基于Flink构建CEP引擎的挑战和实践

The Challenge and Practice of Building CEP Engine Based on Flink

韩 鹏 - 奇安信集团高级研发总监

Han peng - Senior Director of Qianxin Group

FLINK FORWARD # ASIA

实时即未来 # Real-time Is The Future

FLINK FORWARD

Contents
目录

背景及现状

Background and current situation

01

- 技术演进
  Technology evolution
- Flink的痛点
  Shortcomings of the Flink
- 具体的应用场景和数据规模
  Specific application scenarios and data scale

技术演进

Technology evolution

2015

CEP based on Esper

Single server

Limited hardware resources

Poor performance

2017

Dolphin 1.0 implemented by C++

Single server

Limited hardware resources

Implemented by C++

2018

Sabre based on Flink

Capability of multi-source heterogeneous

Distributed expansion capability

Hundreds of semantic expressions

Humanized interactive graphic configuration

Flink的痛点

Shortcomings of the Flink

在企业级受限（硬件资源）环境，规则集数量及种类不确定的情况下，Flink程序运行较难控制，且当前现有库“Flink SQL”和“Flink CEP”均不能满足业务及性能需求。

Due to the hardware-constrained environment applied to enterprise, the number and type of rules are uncertain, the job based on Fink is difficult to work right, and the current existing libraries "Flink SQL" and "Flink CEP" can't meet the product and performance requirements of network security detection.

具体痛点：

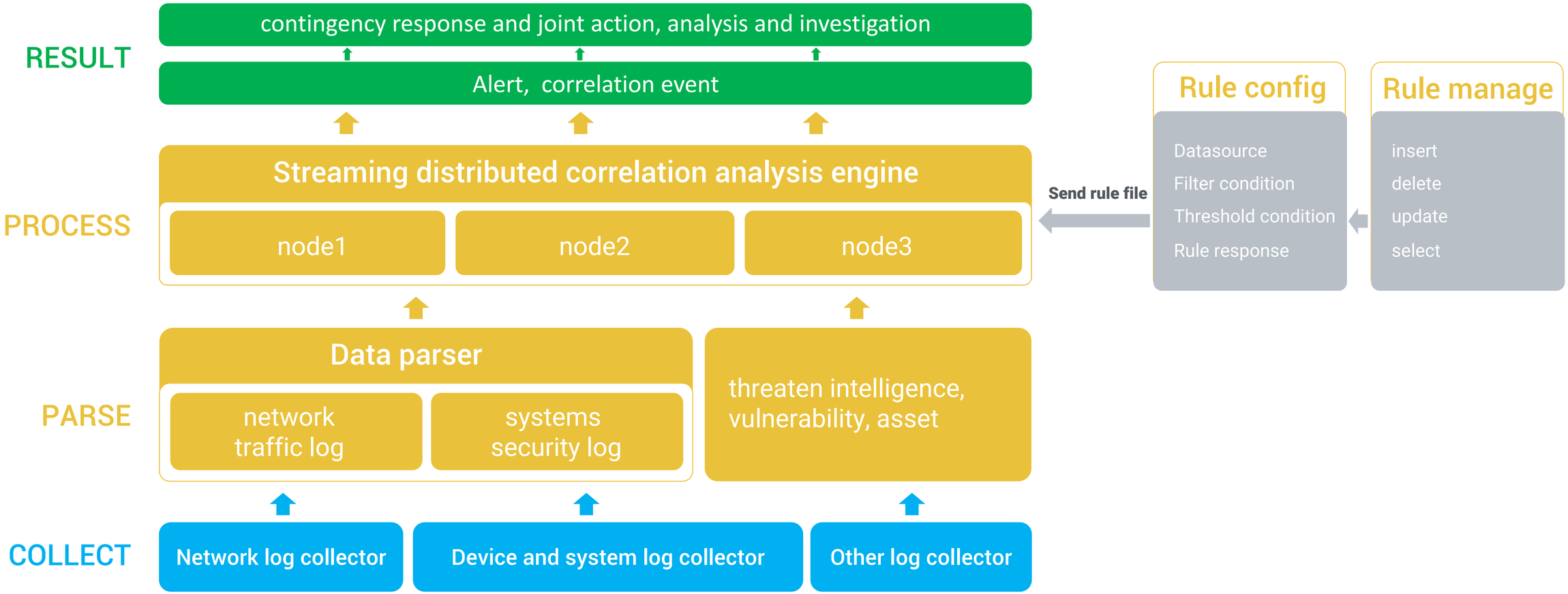
Specific disadvantages

1. 不能进行语义优化
   No semantic optimization
2. 不便于动态更新规则
   Not easy to update rules dynamically
3. 状态监控&高可用
   Status monitoring & High availability
4. CEP网络负载高、CPU利用率低
   High network load, Lower CPU utilization
5. 无资源保护机制
   No resource protection mechanism
6. 重复告警，不支持空值窗口触发
   Repeated alert event, no null trigger window
7. 无不发生算子
   No not occur operator
8. 聚合不保存原始数据
   Aggregation does not save raw data

...

