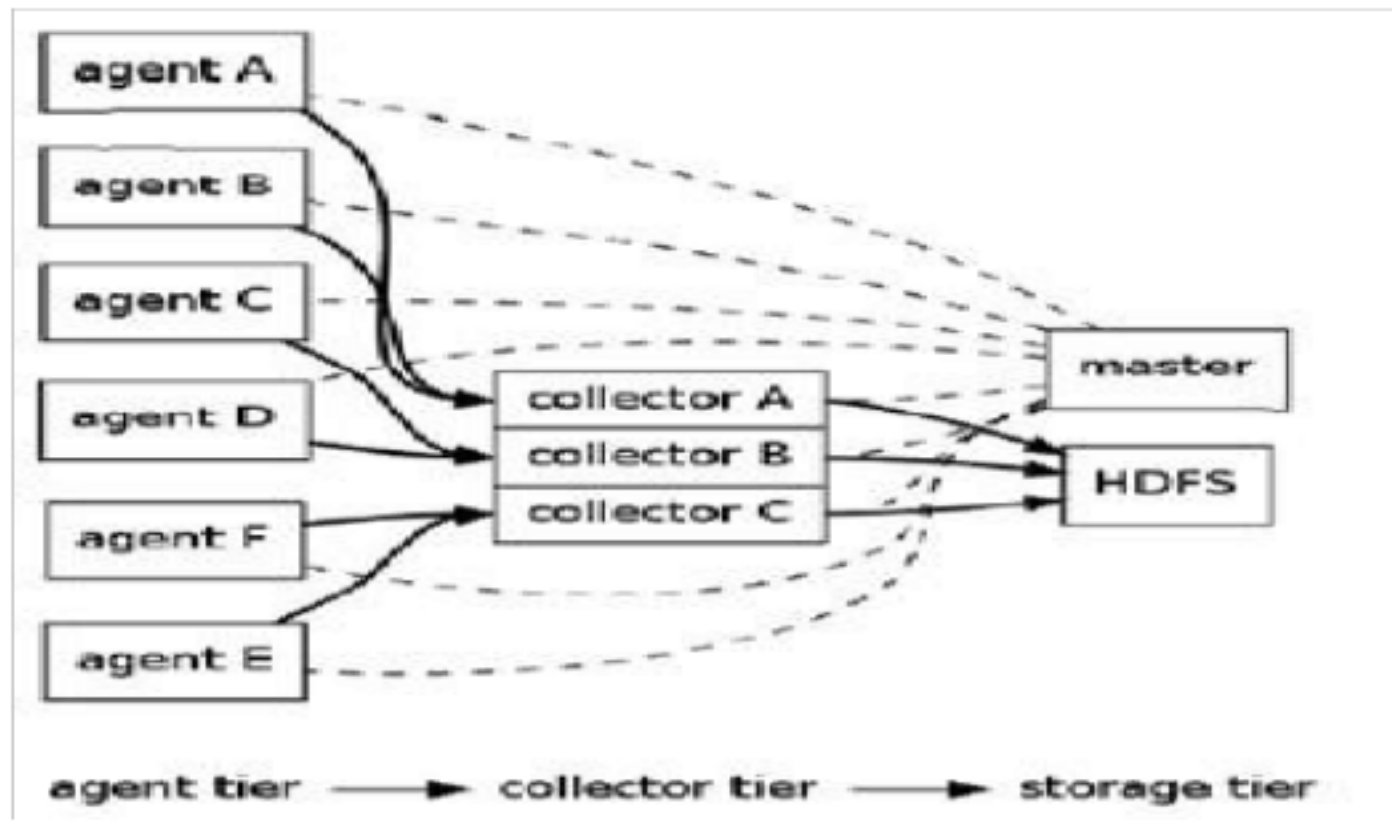
Flume 集群

Flume 介绍

Flume 是一个分布式、可靠、和高可用的海量日志采集、聚合和传输的系统，支持在系统中定制各类数据发送方，用于收集数据；同时，Flume 提供对数据进行简单处理，并写到各种数据接受方（可定制）的能力。

Flume 的逻辑架构：



Flume 逻辑上分三层架构： agent ， collector ， storage

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 。

collector

collector 的作用是将多个 agent 的数据汇总后，加载到 storage 中。

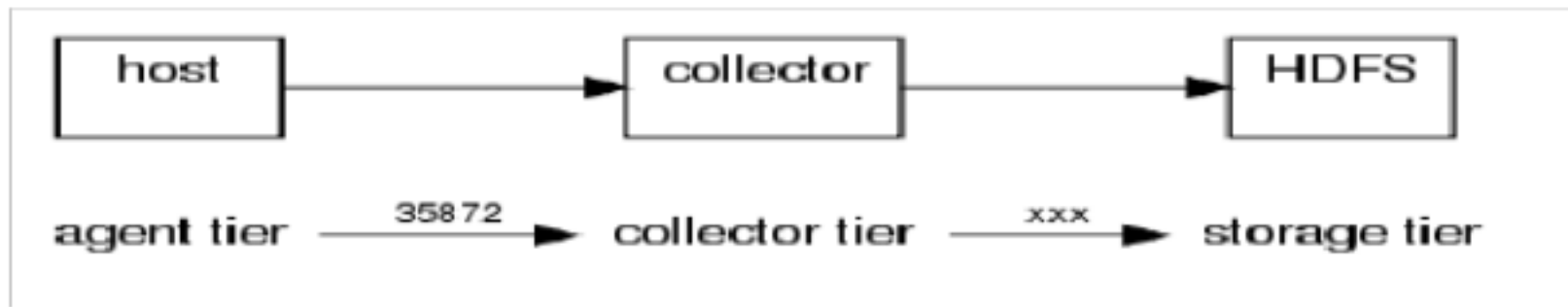
storage

storage 是存储系统，可以是一个普通 file ，也可以是 HDFS，HIVE，HBase 等。

Master

Master 是管理协调 agent 和 collector 的配置等信息，是 flume 集群的控制器。

在 Flume 中，最重要的抽象是 data flow （数据流），data flow 描述了数据从产生，传输、处理并最终写入目标的一条路径。



对于 agent 数据流配置就是从哪得到数据，把数据发送到哪个 collector 。

对于 collector 是接收 agent 发过来的数据，把数据发送到指定的目标机器上。

Flume 的特性

? Reliability ：Flume 提供 3 中数据可靠性选项，包括 End-to-end 、Store on failure 和 Best effort 。其中 End-to-end 使用了磁盘日志和接受端 Ack 的方式，保证 Flume 接受到的数据会最终到达目的，但是效率是最差的。 Store on failure 在目的不可用的时候，数据会保持在本地硬盘，效率会比 end-to-end 高，但是会出现日志丢失的情况。 Best effort 不做任何 QoS保证，效率最高，日志记录没有保证。

