RSQLDB,专为流处理而构建的数据库

倪泽

Apache RocketMQ committer
RSQLDB Maintainer

Agenda

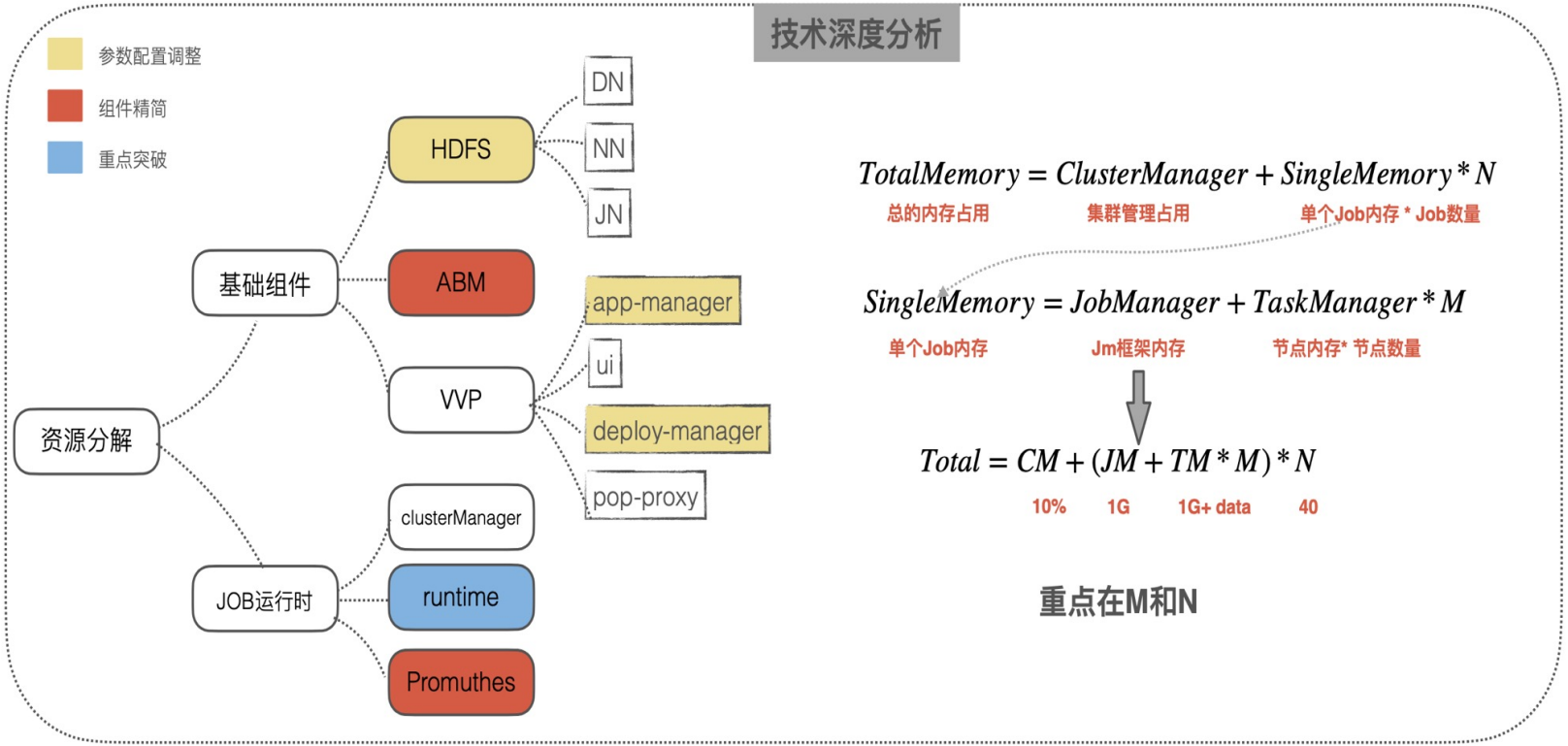
1. 设计背景&目标
2. 架构设计&演进
3. 行业实践

流式计算成本高昂

依赖模块	资源占用(G)
etcd	0.078
Docker	0.514
k8s	3
zookeeper	12
hdfs	44
vvp	11
blink	8
abm	8
abm-write-monitor	2
influxdb	4
pushgateway	2
kube-system	1
总计	95.592