具体的应用场景和数据规模

Specific application scenarios and data scale to be processed



技术架构

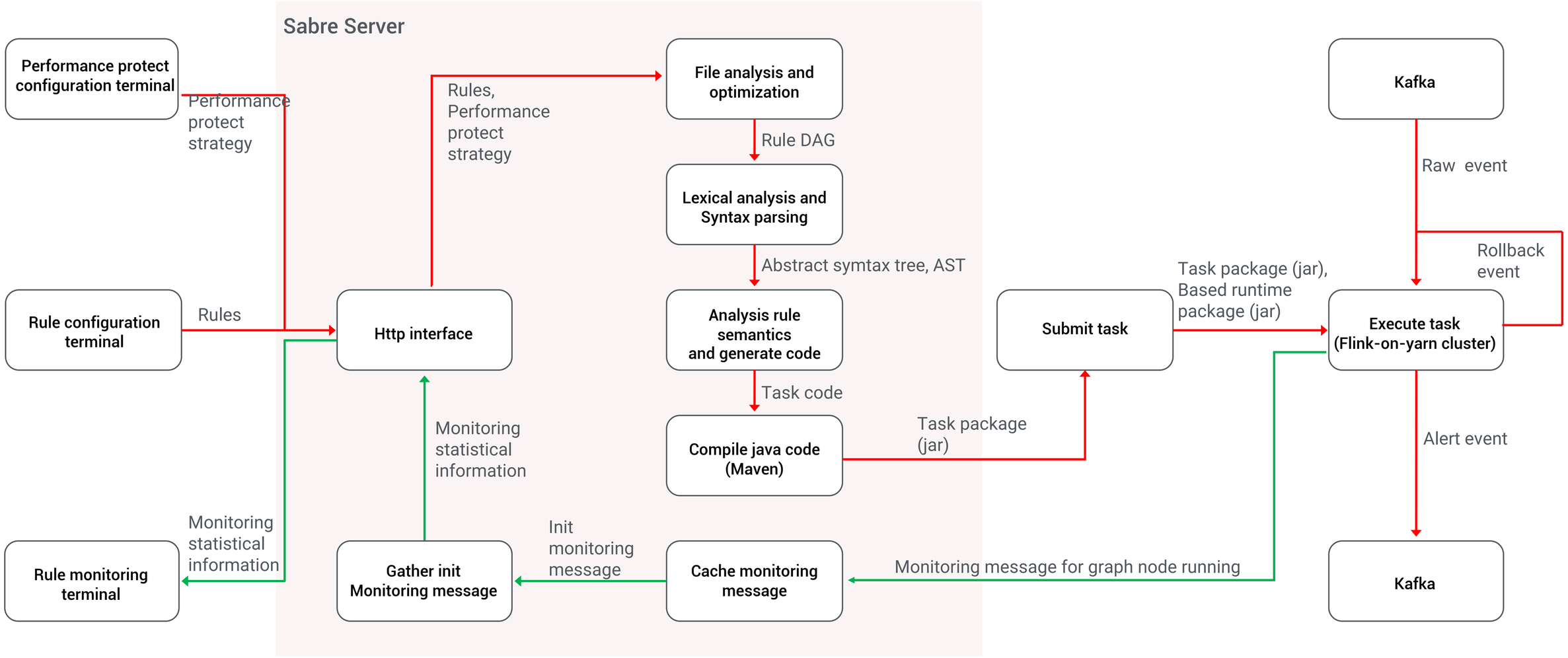
Technology Architecture

02

- 整体架构
  Overall architecture
- 组件依赖与版本兼容
  Component dependencies and version compatibility
- 规则与EPL设计
  Design of the rules and EPL
- 性能优化
  Performance optimization

