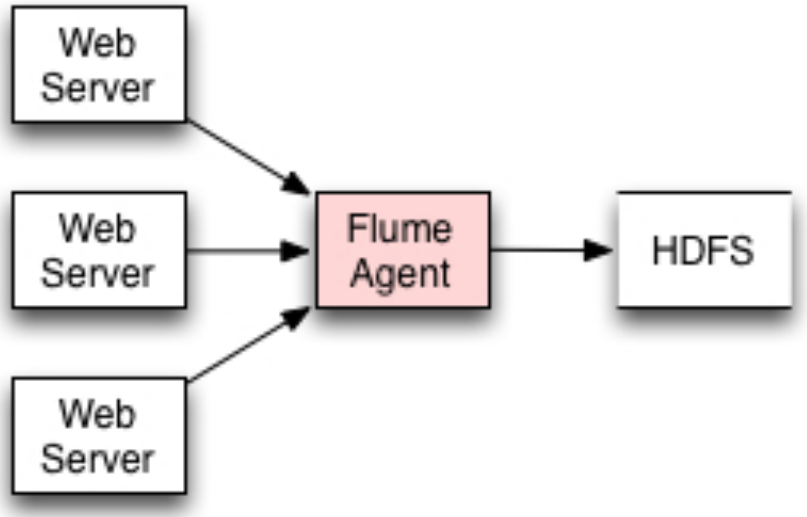


Aggregation Use-Case

Collect Web Server logs

Using Single Flume Agent

- Insulation from destination downtime
  - Managed impact on service
- Quick offload of events from Web Server machines
- Better network utilization

