搜狐 SOHU.com

实时日志处理Flume和kafka技术分享

邵位

2016.5

总纲

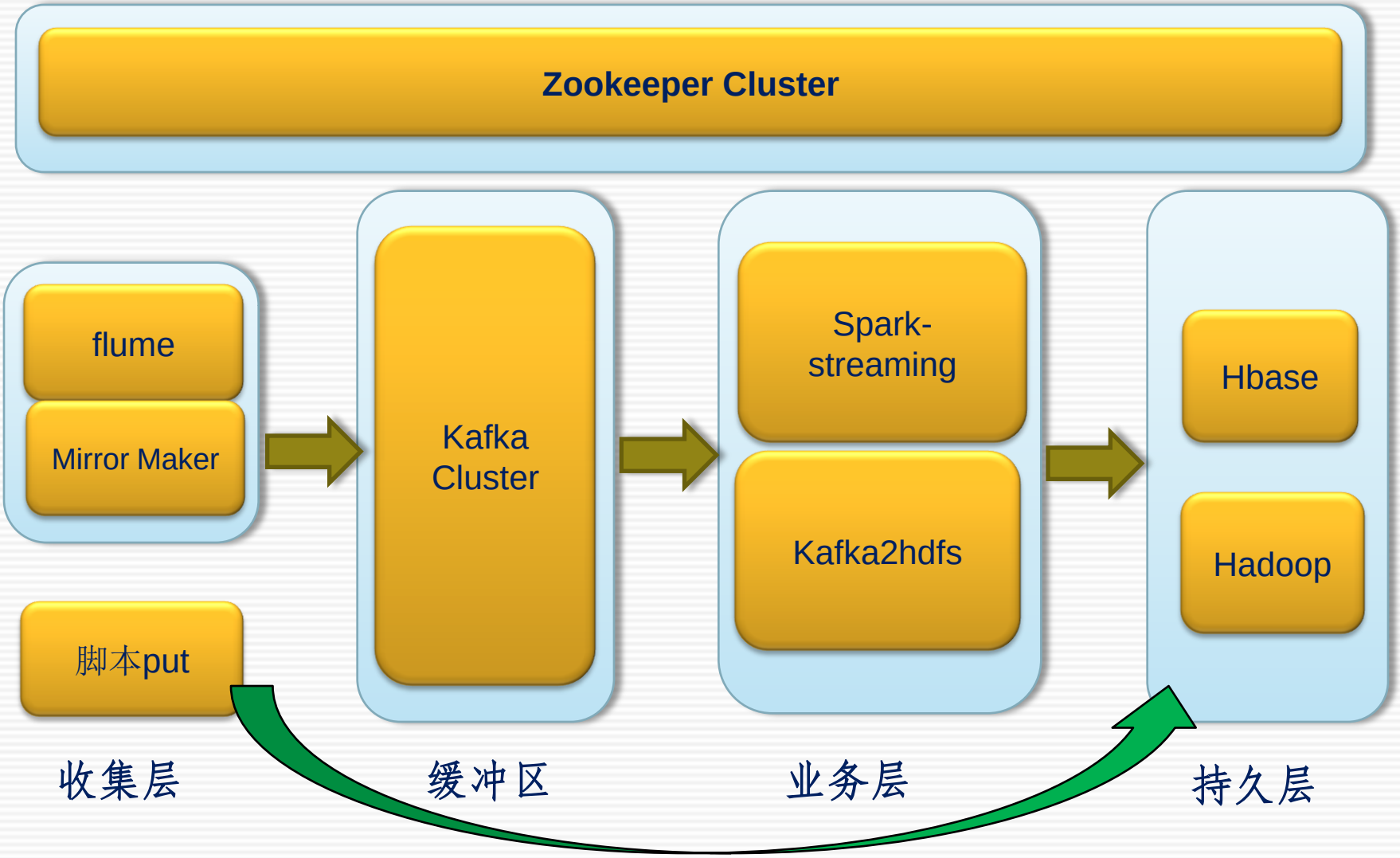
1. Flume 框架、应用以及优化策略

2. kafka框架介绍和应用

Flume目录

应用现状 – 部门架构流程图

应用现状 - 应用流程图


Zookeeper Cluster
flume
Mirror Maker
脚本put
Kafka Cluster
Spark-streaming
Kafka2hdfs
Hbase
Hadoop
收集层
缓冲区
业务层
持久层

