

Application and Practice of Apache Flink® In iQIYI

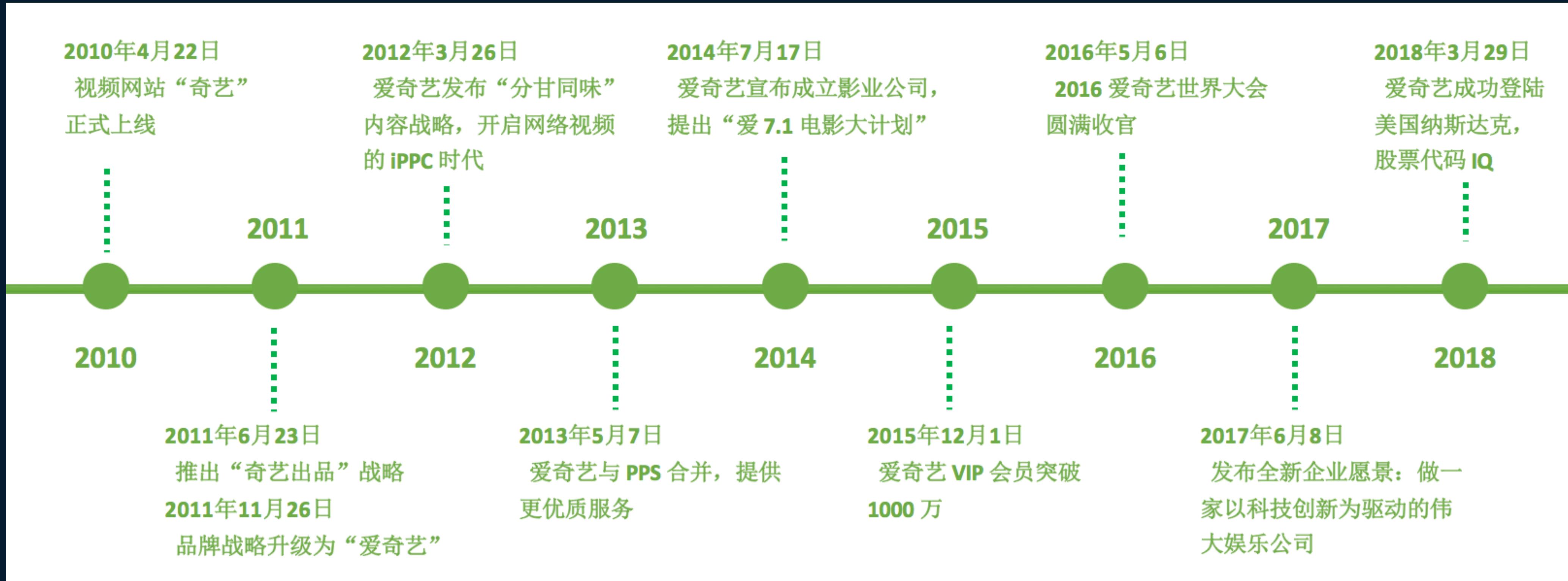
公司：爱奇艺

职位：资深工程师

演讲者：陈越晨



Introduction to iQIYI



Introduction to iQIYI

行业数据领跑 位于行业领先地位

我们拥有规模庞大且高度活跃的用户基础

5.65亿人

月活跃用户数

在线视频领域 No.1

59.08亿小时

在线视频领域 No.1

全网 No.3



移动端

数据来源：QuestMobile，2018年8月

爱奇艺月度总有效时长稳居中国APP榜第三名

Agenda

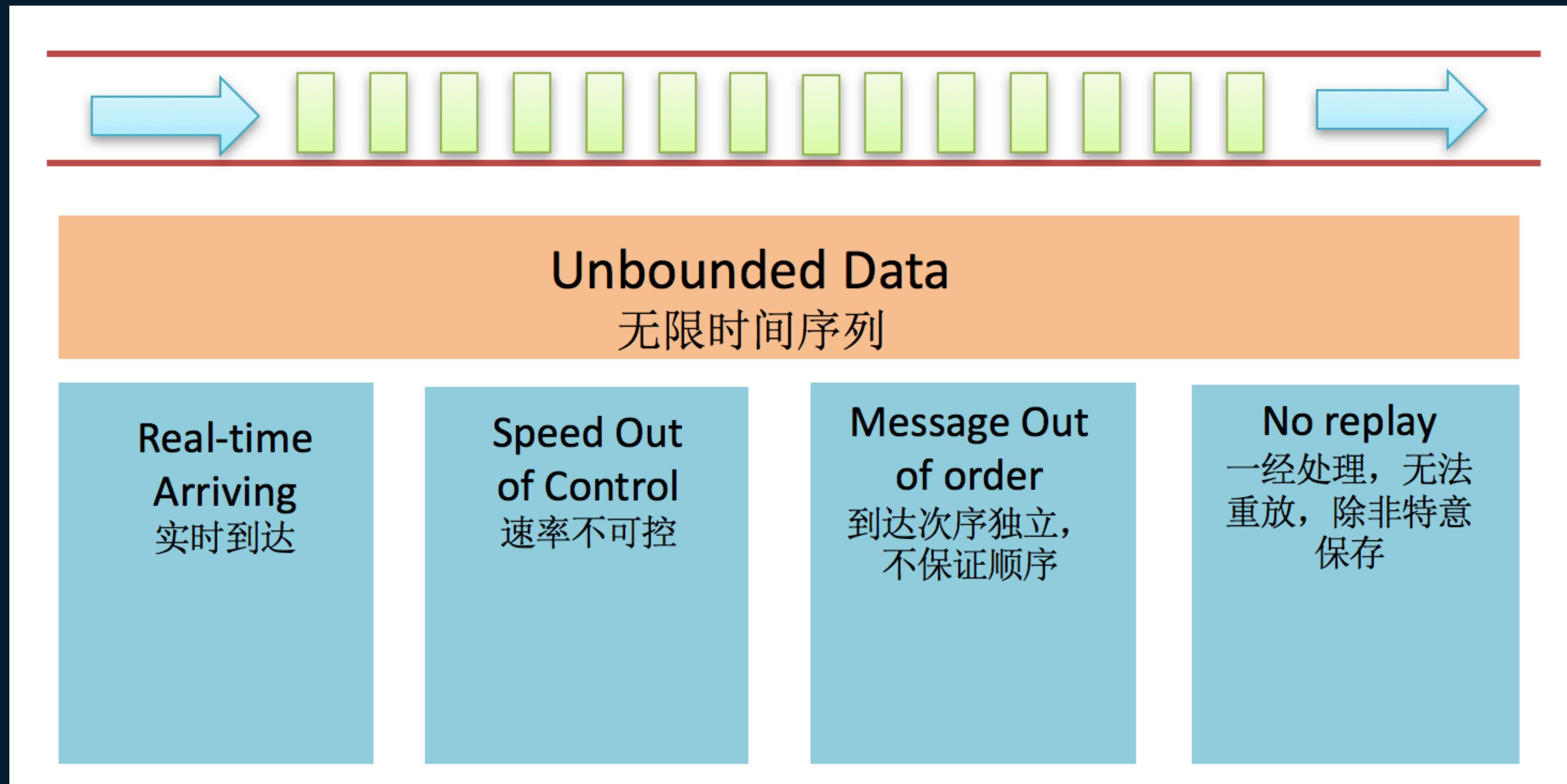
- Challenges and evolution
- Flink use cases @iQIYI
- Platform built for Flink
- Improvements on Flink
- Future Work
- 实时计算的挑战与演化
- 爱奇艺怎么使用Flink
- Flink平台化
- 对Flink的改进
- 未来工作



Challenges and evolution

What's RTC (Real-time computing) ?

实时计算是什么？



Challenges and evolution

Problems 问题

- Out of order 数据乱序
- Out of date 数据延时
- Event time vs Processing time 事件时间 vs 处理时间

Challenges 挑战

- Correctness 正确性
- Fault Tolerant 容错
- Performance 性能
- Latency 延迟
- Throughput 吞吐量
- Scalability 扩展性

Peak events per second: 11 Million/s

峰值事件数: 1100万/s



Challenges and evolution

RTC evolution @ iQIYI

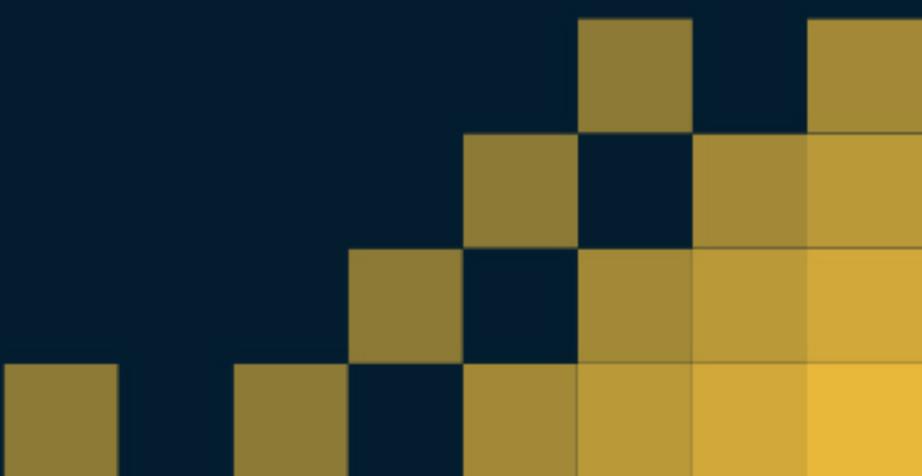
- 2013 : Storm, 3 clusters
- 2015 : Spark streaming on YARN
- 2016 : Developed platform for Spark streaming
- 2017 : Flink standalone & Flink on YARN
- 2018 : Flink co-locate with Kafka, developed Streaming

SQL and real-time analysis platform

爱奇艺实时计算里程碑

- 2013 : 小规模使用Storm, 3个独立集群
- 2015 : 引入Spark Streaming, 部署在YARN
- 2016 : Spark Streaming平台化, 大规模使用
- 2017 : 引入Flink, 部署在独立集群和YARN
- 2018 : Flink与Kafka混合部署, 构建Streaming SQL与实时分析平

台



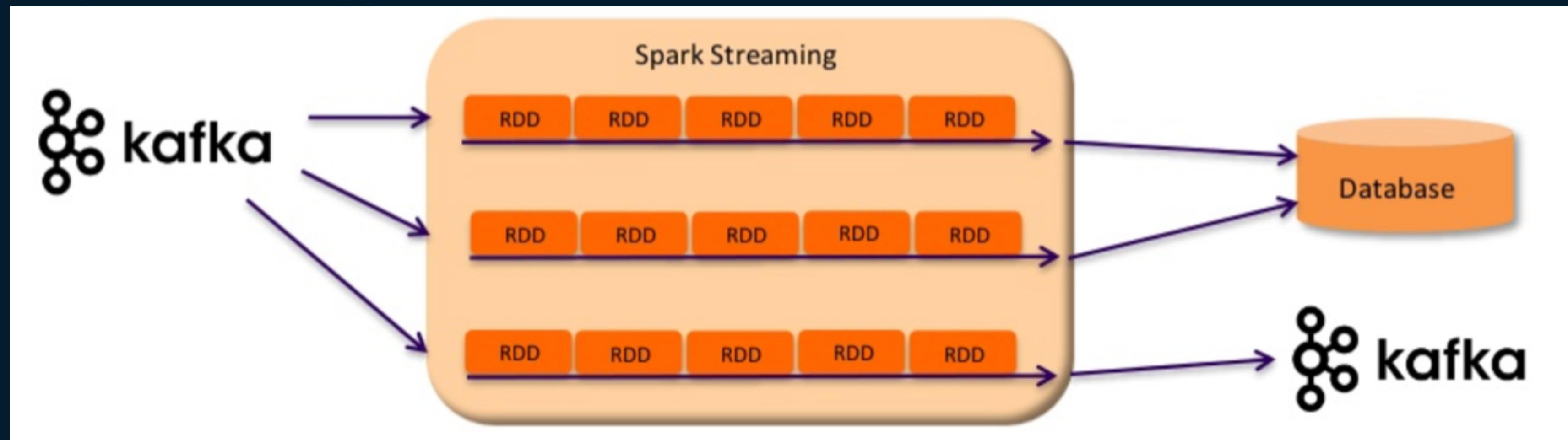
Challenges and evolution

Spark Streaming

- Micro-batch 微批次
- APIs: DStream API, SQL

Problems 问题

- Hard to build REAL real-time, etc.
- event time、late data 很难构建“实时”计算



Challenges and evolution

Flink - Dataflow Model (What Where When How)