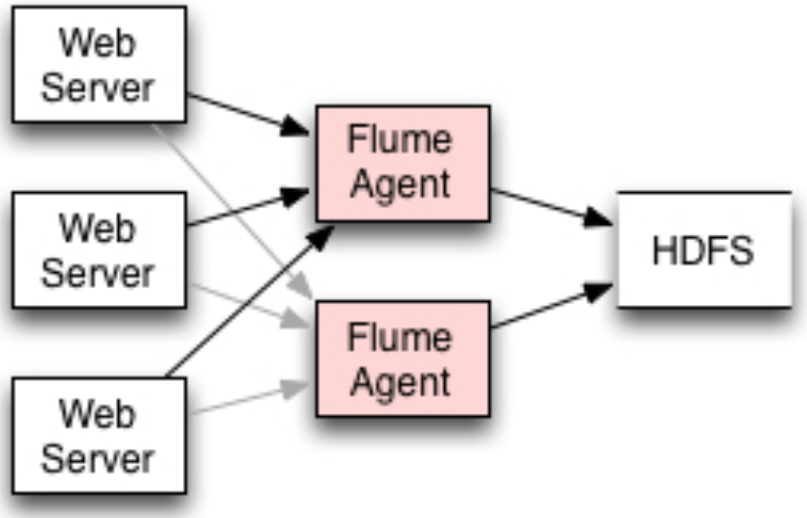


Aggregation Use-Case

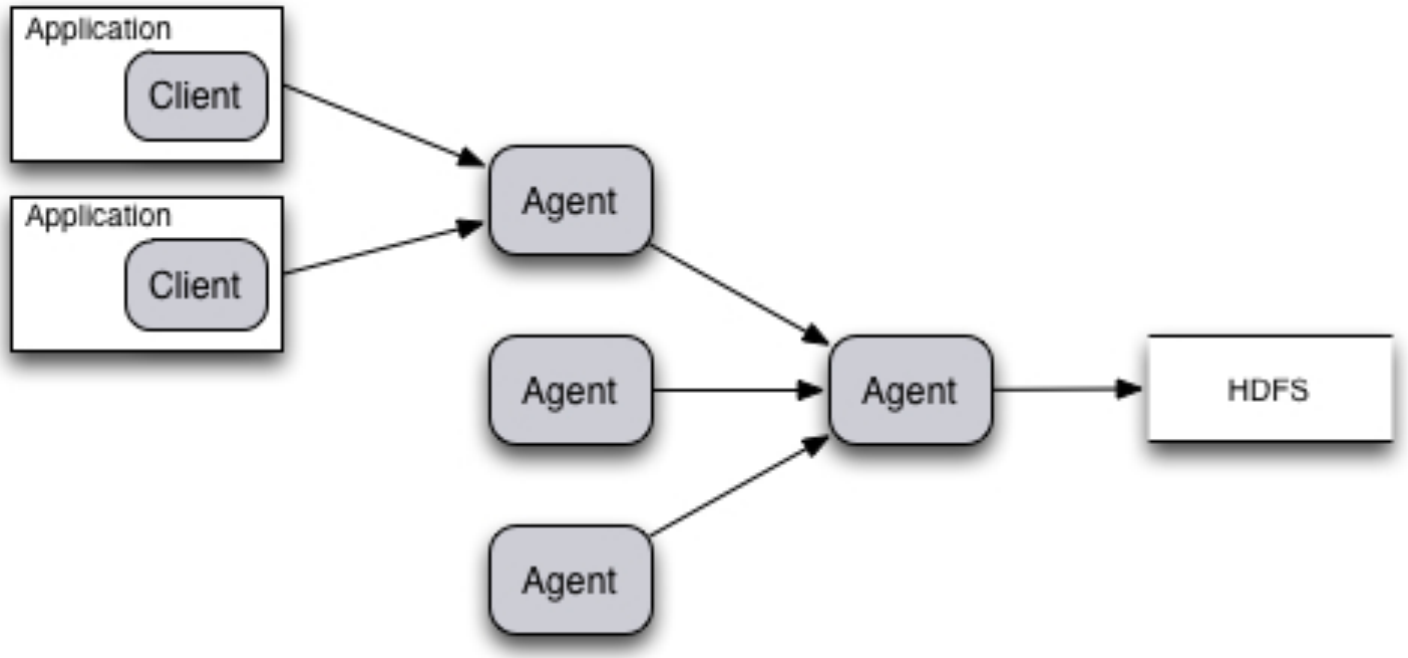
Collect Web Server logs

Using Multiple Flume Agents

- Redundancy is Availability
- Better handling of transient failures/down times with minimal hardware investment
- Automatic contextual routing for planned routing of event traffic