Flume简介

flume组件

Agent

01

Source

数据收集

02

Channel

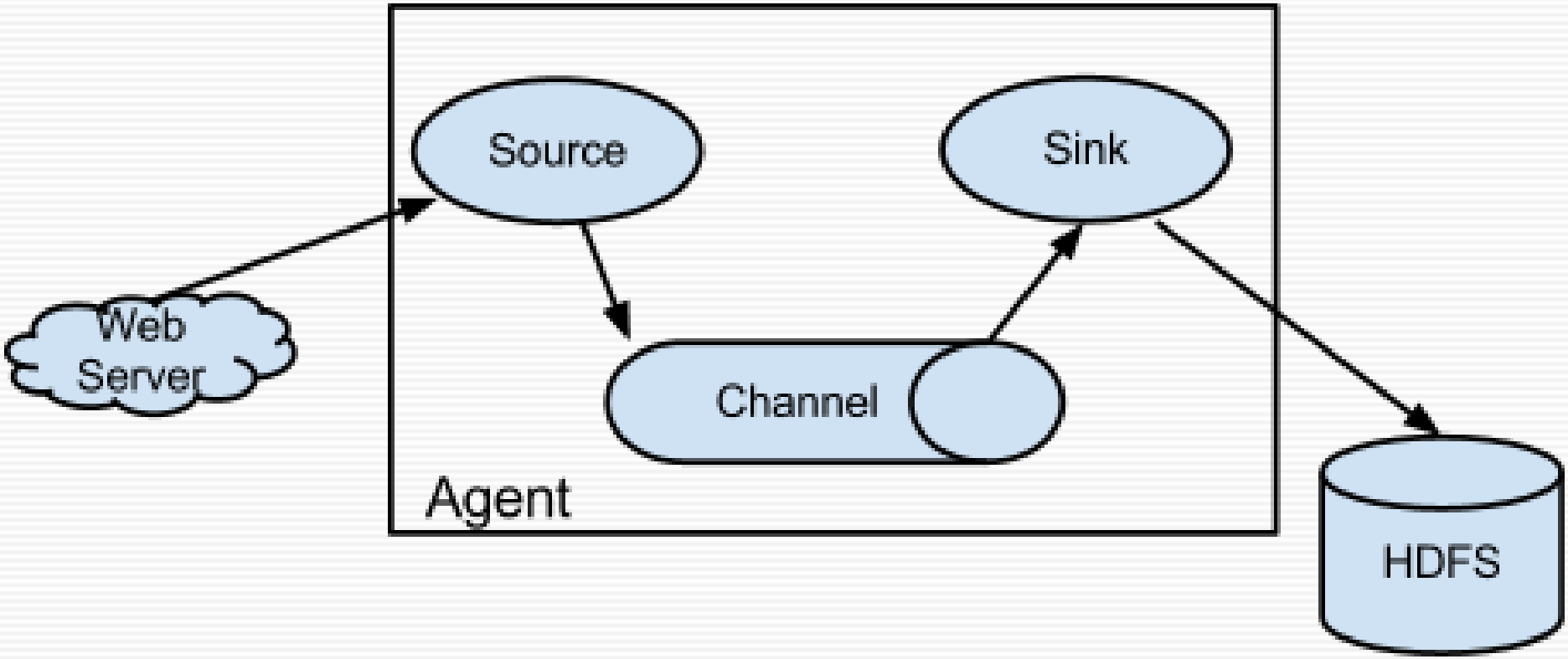
提供队列
缓存

03

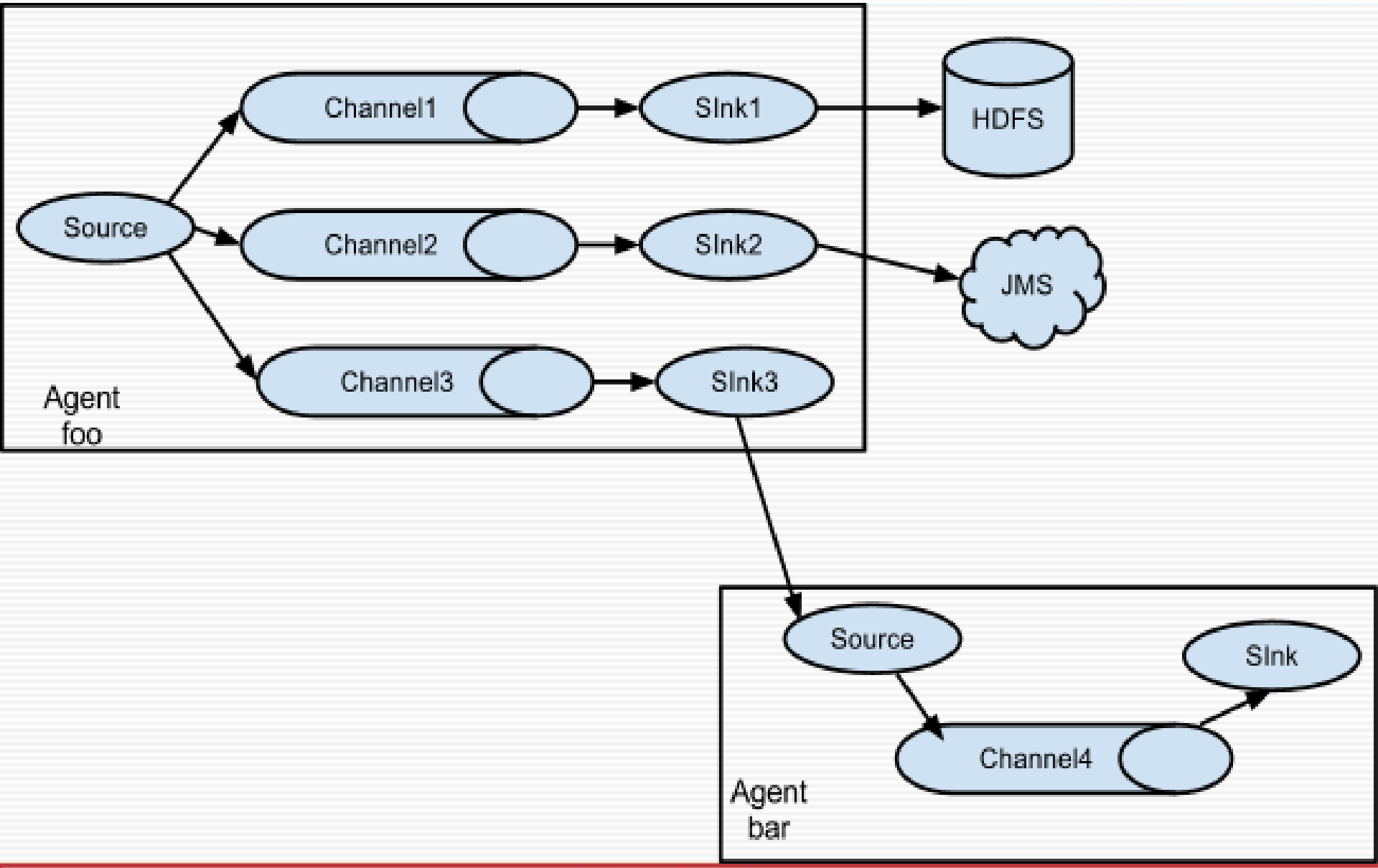
Sink

取数
kafkasink
hdfssink

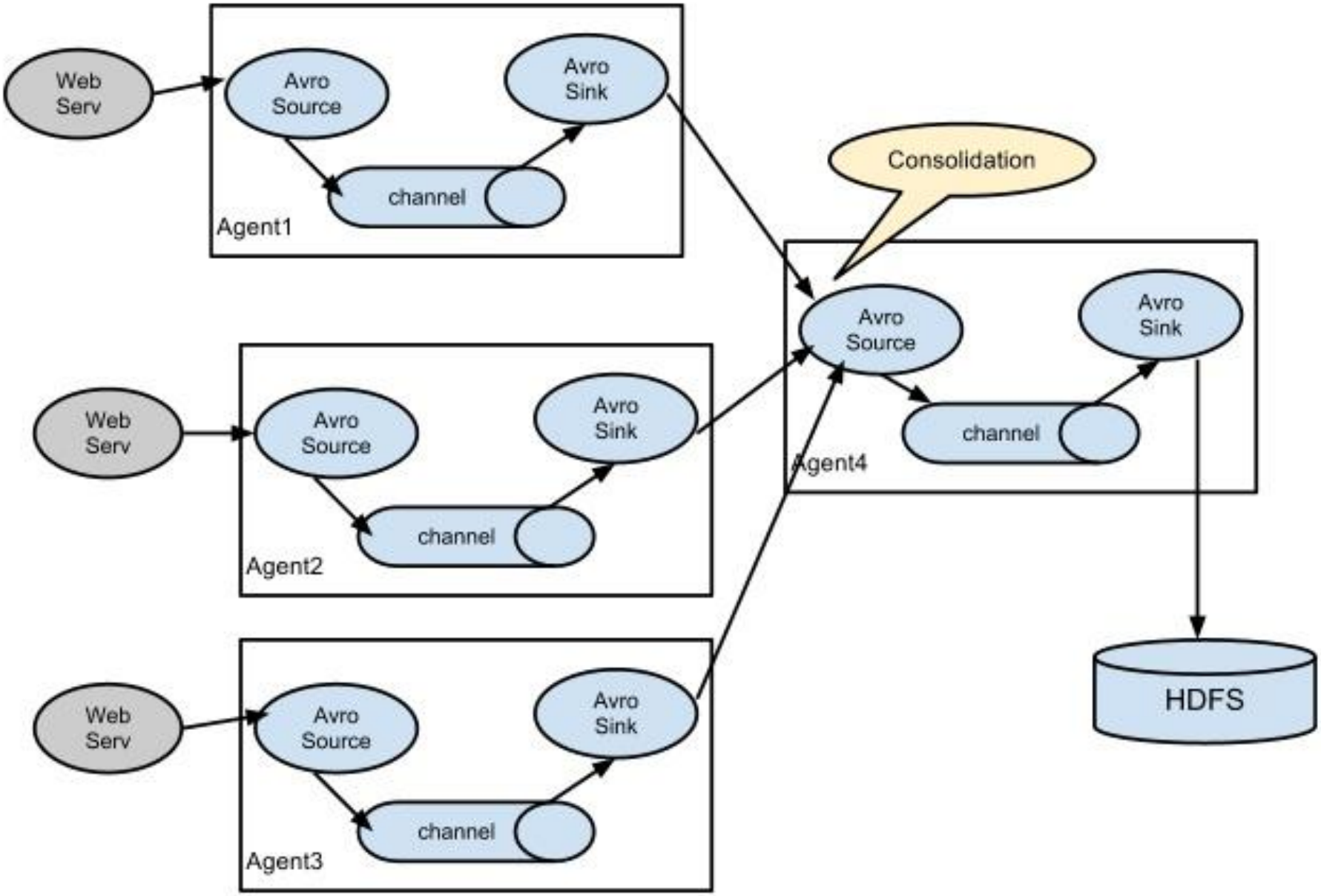
Flume架构



Flume架构


Channel1
Sink1
HDFS
Source
Channel2
Sink2
JMS
Channel3
Sink3
Agent
foo
Source
Sink
Channel4
Agent
bar

Flume架构


Web Serv
Avro Source
Avro Sink
channel
Agent1
Consolidation
Avro Source
Avro Sink
channel
Agent4
Web Serv
Avro Source
Avro Sink
channel
Agent2
Web Serv
Avro Source
Avro Sink
channel
Agent3
HDFS

Flume优点

- 高可靠
- 可扩展性
- 高效
- 可维护性

ExecSource:以运行Linux命令的方式，持续的输出最新的数据,实时传输，如tail -F 文件名指令

```
agent.sources.gather.type =
com.sohu.flume.source.ExecSource
agent.sources.gather.command = tail -F
/opt/countslog/countinfo.log
agent.sources.gather.channels = sendout
```

momerychannel

数据存储在内存队列中
速度最快

```
agent.channels.sendout.type = memory
agent.channels.sendout.capacity = 2000000
agent.channels.sendout.transactionCapacity = 10000
```

Channel Capacity

capacity：Channel可以容纳events数量的最大容量,就是queue的大小

transactionCapacity每次事务处理的events数量，就是putList和takeList的容量大小

kafkasink

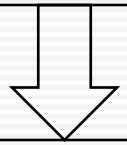
从channel拿数据，发送到kafka

```
agent.sinks.pool.type =
com.sohu.group.flume.sink.KafkaSink
agent.sinks.pool.channel = Mychannel
agent.sinks.pool.topic = countinfo
agent.sinks.pool.broker =
10.16.10.196:8092
agent.sinks.pool.zookeeper =
10.16.10.76:2181/kafka-0.8.1
agent.sinks.pool.num_partitions = 72
```

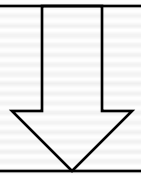
目录

源码的调用关系-put

ExecSource:
flushEventBatch(List<Event> eventList)