? Scalability ：Flume 的 3 大组件： collector 、master 和 storage tier 都是可伸缩的。需要注意的是， Flume 中对事件的处理不需要带状态，它的 Scalability 可以很容易实现。

? Manageability ：master 能够动态管理 flume 集群节点，多 master 情况， Flume 利用 ZooKeeper 和 gossip ，保证配置数据的一致性。

? Extensibility ：基于 Java , 用户可以为 Flume 添加各种新的功能， 如通过继承 Source , 用户可以实现自己的数据接入方式， 实现 Sink 的子类， 用户可以将数据写往特定目标, 同时，通过 SinkDecorator ，用户可以对数据进行一定的预处理。

注：Flume 框架对 hadoop 和 zookeeper 的依赖只是在 jar 包上，并不要求 flume 启动时必须将 hadoop 和 zookeeper 服务也启动。

Flume 的分布式安装

———— 此为目前集群的 flume 安装过程

部署 flume 在集群上，按照如下步骤：

在集群上的每台机器上安装 flume

选择一个或多个节点当做 master

修改静态配置文件

在至少一台机器上启动一个 master ，所有节点启动 flume node

接下来的一章描述如何手动修改配置文件为集群上的节点指定 master ，如何为参数设置默认值，本章的后一部分描述对于大系统的数据流配置， 如何通过增加 collector 来扩充系统容量，如何提高可靠性通过增加更多的 master 。注意： flume 集群整个集群的网络环境要保证稳定， 可靠，否则会出现一些莫名错误（比如： agent 端发送不了数据到 collector ）。

集群每台机器上安装 flume

场景：

操作系统版本： CentOS5.6

Hadoop 版本： 0.20.2

Jdk 版本： jdk1.6.0_26

安装 flume 版本： flume-0.9.4

步骤 1：

下载 flume 最新版本，现在服务器上安装的是 flume-distribution-0.9.4 的版本，下载地址是 https://github.com/cloudera/flume/downloads 目前 flume 的安装包是放在 /data/sysdir/install_tar 文件件的下面。

步骤 2：

解压 flume 安装包到 /data/sysdir 文件夹下面,

在命令行中输入 tar zxvf /data/sysdir/install_tar/flume-distributin-0.9.4.tar.gz -C

/data/sysdir

步骤 3：

修改 etc/profile 文件，加入：

export FLUME_HOME=/data/sysdir/flume-distribution-0.9.4

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

步骤 4：

验证安装及其他

安装完毕后，运行 flume 命令，会看到以下输出：

```
usage: flume command [args...]
commands include:
  dump           Takes a specified source and dumps to console
  node           Start a Flume node/agent (with watchdog)
  master         Start a Flume Master server (with watchdog)
  version        Dump flume build version information
  node_nowatch   Start a flume node/agent (no watchdog)
  master_nowatch Start a Flume Master server (no watchdog)
  class <class> Run specified fully qualified class using Flume environment (no
watchdog)
                for example: flume com.cloudera.flume.agent.FlumeNode
  shell          Start the flume shell
  killmaster     Kill a running master
```

flume 配置文件位置： $FLUME_HOME/conf 下

选择一个或多个节点当做 master

对于 master 的选择情况， 可以在集群上定义一个 master ， 也可以为了提高可用性选择多个节点做为 master,

单点 master 模式：容易管理，但在系统的容错和扩展性有缺陷

多点 master 模式：通常是运行 3/5 个 master ， 能很好的容错

原文如下：（http://archive.cloudera.com/cdh/3/flume/UserGuide/ ）

Standalone mode - this is where the Master runs on a single machine. This is easy to administer, and simple to set-up, but has disadvantages when it comes to scalability and fault-tolerance.

Distributed mode - this is where the Master is configured to run on several machines - usually three or five. This option scales to serve many Flows, and also has good fault-tolerance properties.

Flume master 数量的选择原则

原文如下：

The distributed Flume Master will continue to work correctly as long as more than half the physical machines running it are still working and haven ’t crashed. Therefore if you want to survive one fault, you need three machines (because 3-1 = 2 > 3/2). For every extra fault you want to tolerate, add another two machines, so for two faults you need five machines. Note that having an even number of machines doesn ’t make the Flume Master

any more fault-tolerant - four machines only tolerate one failure, because if two were to fail only two would be left functioning, which is not more than half of four. Common deployments should be well served by three or five machines.

分布式的 master 能够继续正常工作不会崩溃，的前提是正常工作的 master 数量超过总 master 数量的一半。

Flume master 的作用主要有两个：

原文如下：

The Master has two main jobs to perform. The first is to keep track of all the nodes in a Flume deployment and to keep them informed of any changes to their configuration.

The second is to track acknowledgements from the end of a Flume flow that is operating in reliable mode so that the source at the top of that flow knows when to stop transmitting an event.