Typical Converging Flow

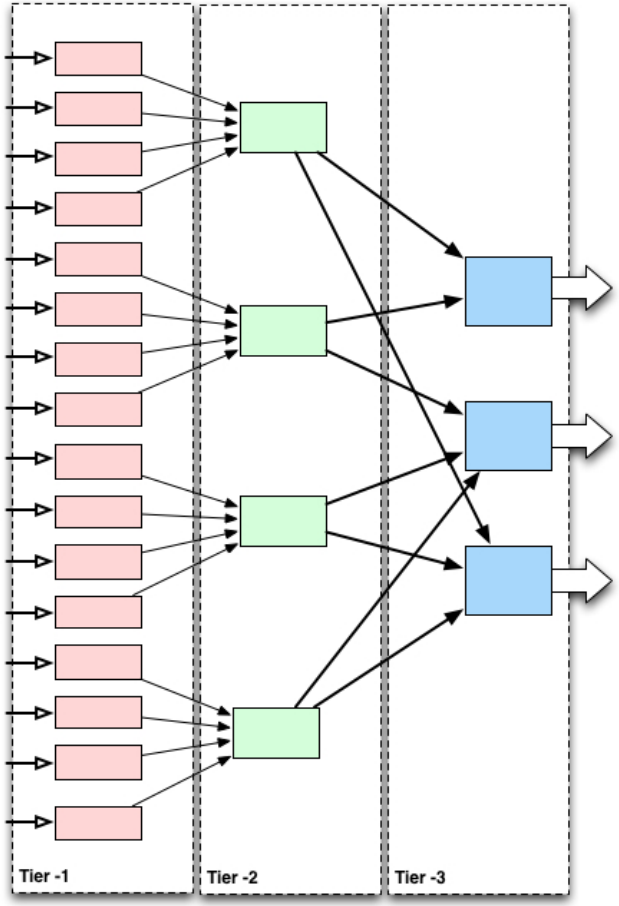


[Client]+ → Agent [→ Agent]* → Destination

Converging Flow Characteristics

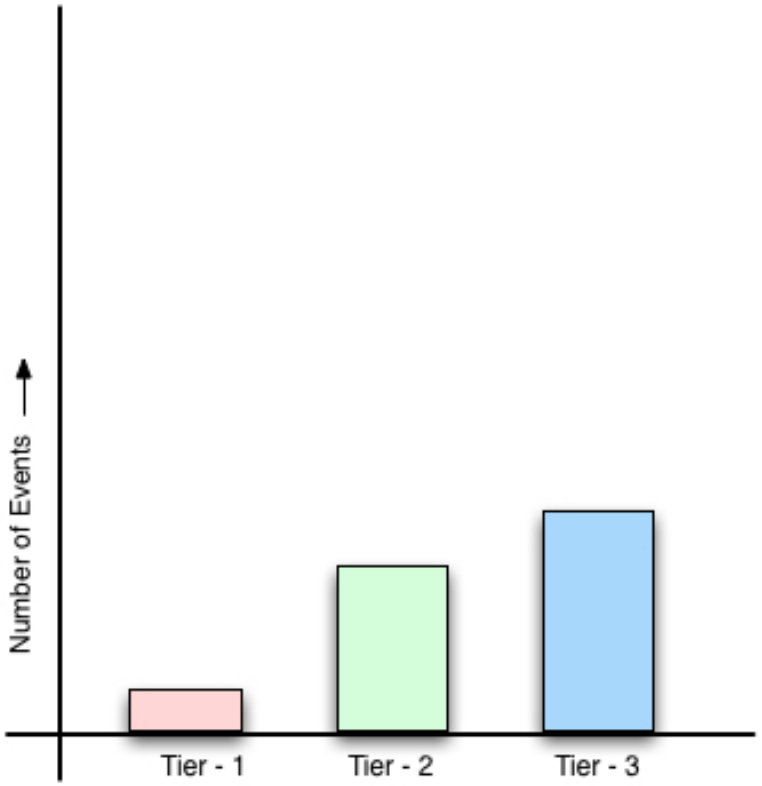
- Events are generated from multiple locations
  - Example: Server Farm producing logs
- Routed through increasingly large pipelines
  - Example: All data center traffic routed through same gateway nodes