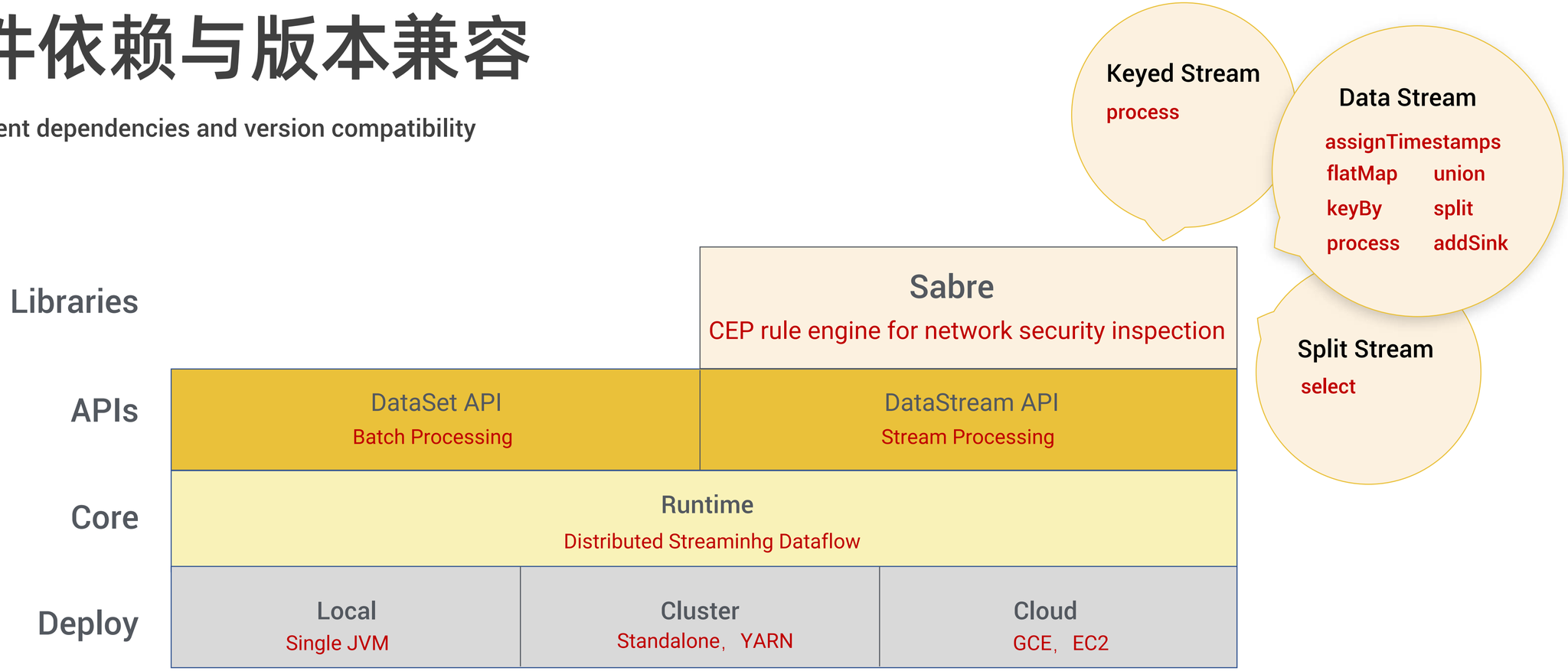
整体框架

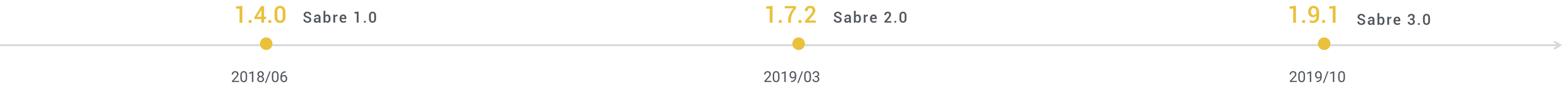
Overall architecture



组件依赖与版本兼容

Component dependencies and version compatibility





Flink 版本兼容
Compatible Flink versions

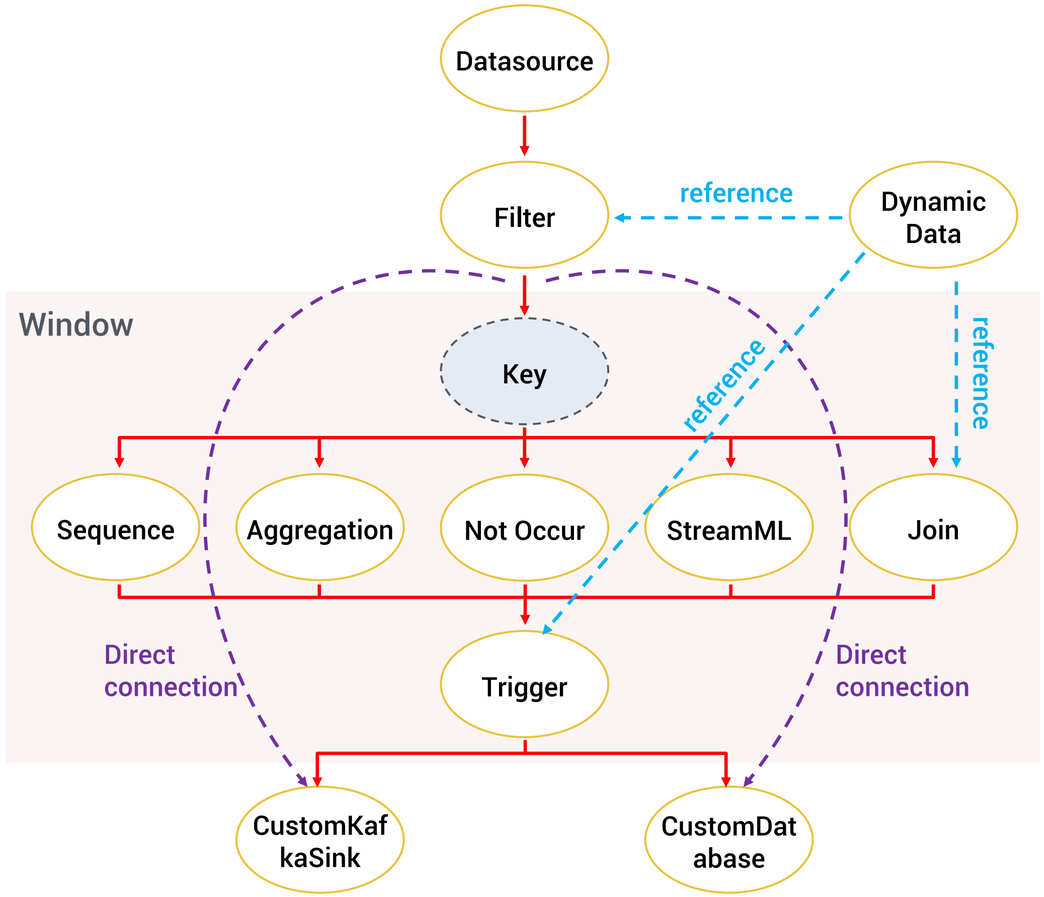
规则与EPL设计

Design of the rules and EPL

Flink	Sabre	Compare
SourceFunction	Datasource	Implement
FilterFunction	Filter	Same
Window 💡	CustomWindow 💡	Optimization 💡
GroupBy/Distinct	Key	Same
Join	Join	Same
Flink CEP 💡	Suquence 💡	Optimization 💡
AggregateFunction	Aggregation	Same
	Notoccur 💡	New 💡
Flink MachineLearning	StreamMachineLearning 💡	New 💡
	Trigger 💡	New 💡
SinkFunction	CustomKafkaSink	Implement
SinkFunction	CustomDatabase	Implement
	DynamicData 💡	New 💡

规则与EPL设计

Design of the rules and EPL




```
dynamicDataObject =
dynamicData(type="database",target="postgresql",data.type="{\"columns\":[{\"name\":
\"host\",\"value.type\": \"string\",\"encrypt\":{\"type\":\"ngsoc\",
\"key\":\"ngsocpassword\"}}]}",path="postgresql.host:5432/ngsoc",user="postgres",pa
ssword="abcd", selector="select host from ioc_table",interval=21600,
notice.zookeeper.connect="10.11.11.1:2181");

read(target="kafka", bootstrap.servers="10.11.11.1:9092",
zookeeper.connect="10.11.11.1:2181", topic="http_flow", group.id="sabreGroupId",
data.type="messagePack", timestamp.type="event", timestamp.field="occur_time",
timestamp.format="yyyy-MM-dd HH:mm:ss:SSS", timestamp.unit="millisecond")
.filter(select * where (rowMatch(object("dynamicDataId"),
match(pattern=rowValue("host"), data=field("domain"), ignorecase=false),
collect="first")) and (field("dns_type") == string(decodeBase64("MQ==")))))
.refrence(dynamicDataObject, "dynamicDataId")
.window(type="slide", stream="union", lateness=0, days=0, hours=0, minutes=10,
seconds=0, milliseconds=0)
.groupBy(sip,dip)
.count(id="aggregationId")
.trigger(object("aggregationId") >= 10)
.sink(target="kafka",bootstrap.servers="10.11.11.1:9092",topic="sabreResult",
data.type="json", event.type="matched", timestamp.type="process",
timestamp.format="yyyy-MM-dd HH:mm:ss:SSS");
```

全新的Window

The whole new custom window operator

1 **实时触发，即刻匹配**

Real-time trigger and match alert event

2 **匹配不重复**

Without repetition alert event

3 **乱序纠正（时间槽、基于流式状态机的局部计算、超时通知）**

Out-of-order correction (time slot, local calculation based on flow state automation engine, timeout notification)

4 **实时资源（CPU/内存）/状态监控**

Real-time hardware resource(CPU, memory) and status monitoring

5 **流量控制，更好地保护下级应用**

Flow control to protect subsequent applications

6 **资源（CPU/内存）保护**

Self-protection of hardware resources(CPU, memory)