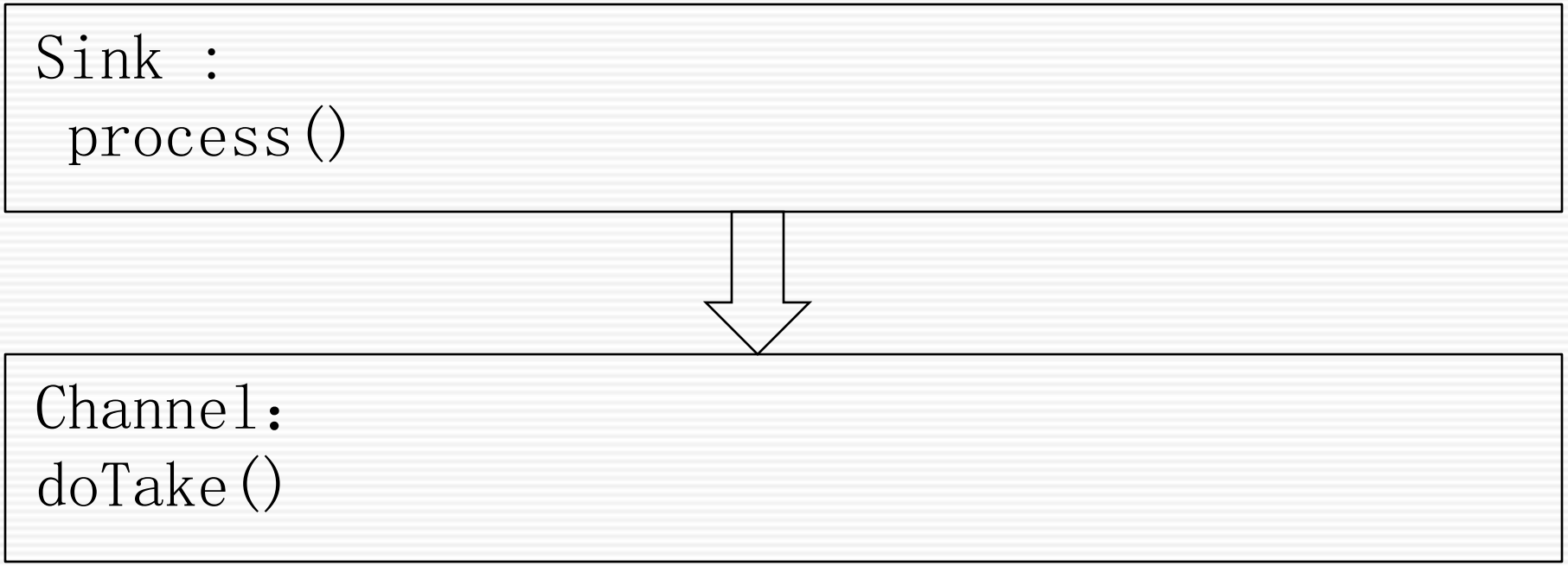


ChannelProcessor（事务）:
processEventBatch(List<Event> events)



Channel：
doPut(Event event)

源码的调用关系-take



MemoryChannel原理

doPut
把event写入数据结构putList

doTake
从queue得到event，放入takeList，返回event,然后发送到kafka
takeList的作用仅仅是commit失败时进行rollback!

Put事务提交

Event从putList放到queue中，
queue容量是capacity
然后清空putList

Take事务提交

清空takeList

Put事务回滚

清空putList，下次事务重新put

Take事务回滚

takeList回吐到queue

目录

KafkaChannel原理

doPut
把event写入List<byte[]>
serializedEvents

doTake
没有使用

KafkaChannel原理

doCommit

把List serializedEvents发送到kafka

发送成功才会事务Commit

doRollback

清空serializedEvents

下次事务再次发送

KafkaChannel优化

agent.channels.ctestfk.kafka.batch.num.messages = 10000
设置为跟transactionCapacity一致

日志每秒条数	kafka.batch.num.messages	延时	每发送20万条耗时
3万	10000	无延时	7.1秒
3万	2000	延时15分钟	12.1秒
3万	不设置(200)	延时1小时多	34-54秒

batchSize

默认是20

agent.sources.s.batchSize = 10000

agent.sinks.k.hdfs.batchSize = 10000

batchSize一定不能大于transactionCapacity

原生的是随机写入

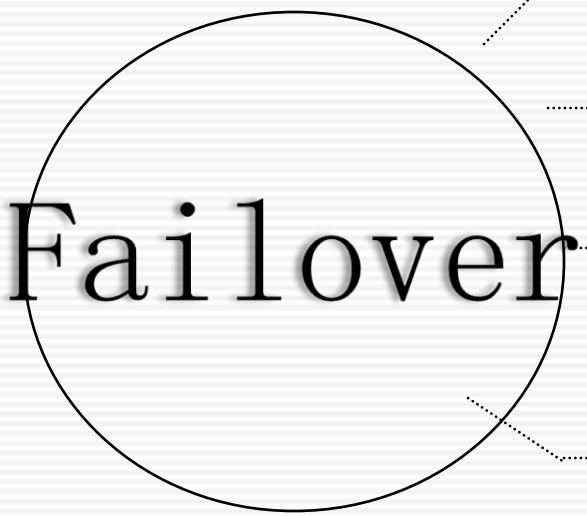
增加了负载均衡功能，轮询写入kafka

乱码问题

自带序列化方式avro, 有乱码问题

目前可通过配置序列化方式来解决乱码问题

Failover



- 以插件的形式开发
- 后台监控进程
- 自动进行channel切换
- 通过配置可启动或关闭failover
- 监控时间间隔可配置

KafkaChannel memoryChanel对比

日志每秒条数	channel	延时	每发送20万条耗时
10万	memorychannel	报错后无响应	8.6秒
10万	kafkachannel	9到25分钟	8.2秒
5万	memorychannel	报错后无响应	8.1秒
3万	memorychannel	无延时	7.1秒
3万	kafkachannel	无延时	7秒

目录

◆OOM 问题

Flume 启动时的最大堆内存大小默认是 20M，线上环境很容易 OOM，因此需要在 flume-env.sh 中添加 JVM 启动参数:

JAVA_OPTS=“-Xms8192m -Xmx8192m …”

