Flume应用场景、原理、基本架构与案例分析

讲师：董西成
博客：dongxicheng.org
微信号：hadoop-123
（二维码见右）

Open Passion Value

目录

Flume是什么？

- 由Cloudera公司开源；
- 分布式、可靠、高可用的海量日志采集系统；
- 数据源可定制，可扩展；
- 数据存储系统可定制，可扩展。
- 中间件：屏蔽了数据源和数据存储系统的异构性

Flume特点

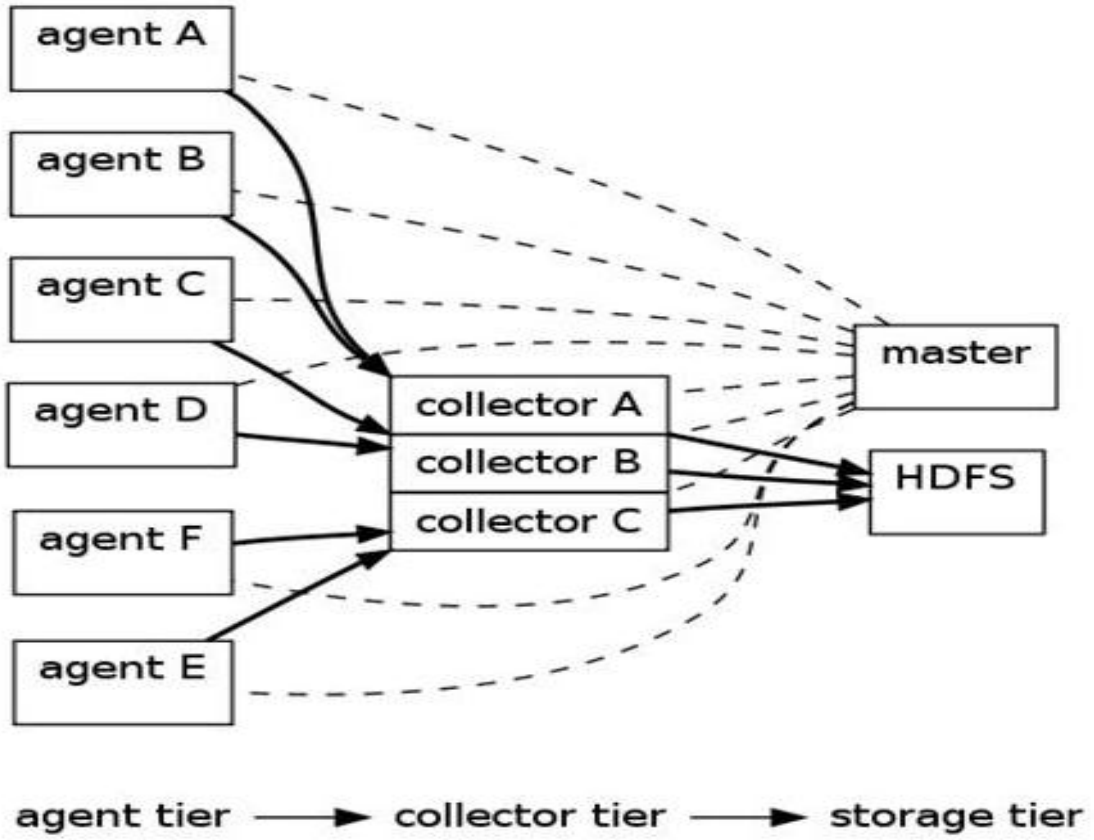
- 可靠性
  - ✓ 保证数据不丢失
- 可扩展性
  - ✓ 各组件数目可扩展
- 高性能
  - ✓ 吞吐率很高，能满足海量数据收集需求
- 可管理性
  - ✓ 可动态增加和删除组件
- 文档丰富，社区活跃
  - ✓ 已成为Hadoop生态系统标配

Flume OG和NG两个版本

- **Flume OG**
  - ✓ OG：“Original Generation”
  - ✓ 0.9.x或cdh3以及更早版本
  - ✓ 由agent、collector、master等组件构成
- **Flume NG**
  - ✓ NG：“Next/New Generation”
  - ✓ 1.x或cdh4以及之后的版本
  - ✓ 由Agent、Client等组件构成
- **为什么要推出NG版本**
  - ✓ 精简代码
  - ✓ 架构简化

目录

Flume OG基本架构



Flume OG基本架构

角色	简介
Master	Master 负责配置及通信管理，是集群的控制器
Collector	Collector 用于对数据进行聚合（数据收集器），往往会产生一个更大的数据流，然后加载到 storage（存储）中
Agent	Agent 用于采集数据，Agent 是 flume 中产生数据流的地方，同时 Agent 会将产生的数据流传输到 Collector

agent

- 用于采集数据
- 数据流产生的地方
- 通常由source和sink两部分组成
  - ✓ Source用于获取数据，可从文本文件，syslog，HTTP等获取数据；
  - ✓ Sink将Source获得的数据进一步传输给后面的Collector。
  - Flume自带了很多source和sink实现
  - ✓ syslogTcp(5140) | agentSink("localhost",35853)
  - ✓ tail("/etc/services") | agentSink("localhost",35853)

Collector

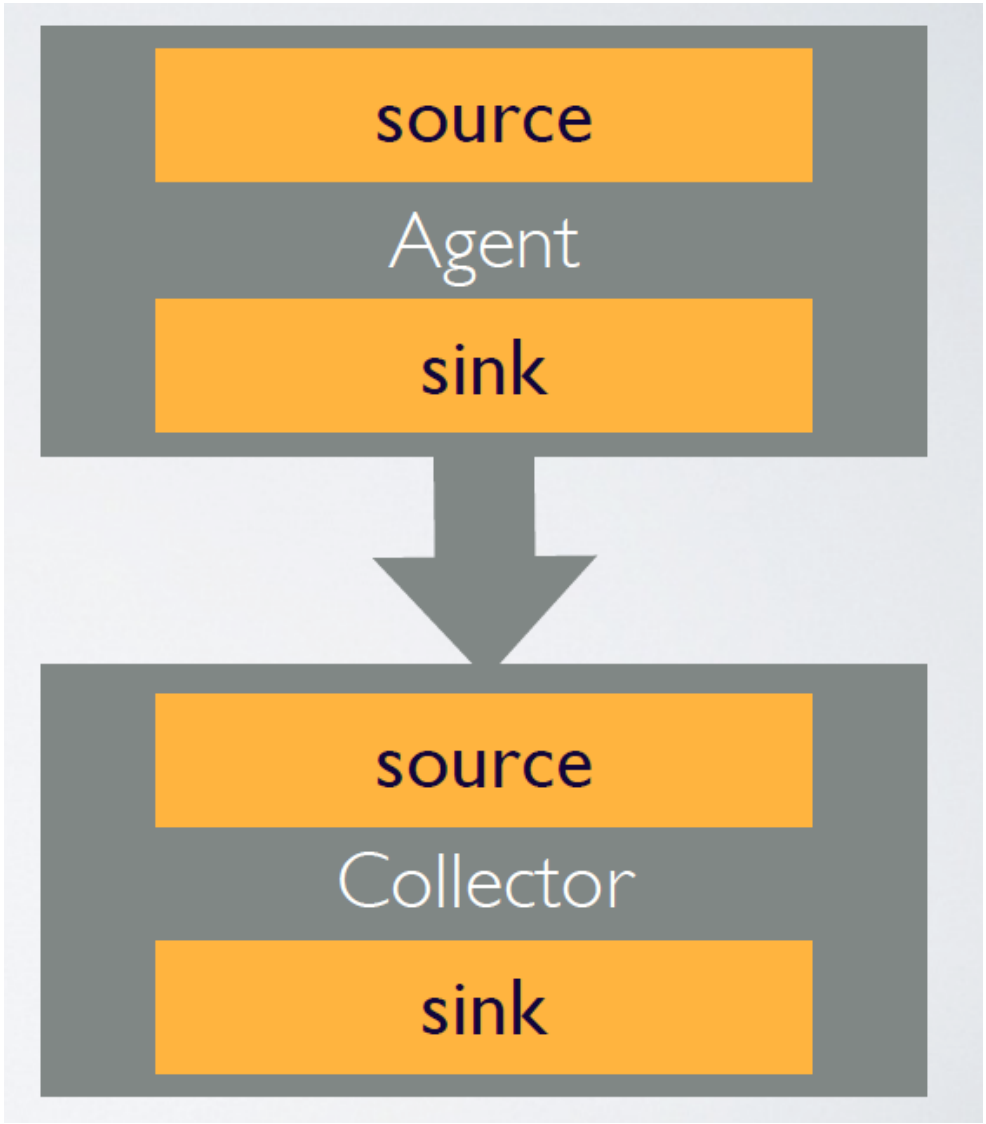
- 汇总多个Agent结果
- 将汇总结果导入后端存储系统，比如HDFS，HBase
- Flume自带了很多collector实现