- Deposited in common location
  - Example: All data populated in HDFS for further processing

Large Converging Flows



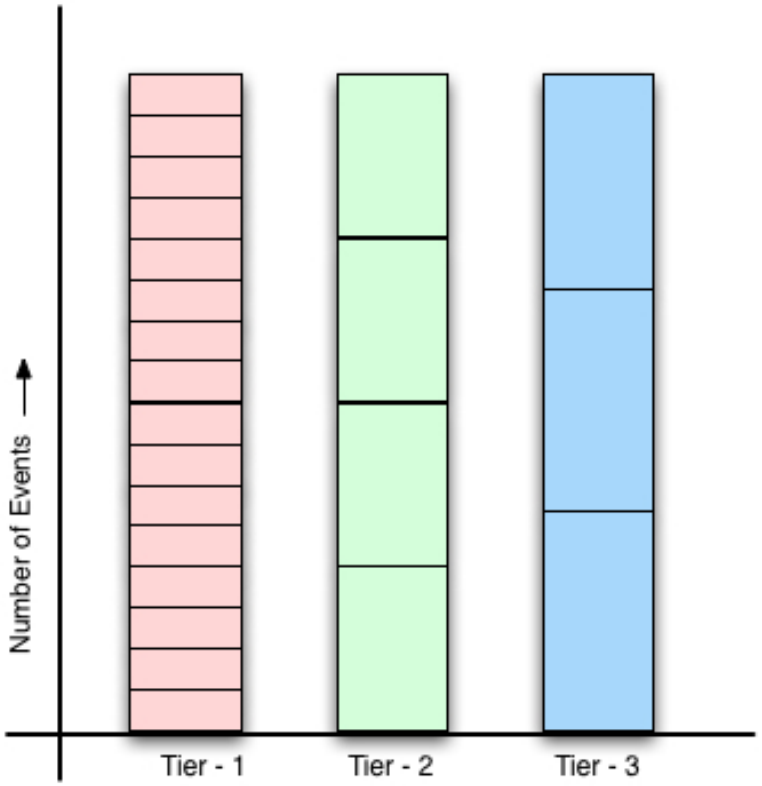
- Outermost tier has maximum number of Agents
- Intermediate tier Agents provide buffering capacity
- Terminal tier Agents deposit aggregated data to end points