◆数据重复写入

回滚后重复写入

可能的解决方法：

1. 关闭批量写入
2. 消费端进行去重

Flume缺点

◆数据丢失：

MomeryChannel
Flume进程挂掉

补救：
hdfs put来补数

预防：
监控：进程和日志监控

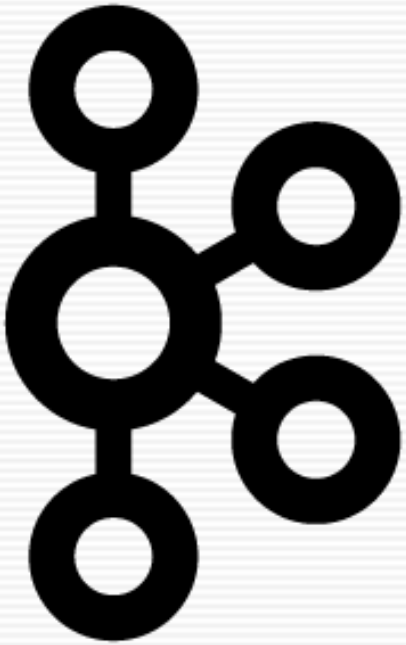
◆日志量比原来大

1. flume.log日志量约60MB/天

2. **监控进程调kafka api产生日志，** 可做日志滚动定期删除。

未来的努力方向

1. 高效稳定的Source
2. Channel参数调优，avro序列化
3. Kafkachannel+其他sink

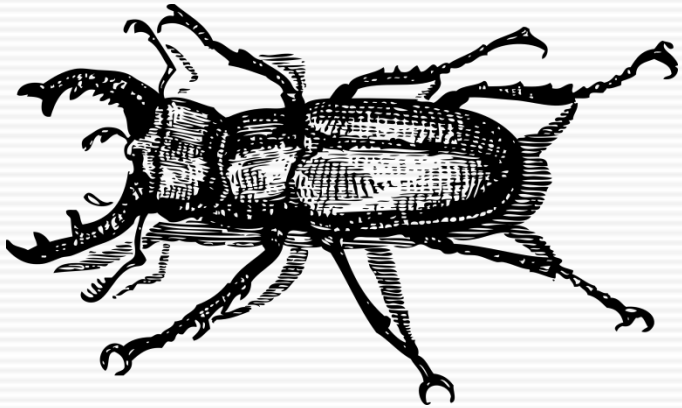


kafka

目录

Kafka 介绍

- LinkedIn开发
- 发布-订阅消息系统