便捷的Sequence

The refreshing and convenient sequence operator

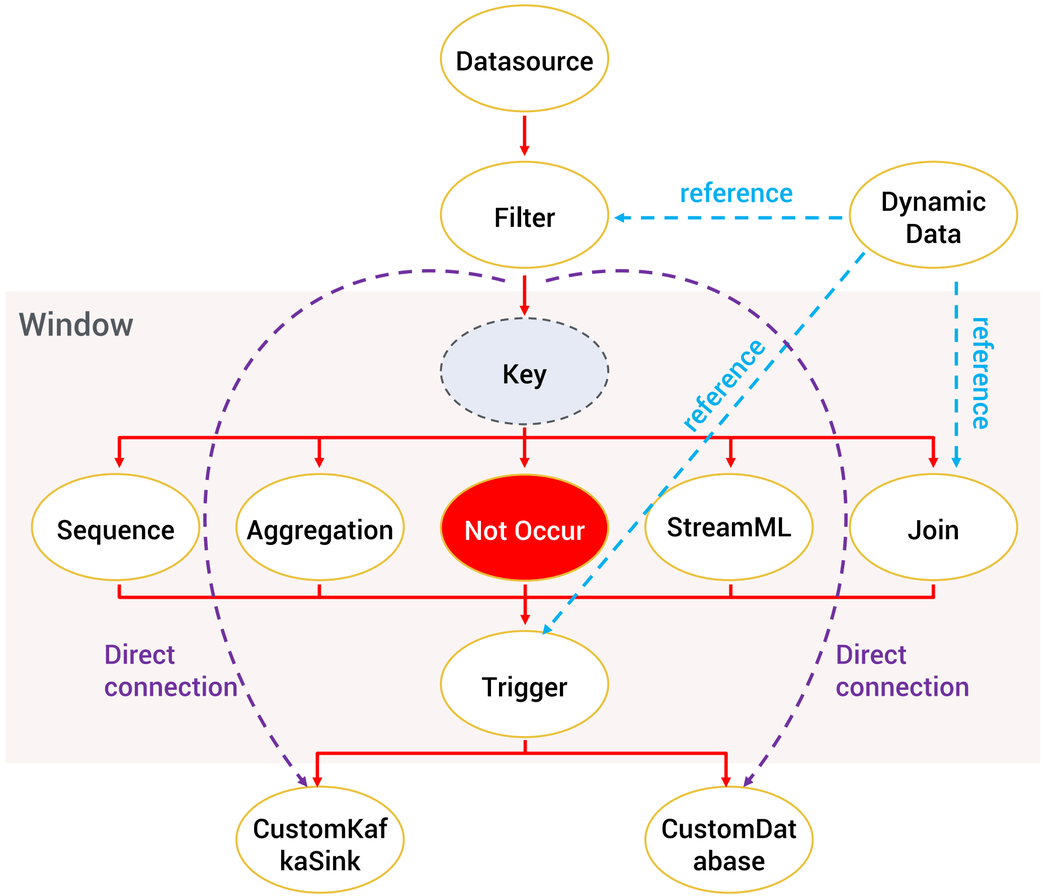
```
Pattern<Event, ?> pattern = Pattern.<Event>begin("start").where(
        new SimpleCondition<Event>() {
            @Override
            public boolean filter(Event event) {
                return event.getId() == 42;
            }
        }
    ).next("middle").subtype(SubEvent.class).where(
        new SimpleCondition<SubEvent>() {
            @Override
            public boolean filter(SubEvent subEvent) {
                return subEvent.getVolume() >= 10.0;
            }
        }
    ).followedBy("end").where(
         new SimpleCondition<Event>() {
            @Override
            public boolean filter(Event event) {
                return event.getName().equals("end");
            }
         }
    );
```

https://ci.apache.org/projects/flink/flink-docs-release-1.9/dev/libs/cep.html#getting-started

Type	Object Id	EPL
Filter	startFilter	select * where field("id") == 42
Filter	middleFilter	select * where field("volume") >= 10.0
Filter	endFilter	select * where field("name") =="end"
Sequence	sequenceId	sequence(alias="startFilter as A, middleFilter as B, endFilter as C", regular.expression="AB.*C")