✓ collectorSource(35853) | console

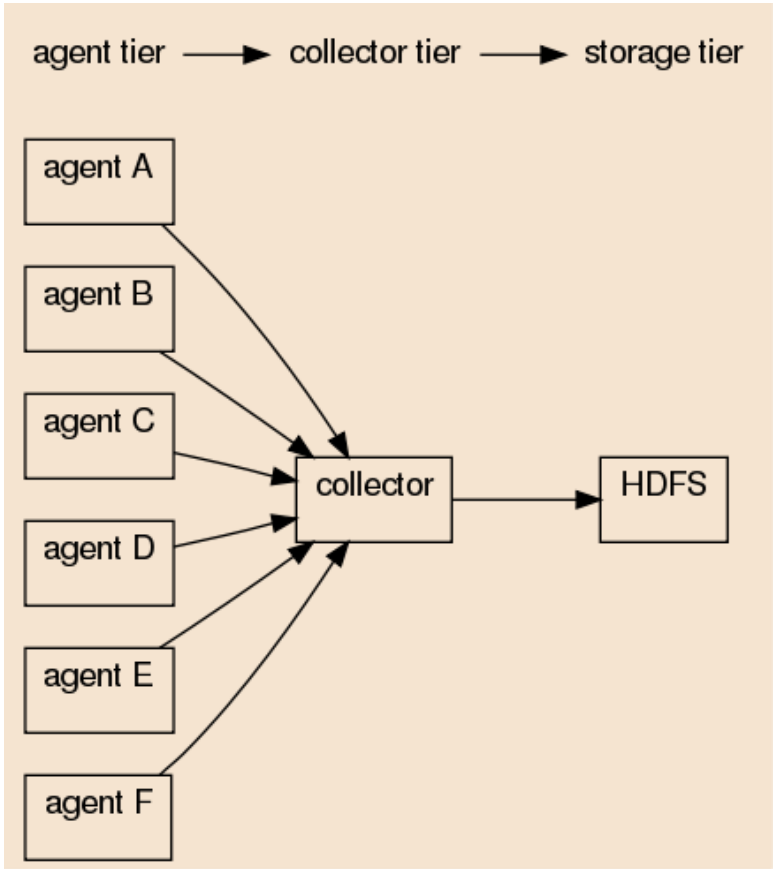
✓ collectorSource(35853) |
collectorSink("file:///tmp/flume/collected", "syslog");

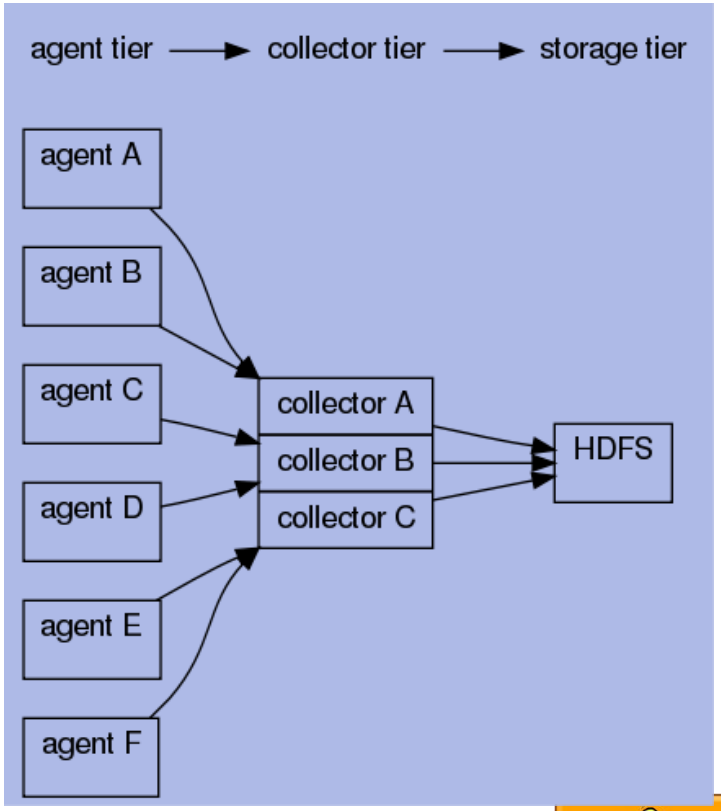
✓ collectorSource(35853) |
collectorSink("hdfs://namenode/user/flume/ ","syslog");

