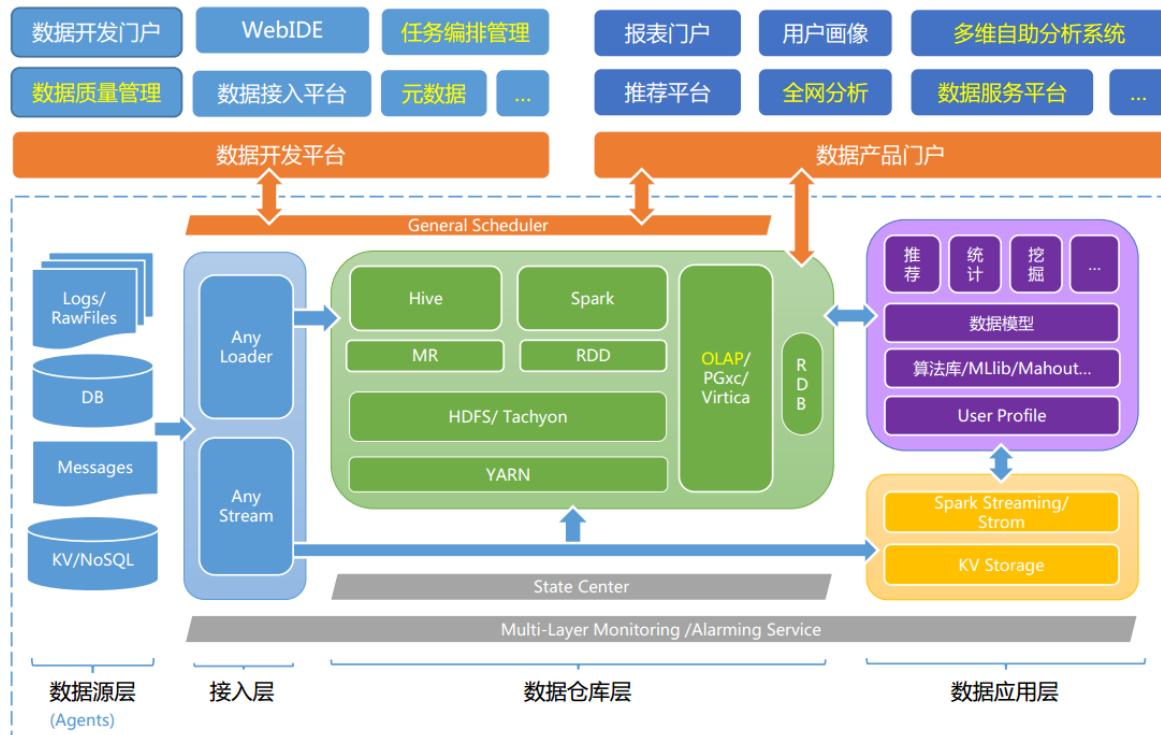


魅族流平台的实践

魅族--沈辉煌

魅族大数据平台的架构



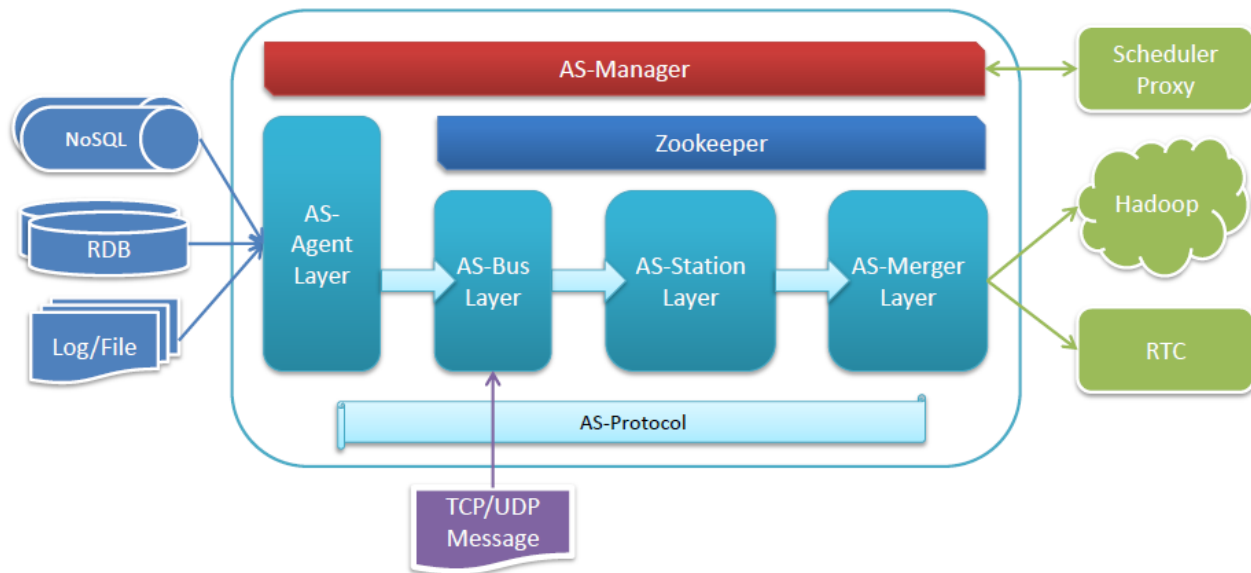