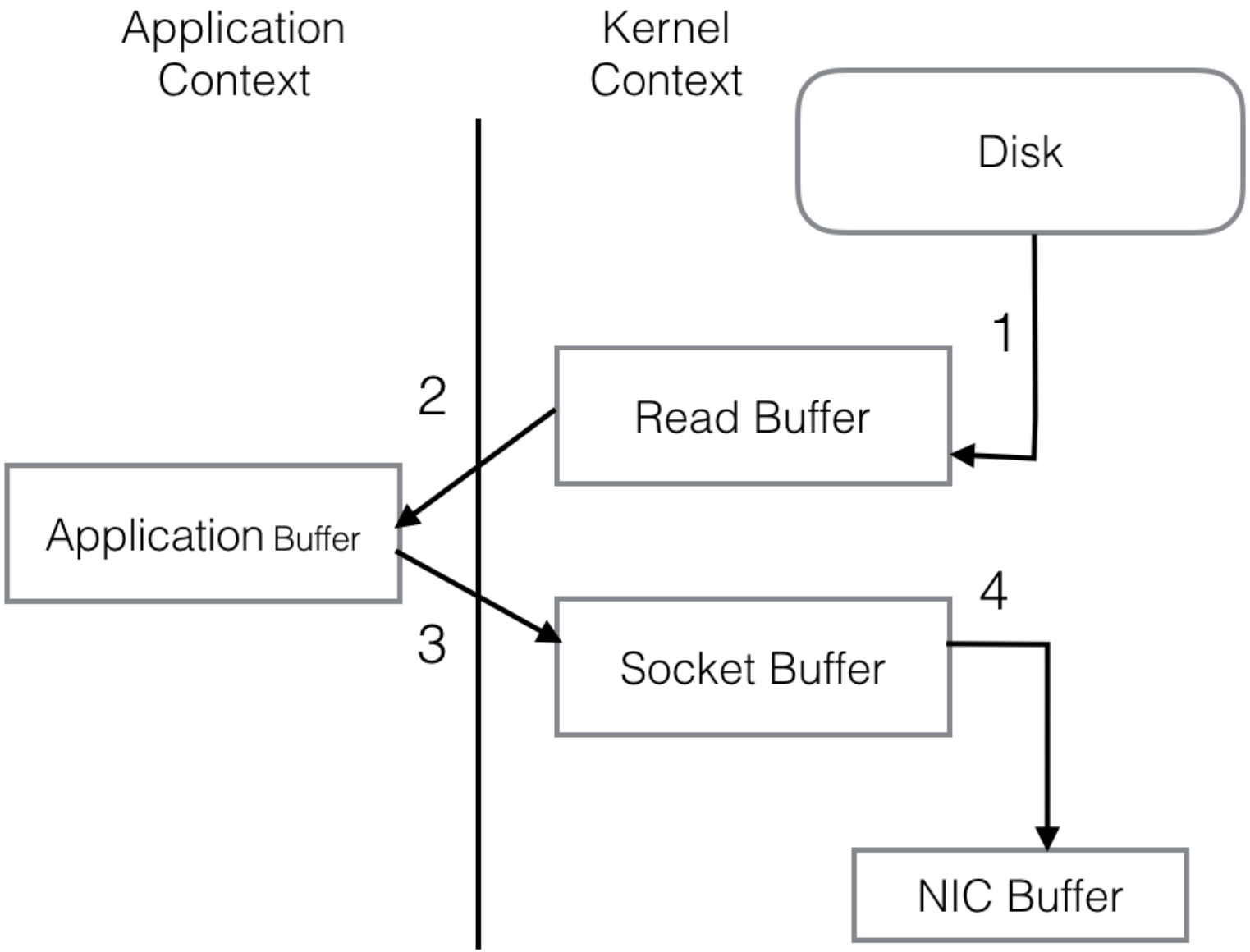


优势

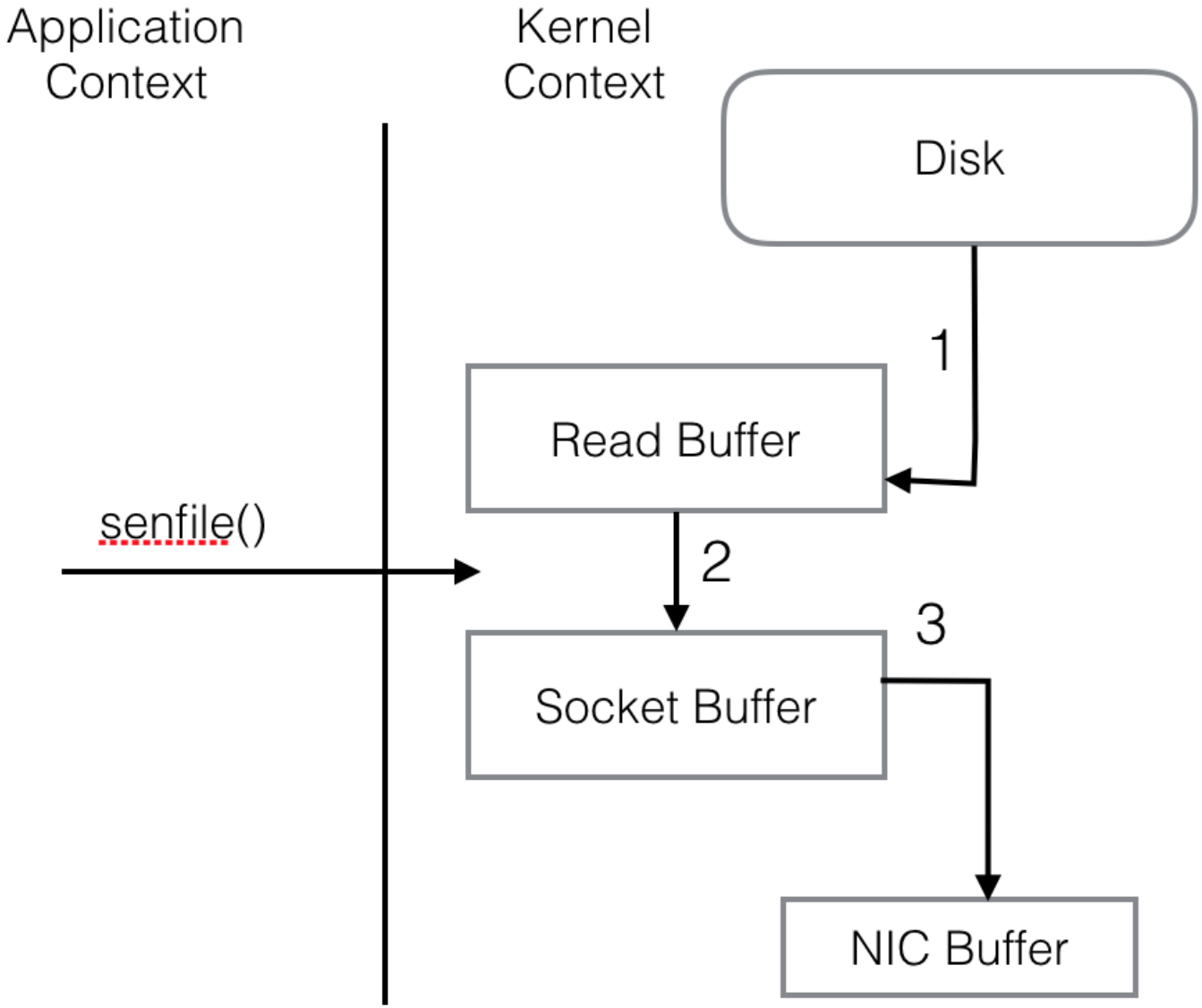
- 可扩展性
- 高吞吐
- 消息持久化
- 分布式

目录

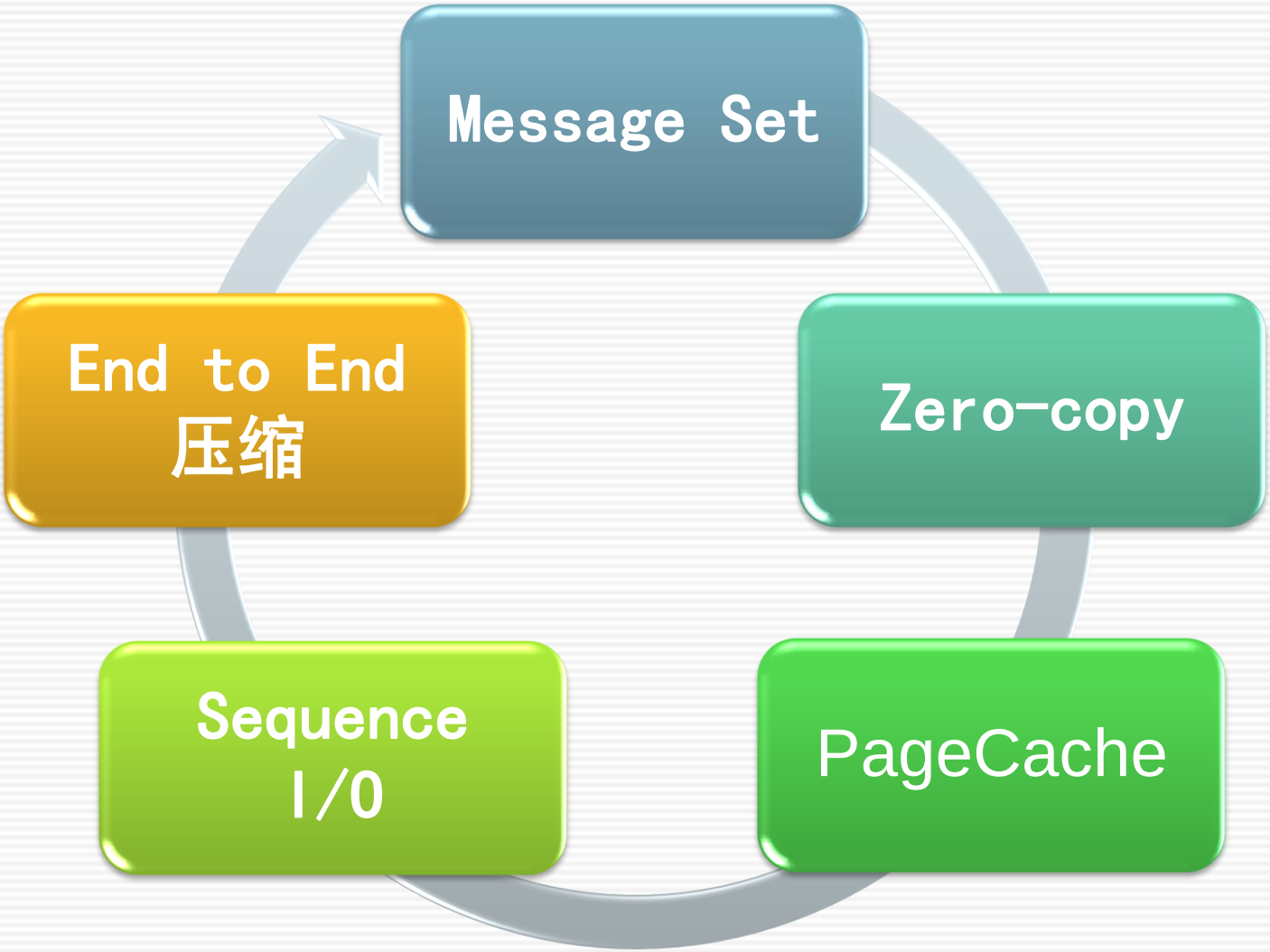
高吞吐量揭秘


Application Context
Kernel Context
Disk
1
Read Buffer
2
Application Buffer
3
Socket Buffer
4
NIC Buffer

ZERO-COPY

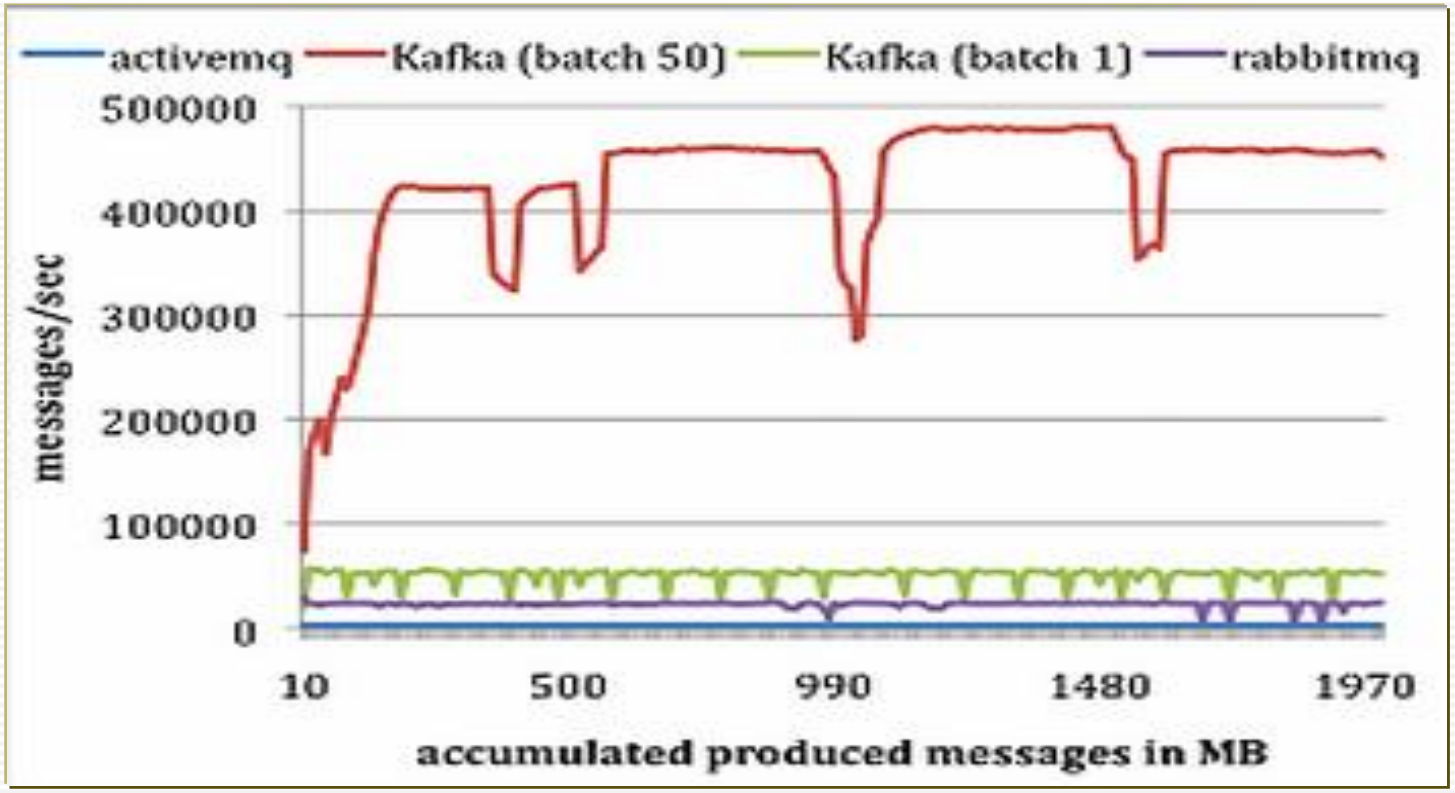

Application Context
Kernel Context
Disk
1
Read Buffer
senfile()
2
3
Socket Buffer
NIC Buffer