Agent与Collector对应关系



Agent与Collector对应关系

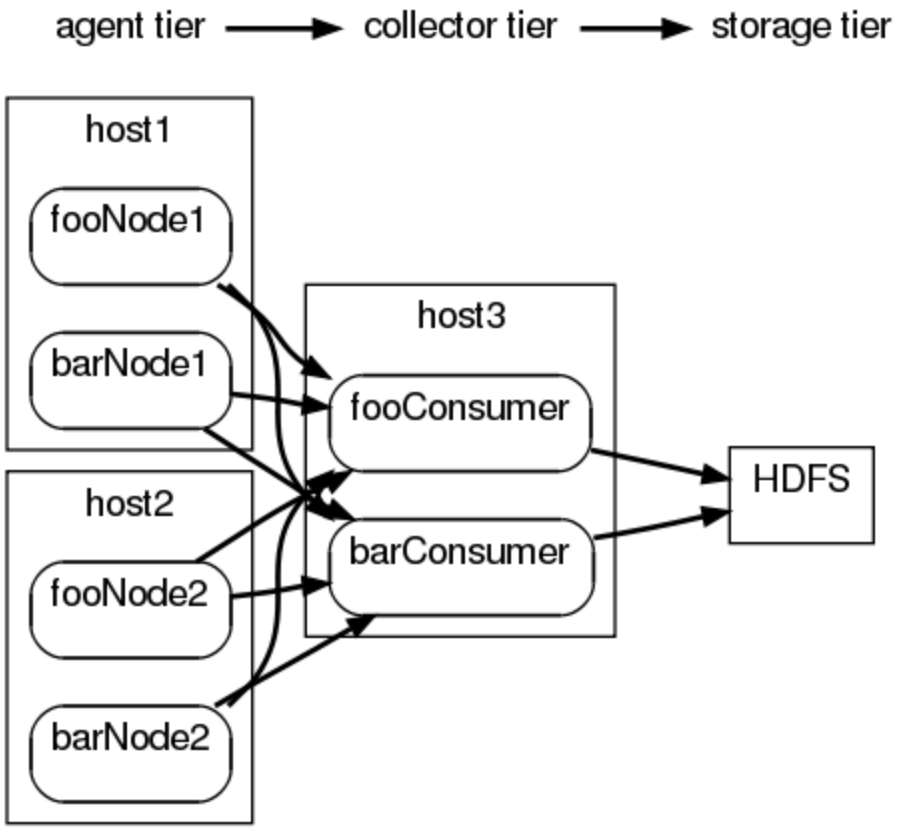
- 可手动指定，也可自动匹配
- 自动匹配的情况下，master会平衡collector之间的负载
  - 



Master

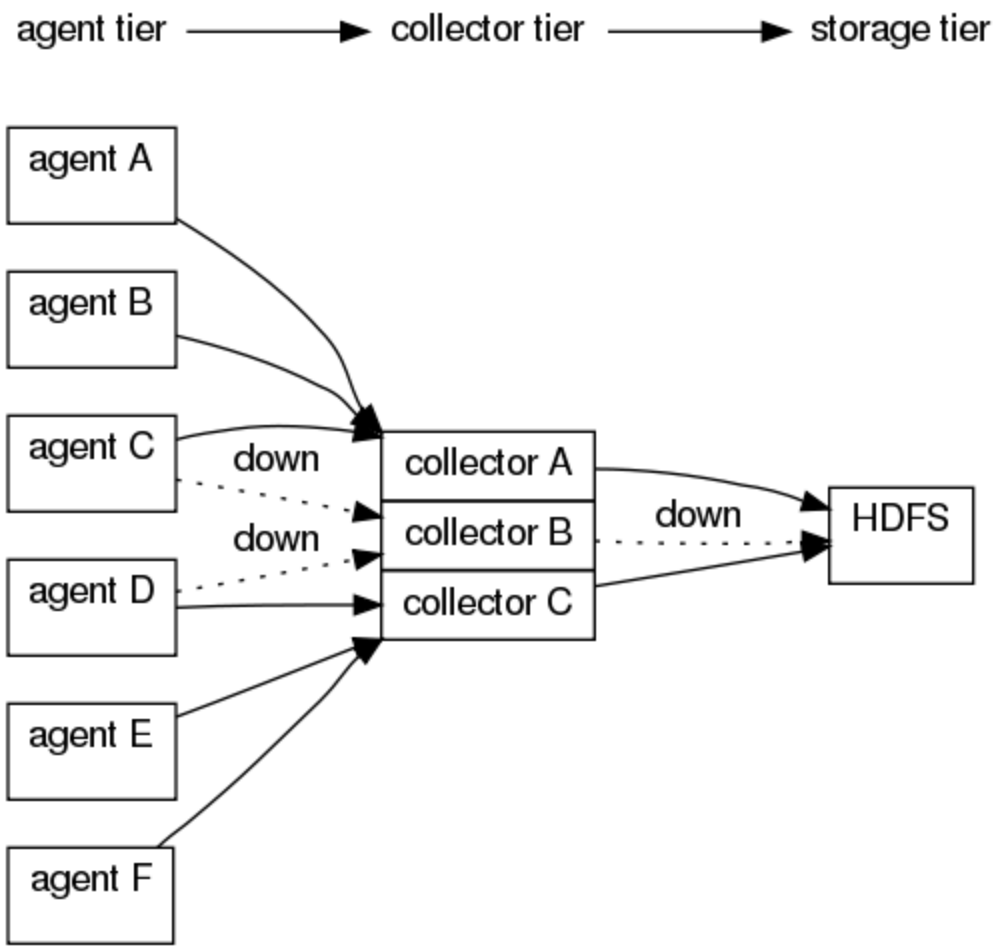
- 管理协调 agent 和collector的配置信息；
- Flume集群的控制器；
- 跟踪数据流的最后确认信息，并通知agent；
- 通常需配置多个master以防止单点故障；
- 借助zookeeper管理管理多Master。