- Layered APIs: DataStream API, process function, SQL
- Guaranteed correctness: exactly once, event-time, late data handing
- Performance: low latency (ms)
- Scales: scale-out architecture, incremental checkpointing
- 分层API : DataStream API, Process Function, SQL
- 正确性保证: Exactly once, 事件时间, 延时数据处理
- 性能 : 低延时 (ms)
- 伸缩性: Scale-out架构, 增量checkpoint

Challenges and evolution

Flink vs Spark Streaming

Framework 计算框架	Flink	Spark Streaming
Streaming Model 流计算模型	Native stream processing 原生流处理	Micro batch 微批处理
Latency 延迟	ms 毫秒	s 秒
Throughput 吞吐量	High 高	High 高
Study curve 学习曲线	Middle 中等	Low 低
Fault-tolerant 容错性	Incremental checkpointing 增量checkpoint	RDD checkpoint
Event Time 事件时间	Support 原生支持	Partial 业务逻辑实现
Late Data 延时数据处理	Support 原生支持	Partial 业务逻辑实现

Flink use cases @iQIYI

- Real-time ETL with massive data 海量数据实时ETL
- Real-time fraud detection 实时风控
- Distributed system tracing 分布式调用链分析

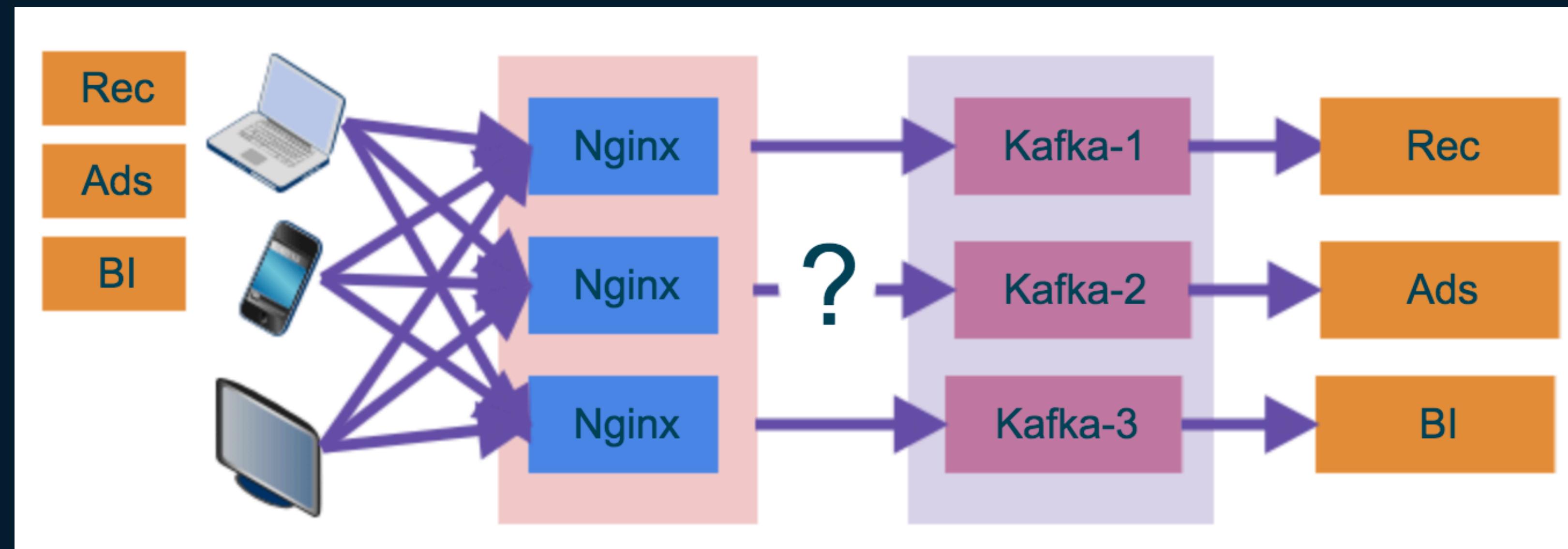
Real-time ETL with massive data

Problem:

- Aggregated event logs in Nginx (Peak: 11 Million/s)
- Sort by category, write to multiple Kafka clusters

问题:

- 将Nginx上聚合日志（峰值流量：1100万/s）根据业务线拆分到多个Kafka集群



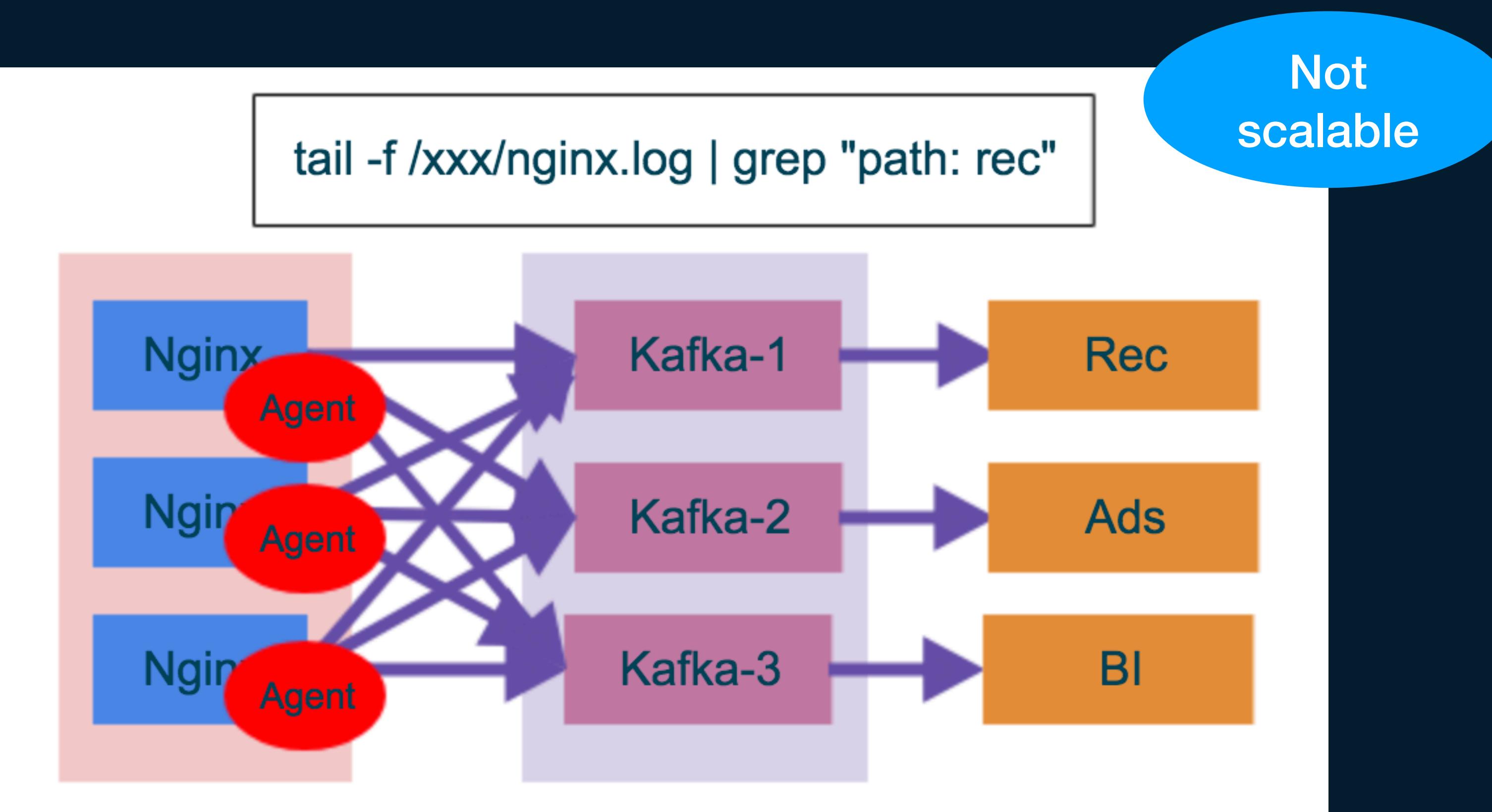
Real-time ETL with massive data

Option 1: Use tail command

选项1：使用tail命令

```
tail -f /xxx/nginx.log | grep "path: rec"
```

Not
scalable



Real-time ETL with massive data

Option 2: ETL

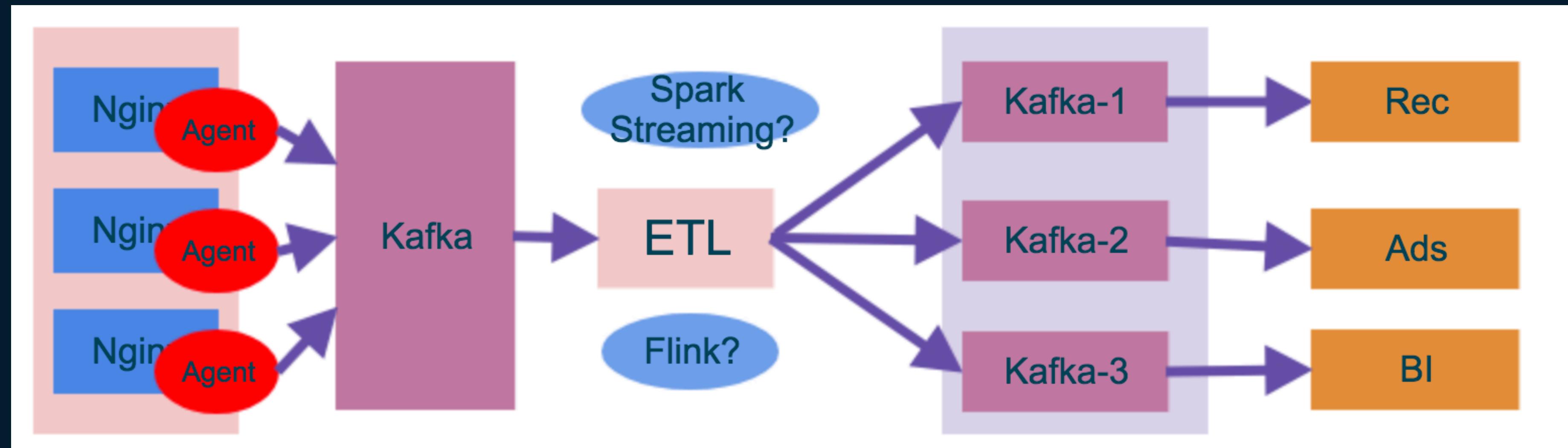
Requirements :

- High Throughput (11 Million/s)
- Low latency (<1s)