大纲

- 流平台介绍
- 流平台设计
- 采集组件
- 流管理平台
- 数据中转
- 实时计算
- Q&A

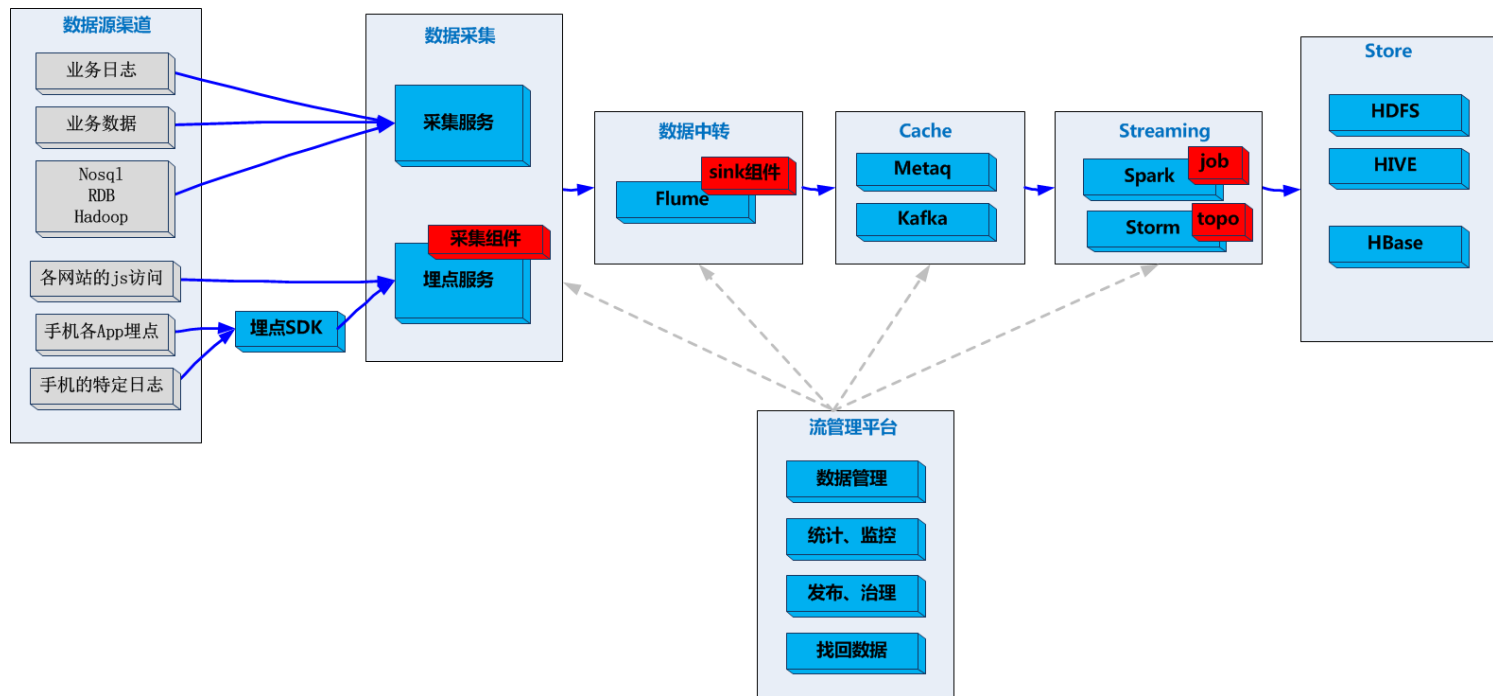
一、流平台介绍