容错机制设计



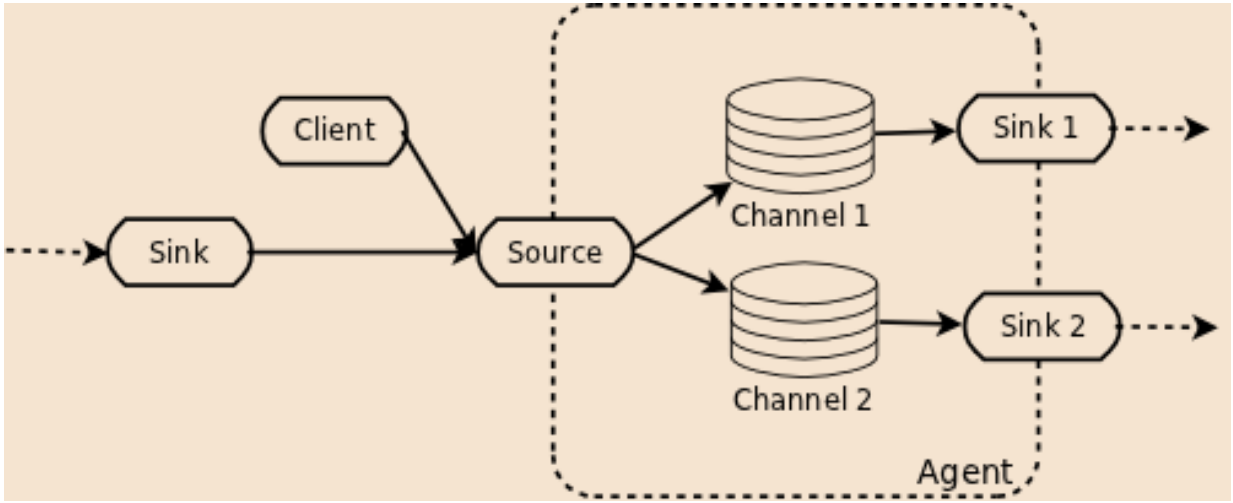
A single flume flow

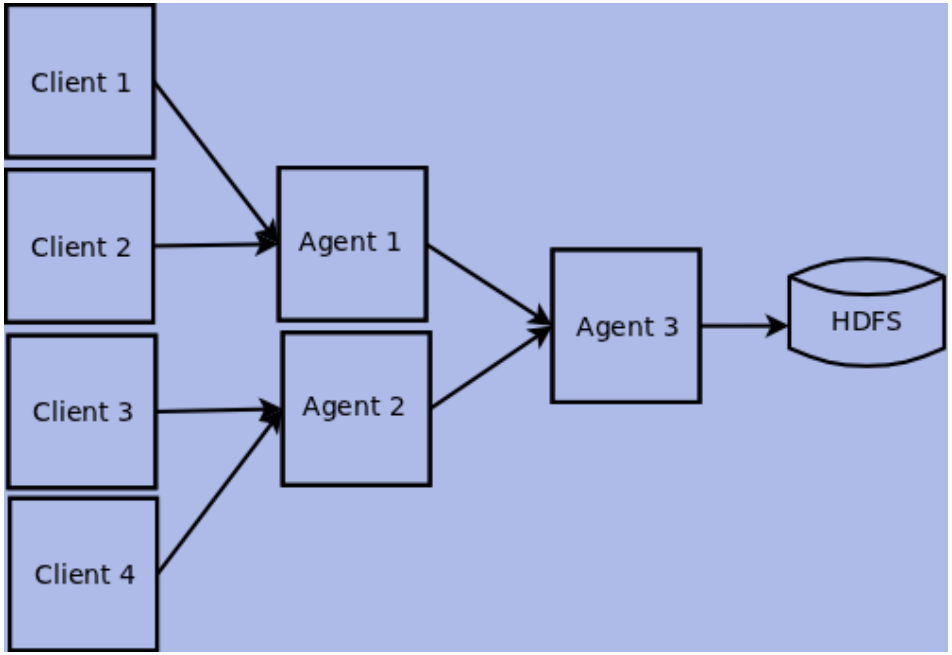
容错机制设计


agent tier
collector tier
storage tier
agent A
agent B
agent C
agent D
agent E
agent F
down
down
collector A
collector B
collector C
down
HDFS

目录

Flume NG基本架构





Flume NG核心概念

- Event
- Client
- Agent
  - ✓ Source
  - ✓ Channel
  - ✓ Sink
  - ✓ 其他组件：Interceptor、Channel Selector、Sink Processor

Event

- Event是Flume数据传输的基本单元
- Flume以事件的形式将数据从源头传送到最终的目的
- Event由可选的header和载有数据的一个byte array构成。
  - ✓ 载有的数据对flume是不透明的
  - ✓ Header是容纳了key-value字符串对的无序集合，key在集合内是唯一的。
  - ✓ Header可以在上下文路由中使用扩展

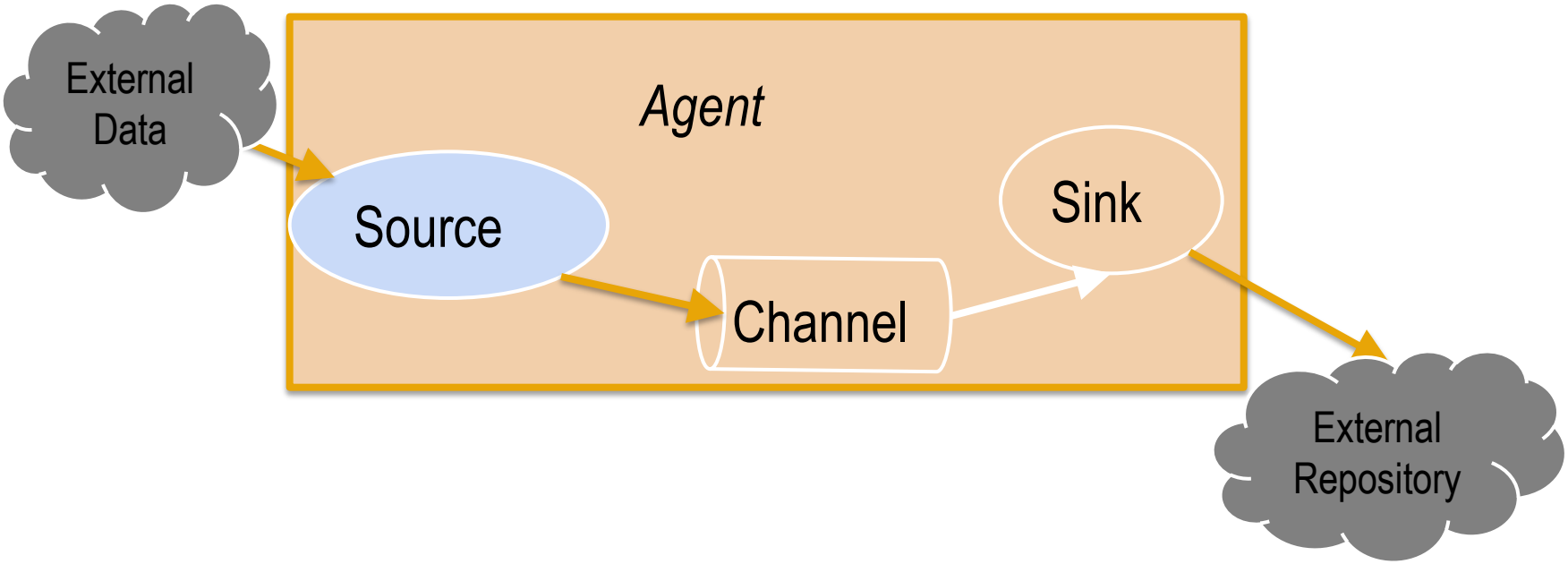
Client

- Client是一个将原始log包装成events并且发送它们到一个或多个agent的实体。
- 目的是从数据源系统中解耦**Flume**
- 在**flume**的拓扑结构中不是必须的
- Client实例
  - ✓ **Flume** log4j Appender
  - ✓ 可以使用Client SDK (org.apache.**flume**.api)定制特定的Client

Agent

- 一个Agent包含Source, Channel, Sink和其他组件；
- 它利用这些组件将events从一个节点传输到另一个节点或最终目的；
- agent是flume流的基础部分；
- flume为这些组件提供了配置、生命周期管理、监控支持。