选项2：ETL拆分

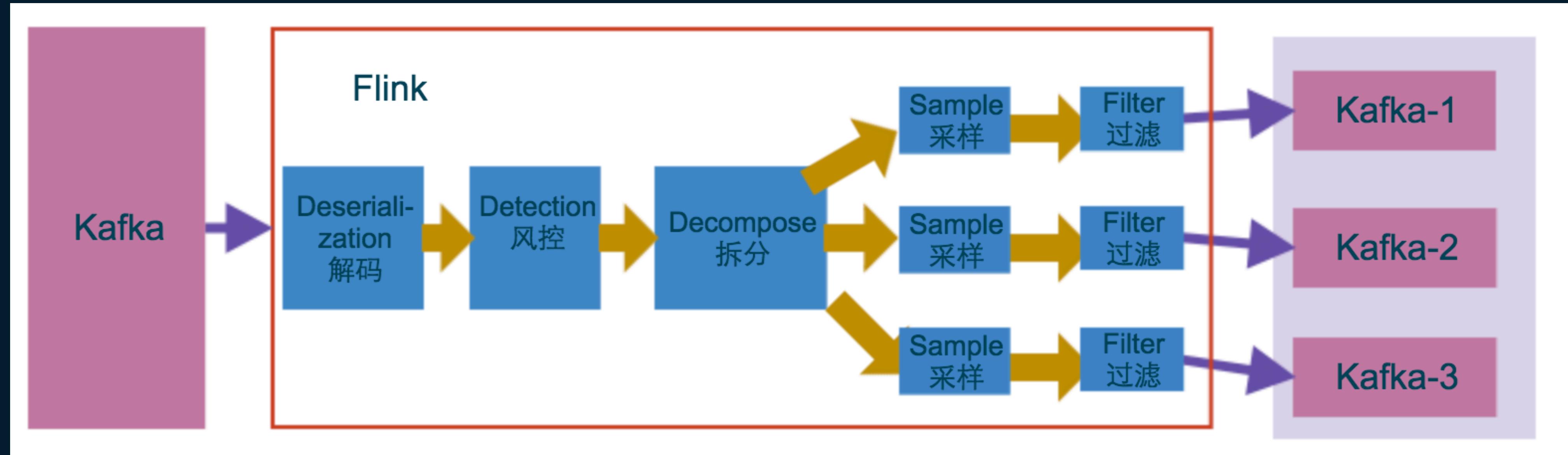
需求：

- 高吞吐 (千万QPS)
- 低延时 (<1s)



Real-time ETL with massive data

Flink ETLs



Real-time ETL with massive data

IP Blacklist Filtering

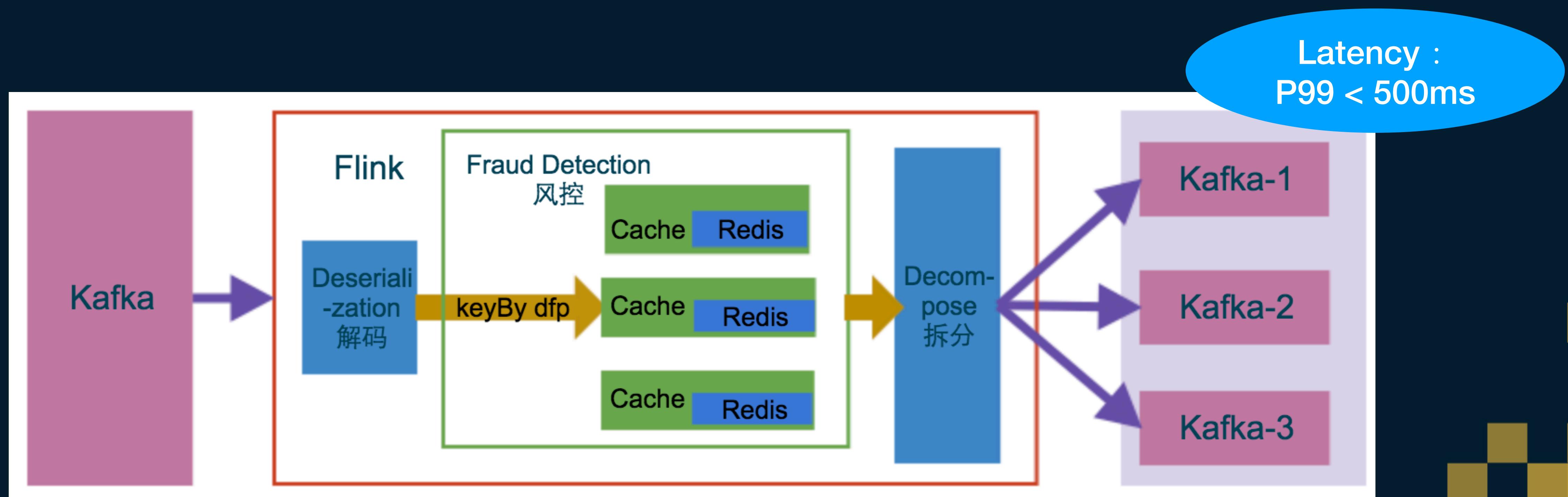
Tuning :

- KeyBy DeviceId
- Local LRU Cache + Redis (4k/s)

IP 黑名单过滤

优化:

- 通过设备ID keyBy, 避免处理倾斜
- 缓存优化: Local LRU Cache + Redis (4k/s)



Real-time ETL with massive data

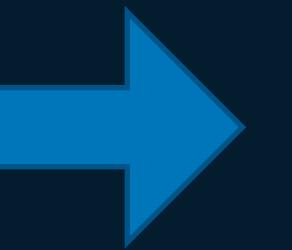
Resource Tuning 资源优化

Flink

- 200+ machines 200+机器
- CPU-bound CPU密集型
- CPU usage: 55% CPU利用率55%

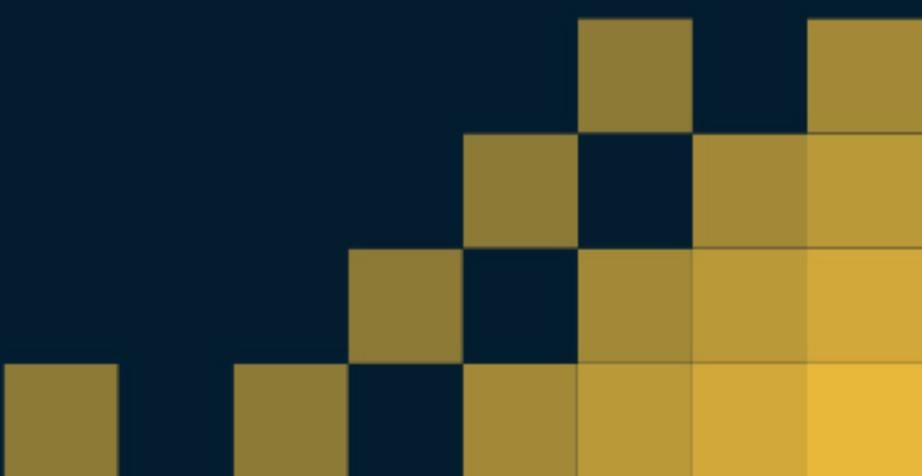
Kafka

- 100+ machines 100+机器
- IO-bound & disk-bound IO吞吐大, 磁盘存储多
- IO Throughput: 8W IO吞吐8w
- CPU usage: 15% CPU利用率15%



Mixed: Flink & Kafka 混合部署

- 200+ machines 200+机器
- CPU usage: 70% CPU利用率70%



Real-time fraud detection

Fraud detection–collision attack

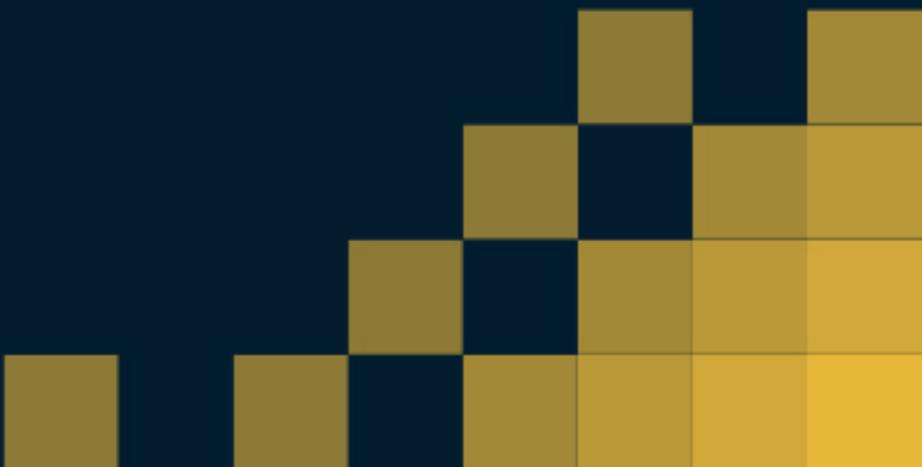
Requirements :

- Detect anomalies
- Post Analysis: Generate IP & Device ID blacklist

风控检测 – 机器撞库盗号攻击

需求：

- 事中过滤: 超高频异常检测
- 事后分析 : 生成IP和设备ID的黑名单



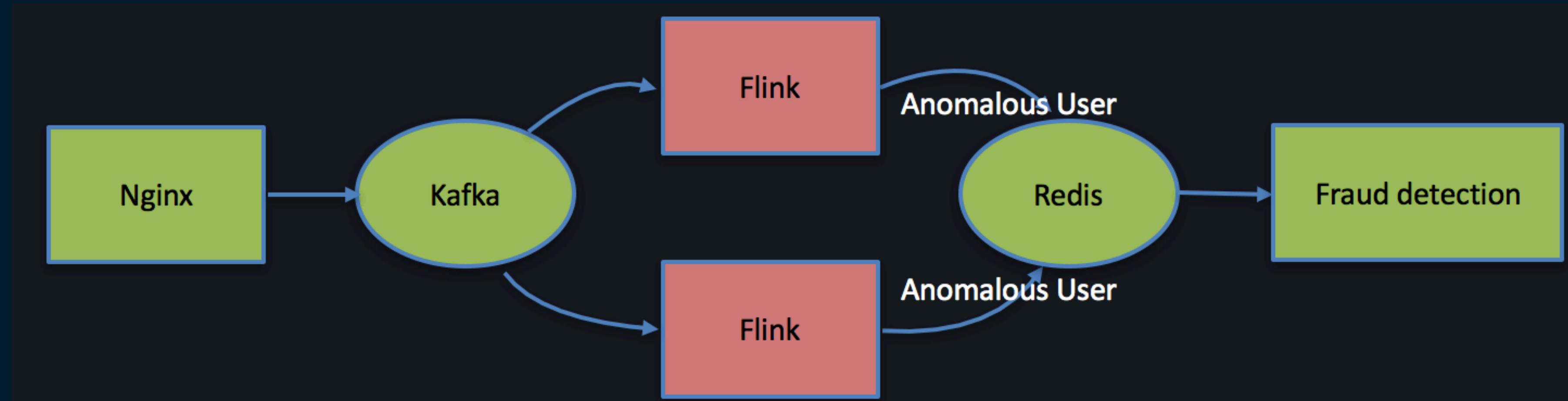
Real-time fraud detection

CEP:

- Recently registered user
- Perform specific actions in short time

复杂事件处理:

- 刚注册的用户
- 短时间内进行一些特定的业务操作



High-frequency anomaly detection:

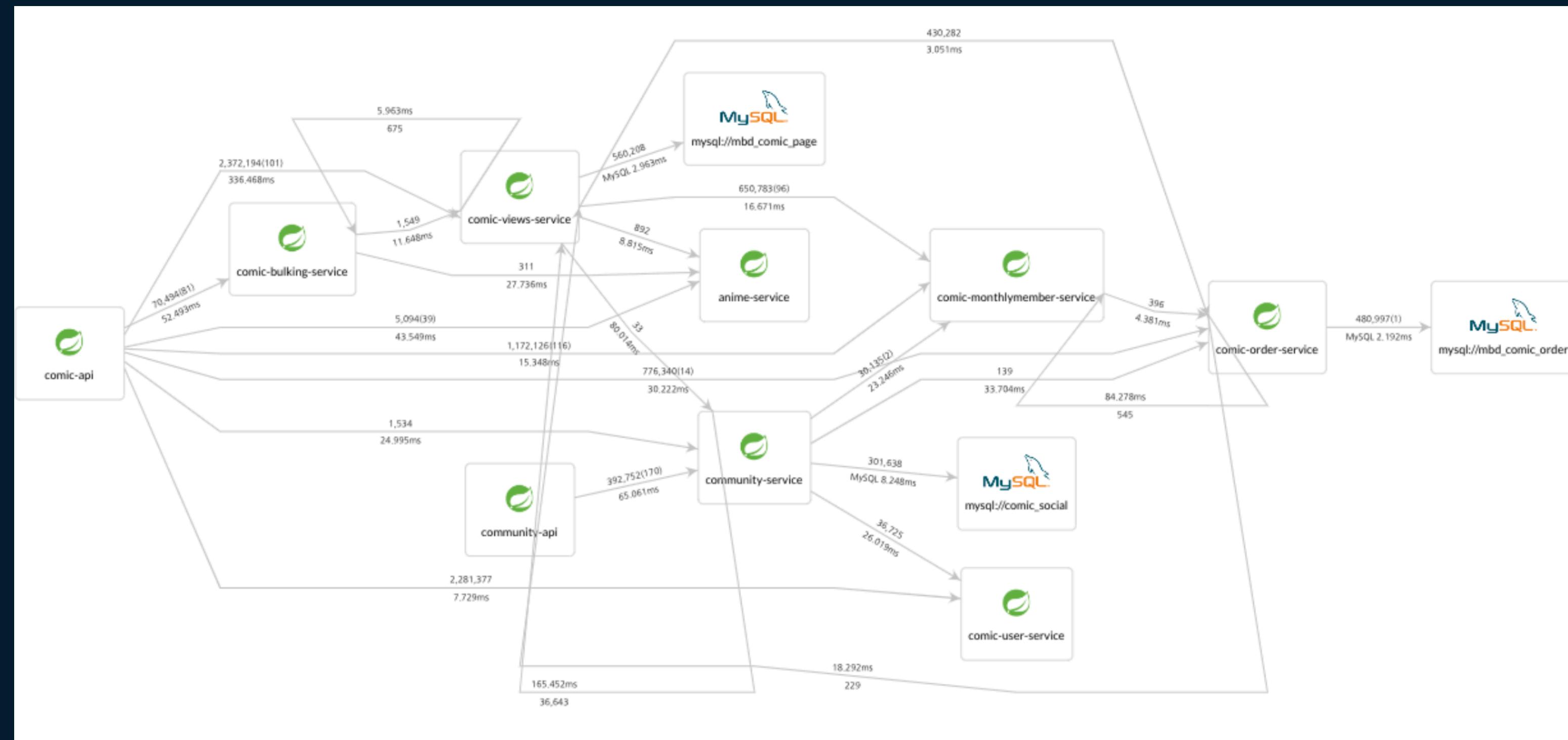
- Count # of requests by one user every second
- Above threshold -> anomalous user

超高频异常检测:

- 对同一个用户在1秒内的请求进行计数
- 超过某个阈值标记为黑产

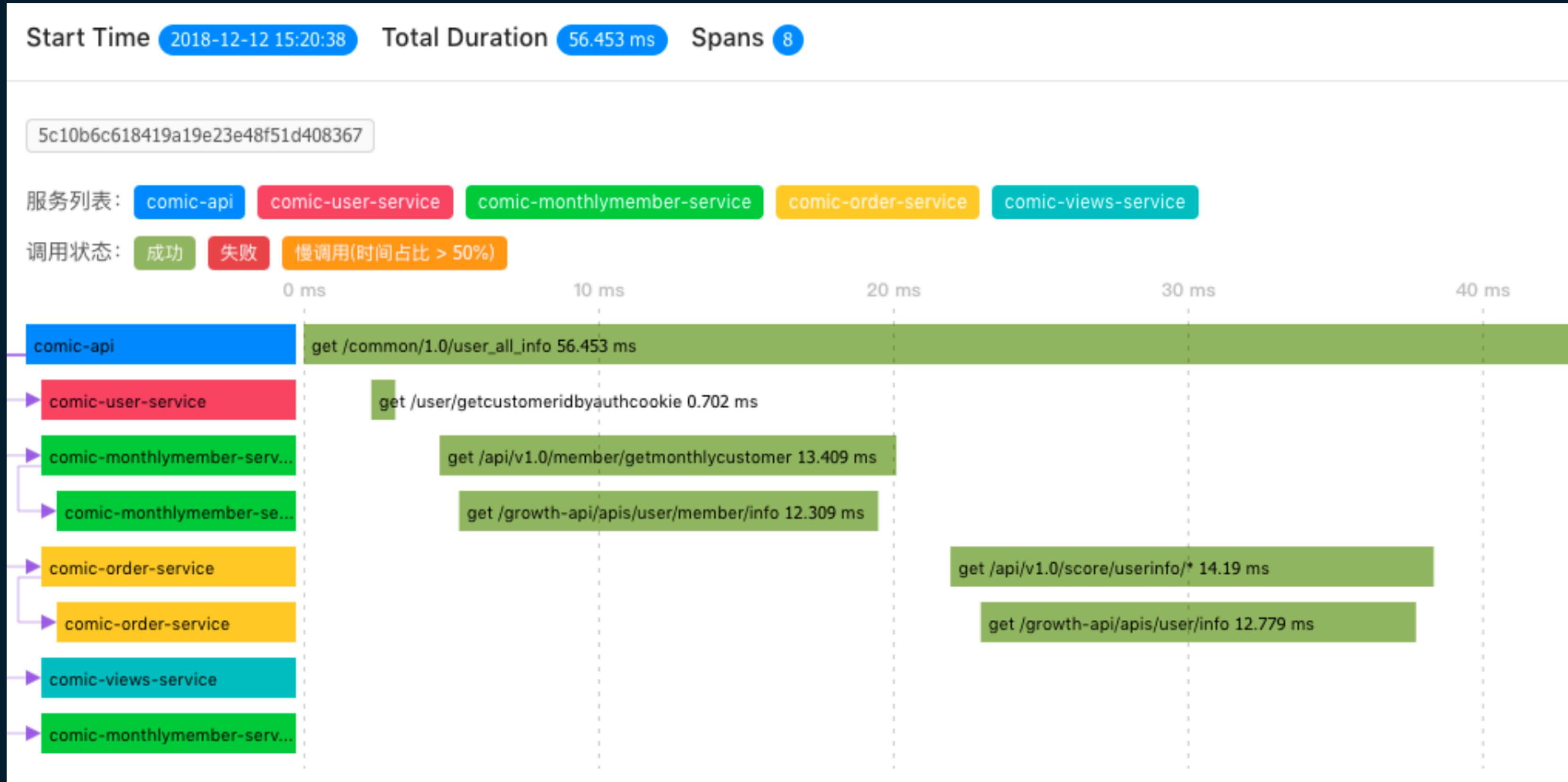
Distributed system tracing

Average duration on tracing topology 整个拓扑的各调用平均耗时



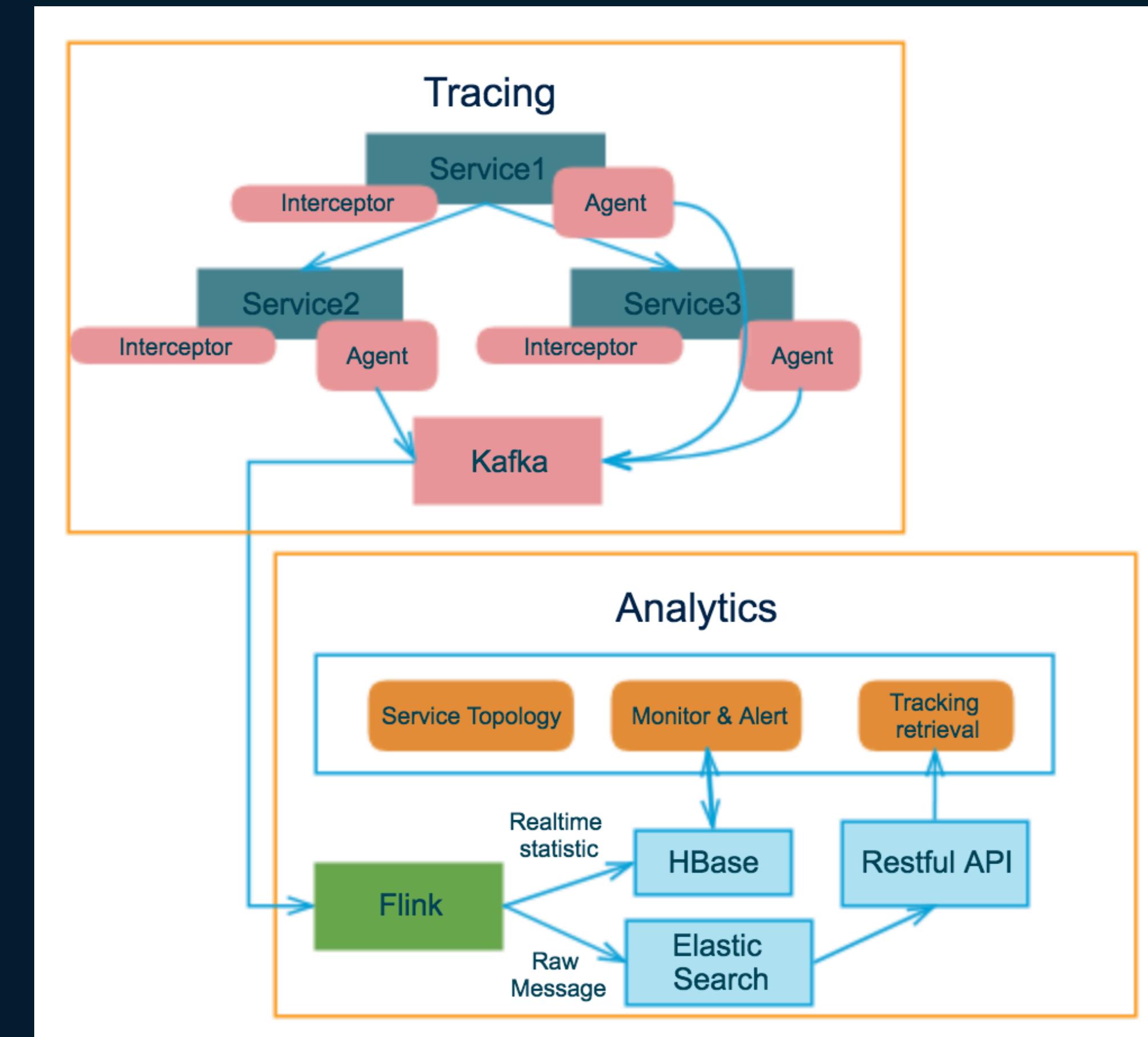
Distributed system tracing

Path and duration of a single call 单次调用的路径和耗时



Distributed system tracing

Architecture 架构



Distributed system tracing

Multiple window aggregations:

- Level 1: build tracing topology of apps
- Level 2: calculate average call duration aggregated by different windows