- 数据采集
- 数据处理
- 数据存储
- 计算能力（实时计算、离线计算）

流平台架构图



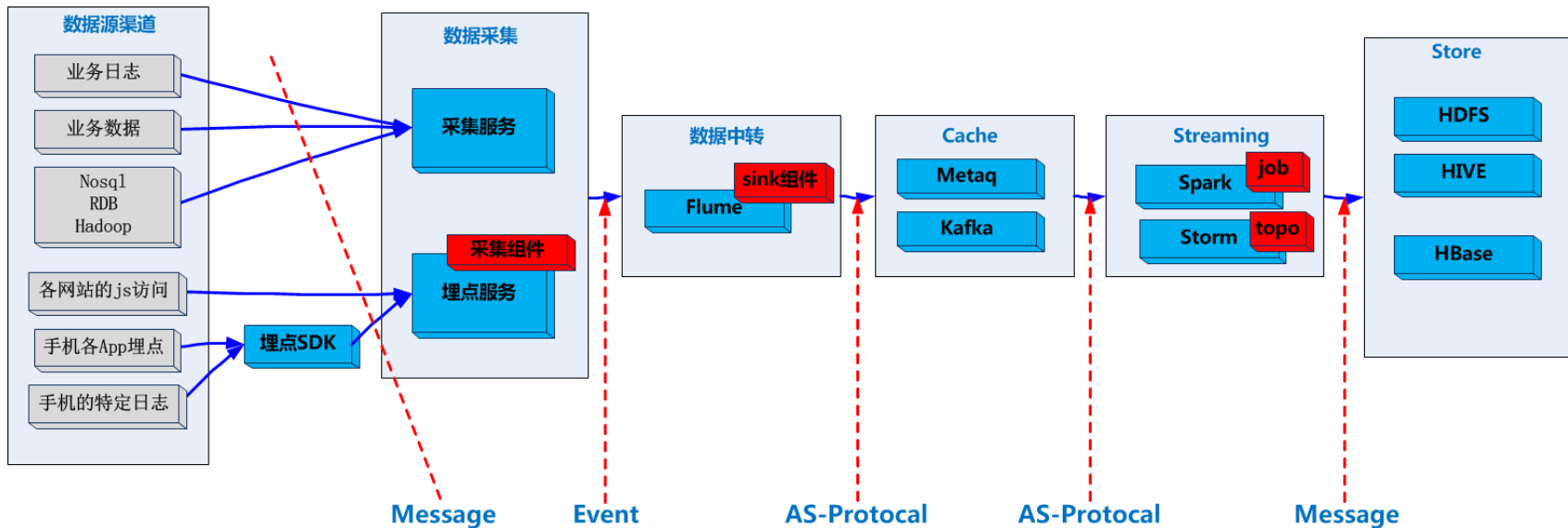
流平台架构图 (cont.)