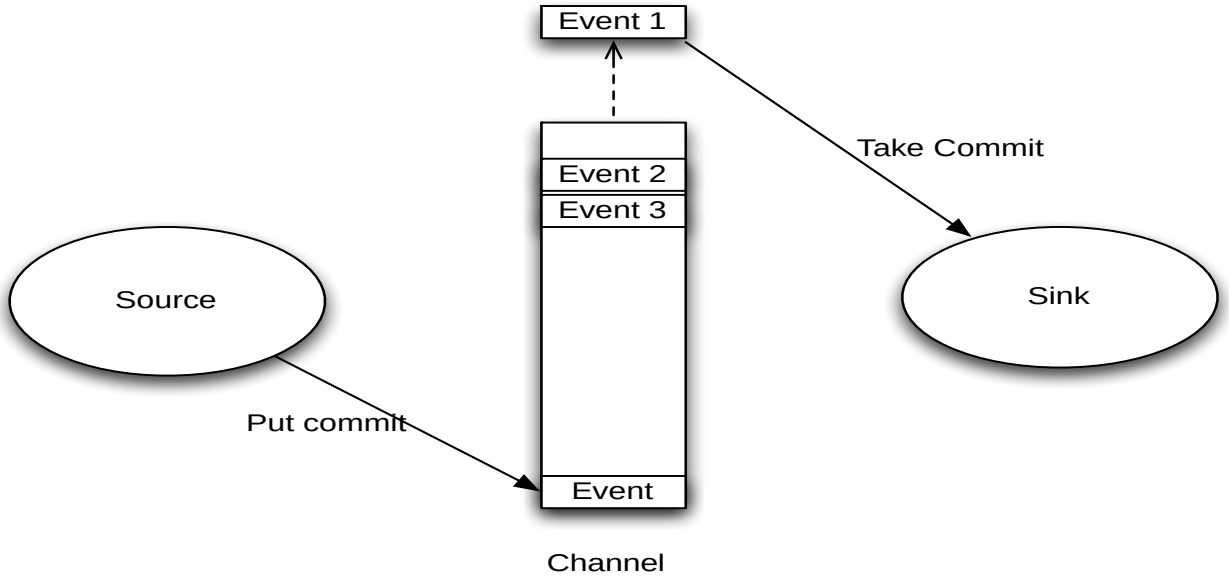
Agent之Source



Agent之Source

- Source负责接收event或通过特殊机制产生event，并将events批量的放到一个或多个Channel。
- 包含event驱动和轮询2种类型
- 不同类型的Source:
  - ✓ 与系统集成的Source: Syslog, Netcat
  - ✓ 自动生成事件的Source: Exec
  - ✓ 用于Agent和Agent之间通信的IPC Source: Avro、Thrift
- Source必须至少和一个channel关联

Agent之Channel与Sink



Agent之Channel

- Channel位于Source和Sink之间，用于缓存进来的event；
- 当Sink成功的将event发送到下一跳的channel或最终目的，event从Channel移除。
- 不同的Channel提供的持久化水平也是不一样的:
  - ✓ Memory Channel: volatile
  - ✓ File Channel: 基于WAL（预写式日志Write-Ahead Logging）实现
  - ✓ JDBC Channel: 基于嵌入Database实现
- Channel支持事务，提供较弱的顺序保证
- 可以和任何数量的Source和Sink工作

Agent之sink

- Sink负责将event传输到下一跳或最终目的，成功完成后将event从channel移除。
- 不同类型的Sink:
  - ✓ 存储event到最终目的的终端Sink. 比如: HDFS, **HBase**
  - ✓ 自动消耗的Sink. 比如: Null Sink
  - ✓ 用于Agent间通信的IPC sink: Avro
- 必须作用于一个确切的channel

其他几个组件

- Interceptor

  作用于Source，按照预设的顺序在必要地方装饰和过滤events。

- Channel Selector

  允许Source基于预设的标准，从所有Channel中，选择一个或多个Channel。

- Sink Processor:

  多个Sink可以构成一个Sink Group。Sink Processor可以通过组中所有Sink实现负载均衡；也可以在一个Sink失败时转移到另一个。

目录

Flume OG部署

步骤1： 下载flume安装包，并解压到各台机器上；

步骤2： 修改etc/profile文件

export FLUME_HOME=/opt/software/flume-0.9.4-cdh3u6.tar.gz

export PATH=.:$PATH::$FLUME_HOME/bin

步骤3： 验证安装

安装完毕后，运行flume命令，会打印flume的用法

步骤4： 修改配置文件$FLUME_HOME/conf

步骤5： 运行命令启动master，agent，collector

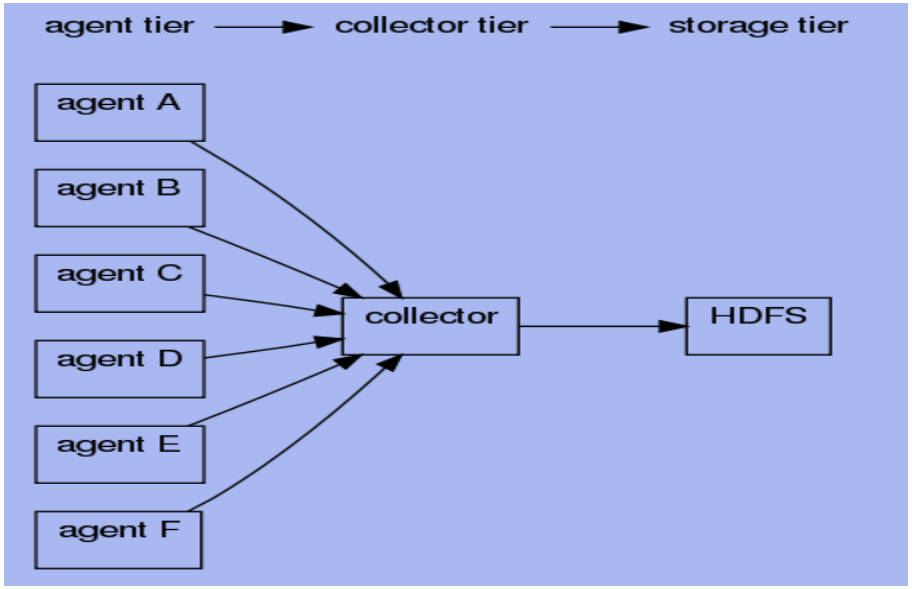
Flume OG集群动态配置

- Agent和Collector均可以动态配置
- 可通过命令行或Web界面配置
- 命令行配置
  - ✓ 在已经启动的master节点上，依次输入”flume shell”➔”connect localhost ”
    如执行 exec config a1 ‘tailDir(“/data/logfile”)’ ‘agentSink’
  - ✓ 默认collector是由flume-conf.xml配置的flume.collector.event.host 和 lume.collector.port两个参数决定
- Web界面

✓ 选中节点，填写source、sink等信息

Flume OG集群拓扑配置(1)

```
agentA : src | agentSink("collector",35853);
agentB : src | agentSink("collector",35853);
agentC : src | agentSink("collector",35853);
agentD : src | agentSink("collector",35853);
agentE : src | agentSink("collector",35853);
agentF : src | agentSink("collector",35853);
collector : collectorSource(35853) |
        collectorSink("hdfs://namenode/flume/","srcdata");
```



Flume OG三种可靠性级别

- agentE2ESink[("machine"[,port])]

  end to end，这个级别是WAL，“relies on an acknowledgement, and will retry if no acknowledgement is received”.