Master 主要有两个工作，第一是跟踪各节点的配置情况，通知节点配置的改变，第二是跟踪来自 flow 的结尾操控在可靠的模式下（ E2E）的信息，以至于让 flow 的源头知道什么时候停止传输 event 。

目前集群 flume master 的选择情况

10.168.0.174 、10.168.0.181 、10.168.0.188 为 flume master

修改静态配置文件

Site-specific 设置对于 flume 节点和 master 通过在每一个集群节点的 conf/flume-site.xml 是可配置的，如果这个文件不存在，设置的属性默认的在 conf/flume--conf.xml 中，在接下来的例子中，在 flume 的节点上设置 master 名，让节点自己去寻找叫“ master ”的 flume Master

conf/flume-conf.xml.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"  href="configuration.xsl"?>
<configuration>
<property>
<name>flume.master.servers</name>
<value>master</value>
</property>
</configuration>
```

在多 master 的情况下需要如下配置：

```
<property>
    <name>flume.master.servers</name>
    <value>hadoopmaster.com,hadoopedge.com,datanode4.com</value>
    <description>A comma-separated list of hostnames, one for each
      machine in the Flume Master.
    </description>
  </property>
```

<property>
<name>flume.master.store</name>
<value>zookeeper</value>
<description>How the Flume Master stores node configurations. Must
be either 'zookeeper' or 'memory'.</description>
</property>

<property>
<name>flume.master.serverid</name>
<value>2</value>
<description>The unique identifier for a machine in a
Flume Master ensemble. Must be different on every
master instance.</description>
</property>

注意：flume.master.serverid 属性的配置主要是针对 master，如在集群上目前是 181,174,188 为 master，这三台机器的 $FLUME_HOME/conf/flume-conf.xml 文件中 flume.master.serverid 必须是不能相同的，该属性的值以 0 开始。

当使用 agent 角色时，你可以通过添加下面的配置文件在 flume-conf.xml 中，来设置默认的 collector 主机：

...

<property>

<name>flume.collector.event.host</name>

<value>collector</value>

<description>This is the host name of the default "remote" collector.

</description>

</property>

<property>

<name>flume.collector.port</name>

<value>35853</value>

<description>This default tcp port that the collector listens to in order to receive

events it is collecting.

</description>

</property>

...

修改配置文件时，根据物理主机在不同的 flume 逻辑层，修改相应的属性的值即可。

在 agent 上，需要修改 flume.collector.event.host 属性，来指定此 agent 默认发送到的 collector

在 cllector 上，不需要修改特定的属性

在 master 上，flume.master.serverid 属性的配置主要是针对 master，如在集群上目前是 181,174,188 为 master，这三台机器的 $FLUME_HOME/conf/flume-conf.xml 文件中 flume.master.serverid 必须是不能相同的，该属性的值以 0 开始。

为简便起见，其他属性保持一致。

关于配置可参见： http://www.cnblogs.com/zhangmiao-chp/archive/2011/05/18/2050443.html 。

其中目前集群调整的配置文件属性：

flume.agent.logdir 该属性是配置 agent 临时存放文件的路径 , 尽可能不要放到 /tmp 下