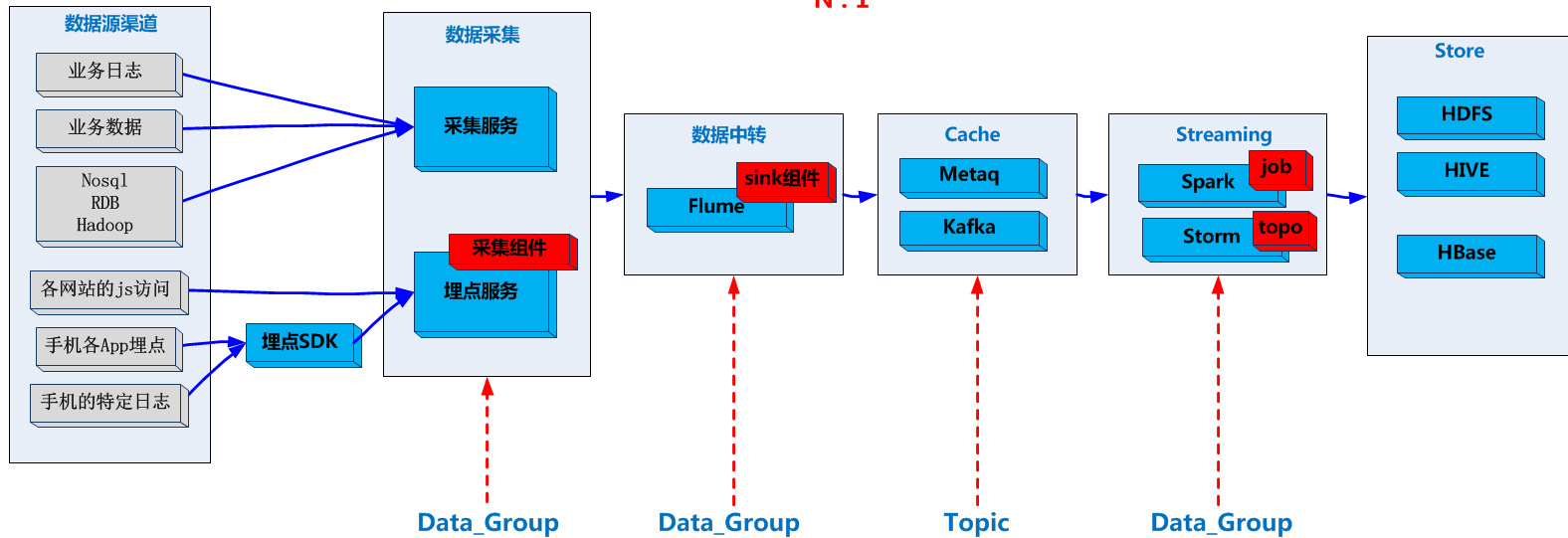
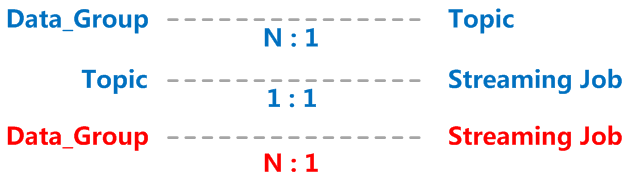
二、流平台设计

- Message
- AS-Protocol
- Event
- Type: 数据格式 (Json/Hive格式)
- Compress
- Data_timestamp
- Send_timestamp
- Unique Key
- Topic
- Data_Group
- Protobuf序列化