强化的NotOccur

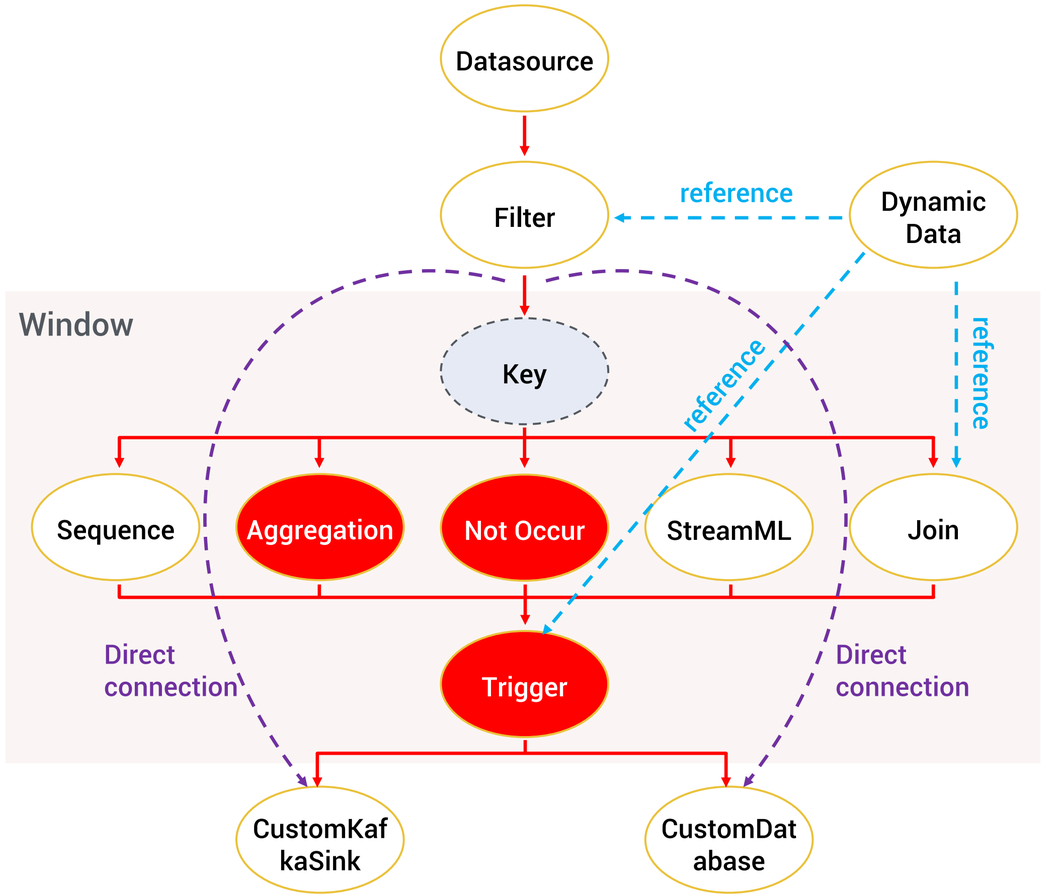
The extended not occur operator



Type	EPL	Trigger	Quantity
notOccur	notOccur(time.range.ms=30000, not.occur.event="singleNotOccurFilterId", time.trigger.interval.ms=10000)	Time	∞
notOccurAll	notOccurAll(time.range.ms=30000, time.trigger.interval.ms=10000)	Time	∞
notOccurAfter	notOccurAfter(time.range.ms=30000, occur.event="occurFilterId", not.occur.event="notOccurFilterId")	Time	1
notOccurBefore	notOccurBefore(time.range.ms=30000, occur.event="occurFilterId", not.occur.event="notOccurFilterId")	Event	1

全局的Trigger

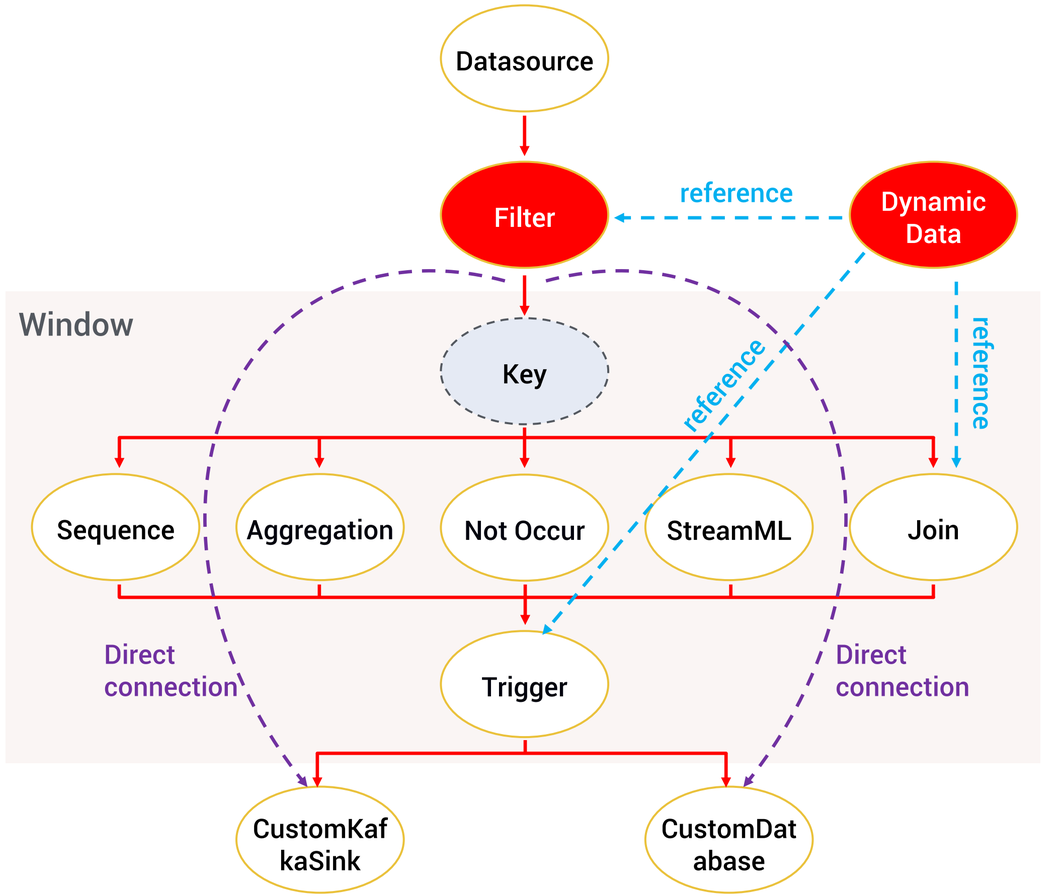
The global trigger operator



Type	Object Id	EPL
Aggregation	aggregationId	count(id="aggregationId")
NotOccur	notOccurId	notOccur(time.range.ms=30000, not.occur.event="filterId", time.trigger.interval.ms=10000)
Trigger	triggerId	object("aggregationId") >= 10 and object("notOccurId")

实用的Dynamic Data

The practical dynamic data operator



Type	Object Id	EPL
Dynamicdata	dynamicdataId	dynamicData(type="database",target="postgresql",data.type="{\"columns\":[{\"name\":\"host\",\"value.type\":\"string\",\"encrypt\":{\"type\":\"AES256\",\"key\":\"sabre\"}}]}",path="postgresql.host:5432/sabredb",user="postgres",password="abcd", selector="select host from ioc_table",interval=21600, notice.zookeeper.connect="10.11.11.1:2181")
Filter	filterId	select * where (rowMatch(object("dynamicDataId"), match(pattern=rowValue("host"), data=field("domain"), ignorecase=false), collect="first")) and (field("dns_type") == string(decodeBase64("MQ=="))))

流式统计与机器学习StreamML

The completely new streaming statistics and machine learning operator

Product business	Baseline	UEBA (User and entity behavior analysis)	
Application scenario	Abnormal behavior detection	Time series anomaly detection	Cluster analysis of group
Machine learning	Statistical learning method	Sequence analysis method	Cluster analysis algorithm

性能优化

Performance optimization

1 全局组件（数据源、动态表）引用优化

Global component (e.g., Datasource, DynamicData) reference optimization

2 流式状态机引擎

Flow state automation engine

3 大规模IP匹配引擎

Large scale IP matching engine

4 大规模串匹配引擎

Large scale string matching engine

5 表计算优化（构建最优计算数据结构，避免全表扫描）

Table evaluation expression optimization
(building the optimal data structure to avoid scanning the entire table)