flume.agent.logdir.maxage 该属性是 agent 日志文件收集信息的时长，根据集群情况可适当调整。
flume.collector.roll.millis 该属性 hdfs 文件切换（关闭后新建）的时长，控制在 hdfs 上生成的文件大小，如果调整该属性需要同时调整 flume.agent.logdir.retransmit 的属性，一般情况下最多是 flume.agent.logdir.retransmit 的一半。
flume.collector.output.format 该属性是 collector 发送数据格式 avro, avrojson(默认), avrodata, , 目前集群是 raw

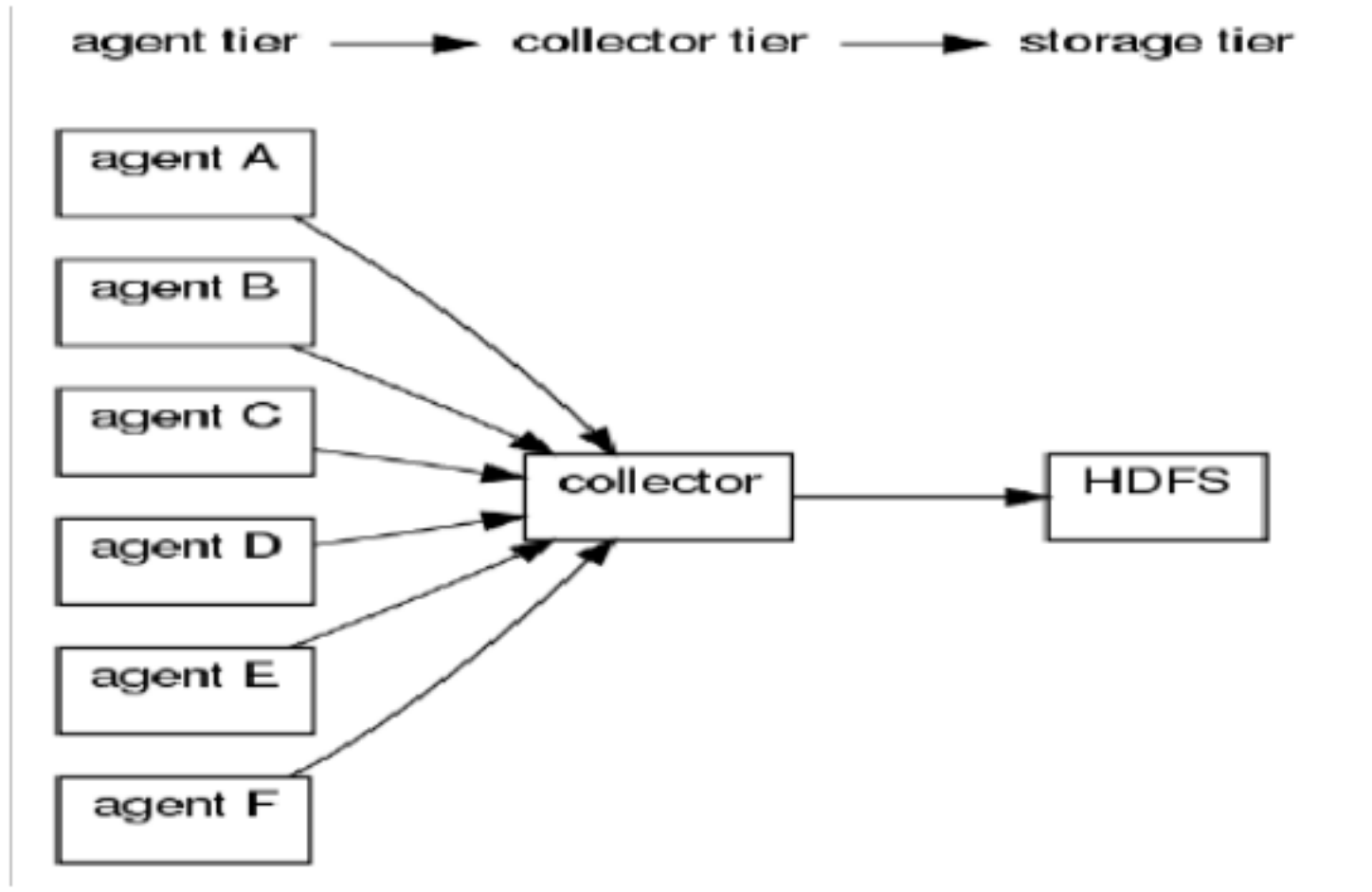
启动 Flume 集群节点

集群上节点启动：

1 在命令行输入： flume master 启动 master 节点
2 在命令行输入： flume node －n nodeName 启动其他节点， nodeName最好根据集群逻辑的划分来取名子，这样在 master 进行配置的时候比较清晰。
如目前集群上的 181 和 188 是 collector 所以 nodeName 可以选择 collector1 collector2
名字规则自己定义，方便记忆和动态配置即可（后续会有介绍动态配置）

Flume 集群动态配置

Flume 的动态配置指的是在 flume shell 下进行配置 agent 和 collector ，具体步骤如下：
在已经启动的 master 节点新开窗口输入 依次输入 ”flume shell “ ”connect localhost ”此时可以进行动态的配置。
如执行 exec config a1 ‘tailDir(“/data/logfile ”)’ ‘agentSink’
表示 nodeName 是 a1 的机器监听 /data/logfile 文件夹下的日志，发送信息到 flume-conf.xml 配置的 flume.collector.event.host 的 flume.collector.port 端口。
如执行配置 collector 的命令： exec config c1 ‘collecotrSource’ ‘collectorSink(“hdfs://hadoopmaster.com:9000/flume/webdata/%Y -%m-%d/%H”，”syslog ”)
表示 nodeName 是 c1 的机器监听配置文件中 flume.collector.port 的端口，发送信息到 hdfs 上
其中 /%Y-%m-%d/%H表示 / 年 - 月 - 日 / 时 建立发送文件到达的文件夹， 这个时间以 agent 读取日志时产生的文件时间为准， syslog 是文件名，创建文件的时间是 flume.collector.roll.millis 属性决定。
其他 command shell 参见 http://www.cnblogs.com/zhangmiao-chp/archive/2011/05/18/2050461.html
接下来的例子是 六个 agent 一个 collector



显式的配置：

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");

可靠性模式

你可以运行 agent 的可靠性级别， 简单的配置不同的 agentSink 就可以， 下面有三个级别的配置

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，如果出现存储在硬盘上的数据， agent 重新进行网络连接，并试图重新发送数据。

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

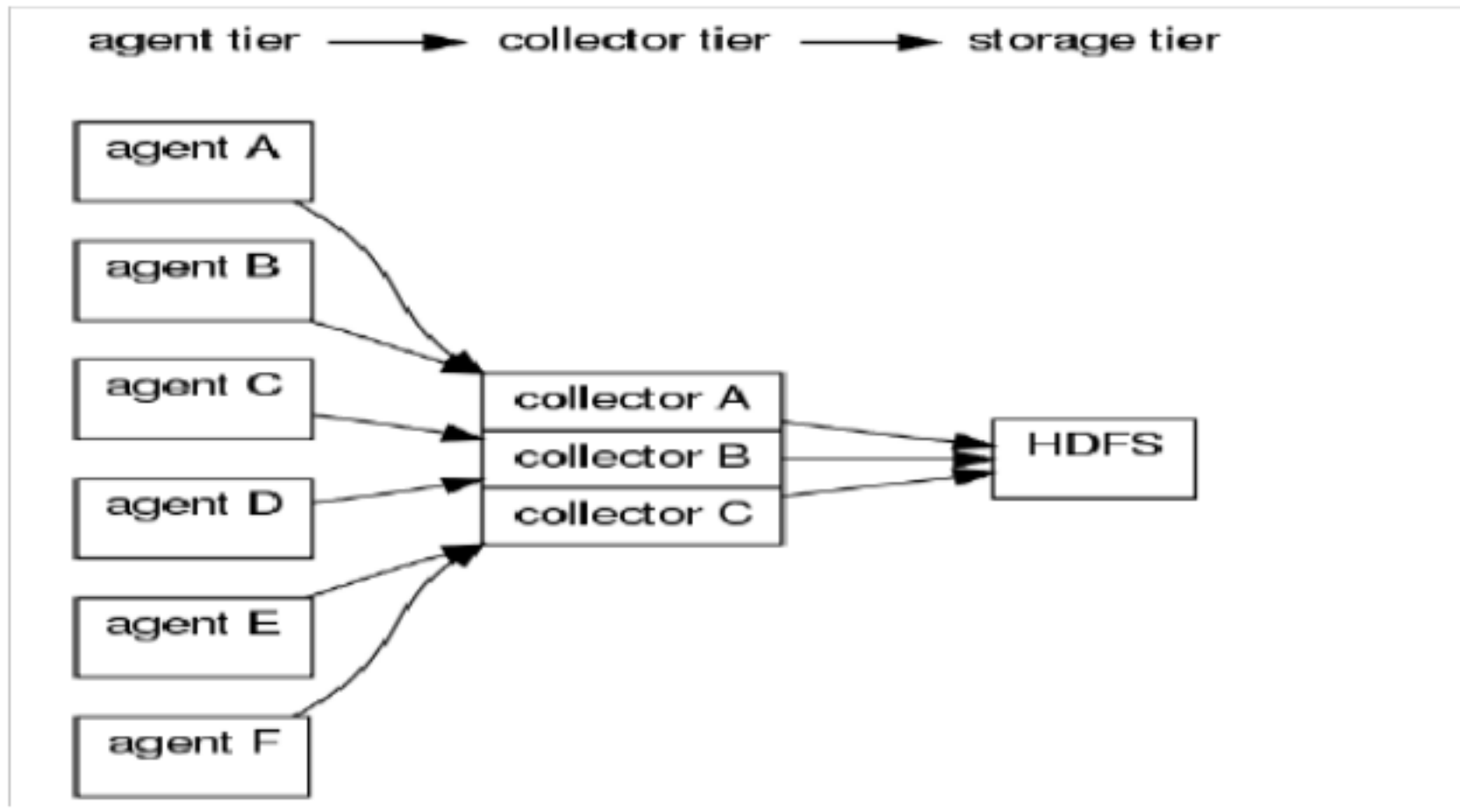
补充说明

多个 collector 能够增加日志收集的吞吐量，提高 collector 的有效性能够提高数据的传输速度， 数据的收集是可并行的， 此外， 来自多个 agent 的数据能够分配到多个 collector 上加载。

多 collector 下 agent 的划分

前面的图展示 flume 节点典型的拓扑结构和数据流， 为了可靠的传输， 当 collector 停止运行或

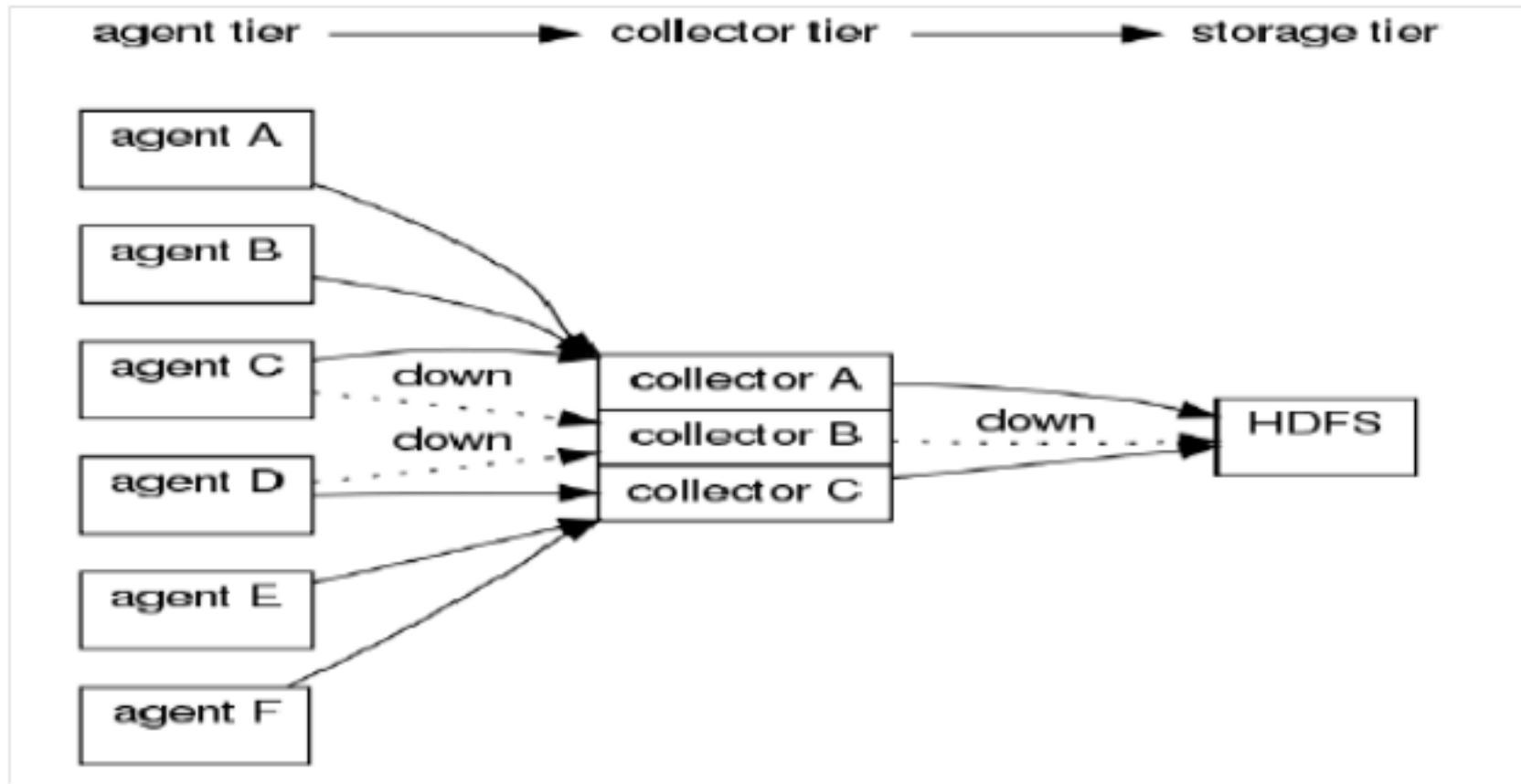
是失去与 agents 的联系的时候, agents 将会存储他们的 events 在各自的本地硬盘上，这些 agents 试图重新连接 collector ，因为 collector 的宕机，任何处理和分析的数据流都被阻塞。



当你有多个 collector 如上图所示, 即使在 collector 宕机的情况下, 数据处理仍然能够进行下去，如果 collector b 宕机了, agent a, agent b, ageng e, 和 agent f 会分别继续传送 events 通过 collector a 和 collector c , agent c 和 agent d 的不得不排在其他 agent 的后面等待日志的处理直到 collector b 重新上线。

接下来的配置划分 agents 在多 collector ，这个例子是每一个 collector 由同一个输出的 dfs 路径和文件的前缀名，聚合所有的日志到同一个目录下