- agentDFOSink[("machine"[,port])]

  DFO，当agent发现在collector操作失败的时候，agent写入到本地硬盘上，当collctor恢复后，再重新发送数据。

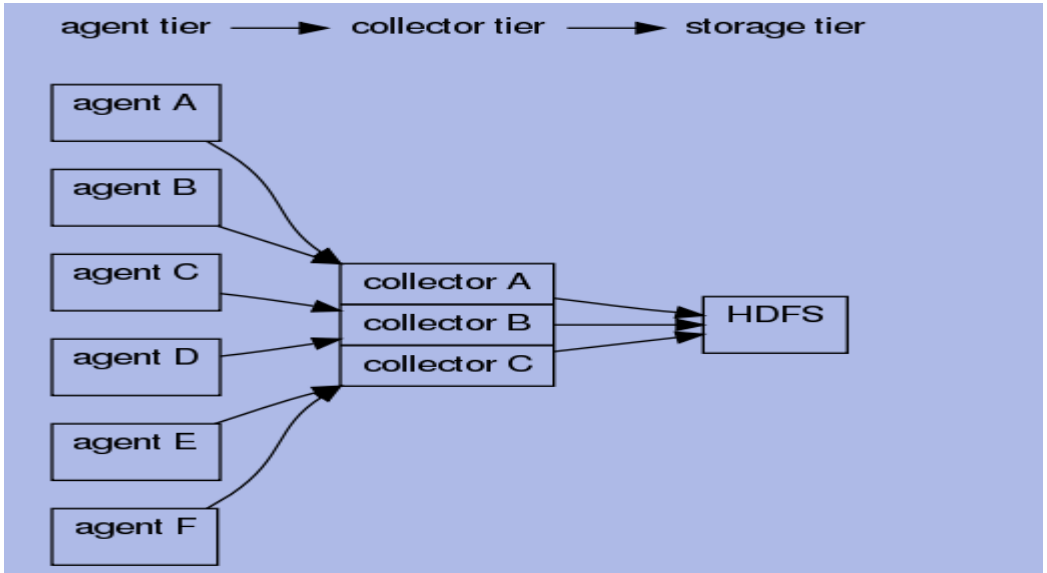
- agentBESink[("machine"[,port])]

  效率最好，agent不写入到本地任何数据，如果在collector 发现处理失败，直接删除消息。

- AgentSink 是agentE2ESink 的别名

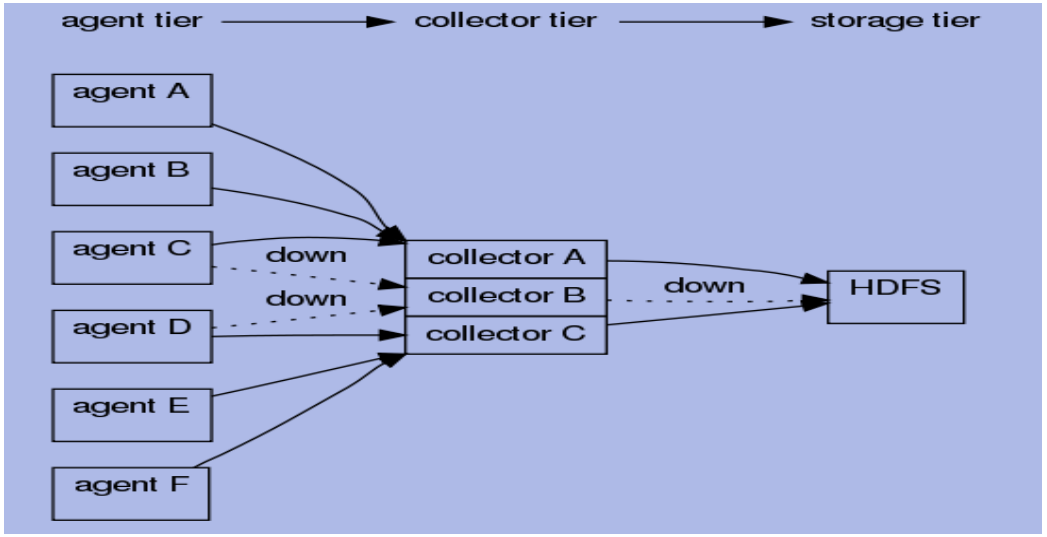
Flume OG集群拓扑配置(2)

agentA : src | agentE2ESink("collectorA",35853);
agentB : src | agentE2ESink("collectorA",35853);
agentC : src | agentE2ESink("collectorB",35853);
agentD : src | agentE2ESink("collectorB",35853);
agentE : src | agentE2ESink("collectorC",35853);
agentF : src | agentE2ESink("collectorC",35853);
collectorA : collectorSource(35853) | collectorSink("hdfs://...","src");
collectorB : collectorSource(35853) | collectorSink("hdfs://...","src");
collectorC : collectorSource(35853) | collectorSink("hdfs://...","src");



Flume OG集群拓扑配置(2)

agentA : src | agentE2EChain("collectorA:35853","collectorB:35853");
agentB : src | agentE2EChain("collectorA:35853","collectorC:35853");
agentC : src | agentE2EChain("collectorB:35853","collectorA:35853");
agentD : src | agentE2EChain("collectorB:35853","collectorC:35853");
agentE : src | agentE2EChain("collectorC:35853","collectorA:35853");
agentF : src | agentE2EChain("collectorC:35853","collectorB:35853");
collectorA : collectorSource(35853) | collectorSink("hdfs://...","src");
collectorB : collectorSource(35853) | collectorSink("hdfs://...","src");
collectorC : collectorSource(35853) | collectorSink("hdfs://...","src");



Flume NG部署

步骤1： 下载flume安装包，并解压到各台机器上；

步骤2： 修改etc/profile文件

export FLUME_HOME=/opt/software/flume-1.4.0.tar.gz
export PATH=.:$PATH::$FLUME_HOME/bin

步骤3： 验证安装

安装完毕后，运行flume命令，会打印flume的用法

步骤4： 修改配置文件$FLUME_HOME/conf

步骤5： 运行命令启动agent

Flume NG部署-配置文件格式

- **Java Properties 文件格式**

```
# 注释
key1 = value
key2 = multi-line \
                value
```

- **层级配置**

agent1.channels.**myChannel**.type = FILE
agent1.channels.**myChannel**.capacity = 1000

- **使用软引用配置链接关系**

agent1.sources.mySource.type = HTTP
agent1.sources.mySource.channels = ***myChannel***

Flume NG部署-简单配置实例

agent1.properties:

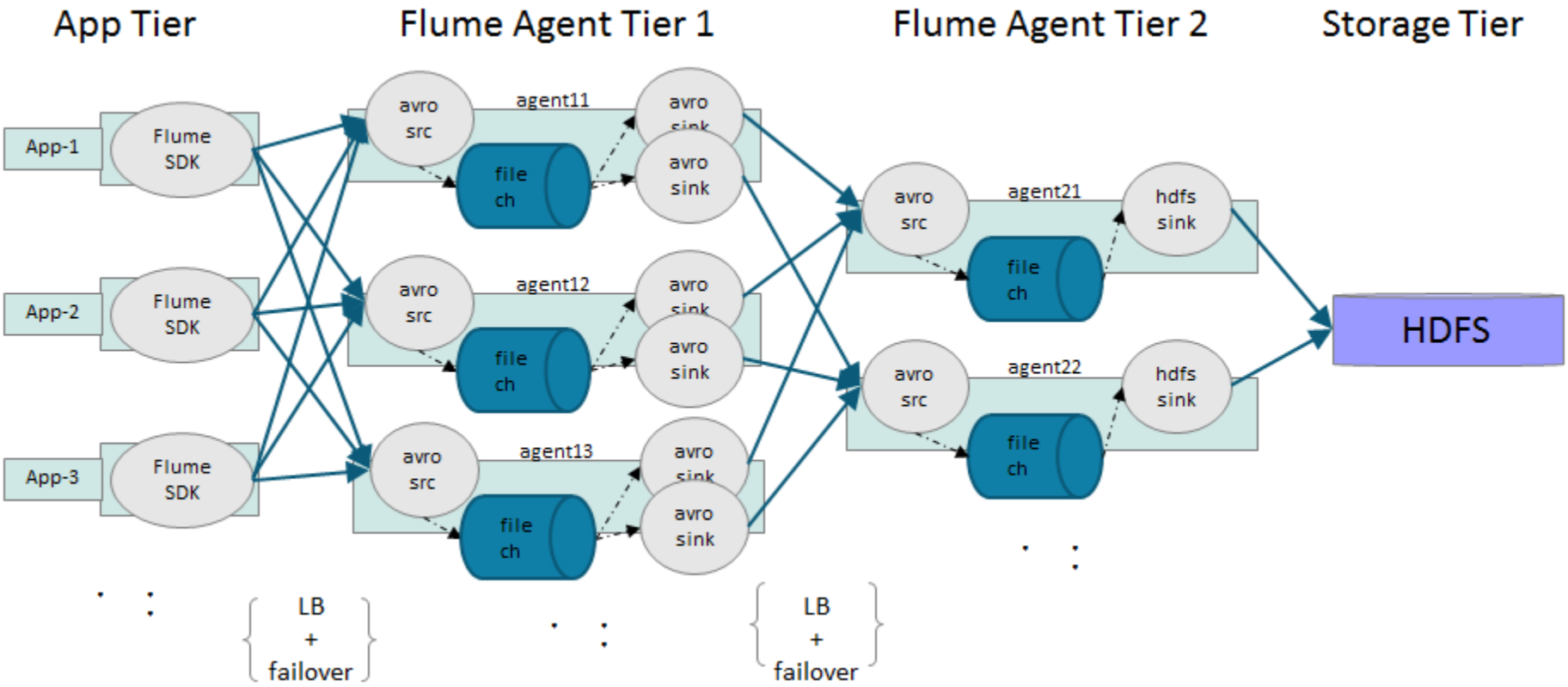
```
# 定义source、channel和sink名称
agent1.sources = src1
agent1.channels = ch1
agent1.sinks = sink1

#定义并配置 src1
agent1.sources.src1.type = netcat
agent1.sources.src1.channels = ch1
agent1.sources.src1.bind = 127.0.0.1
agent1.sources.src1.port = 10112

#定义并配置 sink1
agent1.sinks.sink1.type = logger
agent1.sinks.sink1.channel = ch1

# 定义并配置 ch1
agent1.channels.ch1.type = memory
```

Flume NG部署-实例(1)



Flume NG部署-实例(1)

第一层（Tier 1）配置实例

```
a1.channels = c1
a1.sources = r1
a1.sinks = k1 k2
a1.sinkgroups = g1

a1.sinkgroups.g1.processor.type = LOAD_BALANCE
a1.sinkgroups.g1.processor.selector = ROUND_ROBIN
a1.sinkgroups.g1.processor.backoff = true

a1.channels.c1.type = FILE

a1.sources.r1.channels = c1
a1.sources.r1.type = AVRO
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = 41414
```

```
a1.sinks.k1.channel = c1
a1.sinks.k1.type = AVRO
a1.sinks.k1.hostname = a21.example.org
a1.sinks.k1.port = 41414

a1.sinks.k2.channel = c1
a1.sinks.k2.type = AVRO
a1.sinks.k2.hostname = a22.example.org
a1.sinks.k2.port = 41414
```

Flume NG部署-实例(1)

第二层（Tier 2）配置实例

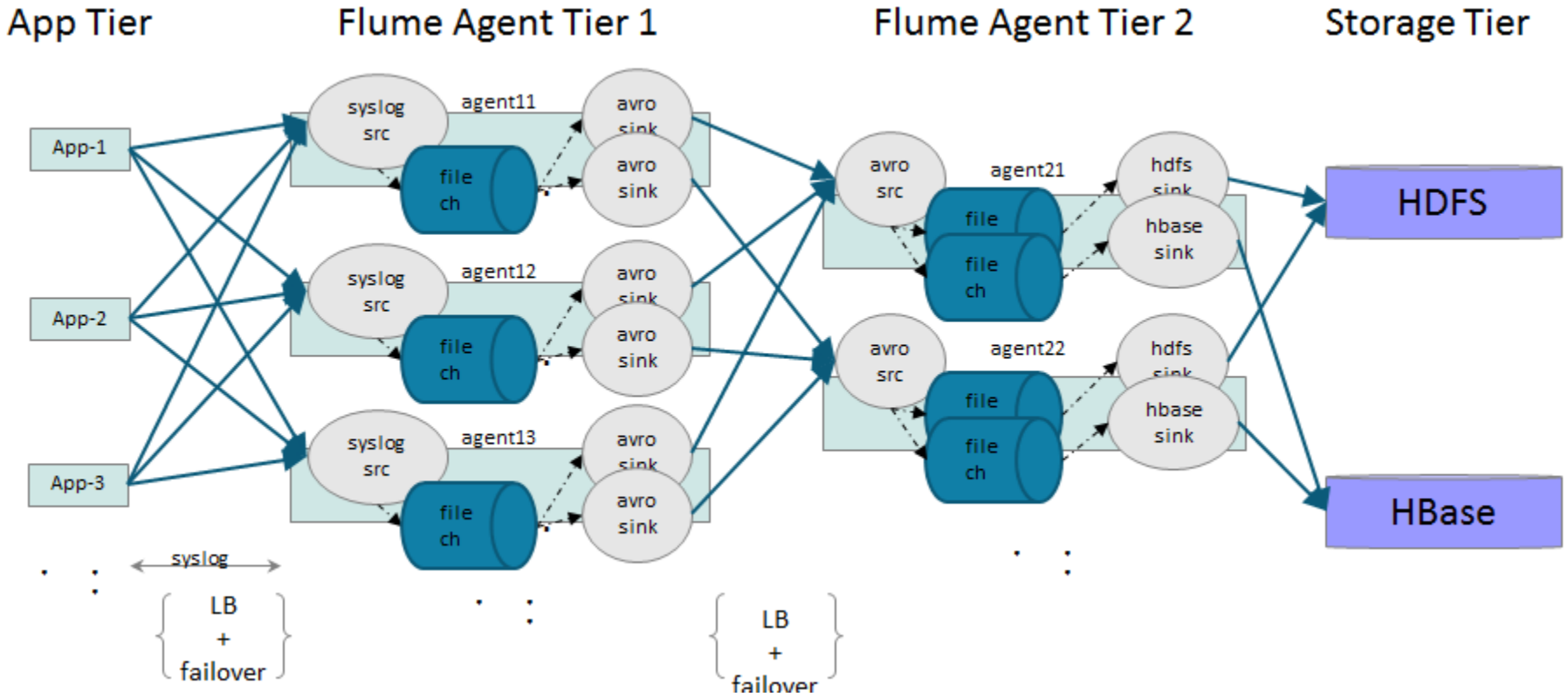
```
a2.channels = c1
a2.sources = r1
a2.sinks = k1

a2.channels.c1.type = FILE

a2.sources.r1.channels = c1
a2.sources.r1.type = AVRO
a2.sources.r1.bind = 0.0.0.0
a2.sources.r1.port = 41414

a2.sinks.k1.channel = c1
a2.sinks.k1.type = HDFS
a2.sinks.k1.hdfs.path = hdfs://namenode.example.org
a2.sinks.k1.hdfs.fileType = DataStream
```