6 自定义Window，实时输出无重复、无遗漏告警

Custom window operator, support for out-of-order correction,
real time output without repetition and omission alert event

7 图上字段自动推导，优化事件结构

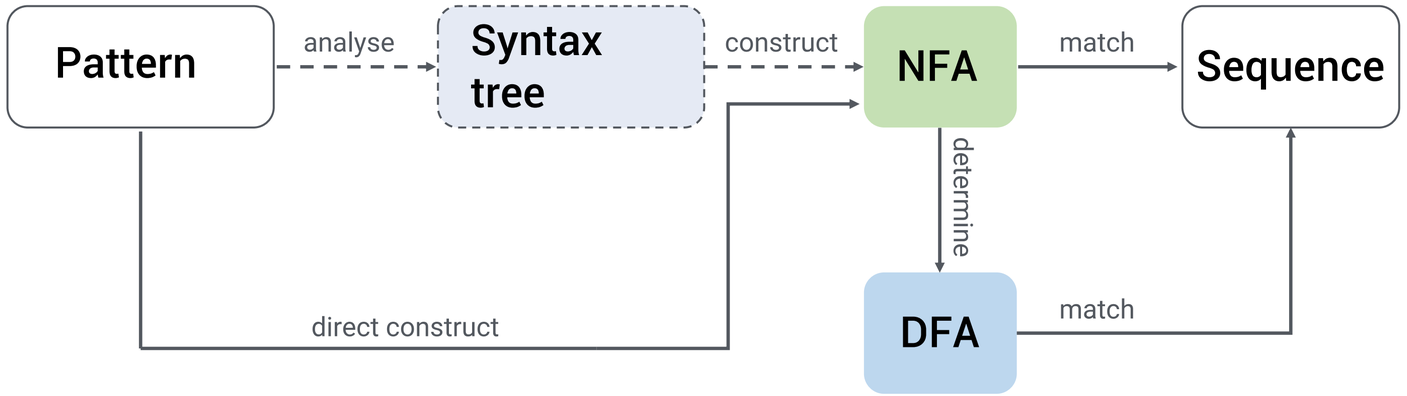
Automatic derivation of fields on the graph to optimize event structure

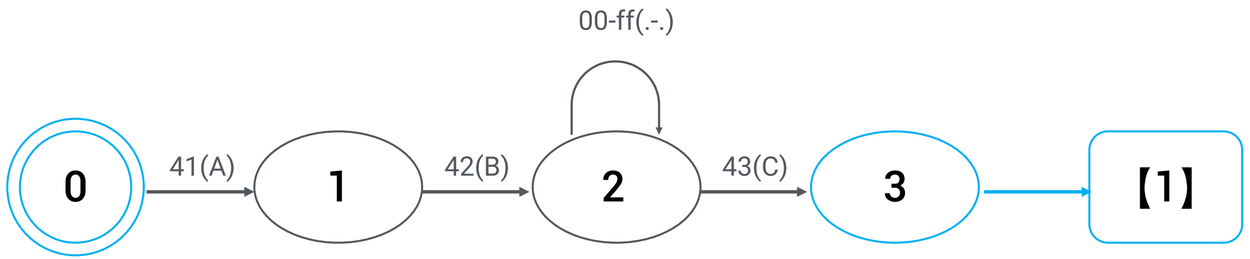
8 图上数据分区自动推导，优化流拓扑

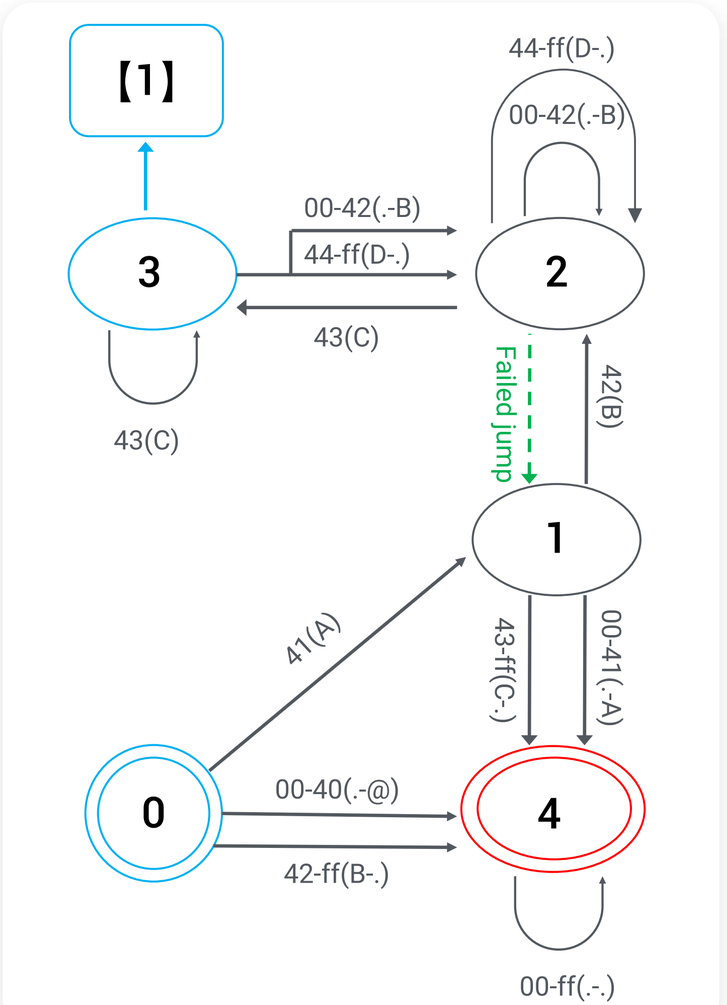
Automatic derivation of data partitions to optimize ExecutionGraph