高性能实现

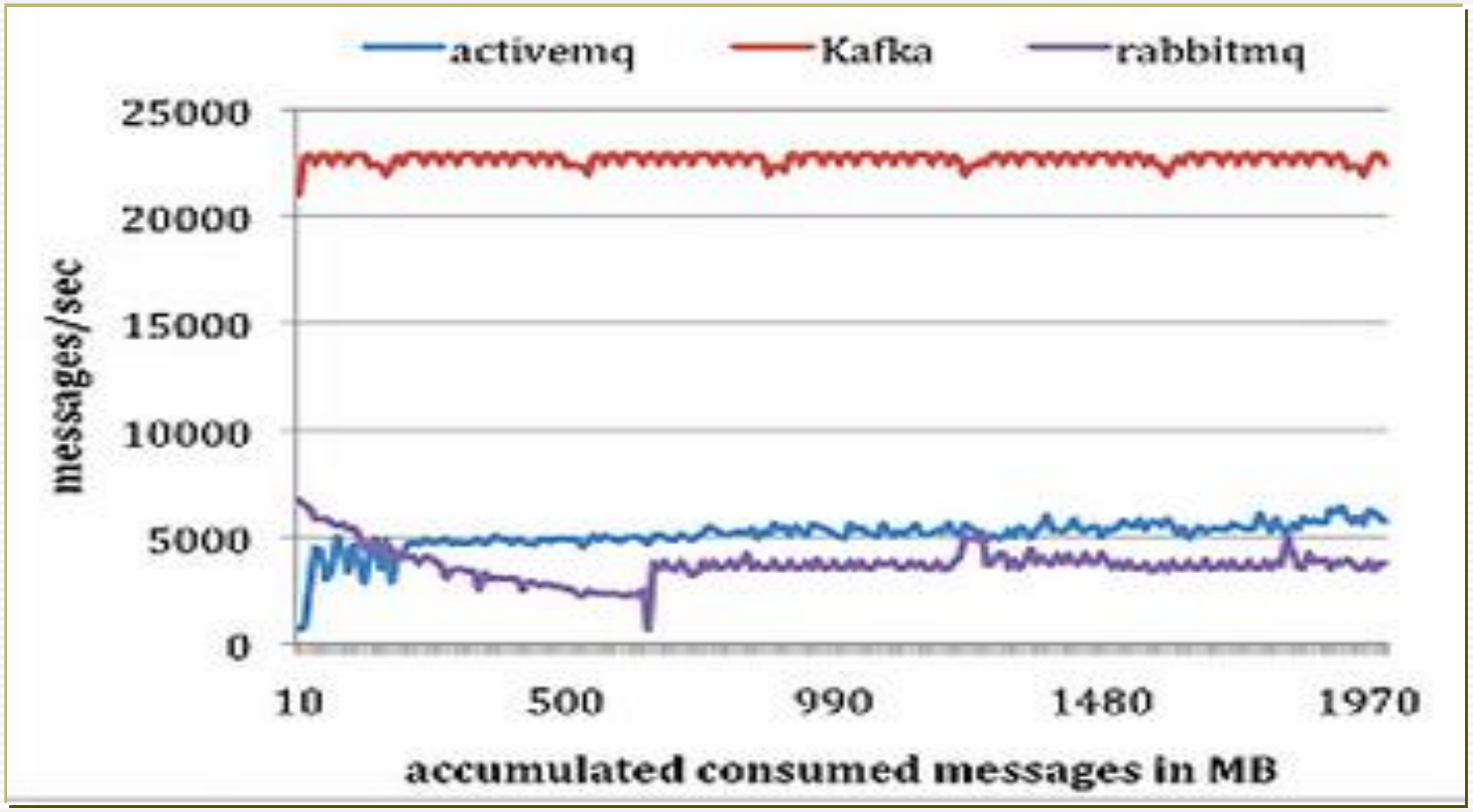


应用场景

性能-示例


activemq
Kafka (batch 50)
Kafka (batch 1)
rabbitmq
500000
400000
300000
200000
100000
0
messages/sec
10
500
990
1480
1970
accumulated produced messages in MB

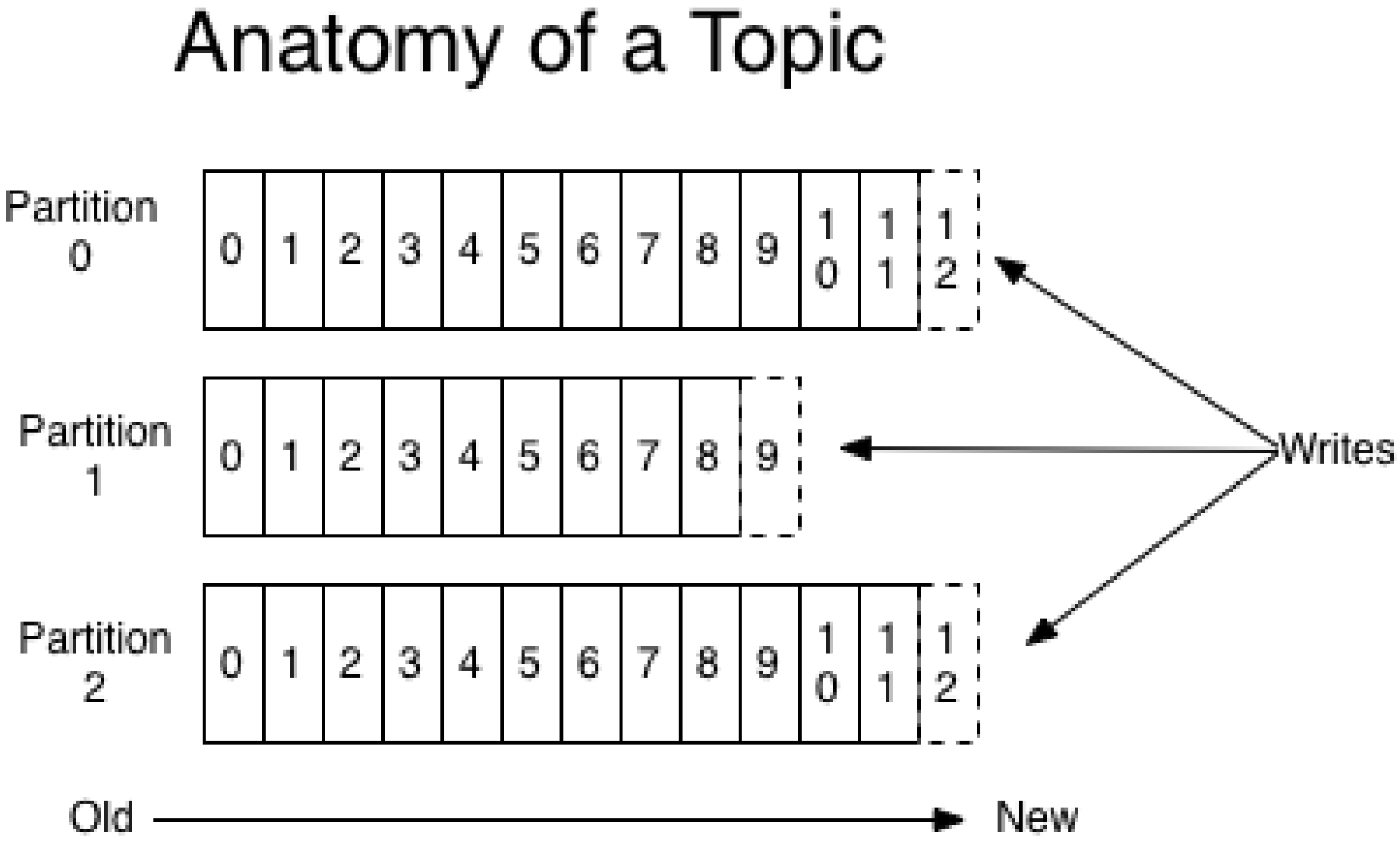
性能-示例


activemq
Kafka
rabbitmq
25000
20000
15000
10000
5000
0
messages/sec
10
500
990
1480
1970
accumulated consumed messages in MB

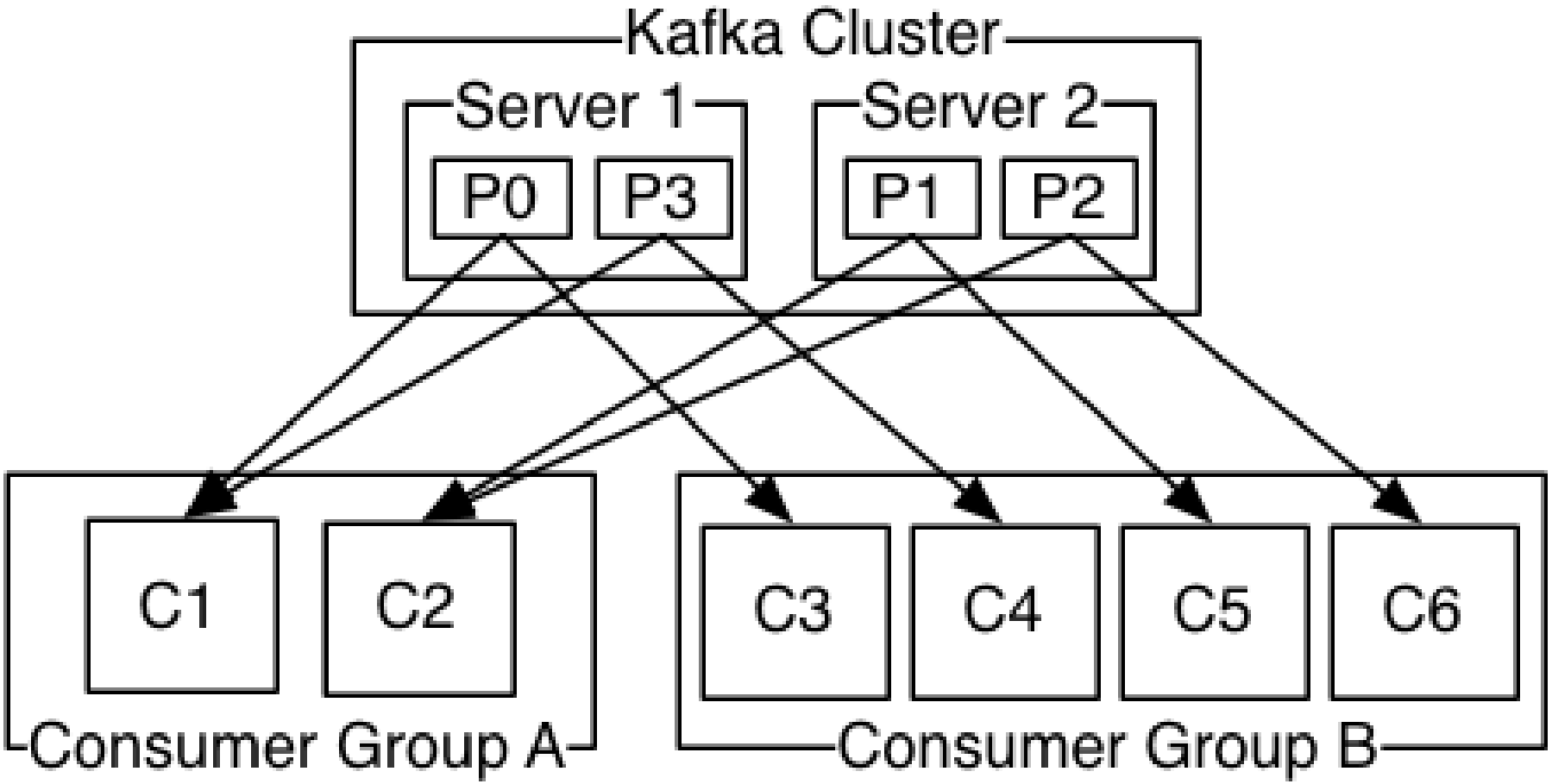
基本概念

- **Broker** Kafka集群中的一个节点服务
- **Topic** 消息源的分类
- **Partition** 基于topic的分组。实现topic负载均衡的方法之一
- **Offset** 可以简单理解为消息在文件中的位置或者累计的量
- **Message** 通信的基本单位