```
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");
```



当多个 collector 写入到相同的存储位置时，你可以设置 agent c agent d 当他们的 collector 出现错误时转向其他的 collector ，可以分别设置 agent c 和 agent d 转移到 collector a 和 collector c 上。

用 agents 的 failover chains 。和指向单独的 collector （agentSink ）类似， failover chains 也有三个可靠性的级别 agentE2EChain, agentDFOChain, and agentBEChain.

下面的例子，手动设置失败转移链表用 agentE2EChain ，有多个失败转移 collector 的 agent 的 end-to-end 的可靠性级别， agentA 默认的将数据发送到 collectorAd 端口 35853，第二个参数在 agentA 's sink 是指定备用的 collector 。你可以定义任意数量的 collector ，（至少一个以上）。

```
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");
```

注意：这章中， agent[A-F] 和 collector[A-B] 是物理节点的名字

注意：自动失败转移链表功能还不能用在多 master 的集群。

Flume 集群测试以及节点失败后的处理

Flume 集群出现错误的节点按照逻辑划分，分别属于 agent ， collector ， master ，这三层的逻辑架构的节点出错可以使单独一个节点出错，也可以是组合出错。

目前集群测试情况

目前集群测试环境：

说明：目前集群的测试主要通过上传文件到 hdfs 上然后通过 hive 查询文件的记录条数。

场景一：三台 master （174/176,181,188 ），两台 collector(181 ， 188) ，一台 agent(235) ，传送的文件有据说有 3000000 条记录， agent 配置 188-181 。

Session1 ：不做任何异常测试， hive 建立表是 logflumeSession1， 结果是：数据条数： 300 万

Session2 ：两个 collector 都宕机，数据条数： 3427971

```
Ended Job =
OK
3427971
Time taken:
```

Session3 ：宕机一个 cllector ，和一个 master ，然后重新启动 collector ， master ，数据量：