流式状态机引擎

The streaming state machine engine

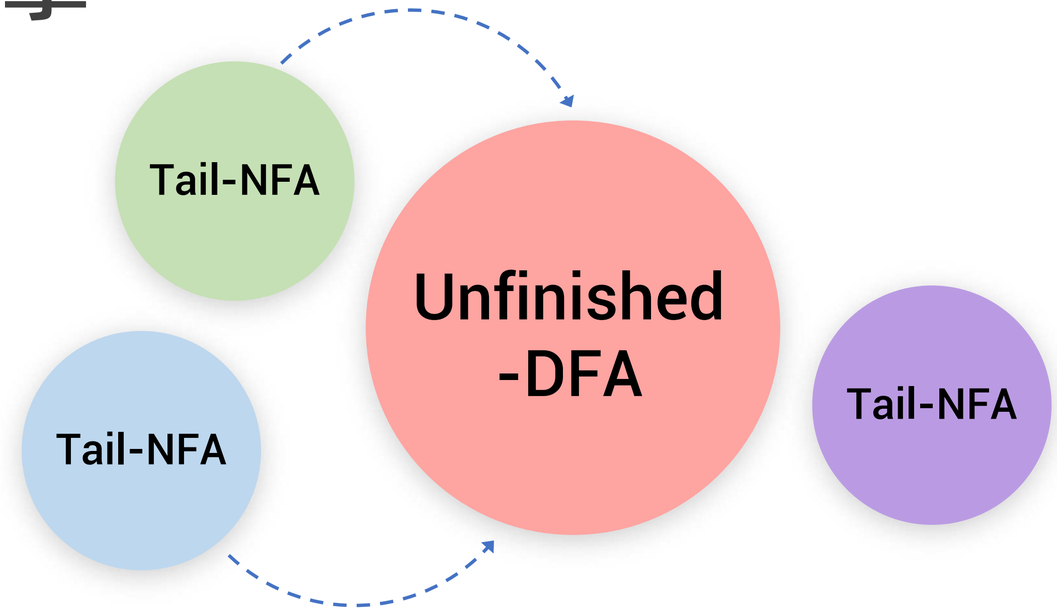


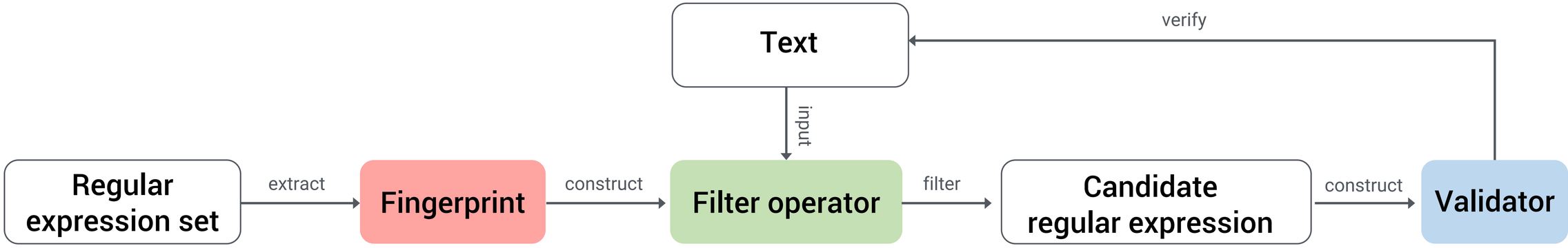




大规模正则引擎

The large scale regular expression





产品及运维

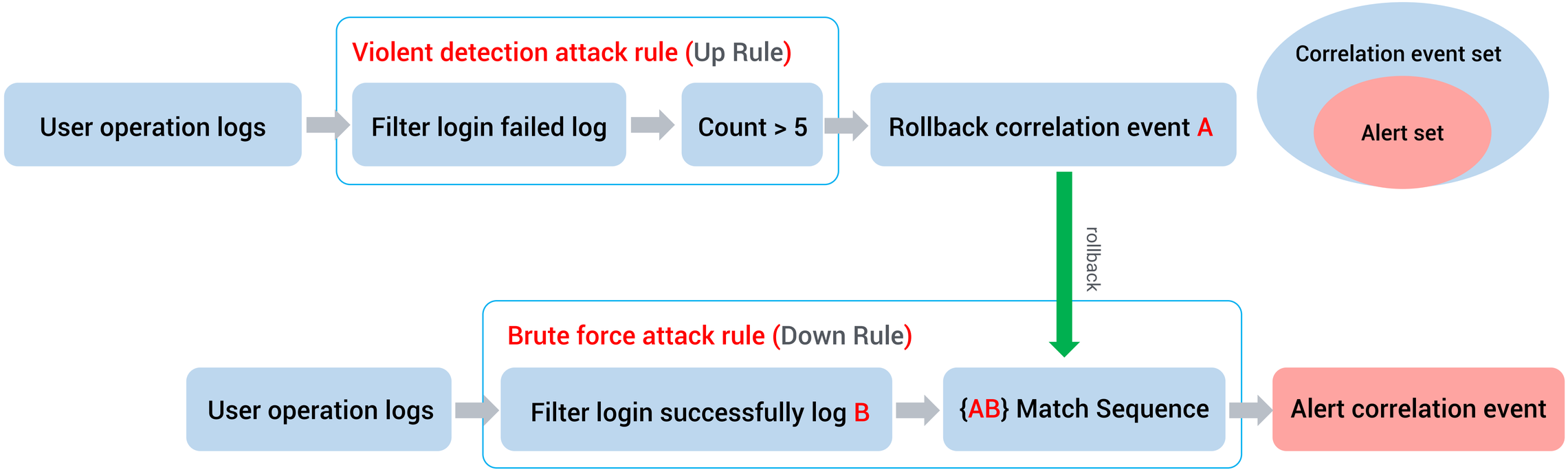
Product and operation and maintenance

03

- 多级规则
  Multi-level rules
- 服务化/多租户/高可用
  Servicisation/Multi-user/Highly available
- 规则级的状态/资源监控
  Rule-level status and resource monitoring
- 整体系统保护
  Overall system protection