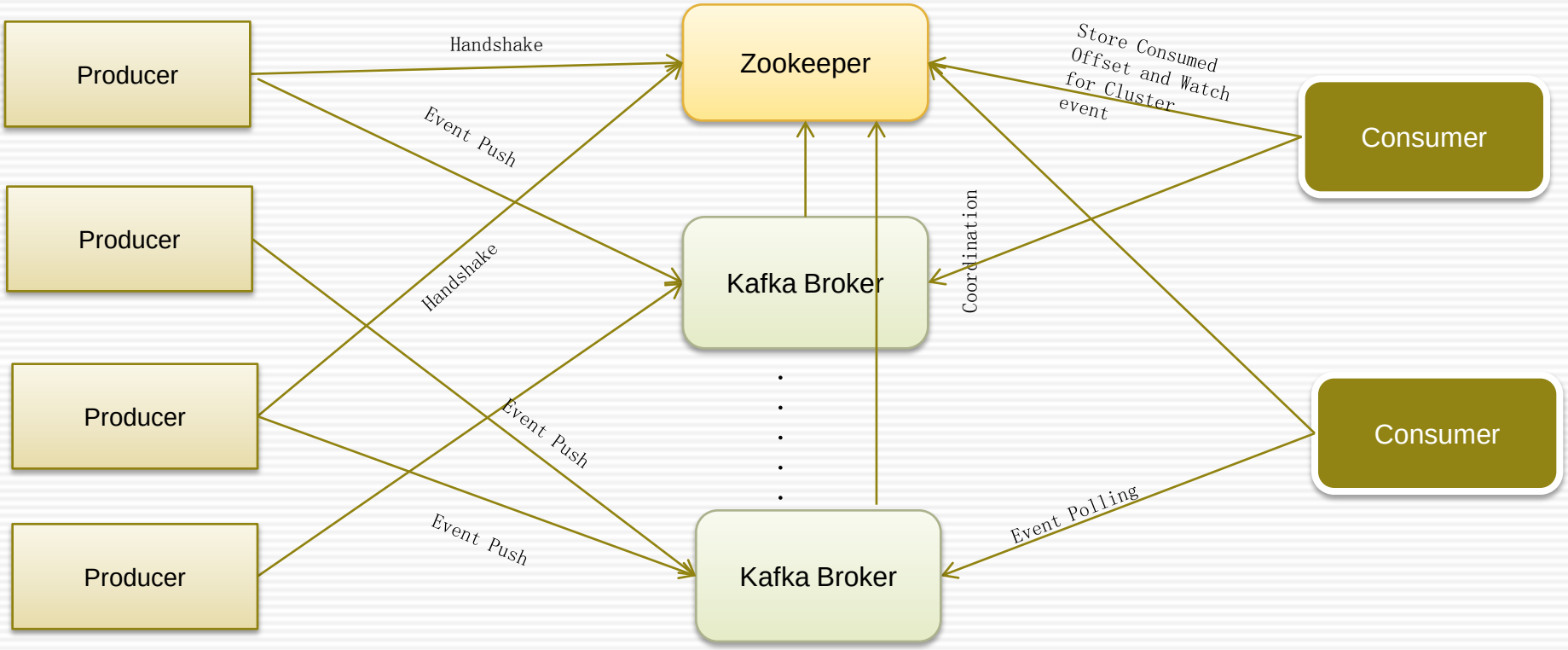
架构流程图


Anatomy of a Topic
Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 1
0 1 2 3 4 5 6 7 8 9
Partition 2
0 1 2 3 4 5 6 7 8 9 10 11 12
Writes
Old
New

架构流程图


Kafka Cluster
Server 1
P0
P3
Server 2
P1
P2
C1
C2
Consumer Group A
C3
C4
C5
C6
Consumer Group B

架构流程图


Producer
Producer
Producer
Producer
Handshake
Event Push
Handshake
Event Push
Event Push
Zookeeper
Kafka Broker
Kafka Broker
Coordination
Store Consumed Offset and Watch for Cluster event
Event Polling
Consumer
Consumer

可靠性

- 基于Partition主从设计
- 1 Leader;(n-1) follower
- 主分区可读写、从分区备份

Consumer API方式

Higher Level

- 自动维护**offset**
- 开发简单,省心

Lower Level

- 消费者单独维护**offset**
- 扩展灵活，开发复杂

目录

应用现状 - kafka集群

- Kafka 18台+6台
- Zookeeper 7台

集群规模

- 12 核、24G、15T、千兆网卡
- kafka为0.8.1版本
- Zk为3.4.5

集群配置

- Available Topic **46**
- Total Topic **81**个
- 入口带宽 30M左右
- 出口带宽 80M左右
- 磁盘空间使用 7T
- 备份3天，副本3

现状

- kafka2hdfs 44个
- spark-streaming 8个
- 其他部门

消费者

存在的问题

1. Partition分配算法简单
2. Topic无法删除
3. 未提供安全机制
4. 没有可视化工具

应用现状 - 监控方式

脚本监控
- 基于zabbix监控

可视化监控工具
- kafka-web-console

应用现状 - ZABBIX

ZABBIX

Help | Get support | Print | Login

Monitoring | Inventory | Reports

10.16.34.54

Dashboard | Overview | Web | Latest data | Triggers |
Events | Graphs | Screens | Maps | IT services

Search

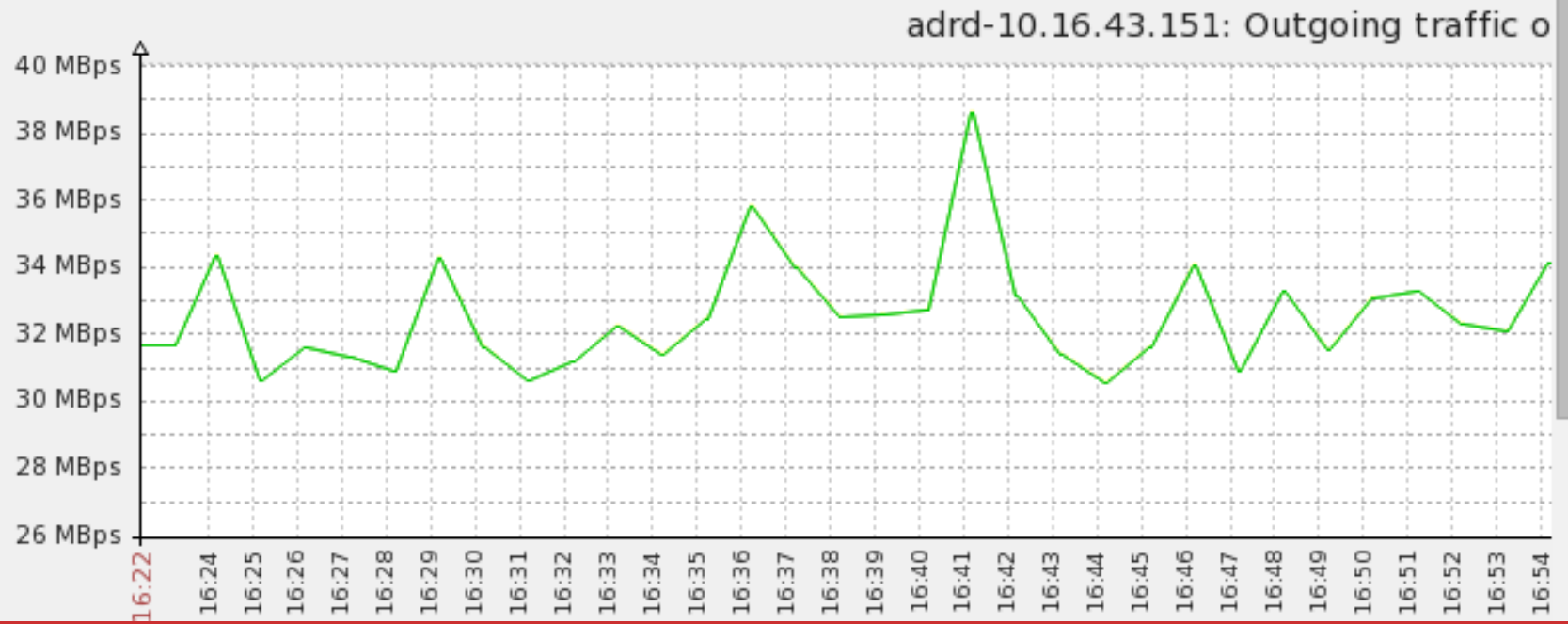
adrd-10.16.43.151: Outgoing traffic on interface eth2