```
Ended Job = job_201110141006_001
OK
3462770
Time taken: 20.033 seconds
```

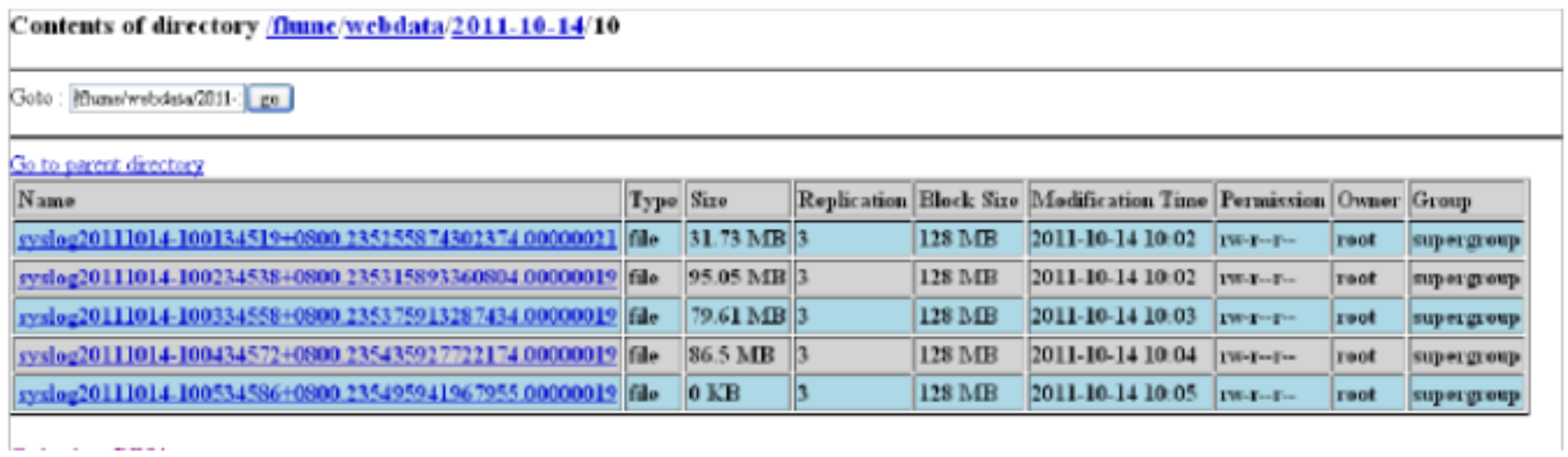
Session4: 宕机 agent ，数据会重新发送

场景 n：三台 master （174/176,181,188 ），一台 collector(188) ，一台 agent(235) ，传送的文件有 6000000 条记录， colletcor 和 agent 不在 174 节点的 master 上

出现的状况： 174 宕机

错误情况：不能正常传送文件到 hdfs 上， collector 报错信息如图

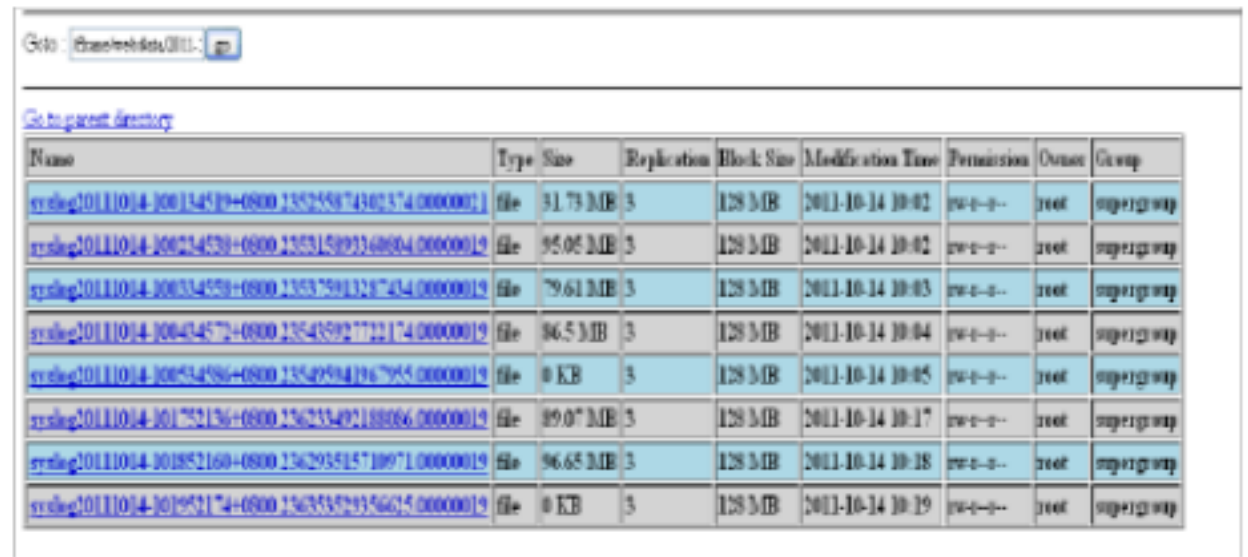
Hdfs 无法写入数据



Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
syslog20111014-100134519+0800.13515587430237 4.00000021	file	31.73 MB	3	128 MB	2011-10-14 10:02	rw-r--r--	root	supergroup
syslog20111014-100234538+0800.13531589336080 4.00000019	file	95.05 MB	3	128 MB	2011-10-14 10:02	rw-r--r--	root	supergroup
syslog20111014-100334558+0800.13537591328743 4.00000019	file	79.61 MB	3	128 MB	2011-10-14 10:03	rw-r--r--	root	supergroup
syslog20111014-100434572+0800.13543592772217 4.00000019	file	86.5 MB	3	128 MB	2011-10-14 10:04	rw-r--r--	root	supergroup
syslog20111014-100534586+0800.13549594196795 5.00000019	file	0 KB	3	128 MB	2011-10-14 10:05	rw-r--r--	root	supergroup

解决办法：重新启动　　collector

结果：数据正常传输



Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
syslog20111014-100134519+0800.13515587430237 4.00000021	file	31.73 MB	3	128 MB	2011-10-14 10:02	rw-r--r--	root	supergroup
syslog20111014-100234538+0800.13531589336080 4.00000019	file	95.05 MB	3	128 MB	2011-10-14 10:02	rw-r--r--	root	supergroup
syslog20111014-100334558+0800.13537591328743 4.00000019	file	79.61 MB	3	128 MB	2011-10-14 10:03	rw-r--r--	root	supergroup
syslog20111014-100434572+0800.13543592772217 4.00000019	file	86.5 MB	3	128 MB	2011-10-14 10:04	rw-r--r--	root	supergroup
syslog20111014-100534586+0800.13549594196795 5.00000019	file	0 KB	3	128 MB	2011-10-14 10:05	rw-r--r--	root	supergroup
syslog20111014-101752136+0800.13623354921888 86.00000019	file	89.07 MB	3	128 MB	2011-10-14 10:17	rw-r--r--	root	supergroup
syslog20111014-101852160+0800.13629351571097 1.00000019	file	96.65 MB	3	128 MB	2011-10-14 10:18	rw-r--r--	root	supergroup
syslog20111014-101952174+0800.13635352935662 5.00000019	file	0 KB	3	128 MB	2011-10-14 10:19	rw-r--r--	root	supergroup

数据的总行数：

数据会增加。

单一逻辑层的节点失败及处理

单一逻辑层的节点出错，情况有四种　　agent ，collector　，master ，storage　，storage　。 storage 层的失败和　collector　　层的失败是一样的，只要数据放不到最终的位置，就认为节点是失败的。

agent 失败

Flume 数据安全级别的配置主要　　Agent 的配置上， Agent 提供三种级别发送数据到　　collector　　：

E2E、DFO、BF。

E2E：agent 会先将收集到的数据写到 agent 本地硬盘上，然后发送 collector ，collector 会将数据发送到最后数据的节点。确定数据已经成功保存到目标机器之后，collector 通知 agent 删除硬盘上文件。