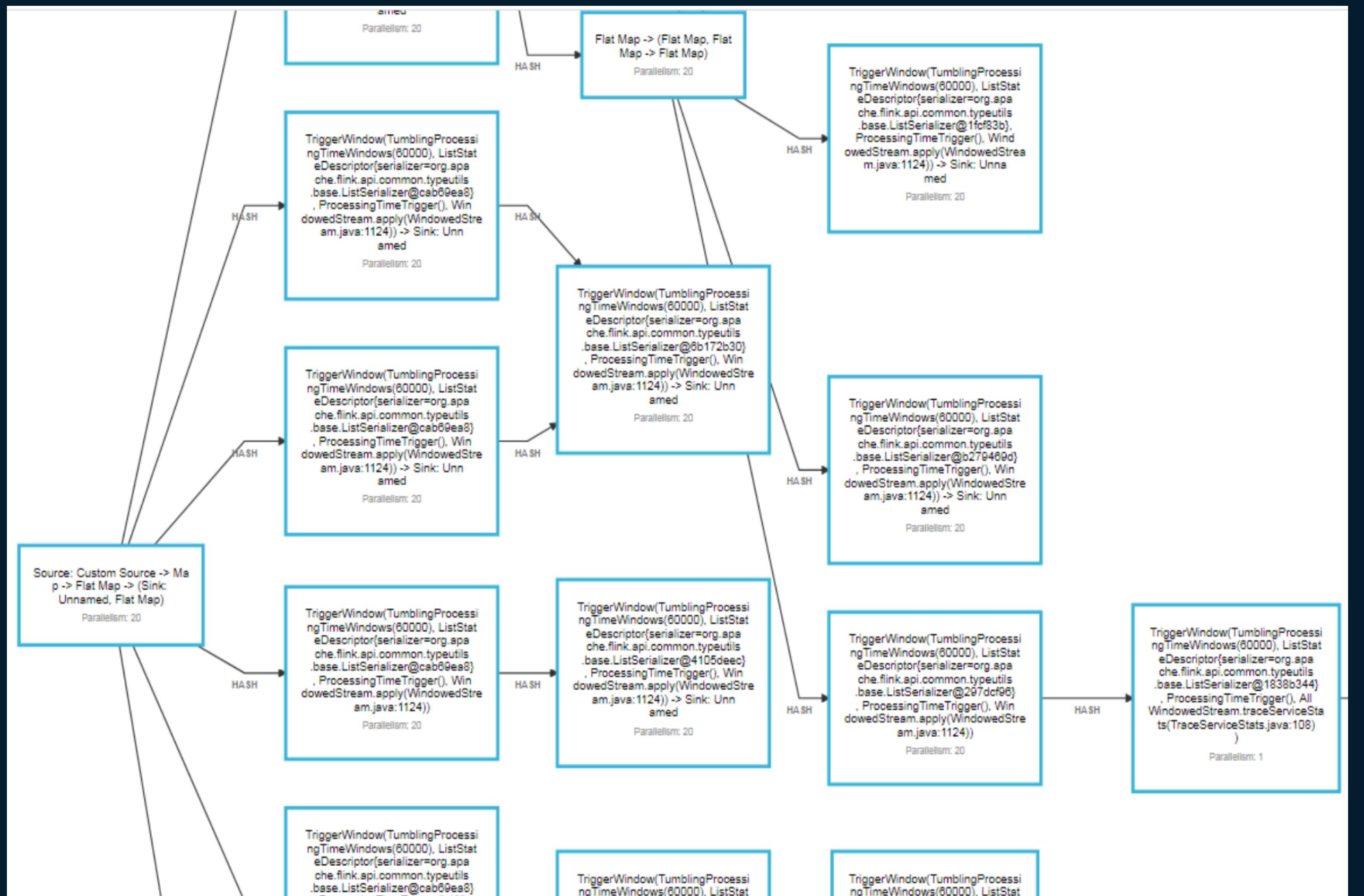
多窗口聚合:

- 第一级：构建App调用拓扑
- 第二级：按窗口计算平均耗时统计



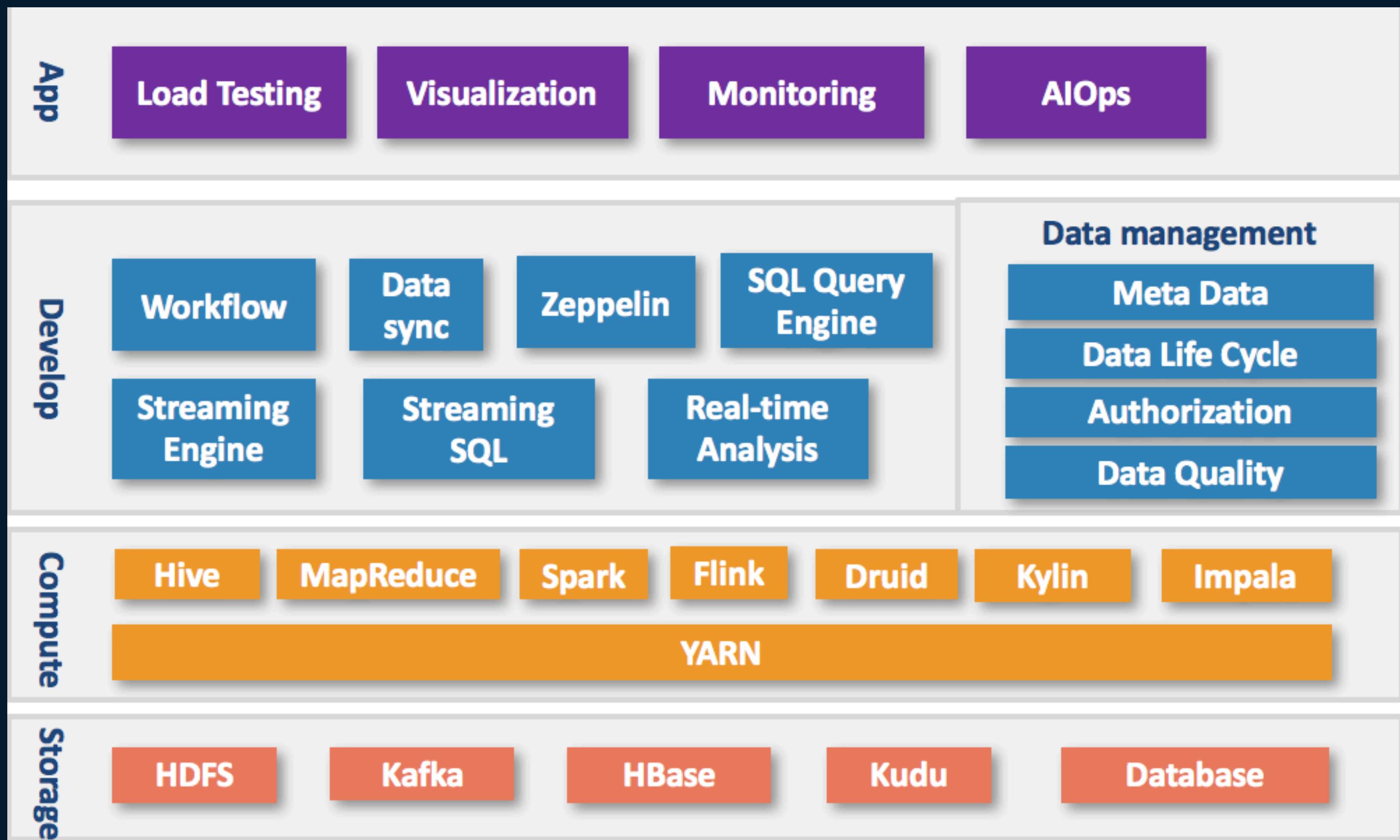
Distributed system tracing

Multiple window triggers 多窗口触发



Platform built for Flink

Overview 概览



Scale

- YARN MapReduce/Spark/Flink

30+ clusters, 6000+ machines

- YARN MapReduce/Spark/Flink

30+集群数, 6000+机器

- Streaming Apps :

Spark Streaming : 2000+

- 流任务:

Spark Streaming : 2000+

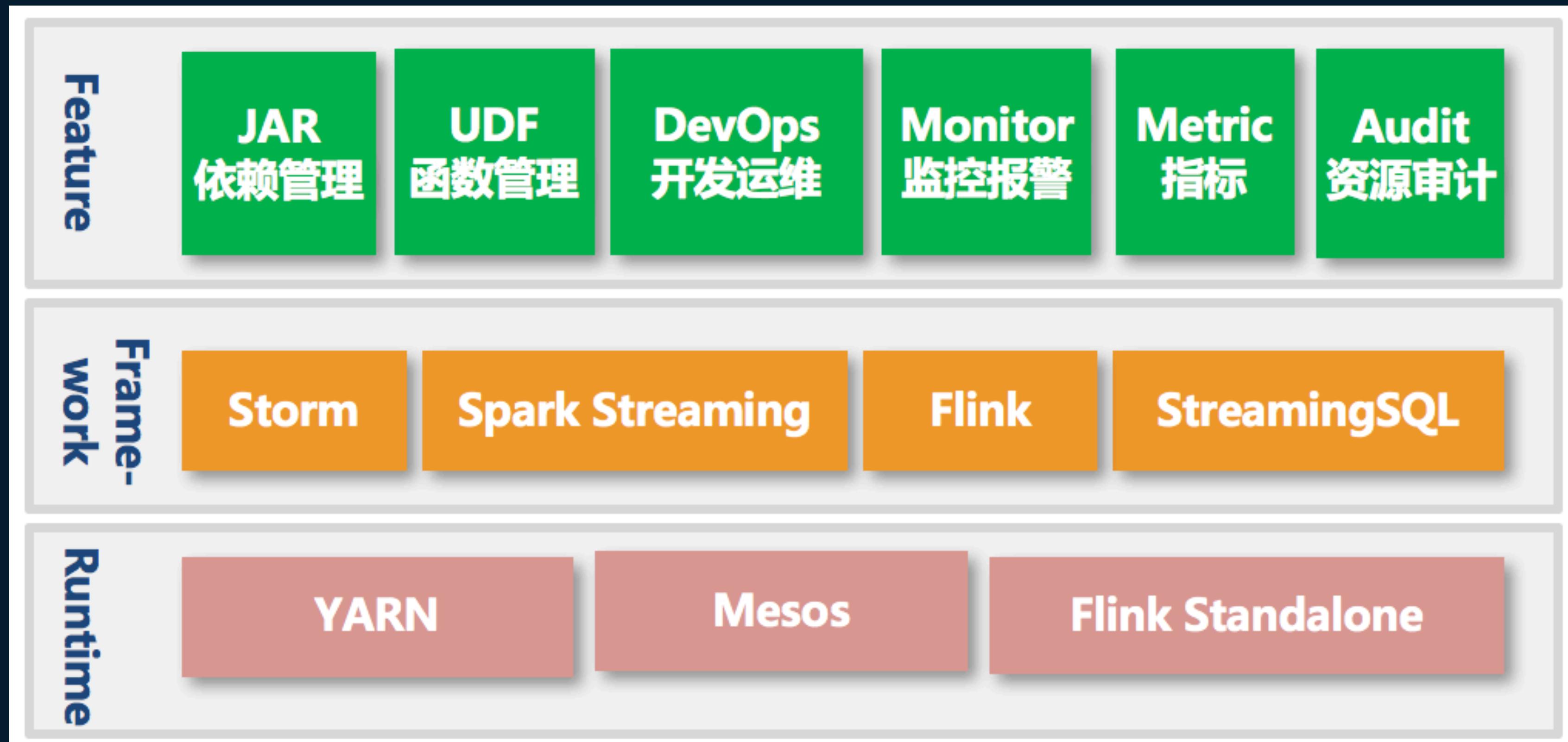
Flink : 300+

Flink : 300+



Streaming Platform

Overview 概览



Streaming Platform

Streaming IDE 流任务IDE



The screenshot shows the configuration for a Flink SQL test task. Key settings include:

- 调度平台: yarn
- 集群: [Cluster icon]
- Flink版本: 1.4.2
- 主类: com.iqiyi.babel.app.FlinkApp
- 程序参数: (empty)
- 编程语言: java
- JobManager内存(G): 4
- TaskManager内存(G): 4
- TaskManager Slot个数: 2
- TaskManager个数: 4



The screenshot shows a list of submitted jobs and a console tab displaying command-line logs for a Flink job submission.