协议设计



数据分组设计



三、采集组件Agent

- Agent-Stub.jar

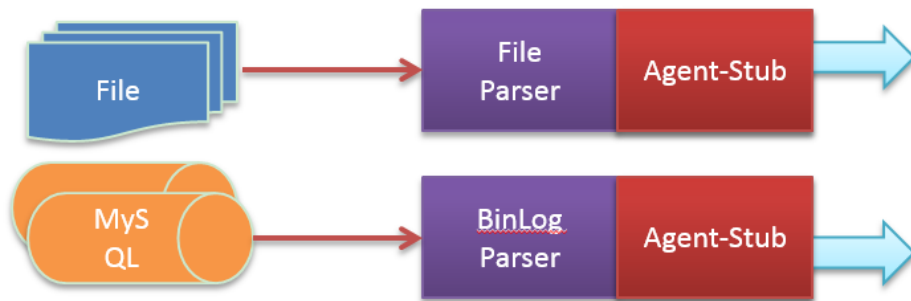


- Agent-File.zip

File-Parser

Binlog-Parser

接入Agent-Stub



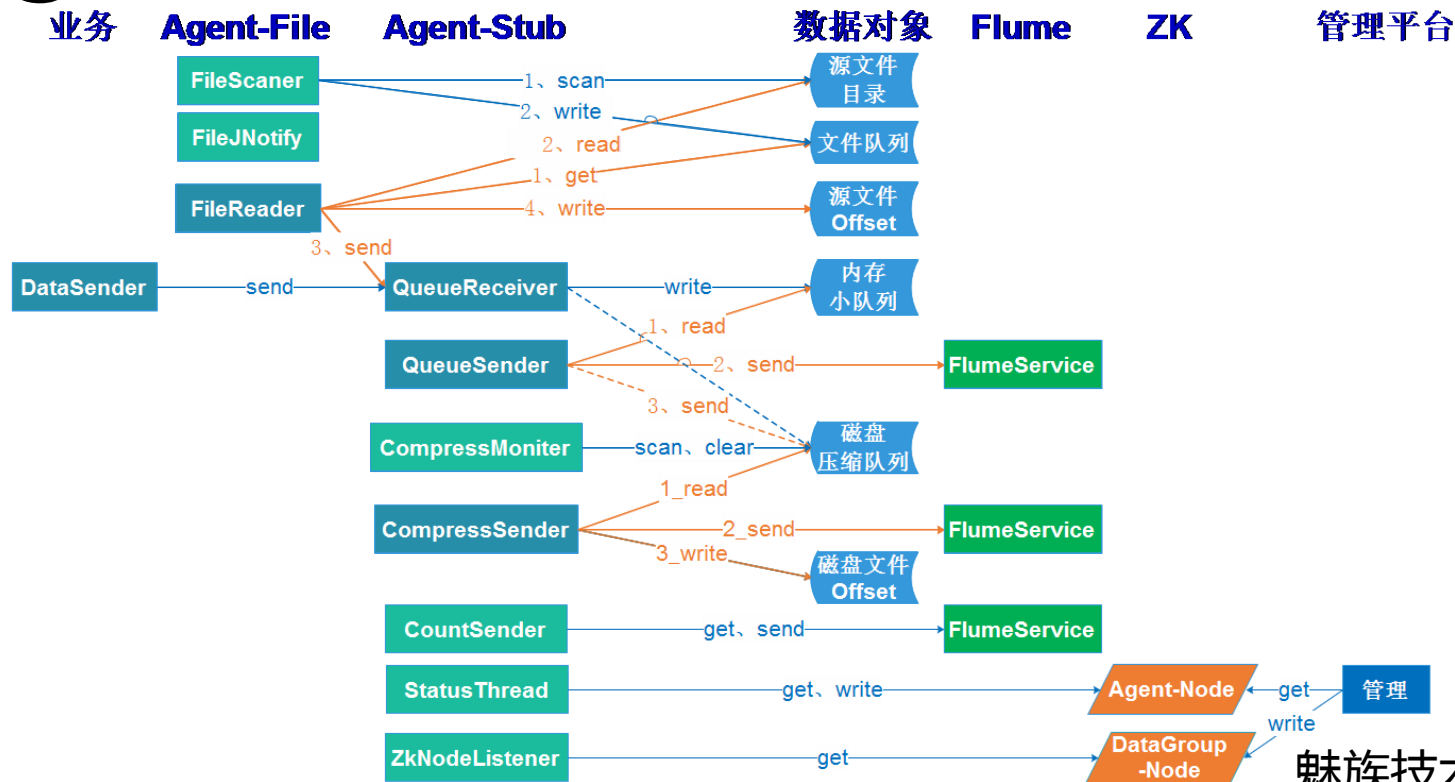
Agent-Stub.jar

- 多线程，异步
- 内存小队列+磁盘压缩队列
- 无损启停
- agent的版本号自动上报平台
- 自动识别接入源，智能归类
- agent的全面实时监控
 - （内存队列数、磁盘队列数、运行状态、出错状态、qps等）
- 支持实时命令
 - （包括限流，恢复限流、停止、调整心跳值等）
- 兼容Docker

Agent-File.zip

- 接入Agent-Stub
- 兼容Docker
- 支持重发历史数据
- 管理平台自助升级
- 文件名正则表达式匹配
- 源目录定时扫描 and Jnotify
- 单文件读取
- 文件方式存储offset，无损启停

Agent示意图



Agent的坑

- 丢数据
- 版本管理
- tailf -f的问题
- 网络原因导致zk删节点问题
- 乱码问题
- 日志问题

四、流管理平台

魅族流管理平台

接入管理 运行管理 系统监控 统一管理

权限申请 沈辉煌 退出

Topic 列表

Topic 标识 Topic 描述 查询

每页显示 10 条记录 [复制](#) [显示/隐藏列](#)

ID	Topic 标识	Topic 描述	Topic 类型	数据源数量	接口数量	目标表数量	操作
149	ANY_TOPIC_OPERATOR	用户操作	业务 Topic				数据源查看 接口配置 重调因子
148	ANY_TOPIC_REQ	请求日志	业务 Topic				数据源查看 接口配置 重调因子
146	ANY_TOPIC_COLLECTION	作业数据收集	业务 Topic				数据源查看 接口配置 重调因子
145	ANY_TOPIC_SEARCH	站内搜索	业务 Topic				数据源查看 接口配置 重调因子
144	ANY_TOPIC_STOCKMAN	ROCKMAN	业务 Topic				数据源查看 接口配置 重调因子
143	ANY_TOPIC_STREAMING_WX		业务 Topic				数据源查看 接口配置 重调因子
142	ANY_TOPIC_STREAMING_WX	直播	业务 Topic				数据源查看 接口配置 重调因子
141	ANY_TOPIC_SEARCH		业务 Topic				数据源查看 接口配置 重调因子
140	ANY_TOPIC_DEVICE_REPAIR		业务 Topic				数据源查看 接口配置 重调因子
139	ANY_TOPIC_POIC		业务 Topic				数据源查看 接口配置 重调因子