Event Volume



- Events volume is the least in outermost tier
- As flow converges, Event volume increases
- Event volume is most in the innermost tier.

Combined Event Volume



- Aggregate event volume remains constant across all tiers
- Any transient imbalance in tier volume is absorbed by the Channel(s)
- When the traffic ebbs the Channel(s) drains out extra events to reach steady state

Topology Planning

Given:

- Number of Points of Origination of Events
- Final Destination of Events

Work Through:

- Ability to cushion load spikes (Impedance mismatch)
- Capacity necessary to handle transient failures

Identify Number of Tiers

- Each tier is an intermediate aggregator
- Aggregators used for:
  - Offloading data quickly from upstream
  - Providing Load Balancing and Failover Support
  - Better network utilization
  - Absorbing Load Spikes
  - Better File Channel utilization (large batch sizes) and thus increased throughput
- Intermediate tiers ideal for contextual routing implementation

Number of Tiers

Rule of Thumb:

One aggregator for every 4 – 16 Agents

- Primarily decided by ingest volume requirement
- Theoretical maximum: number of Agents that saturate the network at projected event rates.
- Can be as high as 100 or more to one on outer tiers
- Should reduce significantly towards inner tiers

Number of Tiers

Other factors and requirements to consider:

- Load Balancing
- Failover
- Contextual routing

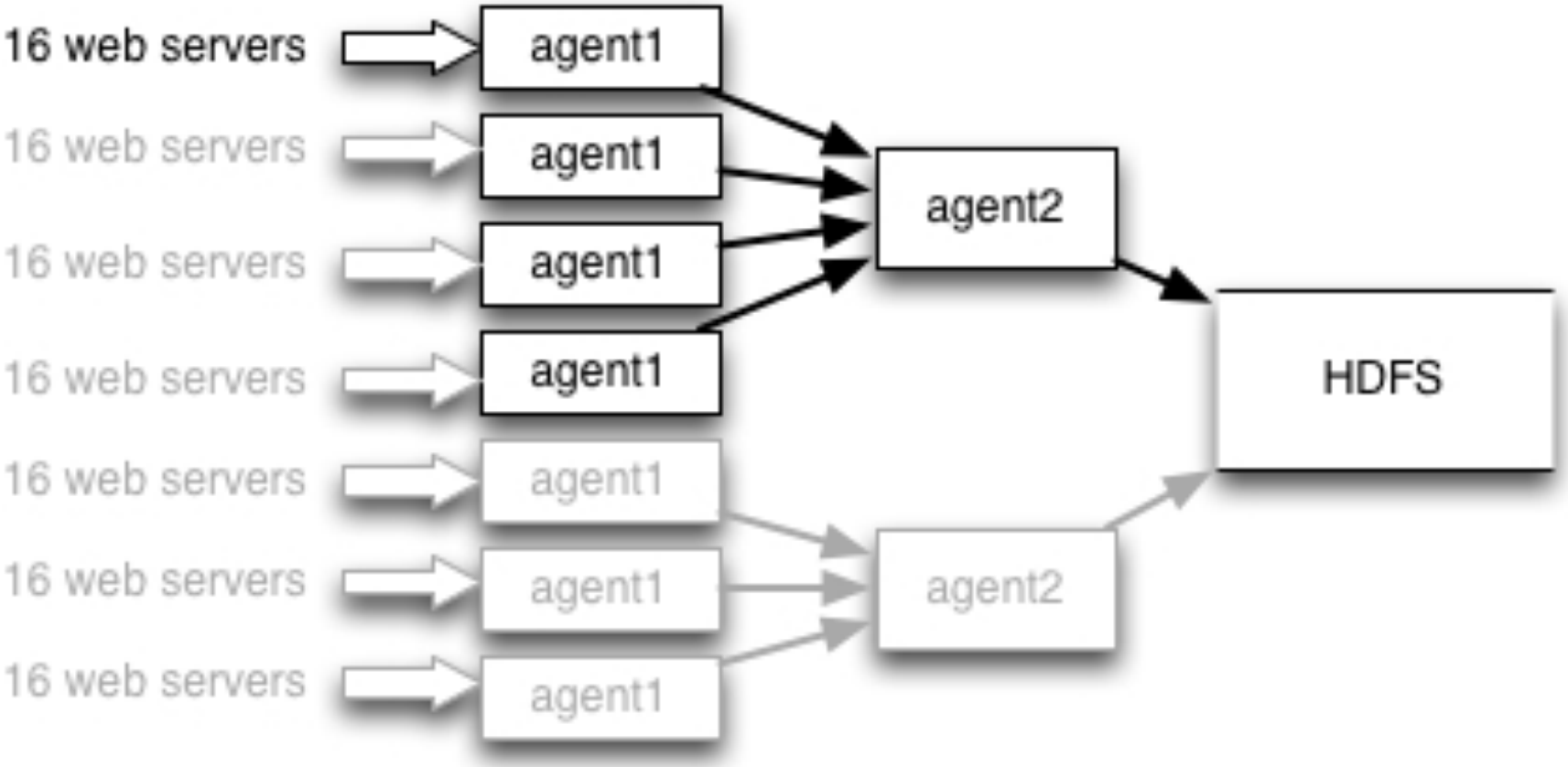
Number of Tiers

For aggregation flow from 100 web servers

- Using ratio of 1:16 for outer tier, no load balancing or failover requirements
  - Number of tier 1 agents = 100/16 ~ 7
- Using ratio of 1:4 for inner tier, no load balancing or failover
  - Number of tier 2 agents = 7/4 ~ 2

Total of 2 tiers, 9 agents

Number of Tiers



Sink Batch Size

Dictated by steady state requirements

For example:

- For agent1 receiving data from 16 web server clients
- Assuming each server sends a batch of 100 events per cycle

  ➔ Exit Batch Size = 16 x 100 = 1600
- If batch size is too large (> 2500), Use multiple sinks

Sink Batch Size

Similarly, for agent2:

- Receiving a batch of 1600 events from each of four upstream agents

  ➔ Batch size = 1600 x 4 = 6400

- Since this is very large, we split it into 3 sinks with batch size 2150 each
- Larger the batch size, greater the risk of duplication