作业ID

(与Europa一致)	作业名称
18971071	S-babel_flink_FlinkDemo-121
18969067	S-babel_flink_FlinkDemo-120
18944451	S-babel_flink_FlinkDemo-119
18938817	S-babel_flink_FlinkDemo-118
18935276	S-babel_flink_FlinkDemo-117
18934046	S-babel_flink_FlinkDemo-116

CONSOLE Stdout

```

cd /europa/local/cmd
export HADOOP_CLASSPATH=/europa-tmp/install/EuropaServer-europa-online-907-20181212173610/europa-server/europa-hadoop/hadoop2.6.0-cdh5.11.0/target/europa-hadoop2.6.0-cdh5.11.0-2.9.3.jar
export HADOOP_CONF_DIR=/europa-tmp/install/EuropaServer-europa-online-907-20181212173610/europa-server/europa-master/data/HadoopConf
/europa-tmp/install/EuropaServer-europa-online-907-20181212173610/europa-server/europa-master/script/submit-flink.sh \
/europa/local/flinks/flink-1.4.2-cdh5.11.0-qiyi/bin/flink-class \
/europa/local/flinks/flink-1.4.2-cdh5.11.0-qiyi/europaconf \
com.iqiyi.europa.EuropaBootstrapper \
org.apache.flink.client.CliFrontend \
europa@HADOOP.QIYI.COM \
/europa/local/kerberos/2001044/europa.keytab \
run \
-m yarn-cluster \
-ynm "Europa_2001044_S-babel_flink_FlinkDemo-121_18971071_1544972565171" \
-c com.qiyi.realtime.flink.calculate.ImpClkEventJoin \
-ytm 4096 \
-ys 2 \
-yn 5 \
-yjm 4096 \

```

Streaming Platform

Monitoring 指标监控

添加报警策略

* 报警策略名称

* 报警范围 本人订阅 项目组全部订阅 自定义范围

* 报警规则 停止 失败重试 Spark任务SLA Flink任务SLA

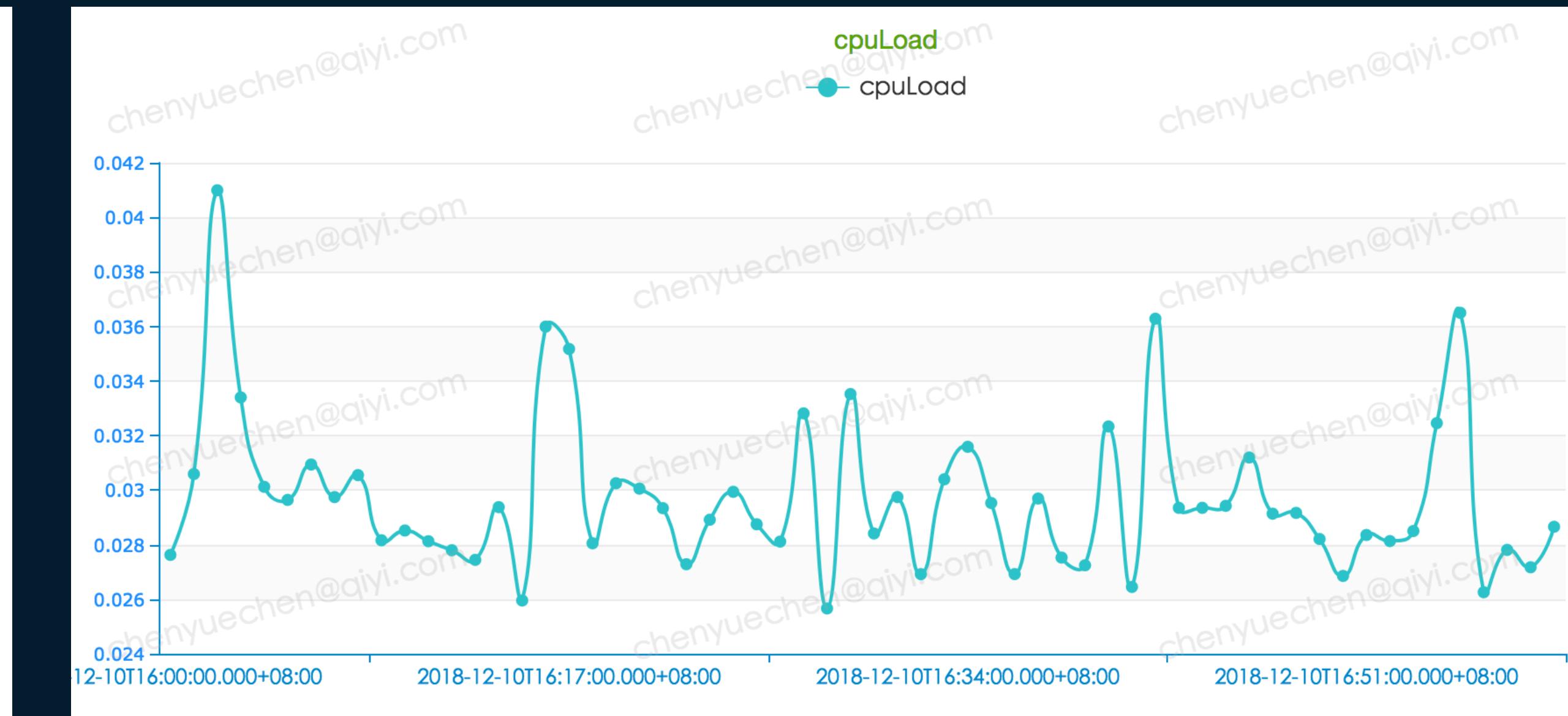
BackPressure过大

检查点失败数量

CPU负载

故障时长

* 报警方式 邮件 热聊 短信



StreamingSQL

Define DDL & DML

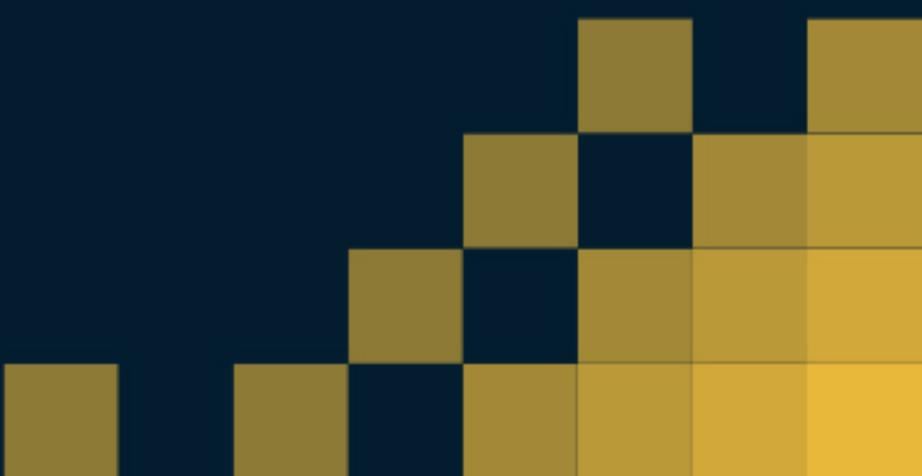
- Streaming Table : MQ Access, Decoding
- Dimension Table : Static Table joining with streaming table
- Temporary Table : Intermediate Result
- Result Table : Output
- UDF

Data Sink : Kafka, MySQL, Kudu, ES,
Druid, HBase

DDL & DML 定义

- 流表 : 定义MQ访问方式, 解码方式
- 维度表: 定义静态表, 用于与流表Join
- 临时表: 定义中间结果
- 结果表: 定义输出数据源信息
- 用户自定义函数

输出源支持: Kafka, MySQL, Kudu, ES,
Druid, HBase



StreamingSQL

Case 示例

```
CREATE STREAM TABLE t1
(`timestamp` long) WITH (
    type = "kafka",
    brokers = "xxxxxxxxxx",
    topics = "xxxxxxxx",
    deserializer = "json");

CREATE TMP TABLE t2 AS
SELECT floor(`timestamp` / 1000) AS
`timestamp` FROM t1;
```

```
CREATE RESULT TABLE r (`timestamp`  

timestamp, min_time timestamp, p99  

timestamp,  

p50 timestamp, avg_time timestamp,  

max_time timestamp) WITH  

(type = "console");

INSERT INTO r  

SELECT  

current_timestamp() AS `timestamp`,  

min(`timestamp`) AS min_time,  

percentile(`timestamp`, 0.01) AS p99,  

percentile(`timestamp`, 0.5) AS p50,  

avg(`timestamp`) AS avg_time,  

max(`timestamp`) AS max_time  

FROM t2;
```

StreamingSQL

Web IDE

故障数 ×

故障数 ×

```

1 # 设置sql
2
3 # set your sql
4
5 CREATE STREAM TABLE t1 (rawMessage string) WITH (
6   type="kafka",
7   brokers =
"10.10.10.10:9092,10.10.10.10:9092,10.10.10.10:9092,10.10.10.10:9092",
8   topics = "ext-example-1.0",
9   deserializer = "json");
10
11 create tmp table t2 as
12   select m['ip'] as ip, unix_timestamp(m['datetime'], 'dd/MMM/yyyy:HH:mm:ss') as ts, pingback(rawMessage) as m from t1;
13
14 create result table r (ip string, city string, isp string, d_dm string, p
15   partitioned by (d_dd string, d_dh string)
16   with (
17     type="hive",
18     database="ext-example-1.0",
19     table="r");
20
21 insert into r
22 select ip, ip2city(ip) as city, ip2isp(ip) as isp, from_unixtime(ts, 'YYYY-MM-DD HH:mm:ss') as d_dd, from_unixtime(ts, 'mm') as d_dm, parse_url(url) as params from t2;
23

```

故障数 ×

```

31 #####
32 # 权限相关
33 #####
34 # 指定任务所在队列
35 # 类型为字符串
36 # 必填
37 ENV stream.queue=
38 # 指定任务使用的key
39 # 类型为字符串
40 # 必填
41 ENV stream.principality=
42 # 其他
43 #####
44 ENV stream.interval=60;
45 #####
46 #####
47 # 选填, 默认为300
48 #####
49 # 指定Batch的间隔时间
50 # 类型为整型
51 # 选填, 默认为300
52 ENV stream.interval=60;
53 ENV stream.principality=
54 #####

```

set your udf

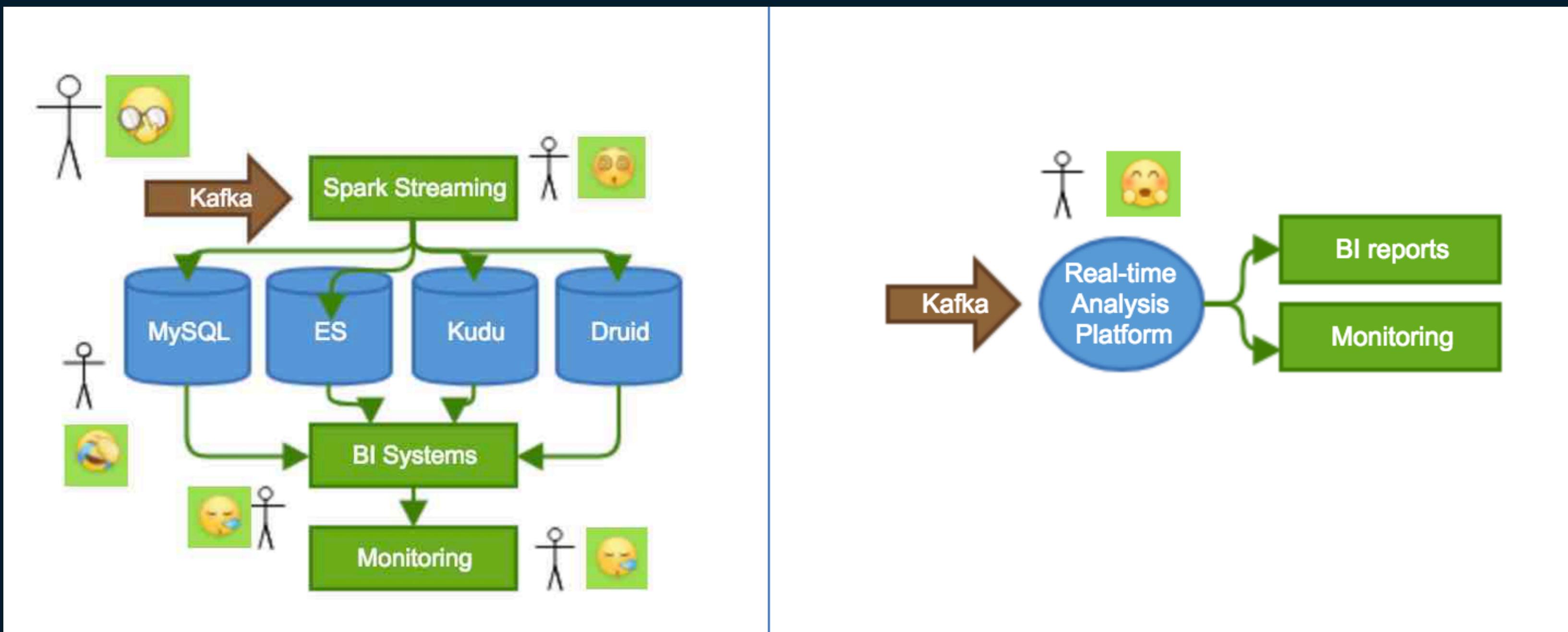
程序jar包 :