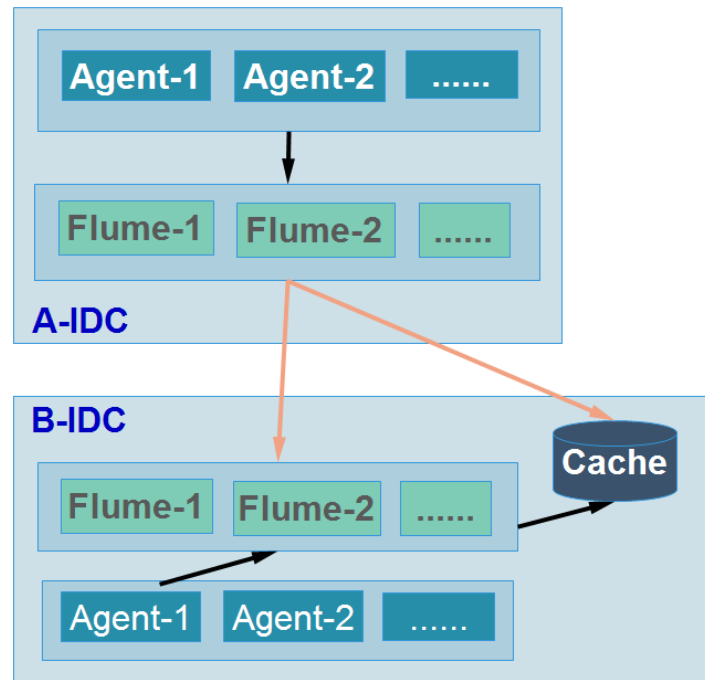
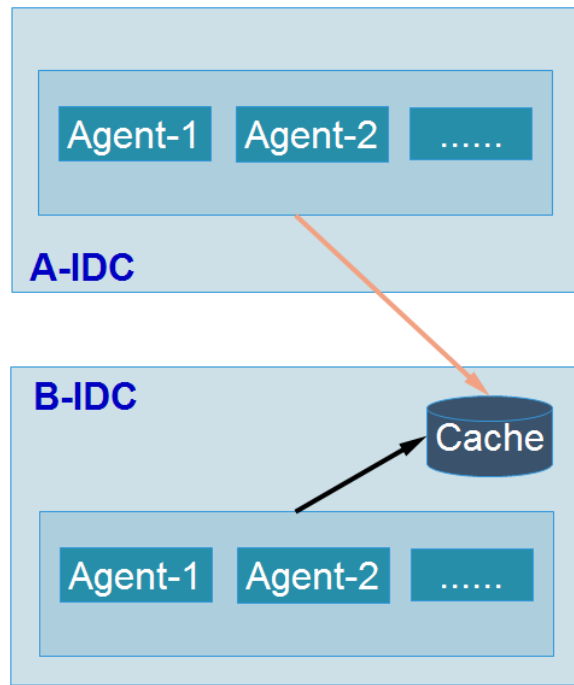
第 1 到第 10 条，共 56 条记录

首页 前页 1 2 3 4 5 6 后页 尾页

流管理平台核心功能

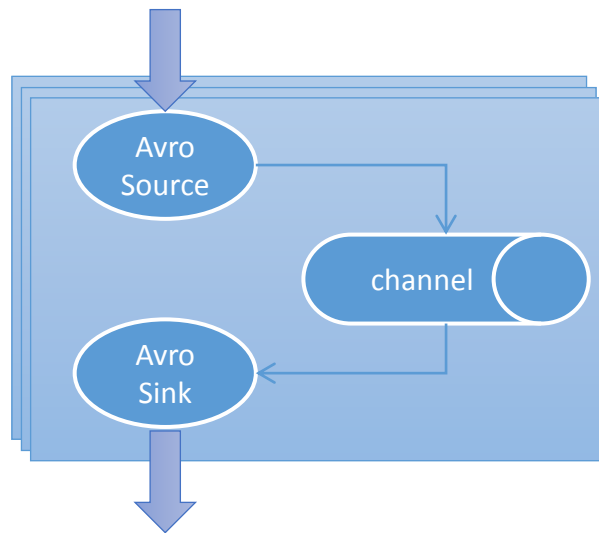
- 接入业务的管理、发布、上线
- 对Agent节点进行实时监测、管理、命令
- 对Flume的集群监测、管理
- 实时计算的Job管理
- 全链路的数据流量对帐
- 智能监控报警