- 依赖多
- 软硬件成本高

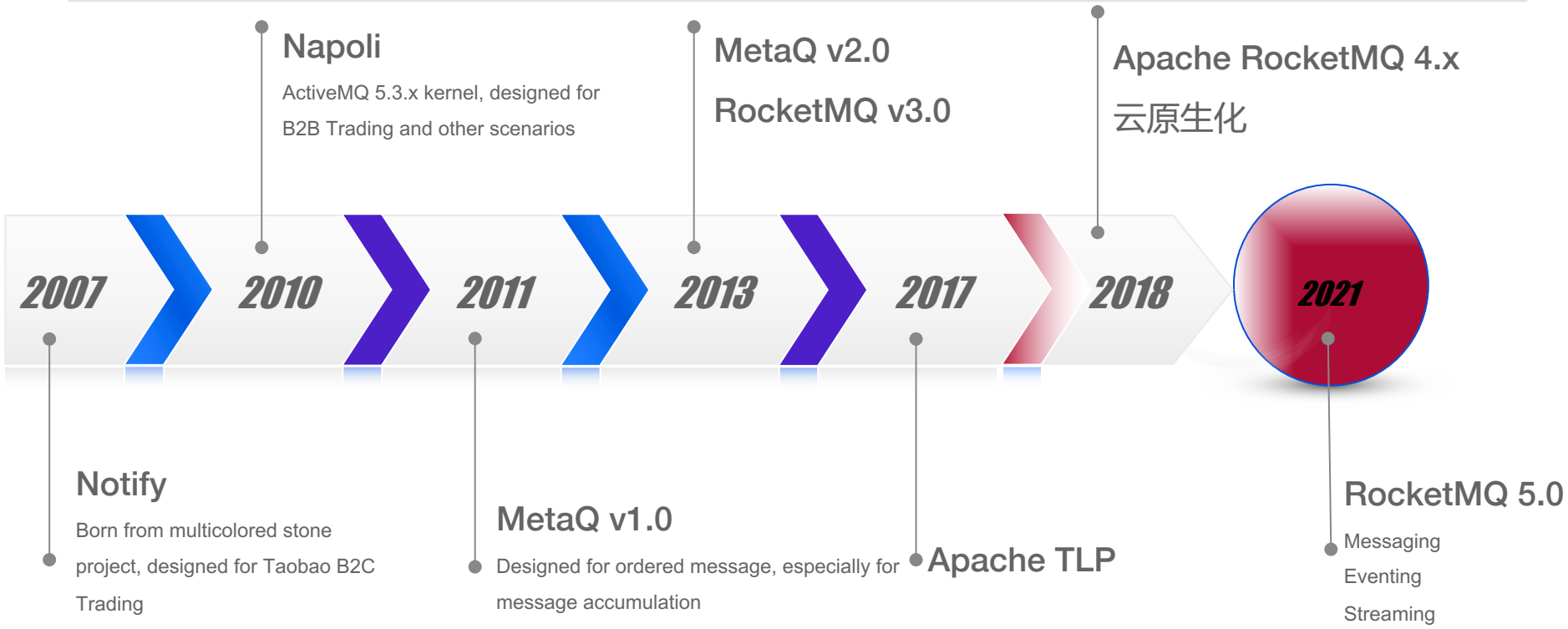
资源使用线性增加


技术深度分析
参数配置调整
组件精简
重点突破
资源分解
基础组件
HDFS
DN
NN
JN
ABM
VVP
app-manager
ui
deploy-manager
pop-proxy
JOB运行时
clusterManager
runtime
Promuthes
$TotalMemory = ClusterManager + SingleMemory * N$
总的内存占用
集群管理占用
单个Job内存 * Job数量
$SingleMemory = JobManager + TaskManager * M$
单个Job内存
Jm框架内存
节点内存* 节点数量
$Total = CM + (JM + TM * M) * N$
10%
1G
1G+ data
40
重点在M和N