DFO:当数据无法正常保存到目标主机时，agent 会将收集到的数据保到 agent 安装时指定的目录下，错误恢复之后，将记录的数据发送到 collector ，进入目标机器。

BF：agent 不做任何的日志记录，如果目标地址不可达则数据丢失。

agent 节点监控日志文件夹下的所有文件，每一个 agent 最多监听 1024 个文件，每一个文件在 agent 的都会有一个类似游标的东西，记录监听文件读取的位置，这样每次文件有新的记录产生，那么游标就会读取增量记录，根据 agent 配置发送到 collector 的安全层级属性有 E2E,DFO。如果是 E2E 的情况那么 agent 节点会首先把文件写入到 agent 节点的文件夹下，然后发送给 collector ，如果最终数据最终成功存储到 storage 层，那么 agent 删除之前写入的文件，如果没有收到成功的信息，那么就保留信息。

如果 agent 节点出现问题，那么相当于所有的记录信息都消失了，如果直接重新启动，agent 会认为日志文件夹下的所有文件都是没有监听过的，没有文件记录的标示，所以会重新读取文件，这样，日志就会有重复，具体恢复办法如下

将 agent 节点上监听的日志文件夹下已经发送的日志文件移出，处理完故障重新启动 agent 即可。

注：在 agent 节点失败的情况下，按照失败的时间点，将时间点之前的数据文件移出，将 flume.agent.logdir 配置的文件夹清空，重新启动 agent 。

collector 失败

collector 端做数据的合并，并且将数据发送到目标机器上去，如果 collector 端失败那么不用担心，因为 agent 的 E2E 模式会将没有发送成功的数据保存到 agent 的本地，等到 collector 恢复之后，数据会重新发送到 collector 上，数据不会丢失。

除了恢复故障节点之外不用做其他的额外处理，collector 的性能影响的是整个 flume 集群的数据吞吐量，所以 collector 最好单独部署。

master 失败

master 宕机，整个集群将不能工作，在重新启动集群，将 agent 监听的日志文件夹下的所有文件

移出，然后重新启动 master 即可。

在多 master 节点情况下，只要集群上正常工作的 master 大于总 master 数量的一半，集群就能正常工作，那么只要恢复其中宕机的 master 即可。

多逻辑层节点组合失败及处理

两个逻辑层组合出错，情况有 agent-collector , collector-master , agent-master , 实际上 agent

agent-collector　　组合

agent 和 collector　组合出现错误，　collector　和 stroage　出错一样，姑且将　collector　和 storage　放在一起考虑。

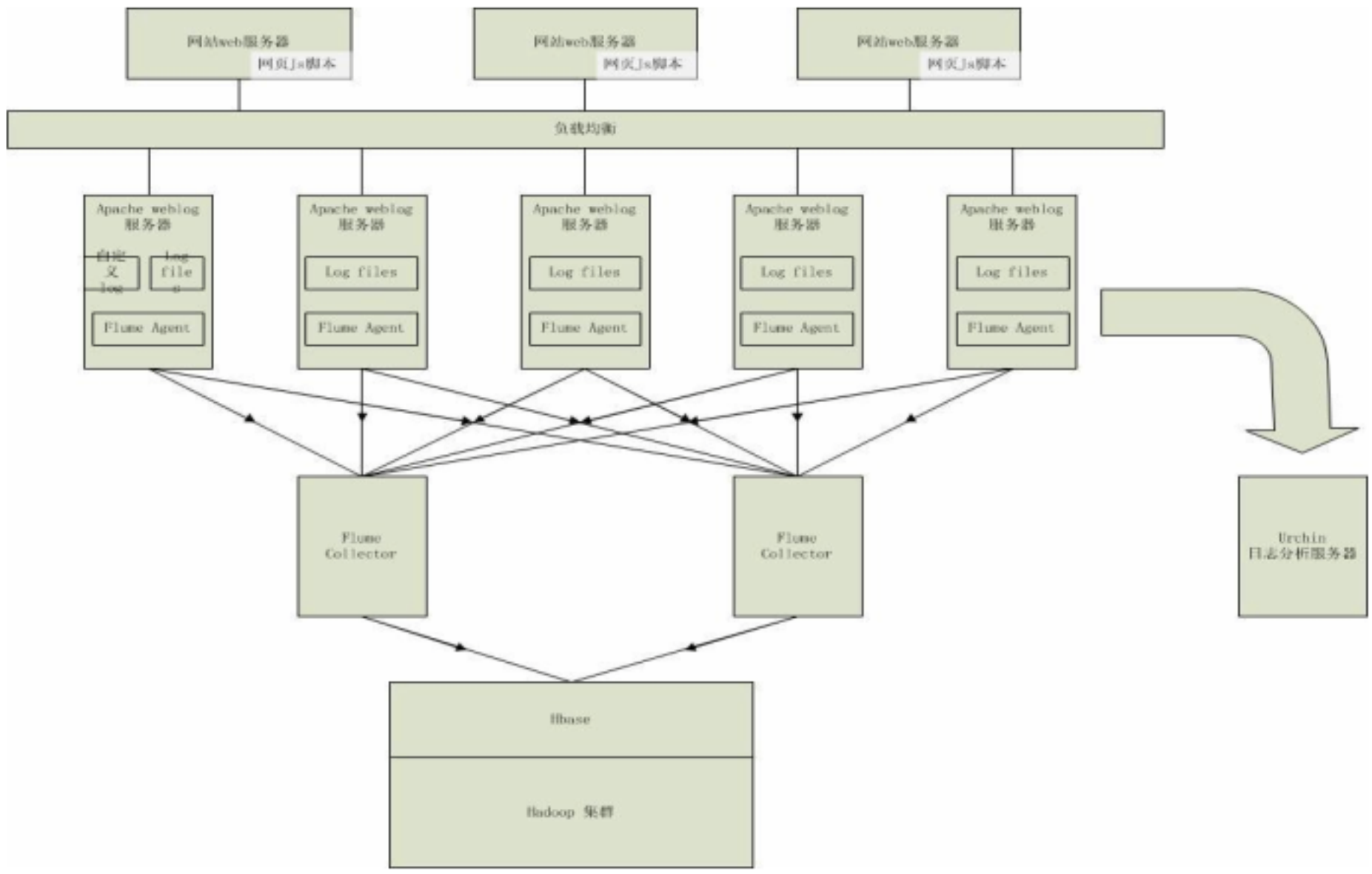
collector-master　　组合

agent-master　组合

Flume 插件开发

Flume 插件开发介绍

Flume 插件开发举例（　hbase）



Flume 集群遇到的问题

1，　Flume 在　agent　端采集数据的时候默认会在　　/tmp/flume-{user}　　下生成临时的目录用于存放 agent 自己截取的日志文件，如果文件过大导致磁盘写满那么　　agent　端会报出 Error closing logicalNode a2-18 sink: No space left on device　　，所以在配置　agent　端的时候需要注意

<property>
<name>flume.agent.logdir</name>
<value>/data/tmp/flume-${user.name}/agent</value>
</property>

属性，只要保证 flume 在 7*24 小时运行过程 agent 端不会使该路径 flume.agent.logdir 磁盘写满即可。

2, Flume 在启动时候会去寻找 hadoop-core-*.jar 的文件，需要修改标准版的 hadoop 核心 jar 包的名字 将 hadoop-*-core.jar 改成 hadoop-core-*.jar 。

3, Flume 集群中的 flume 必须版本一致。否则会出现莫名其妙的错误。

4, Flume 集群收集的日志发送到 hdfs 上建立文件夹的时间依据是根据 event 的时间，在源代码上是 Clock.unixTime() ，所以如果想要根据日志生成的时间来生成文件的话，需要对 com.cloudera.flume.core.EventImpl 类的构造函数

public EventImpl(byte[] s, long timestamp, Priority pri, long nanoTime,

String host, Map<String, byte[]> fields) 重新写，解析数组 s 的内容取出时间，赋给 timestamp 。

注意： flume 的框架会构造 s 内容是空的数组，用来发送类似简单验证的 event ，所以需要注意 s 内容为空的时候 timestamp 的问题。

5, Flume 在读取 utf-8 格式的文件时会出现解析不了时间戳，因为 utf-8 的文件格式在文件头会加上 utf-8 的文件标识，解决办法：

flume-core-0.9.4-modify.jar 包 修改了 EventImpl 类

使文件创建的时间是根据日志中的时间来确定的 ，修改的代码部分在 FlumeSource 中

修改的部分是在构造函数上， 支持 utf-8 的文件格式， 也支持其他普通的文件格式的文件

主要代码如下：