五、数据中转



Flume介绍

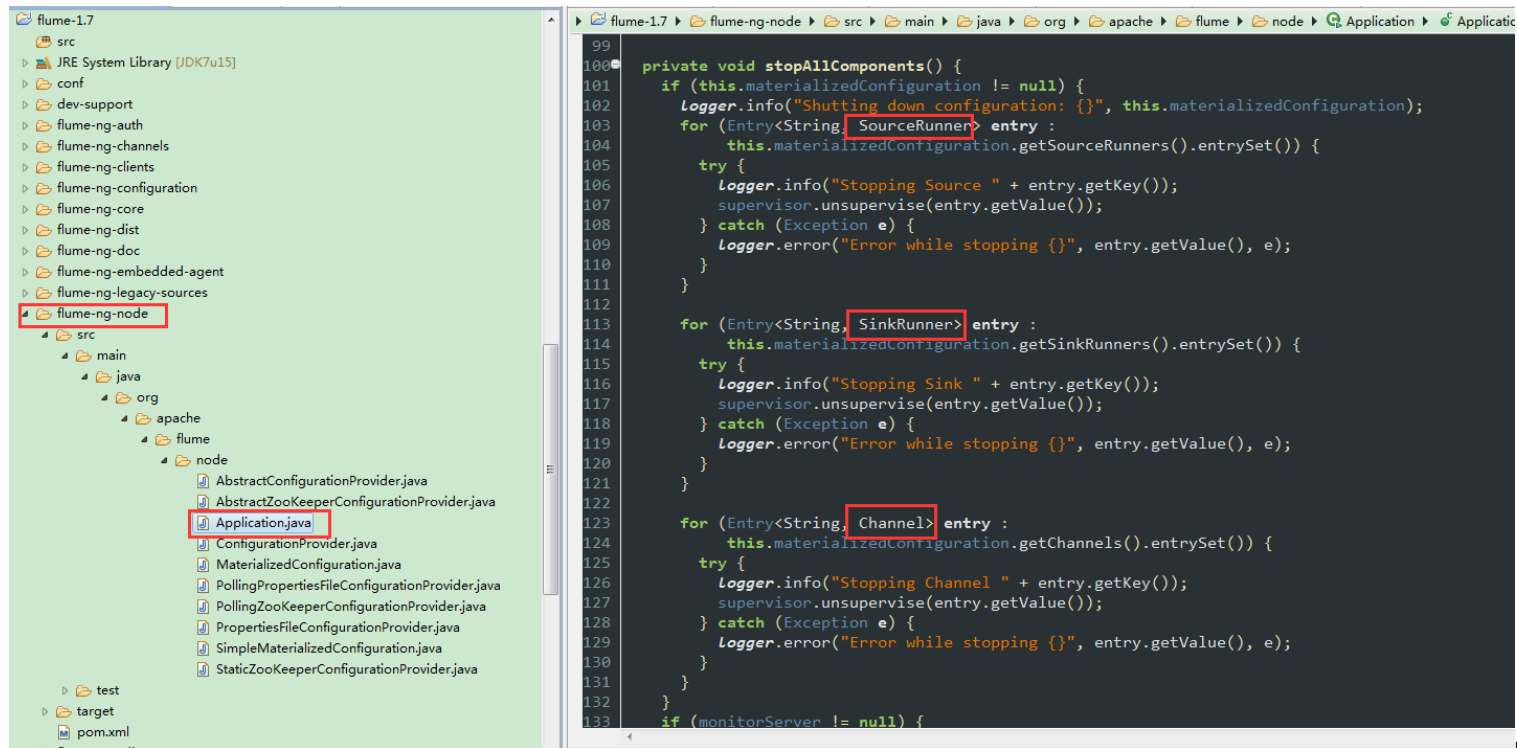
- Avro、HTTP、Thrift...
- Memory、File、Spillable Memory ...
- Avro、Thrift、Hdfs、Hive...