专有云面临问题

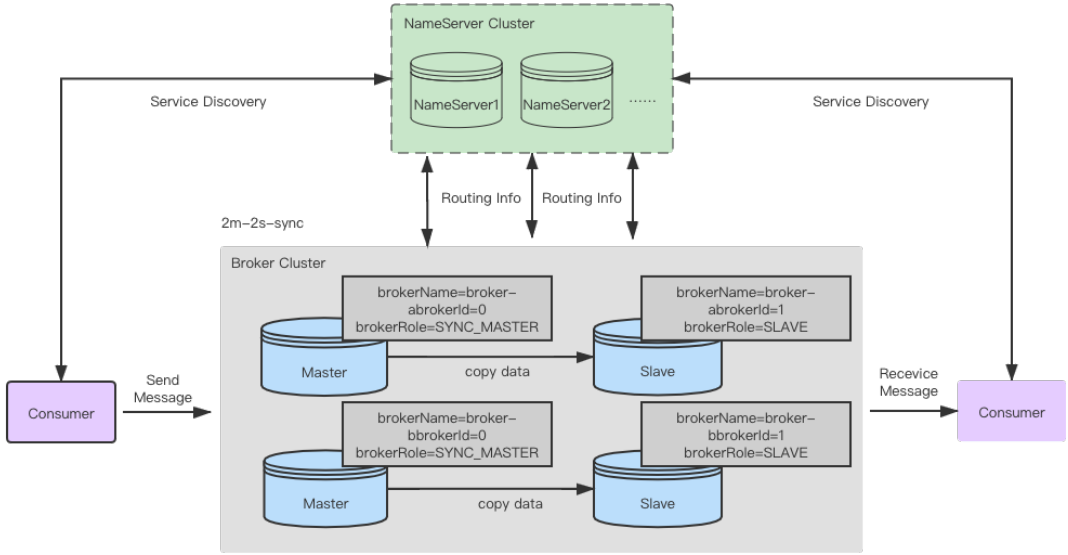
- 用户不愿意单独购买流式计算产品
- 内置流计算产品成本高，用户购买成本上升
- 专有云扩容成本高

RocketMQ 发展历程



业务消息领域首选

微服务业务消息领域首选



让人睡得着觉的消息产品

- 金融级高可靠
- 极简架构、极低运维成本
- 丰富的消息类型
- 高吞吐、低延迟

Jar:

1. nohup sh bin/mqnamesrv &
2. nohup sh bin/mqbroker -n localhost:9876 &

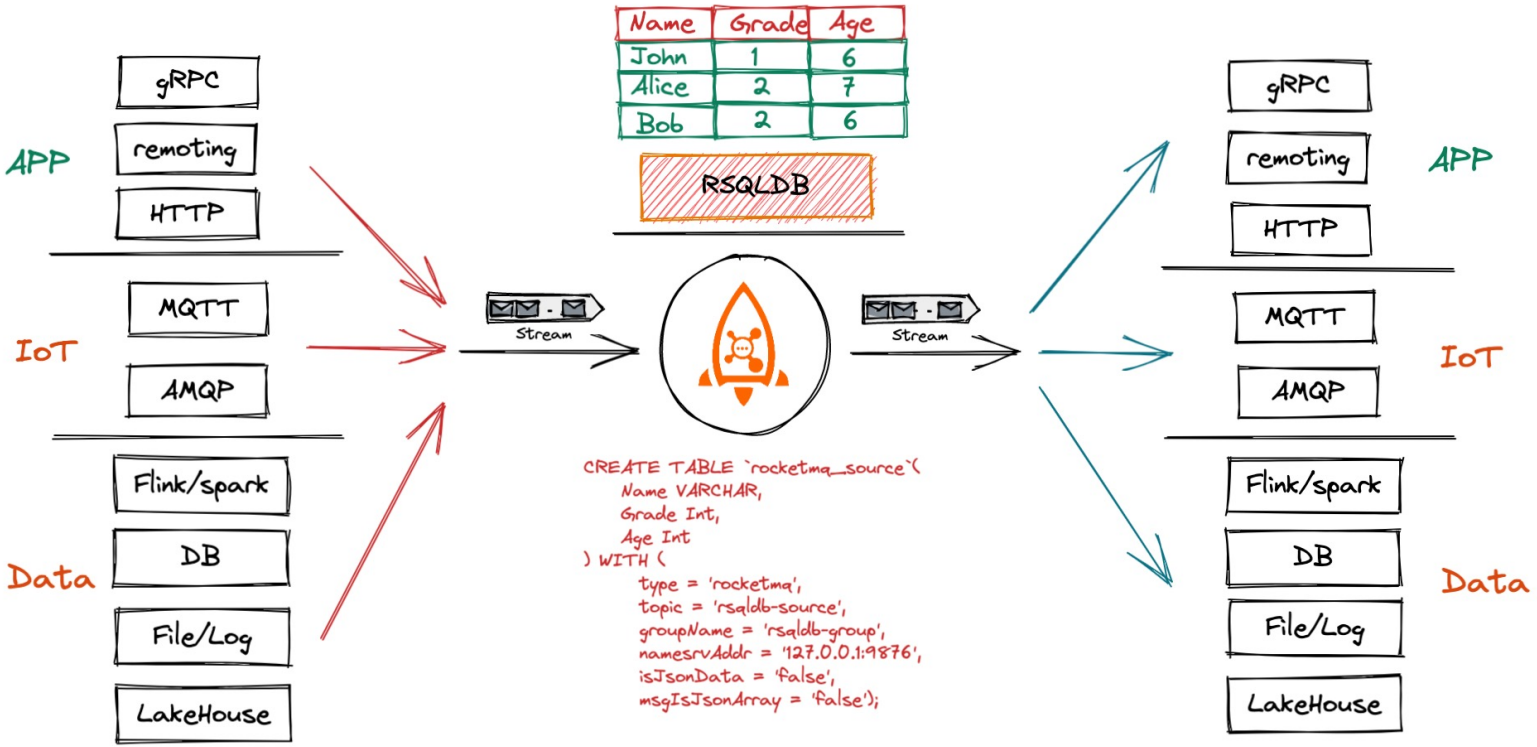
On K8S:

kubectl apply -f example/rocketmq_cluster.yaml

RocketMQ 流计算特性

特性	解决问题
Compact topic	RocketMQ以KV形式存储流计算中间状态
Logic queue	解耦Broker和分区，Broker阔所容不影响分区数量
Schema	增加数据格式，完善数据治理
Batch	批量发送，提升效率
生产者事务	保障数据消费和生产在一个原子单元完成，pull-process-commit原子，用于实现流计算Exactly-Once

设计目标


Name Grade Age
John 1 6
Alice 2 7
Bob 2 6
RSQLDB
APP
gRPC
remoting
HTTP
IoT
MQTT
AMQP
Data
Flink/spark
DB
File/Log
LakeHouse
Stream
Stream
CREATE TABLE `rocketmq_source`(
Name VARCHAR,
Grade Int,
Age Int
) WITH (
type = 'rocketmq',
topic = 'rsqldb-source',
groupName = 'rsqldb-group',
namesrvAddr = '127.0.0.1:9876',
isJsonData = 'false',
msgIsJsonArray = 'false');
gRPC
remoting
HTTP
APP
MQTT
AMQP
IoT
Flink/spark
DB
File/Log
LakeHouse
Data