多级规则

Multi-level rules



服务化/多租户/高可用

Servicisation/Multi-user/Highly available

1 **微服务**

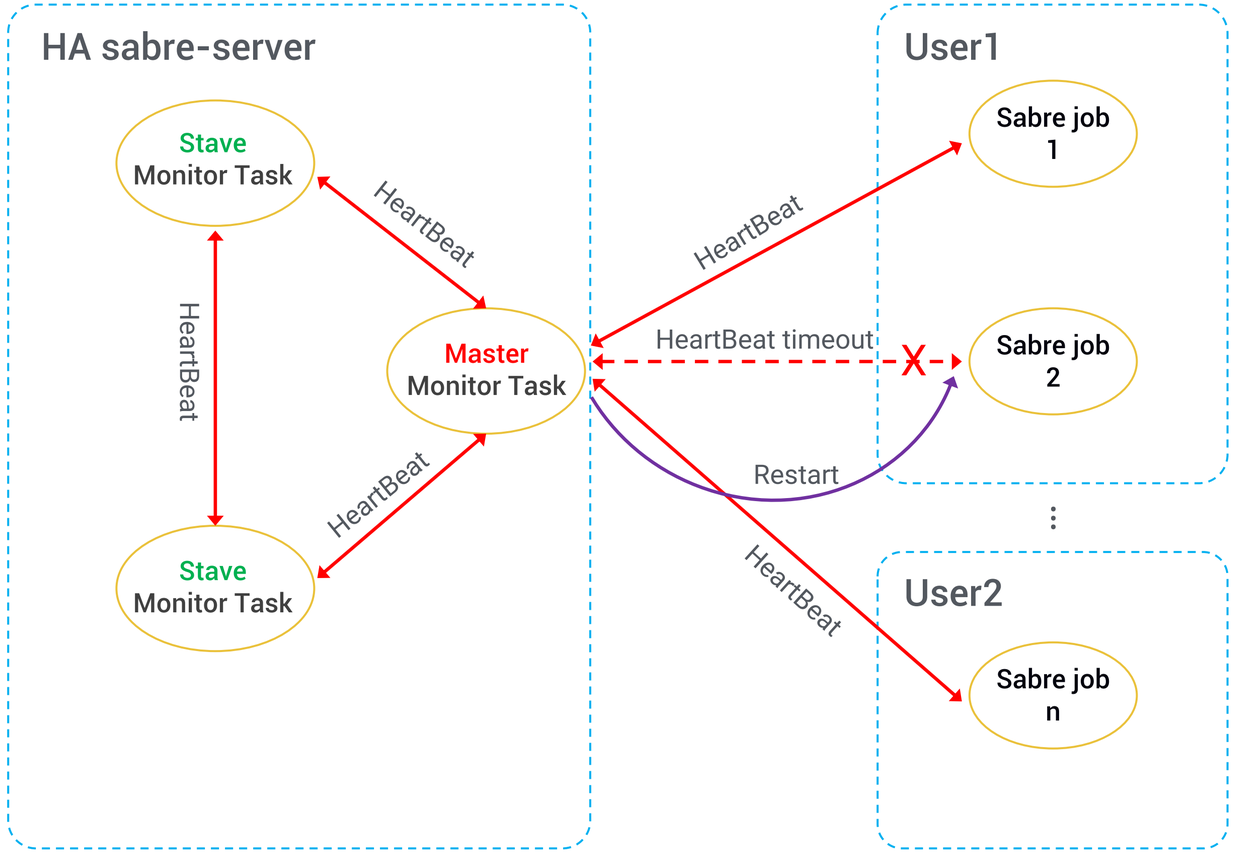
Micro-service

2 **多用户，多任务**

Multi-user, multi-task

3 **实时监控，稳定运行**

Real-time monitoring, stable operation



规则级的状态/资源监控

Rule-level status and resource monitoring

1 **分布式监控**

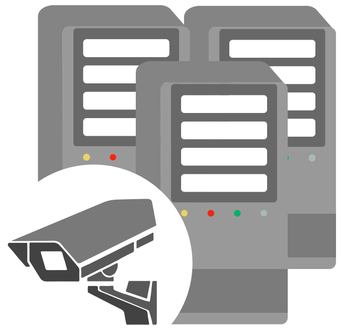
Distributed monitoring system

2 **提供三级（用户->任务->规则）分布式监控能力**

Capability of the three level distributed monitoring, user -> task -> rule

3 **吞吐总量、 EPS、 CPU、内存**

Throughput, EPS, CPU, memory



规则序号	规则名	规则类型	模版类型	告警动作	启用状态	运行状态	生成结果	告警计数	日志计数	内存占用/MB	CPU利用率
426	预置-流量-特定端口扫描	端口扫描	统计规则		启用	运行中	告警	0	956913128	23	0.0444%
474	预置-流量-端口漫扫描	端口扫描	统计规则		启用	运行中	告警	0	956913128	23	0.0444%
434	预置-流量-地址扫描	地址扫描	日志关联规则		启用	运行中	告警	0	956913128	23	0.0444%
405	预置-内部主机向外部发送	可疑事件	统计规则		启用	运行中	告警	0	510552556	23	0.0599%

整体系统保护

Overall system protection

流量控制

流量控制，防止大量告警，更好地保护下级应用。

Flow control：prevent the generation of a large amount of alarms to protect subsequent applications.

自我保护

内存及cpu自我保护，实时监控CPU及内存占用，防止出现CPU长期过高及内存OOM问题。

Self-protection of memory and CPU：monitor CPU and memory usage in real time to prevent long-time over high CPU utilization and memory OOM problems.

```
memoryMonitor(
        max.events.count=0,
        max.memory.size=0,
        max.memory.ratio=0.5,
        min.jvm.free.memory.size=2147483648,
        max.jvm.memory.ratio=0.8)

computingMonitor(
        computing.threshold.ns=100000000,
        min.computing.count=5)
```

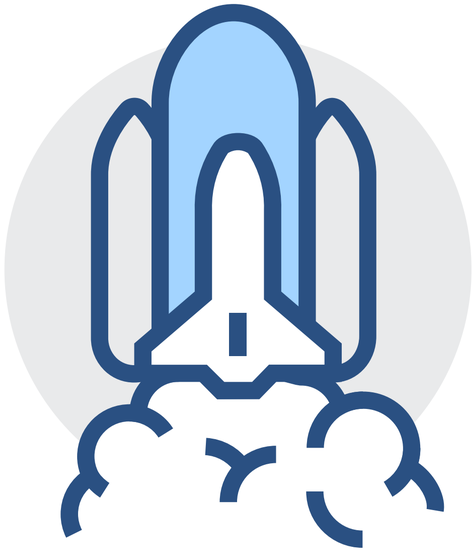
未来发展与思考

Future development and thinking

04

未来发展及思考

Think about the future development



持续进行性能及功能优化

Continuous performance and function optimization

优秀实践回馈社区

Contribute excellent practices to Apache Flink Community


FLINK
FORWARD
THANKS
Q&A