Flume实践

- 无Group，采用Zookeeper做集群
- Agent采用LB做负载均衡，动态感知
- 硬盘缓存、无损启停
- 停止顺序优化
- 多种转发方式
- 自定义Sink，多线程发送（channel的get只能单线程）

停止顺序



The image shows a screenshot of an IDE with two panels. The left panel displays the project structure for flume-1.7, with the file `Application.java` highlighted in the `flume-ng-node` directory. The right panel shows the code for the `stopAllComponents()` method in `Application.java`. The code is as follows:

```
99 private void stopAllComponents() {
100     if (this.materializedConfiguration != null) {
101         Logger.info("Shutting down configuration: {}", this.materializedConfiguration);
102         for (Entry<String, SourceRunner> entry :
103             this.materializedConfiguration.getSourceRunners().entrySet()) {
104             try {
105                 Logger.info("Stopping Source " + entry.getKey());
106                 supervisor.unsupervise(entry.getValue());
107             } catch (Exception e) {
108                 Logger.error("Error while stopping {}", entry.getValue(), e);
109             }
110         }
111     }
112
113     for (Entry<String, SinkRunner> entry :
114         this.materializedConfiguration.getSinkRunners().entrySet()) {
115         try {
116             Logger.info("Stopping Sink " + entry.getKey());
117             supervisor.unsupervise(entry.getValue());
118         } catch (Exception e) {
119             Logger.error("Error while stopping {}", entry.getValue(), e);
120         }
121     }
122
123     for (Entry<String, Channel> entry :
124         this.materializedConfiguration.getChannels().entrySet()) {
125         try {
126             Logger.info("Stopping Channel " + entry.getKey());
127             supervisor.unsupervise(entry.getValue());
128         } catch (Exception e) {
129             Logger.error("Error while stopping {}", entry.getValue(), e);
130         }
131     }
132 }
133 if (monitorServer != null) {
```

Memory的capacity

参数					结果						
包大小	channel 容量	agent 个数	agent 线程	TPS	agent 流量 (M)	flume 流量 (M)	flume CPU%	metaq CPU%	Channel 占用 (%)	JVM MEM 占用 (OLD)	
10k	10000	1	5	5113	35	35	2-3	2-3	1-38	OC:311M OU:5M	
	10000	2	10	9691	68	68	4-6	3-6	1-70	OC:311M OU:5M	
	50000	1	5	5123	35	35	2-3	2-3	0.2-6	OC:311M OU:5M	
	100000	1	5	5278	35	35	2-4	2-3	0.1-3	OC:311M OU:5M	
30k	10000	1	5	2647	170	170	2-6	2-5	30-60	OC:311M OU:259M	
	10000	2	10	2712	200	200	3-7	3-7	99-100	OC:719M OU:387M	
	50000	1	5	2624	170	170	2-4	2-6	5-10	OC:369M OU:276M	
	50000	2	10	2749	200	200	3-6	3-6	99-100	OC:2497M OU:2000M	
	100000	1	5	2685	170	170	2-4	2-6	2-4	OC:311M OU:198M	
	100000	2	10	2544	200	200	3-8	3-6	99-100	OC:3473M OU:3039M	
50k	10000	1	5	1791	200	200	3-5	3-6	85-100	OC:1212M OU:778M	
	50000	1	5	1846	200	200	3-6	3-6	97-100	OC:3473M OU:3013M	
	100000	1	5	1824	200	200	3-7	3-7	64-66	OC:3473M OU:3304M	

六、实时计算

维度/组件	实时性	容错(重复、可靠)	数据流	HA	性能	集群规模	开发易用性	Api易用性	流行度	发展性
Storm 1.x	单个事件 (trident支持batch)	ACK机制 Trident事务机制	多种分流方式	支持	1.x宣称 提速16倍	zookeeper (Pacemaker) Storm On Mesos Storm On Yarn (非原生)	强大简便的本地模式 (不用搭建任何东西)	Java版的API Trident API 通过JNI支持其他语言	-	Apache 2011年出现 另有阿里的Jstorm加入 Storm只有流处理能力
Spark-Streaming	一个时间窗口内的所有事件	RDD的简易 重新生成机制	RDD	支持	官测速度不错	standalone (zookeeper) spark on mesos spark on YARN	无本地模式	Java版 python版的api scala版	-	Apache 2013年出现 Spark兼有MR和流处理 2种能力

实时计算集群

- Metaq+Spark (Standalone , ZK做HA)
- Kafka+Strom (ZK)

Spark实践

- 直接写HDFS底层文件
- 自动创建不存在的Hive分区
- Metaq的日志切割
- 不要定时Kill Job

Q&A

Thanks