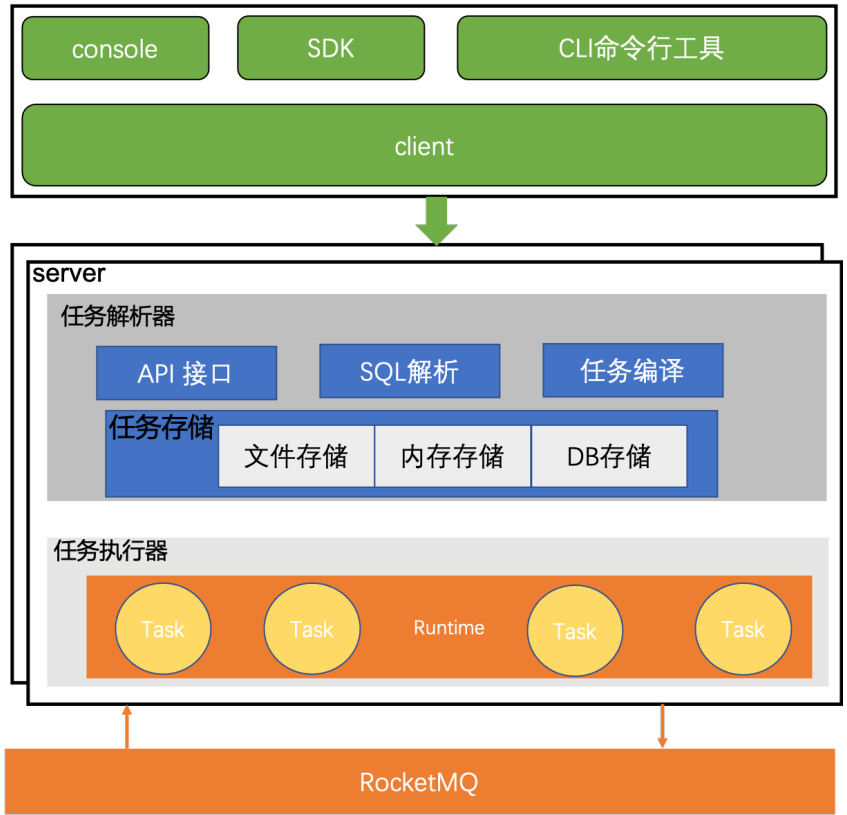
Agenda

1 设计背景&目标

2 架构设计&演进

3 行业实践

技术架构



- C-S架构支持多种客户端提交任务
- 资源利用效率高

SQL功能支持

类别	功能点
DDL语句	RocketMQ数据源表/结果表，Mysql数据源表/结果表
DML语句	INSERT INTO语句，EMIT语句
QUERY语句	SELECT、WHERE、HAVING、GROUP BY、双流JOIN、维表JOIN
内置函数	聚合函数、逻辑函数
窗口函数	滚动窗口、滑动窗口、会话窗口
自定义函数	自定义标量函数（UDF）、自定义表值函数（UDTF）

典型SQL示例

```
CREATE TABLE movie_ticket
(
    `id`                    INT,
    `purchaser_id`          INT,
    `movie_name`            VARCHAR,
    `position`              VARCHAR,
    `time`                  TIMESTAMP,
    primary key (id)
) WITH (
      type = 'rocketmq',
      topic = 'movie_ticket',
      groupName = 'movie_ticket',
      namesrvAddr = '127.0.0.1:9876',
      isJsonData = 'true'
      );
```

```
CREATE TABLE purchaser_dim
(
    `purchaser_id`          INT,
    `name`                  VARCHAR,
    `gender`                VARCHAR,
    `age`                   INT,
    primary key (purchaser_id)
)WITH (
    type = 'db',
    url='jdbc:mysql://xxx',
    userName='',
    password='',
    tableName='purchaser_dim'
     );
```