```
System.arraycopy(s, 3, tmpByte, 0, 13);
    tmpStr = new String(tmpByte);
    m = p.matcher(tmpStr);
    if(m.matches())//          表示符合 utf-8 的文件格式
    this.timestamp = Long.parseLong(tmpStr);
    else
    this.timestamp                                  =Long.parseLong(new String(s).split("\t")[0].toString());
```

性能没有测试。

6, 与 5 不同的是时间戳不是 long 的，而是字符串如：“ 2011 -10- 21 10:12:65.125 ”，修改办法如下： System.arraycopy(s, 3, tmpByte, 0, 25);

```
t mpStr = new String(tmpByte);
t ry {
        //this.timestamp=sf.parse(tmpStr.split("\t")[0].toString()).getTime();

this.timestamp=sf.parse(tmpStr.substring(0,tmpStr.indexOf("\t"))).getTime();
    } catch (ParseException e) {
        e.printStackTrace();
    }
```

7, 如果 collector 和 agent 不在一个网段的话会发生闪断的现象，这样的话，就会造成 agent 端不能传送数据个 collector 所以，在部署 agent 和 collector 最好在一个网段。

8, 如果在启动 master 时出现：“ 试着启动 hostname ，但是 hostname 不在 master 列表里的错误 “ ，这是需要检查是否主机地址和 hostname 配置的正确与否。