Flume NG部署-实例(2)



Flume NG部署-实例(2)

第一层（Tier 1）配置实例

```
a1.channels = c1
a1.sources = r1
a1.sinks = k1 k2
a1.sinkgroups = g1
a1.sinkgroups.g1.sinks = k1 k2
a1.sinkgroups.g1.processor.type = LOAD_BALANCE
a1.sinkgroups.g1.processor.selector = ROUND_ROBIN
a1.sinkgroups.g1.processor.backoff = true
a1.channels.c1.type = FILE

a1.sources.r1.channels = c1
a1.sources.r1.type = SYSLOGTCP
a1.sources.r1.host = 0.0.0.0
a1.sources.r1.port = 41414

a1.sinks.k1.channel = c1
a1.sinks.k1.type = AVRO
a1.sinks.k1.hostname = a21.example.org
a1.sinks.k1.port = 41414
a1.sinks.k2.channel = c1
a1.sinks.k2.type = AVRO
a1.sinks.k2.hostname = a22.example.org
a1.sinks.k2.port = 41414
```

Flume NG部署-实例(2)

第二层（Tier 2）配置实例

```
a2.channels = c1  c2
a2.sources = r1
a2.sinks = k1 k2
a2.sinkgroups = g1

a2.sinkgroups.g1.sinks = k1 k2
a2.sinkgroups.g1.processor.type = LOAD_BALANCE
a2.sinkgroups.g1.processor.selector = ROUND_ROBIN
a2.sinkgroups.g1.processor.backoff = true

a2.channels.c1.type = FILE
a2.channels.c1.checkpointDir = /var/run/flume-ng/.flume/ch-1/checkpoint
a2.channels.c1.dataDirs = /var/run/flume-ng/.flume/ch-1/data

a2.channels.c2.type = FILE
a2.channels.c2.checkpointDir = /var/run/flume-ng/.flume/ch-2/checkpoint
a2.channels.c2.dataDirs = /var/run/flume-ng/.flume/ch-2/data
```

```
a2.sources.r1.channels = c1 c2
a2.sources.r1.type = AVRO
a2.sources.r1.bind = 0.0.0.0
a2.sources.r1.port = 41414
a2.sources.r1.selector.type = MULTIPLEXING
a2.sources.r1.selector.header = Severity
a2.sources.r1.selector.default = c1
a2.sources.r1.selector.mapping.0 = c1 c2
a2.sources.r1.selector.mapping.1 = c1 c2
a2.sources.r1.selector.mapping.2 = c1 c2
a2.sources.r1.selector.mapping.3 = c1 c2
a2.sinks.k1.channel = c1
a2.sinks.k1.type = HDFS
a2.sinks.k1.hdfs.path = hdfs://nn.example.org/demo/%Y-%m-%d/%H%M/
a2.sinks.k1.hdfs.filePrefix = FlumeData-%{host}-
a2.sinks.k1.hdfs.fileType = DataStream
a2.sinks.k1.hdfs.round = true
a2.sinks.k1.hdfs.roundUnit = minute
a2.sinks.k1.hdfs.roundValue = 10
a2.sinks.k2.channel = c2
a2.sinks.k2.type = org.apache.flume.sink.hbase.AsyncHBaseSink
a2.sinks.k2.table = mytable1
a2.sinks.k2.columnFamily = mycolfam1
```

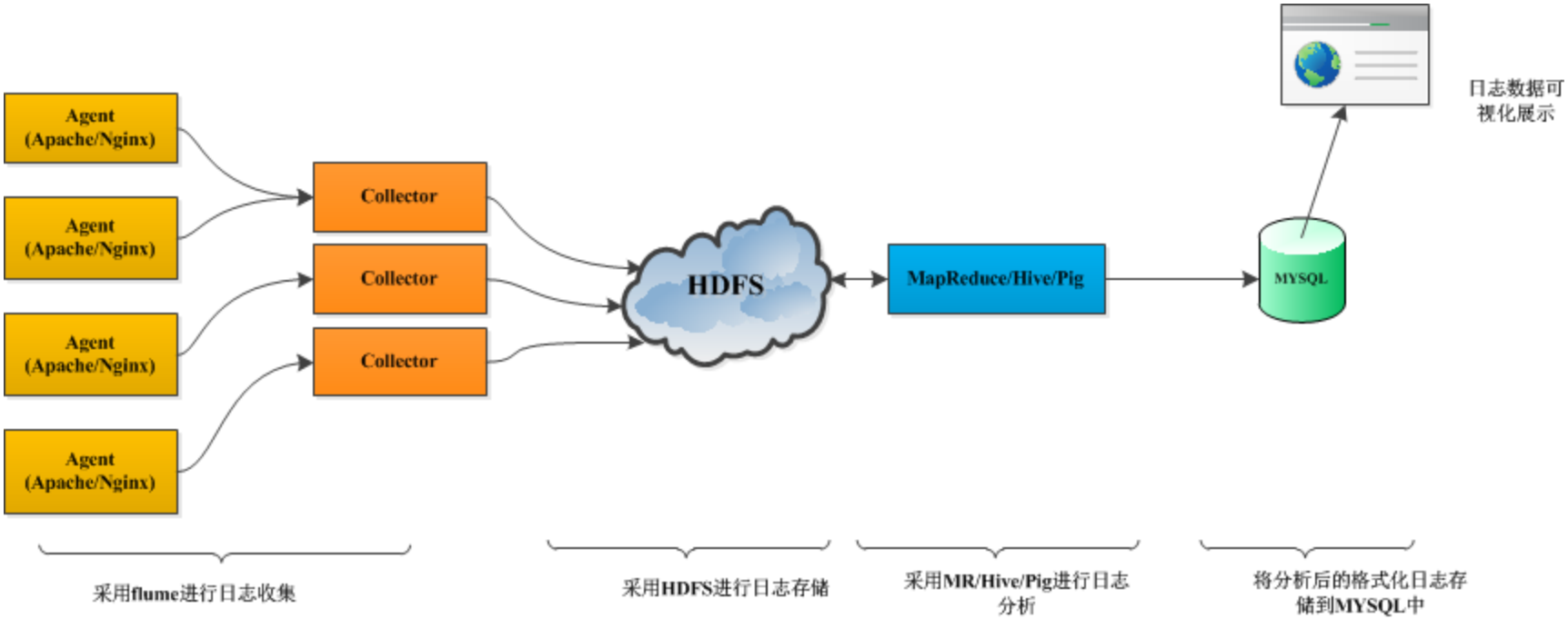
目录

日志集群构建

Flume：日志收集

HDFS/HBase：日志存储

Hive：日志分析



总结

1. Flume背景及应用场景
2. Flume OG基本架构
3. Flume NG基本架构
4. Flume部署
5. Flume案例分析