File Channel Capacity

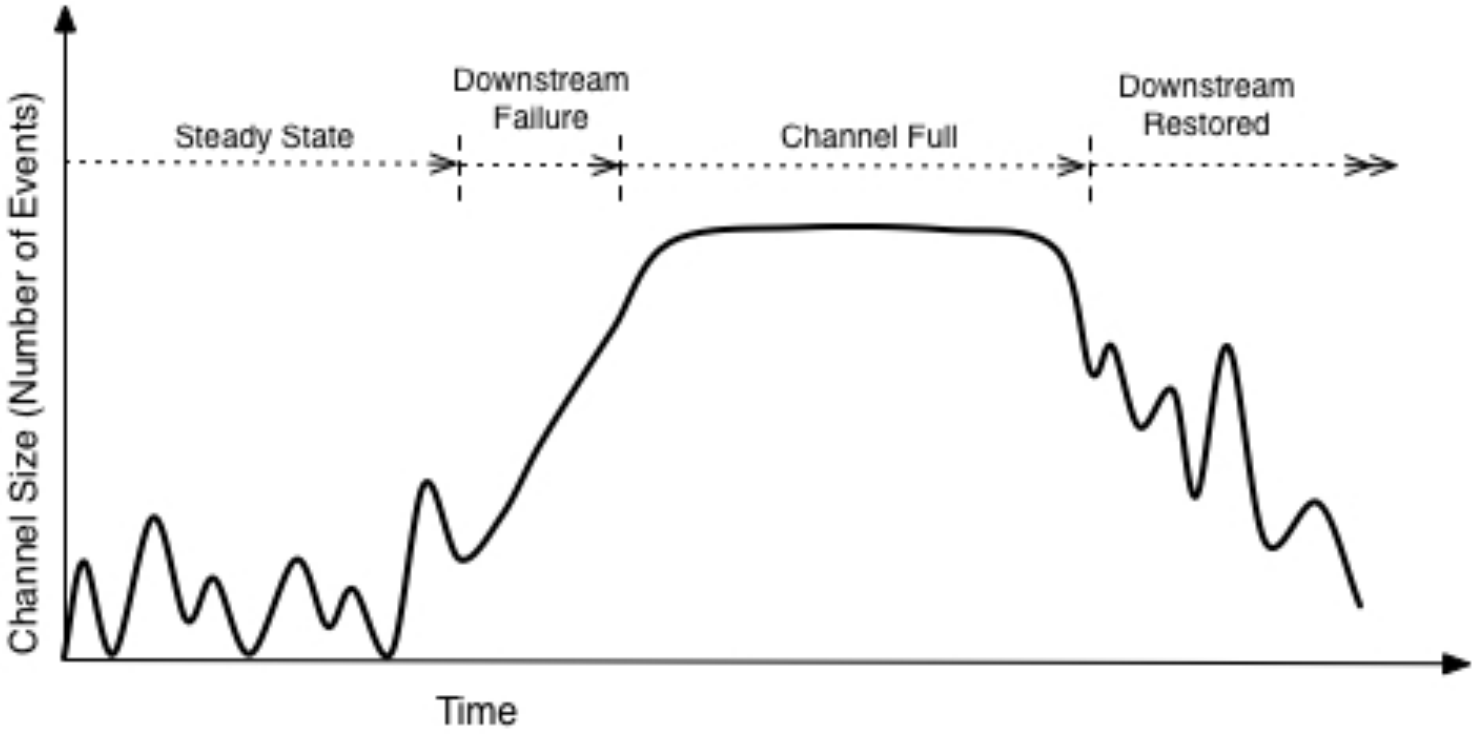
- Dictated by Failure handling requirements
- For example:
  - Assume expected resolution for downstream failure to be 2 hours
  - Expected worst case event rate arriving at a single Flume Agent is 100 per sec
  - Number of events in two hours at this rate

    = 100 x 60 x 60 x 2 = 720,000.
  - Increase by safety padding factor say 1.5

    = 720,000 x 1.5 = 1,080,000

Required File Channel Capacity = 1,080,000

Channel Capacity Utilization



Hardware Sizing

- Start with this rule of thumb:

  # Cores = (# Sources + # Sinks) / 2

- If using Memory Channel, RAM should be sufficient to hold maximum channel capacity
- If using File Channel, more disks are better for throughput

Getting Ready for Production

- Before going to production:
  - Stage/test worst case scenarios
  - If needed change configuration parameters, topology, or hardware
  - Repeat Testing until system is verifiably meeting expected load requirements

- Monitor system closely in production
  - Pay attention to Channel Sizes
  - Adjust configuration if needed

For More Information

http://flume.apache.org/

https://cwiki.apache.org/confluence/display/FLUME/Home

https://issues.apache.org/jira/browse/FLUME

User Mailing List:

- user@flume.apache.org
- user-subscribe@flume.apache.org

Developer Mailing List:

- dev@flume.apache.org
- dev-subscribe@flume.apache.org