Graph

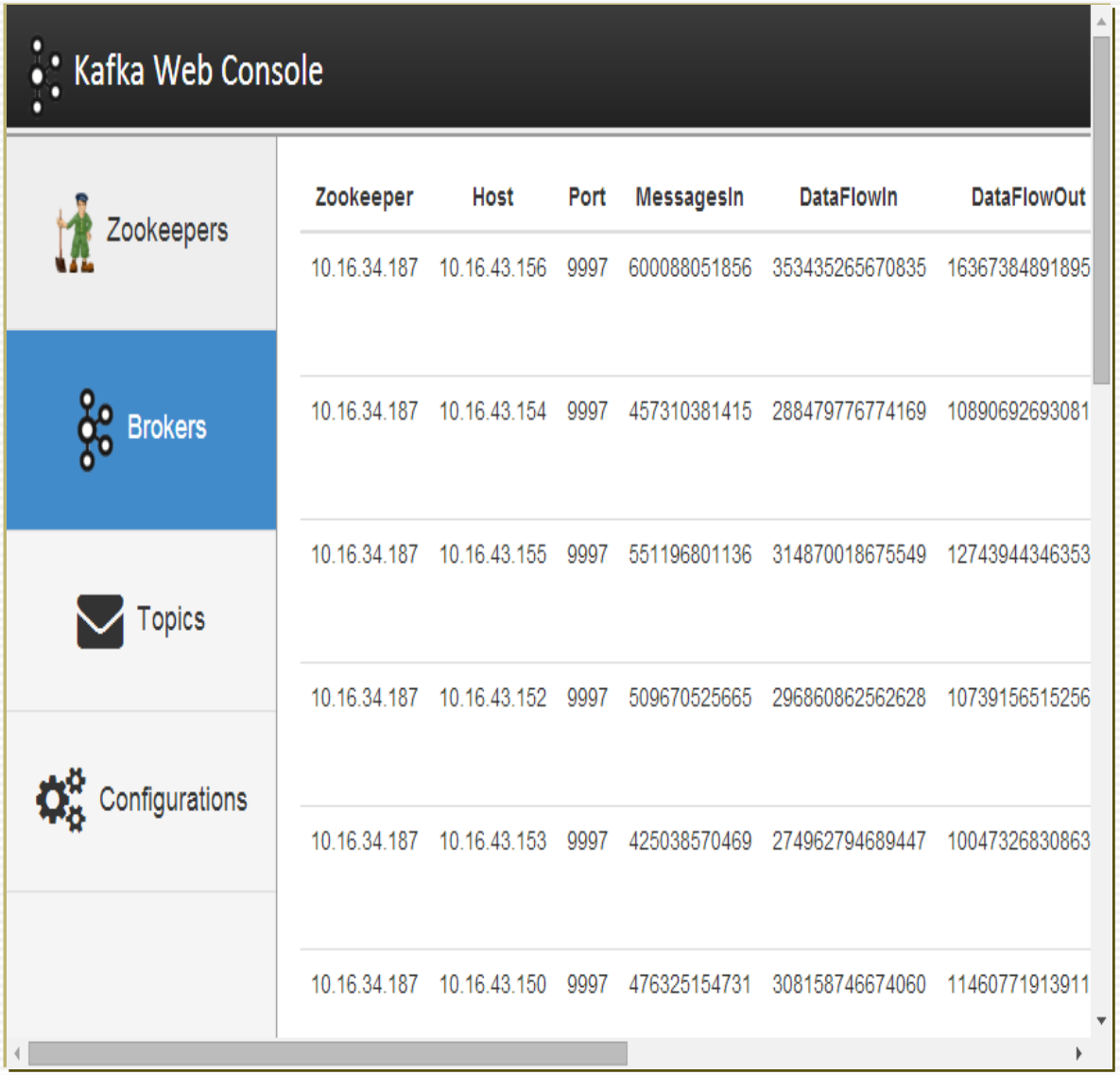
Hide filter

Zoom: 1h 2h 3h 6h 12h 1d 7d 14d 1m 3m 6m All

«« 6m 1m 7d 1d 12h 1h | 1h 12h 1d 7d 1m 6m »»



应用现状 - 可视化监控工具



- **技术方案**

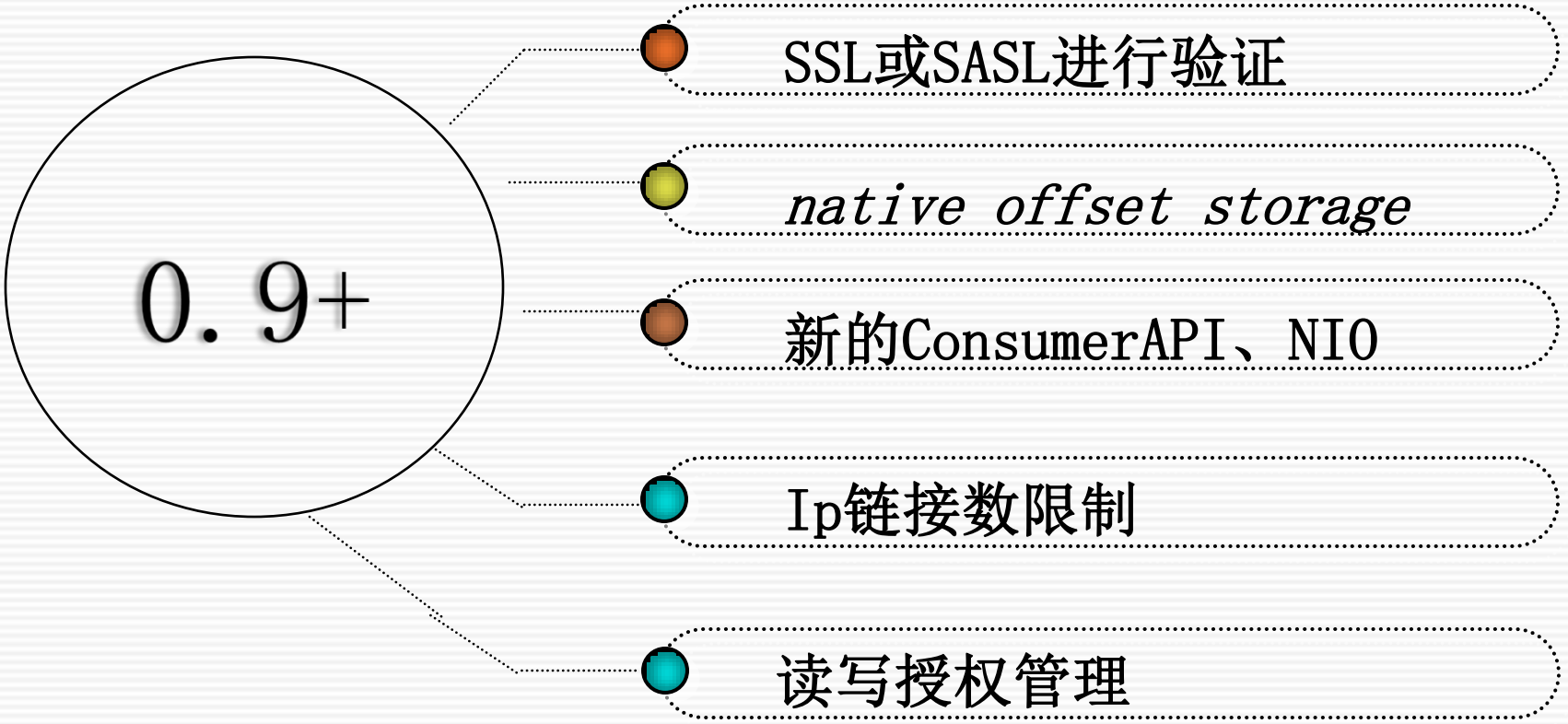
Scala play、zookeeper、jmx、sbt

- **功能**

Brokers、Topic的监控
监控项配置
告警等功能

目录

其他 - 新版本features



搜狐
SOHU.com

Thank You !