+ ext-example-1.0-SNAPSHOT.jar

环境变量 SQL 自定义函数

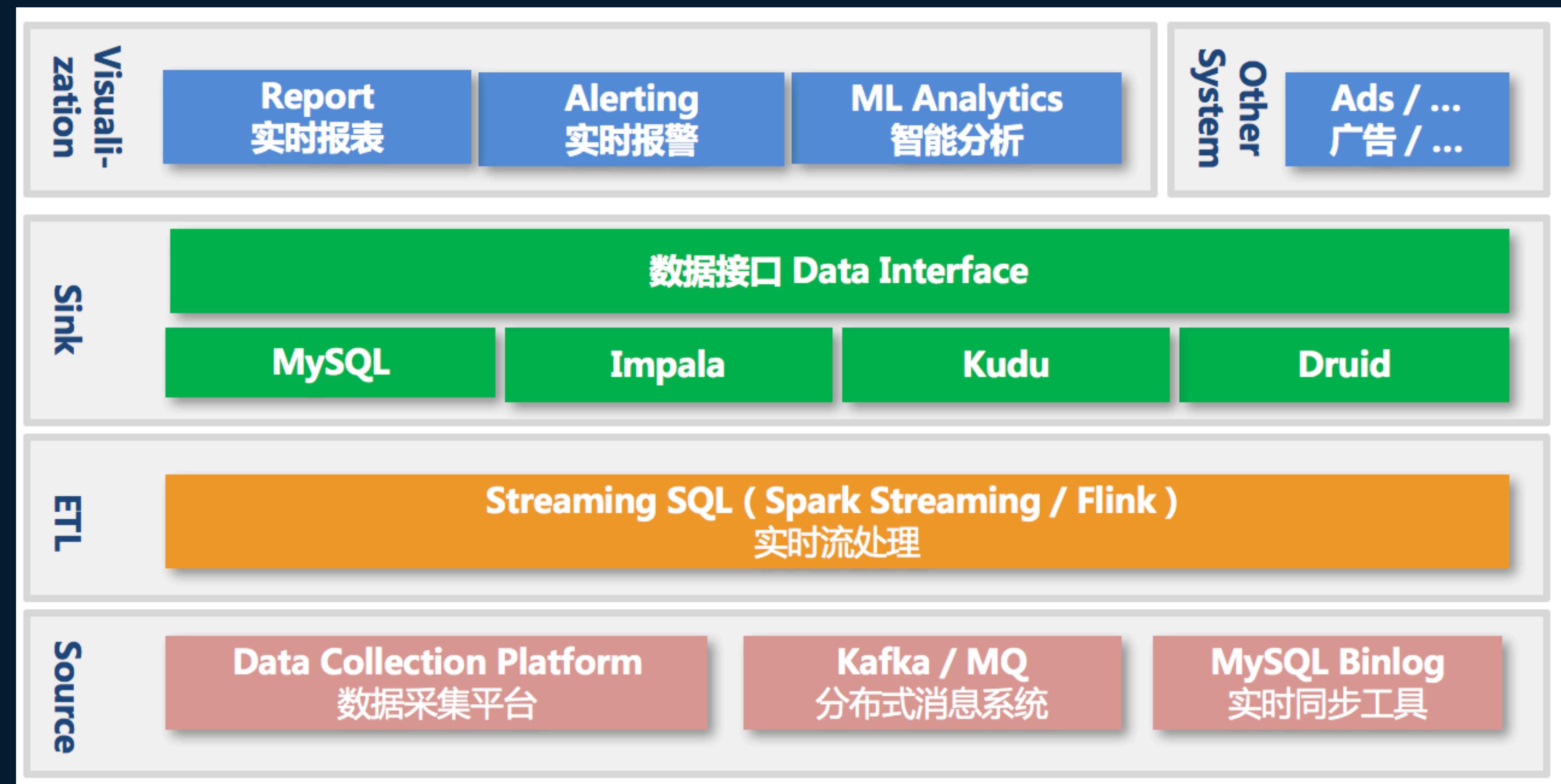
Real-time Analysis Platform

- User-guided configuration
- Compute & Storage invisibility
- Low latency
- Sub-second query response time
- 全向导配置
- 计算存储透明
- 低延时
- 亚秒级查询



Real-time Analysis Platform

Architecture 架构



Real-time Analysis Platform

User-guided configuration 用户向导配置

数据接入 > 数据处理 > 模型配置 > 报表配置

时间戳定义 ②

时间戳/毫秒

指标定义 ②

计数(count)

独立计数(distinct)

独立计数(distinct)

求和(sum)

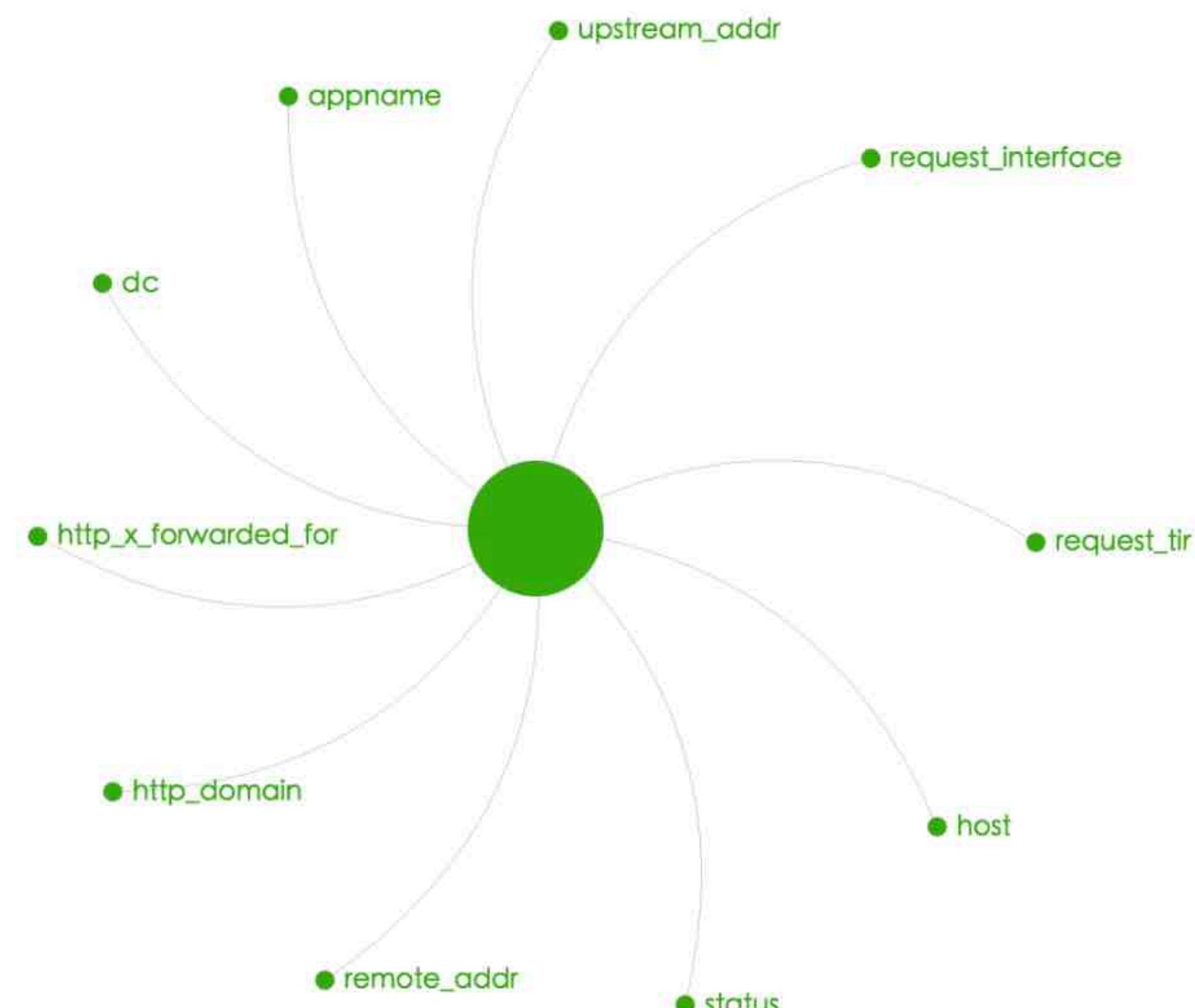
+

维度定义 ②

字典维度

字典维度

字典维度



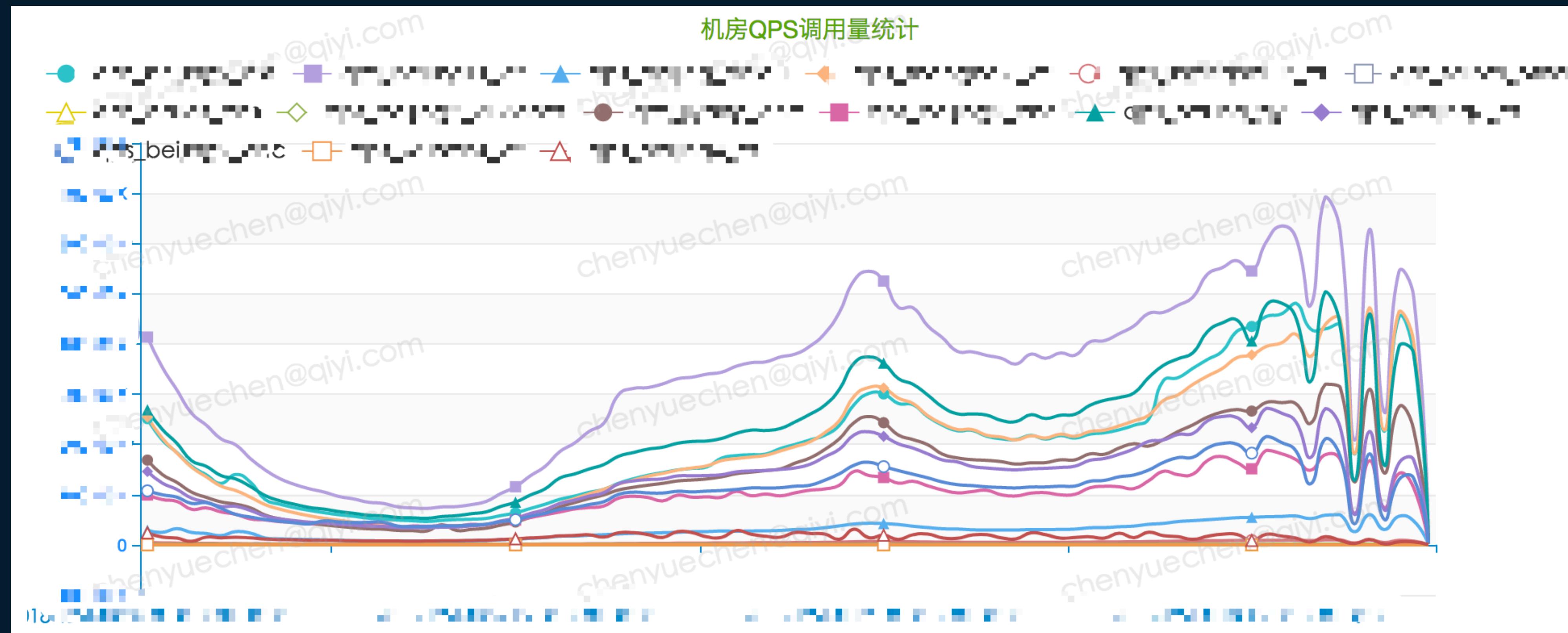
The diagram illustrates the relationships between various data dimensions. A central green circle is connected to several other green circles, each representing a different dimension or feature:

- upstream_addr
- appname
- dc
- http_x_forwarded_for
- request_interface
- request_time
- host
- status
- remote_addr
- http_domain
- dc

Real-time Analysis Platform



Reports 报表



Real-time Analysis Platform

Data processing 数据处理配置

添加临时表

单表处理

转换过程

处理表	stream_table0				
字段转换	remote_addr	Ipv4转省份(Cn)	as	remote_addr	<input checked="" type="checkbox"/>
	http_x_forwar...	Ipv4转省份(Cn)	as	http_x_forwarded_for	<input checked="" type="checkbox"/>
	upsteam_name	不转换	as	upsteam_name	<input checked="" type="checkbox"/>
	appname	不转换	as	appname	<input checked="" type="checkbox"/>
	http_domain	不转换	as	http_domain	<input checked="" type="checkbox"/>

取消 确定

添加临时表

自定义SQL

SQL编辑器

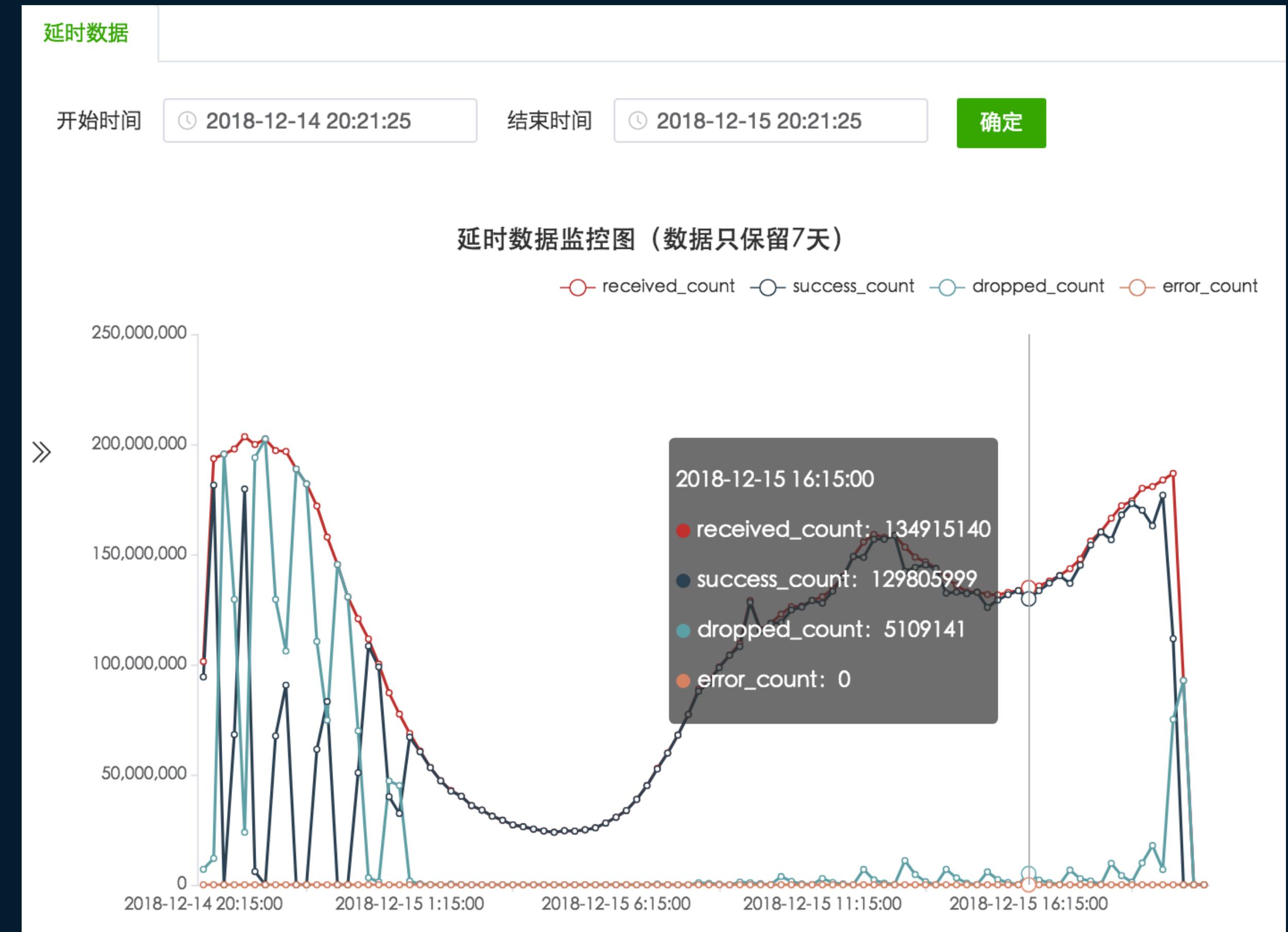
```
select
    remote_addr,http_x_forwarded_for,upsteam_name,appname,http_domain,host,dc,
    request_interface,upstream_addr,request_time,upstream_response_time,status,timestamp,
    case
        when request_time >= 0.000 and request_time < 0.020 then "0-20"
        when request_time >= 0.020 and request_time < 0.050 then "20-50"
        when request_time >= 0.050 and request_time < 0.100 then "50-100"
        when request_time >= 0.100 and request_time < 0.500 then "100-500"
        when request_time >= 0.500 then "500+"
        else "exception"
    end as request_time_interval
from temp_table0
```

输出字段集

remote_addr	string
http_x_forwar...	string
upsteam_nam...	string
appname	string

Real-time Analysis Platform

Data quality monitor 数据质量监控



Improvements on Flink

Improvement - Resuming from latest checkpoint gracefully

改进 – 优雅恢复checkpoint

Problem (问题) :

- Savepoint path required for resuming the job from externalized checkpoints

从checkpoint恢复需指定savepoint路径，无法通过用户代码控制

- Command : flink run -s :savepointPath [:runArgs]

```

val ssc = StreamingContext.getOrCreate(checkpointDirectory, () => {
    createStreamingContext(checkpointDirectory, uri, user, password, subject, seconds)
})

ssc.start()
ssc.awaitTermination()
}

def createStreamingContext(checkpointDirectory: String, uri: String, user: String, password: String, subject: String,
                           seconds: Int) : StreamingContext = {
    val sparkConf = new SparkConf()
    sparkConf.setAppName("AmqReader")
    sparkConf.setIfMissing("spark.streaming.receiver.maxRate", "100")
    sparkConf.setIfMissing("spark.master", "local[4]")

    // 如果需要保证数据不丢，必须在设置Checkpoint的同时设置WAL
    sparkConf.setIfMissing("spark.streaming.receiver.writeAheadLog.enable", "true")
}

```

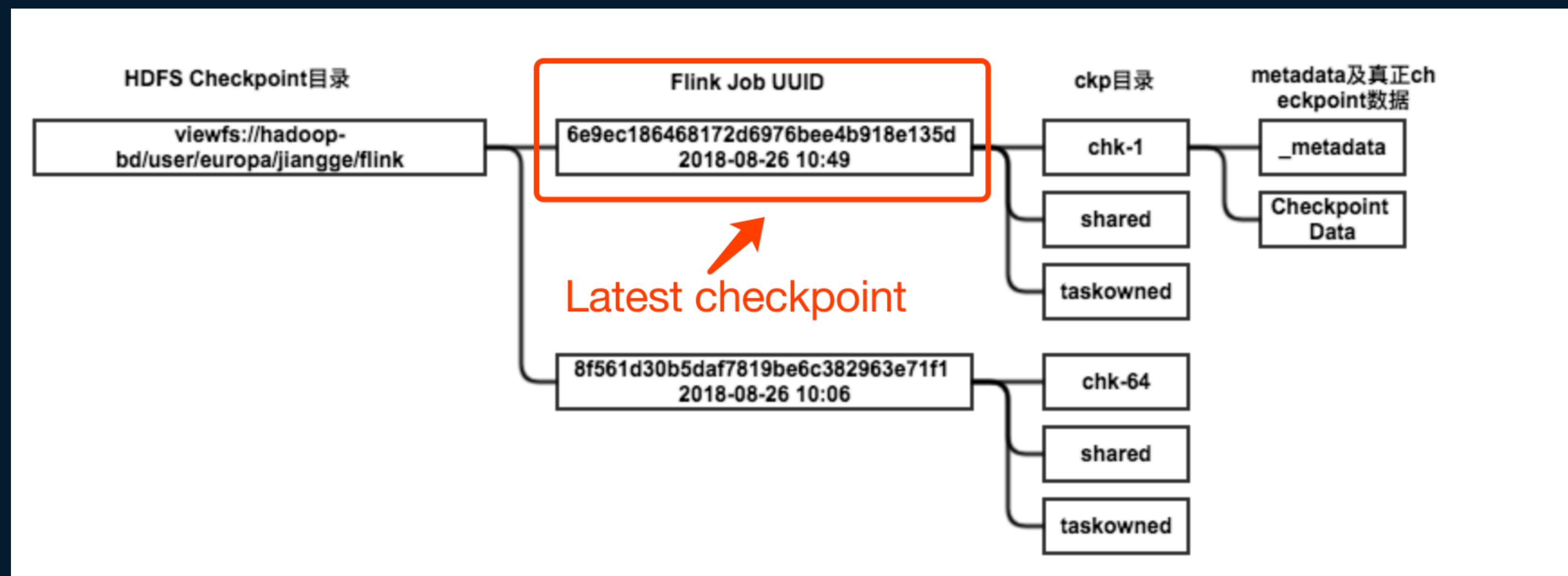
Spark Streaming: Resuming from codes

Improvements on Flink

Improvement - Resuming from latest checkpoint gracefully

改进 – 优雅恢复checkpoint

Command : flink run -s [:runArgs]



Improvements on Flink

Improvement – Kafka Broker HA

改进 - Kafka Broker HA

Problem 问题:

Job failed when one Kafka broker shutdown

一台Kafka Broker宕机后任务会失败

Improvement 改进:

- Handle socketTimeOutException
- Make retry times configurable

处理socketTimeOutException, 支持重试次数配置

```

try {
    fetchResponse = consumer.fetch(fetchRequest);
}
catch (Throwable cce) {
    //noinspection ConstantConditions
    if (cce instanceof ClosedChannelException) {
        LOG.warn("Fetch failed because of ClosedChannelException.");
        LOG.debug("Full exception", cce);

        // we don't know if the broker is overloaded or unavailable.
        // retry a few times, then return ALL partitions for new leader
        if (++reconnects >= reconnectLimit) {
            LOG.warn("Unable to reach broker after {} retries. Returning all partitions to unassigned state.", reconnects);
            for (KafkaTopicPartitionState<TopicAndPartition> fp: this.partitions.values()) {
                fp.setUnassigned();
            }
            unassignedPartitions.add(fp);
        }
        this.partitions.clear();
        continue; // jump to top of loop: will close thread or subscribe to topic
    }
}
try {
    consumer.close();
} catch (Throwable t) {
}

```

Future Work

- StreamingSQL on Flink
- Flink Resource Audit
- Exactly once on Flink 1.7 & Kafka 2.0
- Flink batch
- StreamingSQL集成Flink SQL
- Flink资源审计
- Flink 1.7 & Kafka 2.0上的Exactly once
- Flink 批处理模式

THANKS

