**What is meant by FlumeNG ?**

At a high-level, Flume NG uses a single-hop message delivery guarantee semantics to provide end-to-end reliability for the system.

The purpose of Flume is to provide a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store. The architecture of Flume NG is based on a few concepts that together help achieve this objective.

It achieves reliability by the following mechanism

This mechanism also forms the basis for failure handling in Flume NG. When a flow that passes through many different agents encounters a communication failure on any leg of the flow, the affected events start getting buffered at the last unaffected agent in the flow. If the failure is not resolved on time, this may lead to the failure of the last unaffected agent, which then would force the agent before it to start buffering the events. Eventually if the failure occurs when the client transmits the event to its first-hop destination, the failure will be reported back to the client which can then allow the application generating the events to take appropriate action.

On the other hand, if the failure is resolved before the first-hop agent fails, the buffered events in various agents downstream will start draining towards their destination. Eventually the flow will be restored to its original characteristic throughput levels. Figure 4 below illustrates a scenario where a flow comprising of two intermediary agents between the client and the central store go through a transient failure. The failure occurs between agent 2 and the central store, resulting in the events getting buffered at the agent 2 itself. Once the failing link has been restored to normal, the buffered events drain out to the central store and the flow is restored to its original throughput characteristics.

Can Flume provides 100 % reliability to the data flow?

Yes it provides by transaction mechanism

(ie) by the concept like either full or nothing if by chance a failure occurs in the data flow

It is like money transaction mechanism

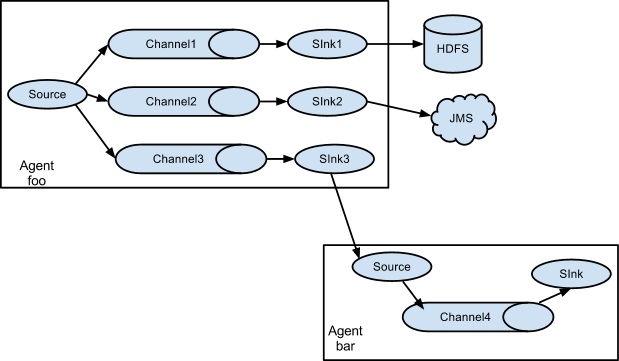
In Flume, for each event, two transactions take place: one at the sender and one at the receiver. The sender sends events to the receiver. Soon after receiving the data, the receiver commits its own transaction and sends a “received” signal to the sender. After receiving the signal, the sender commits its transaction. (Sender will not commit its transaction till it receives a signal from the receiver.)

• If for some unlikely reason the events could not be logged, the transaction would be rolled back and the events would remain in the channel for later redelivery.

• The channel we are using is a file channel, which has the property of being durable: once an event has been written to the channel, it will not be lost, even if the agent restarts

Can Flume can distributes data to multiple destinations?

Yes it can done by fan-out as shown



The source sends the event to 3 channels and is written to HDFS,JMS AND IN THIRD IT IS TASKEN AS A SOURCE.

There are 2 types of FAN-OUT

1.REPLICATING

2.MULTIPLEXING

Replicating − The data flow where the data will be replicated in all the configured channels.

Multiplexing − The data flow where the data will be sent to a selected channel which is mentioned in the header of the event

**Explain about the different channel types in Flume. And which channel type is faster?**

There are 3 types

Memory Channel

JDBC Channel

File Channel

Memory Channel

Memory Channel is an in-memory channel that stores events written to it on the heap. For all practical purposes, the Memory Channel is an in-memory queue—the sources write to its tail and sinks read off its head. The Memory Channel supports very high throughput, as it holds all data in memory. As mentioned earlier, the channel is thread-safe and can handle writes from several sources and reads from several sinks at the same time. .

**The Memory Channel should be used when data loss is not a concern, since the channel does not persist the data to disk. If data loss is a concern, then the Memory Channel should not be used, since process death or machine crashes or restarts can cause data to be lost.**

**JDBC Channel**

JDBC Channel stores the events in an embedded Derby database.

File Channel

The File Channel is Flume’s persistent channel. It writes out all events to disk and thus does not lose data on process or machine shutdown or crash. The File Channel ensures that any events committed into the channel are removed from the channel only when a sink takes the events and commits the transaction, even if the machine or agent crashed and was restarted. It is designed to be highly concurrent and to handle several sources and sinks at the same time.

**MEMORY Channel is the fastest channel among the three however has the risk of data loss. The channel that you choose completely depends on the nature of the big data application and the value of each event**