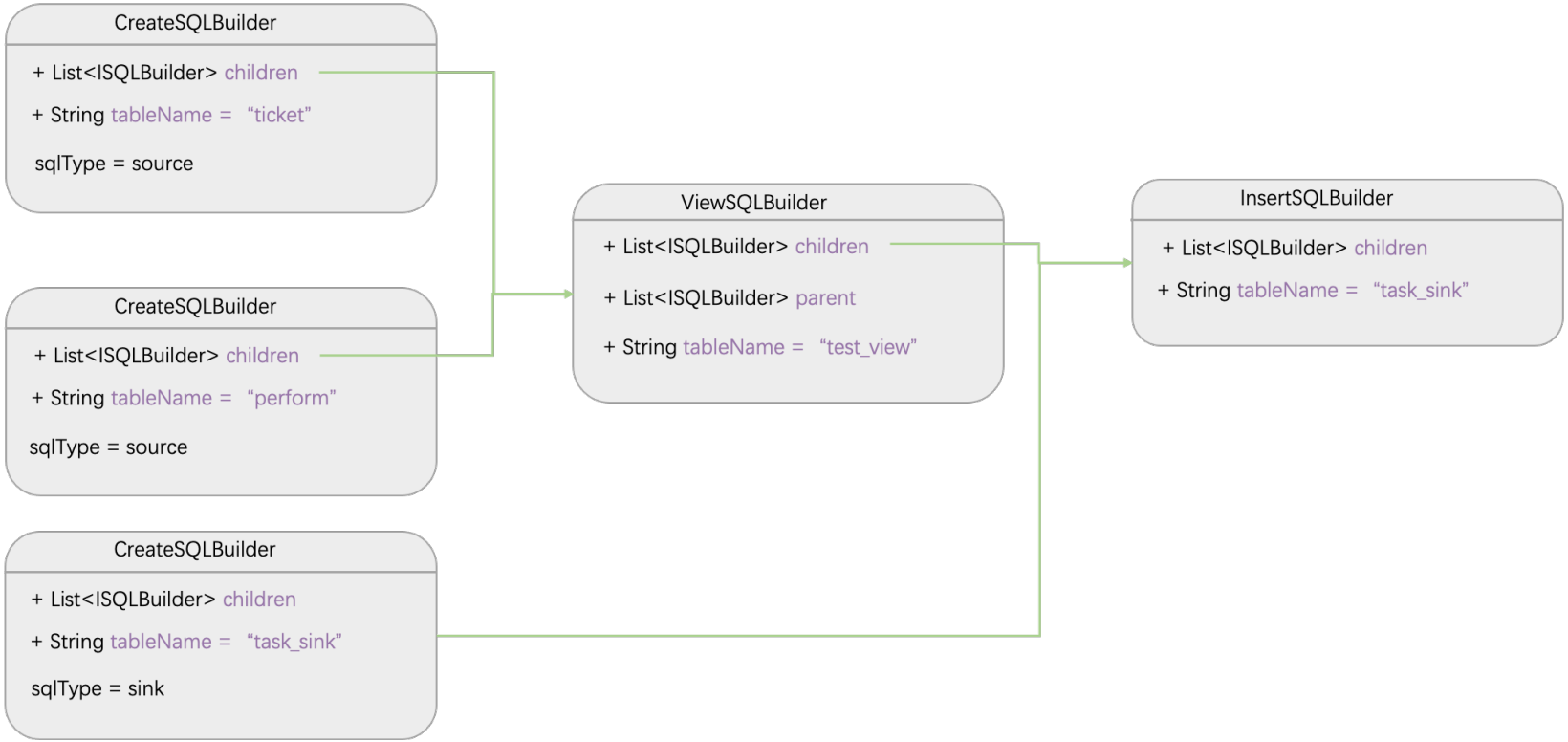
```
CREATE TABLE result_table
(
    `purchaser_id`
INT,
    `name`                  VARCHAR,
    `gender`                VARCHAR,
    `movie_name`            VARCHAR
) WITH (
      type = 'print'
     );
```

```
CREATE VIEW result_view AS
SELECT
    t.purchaser_id          AS purchaser_id,
    pd.name                 AS name,
    pd.gender               AS gender,
    t.movie_name            AS movie_name

FROM movie_ticket as t JOIN purchaser_dim FOR SYSTEM_TIME AS OF PROCTIME() AS pd
ON t.purchaser_id = pd.purchaser_id;
```

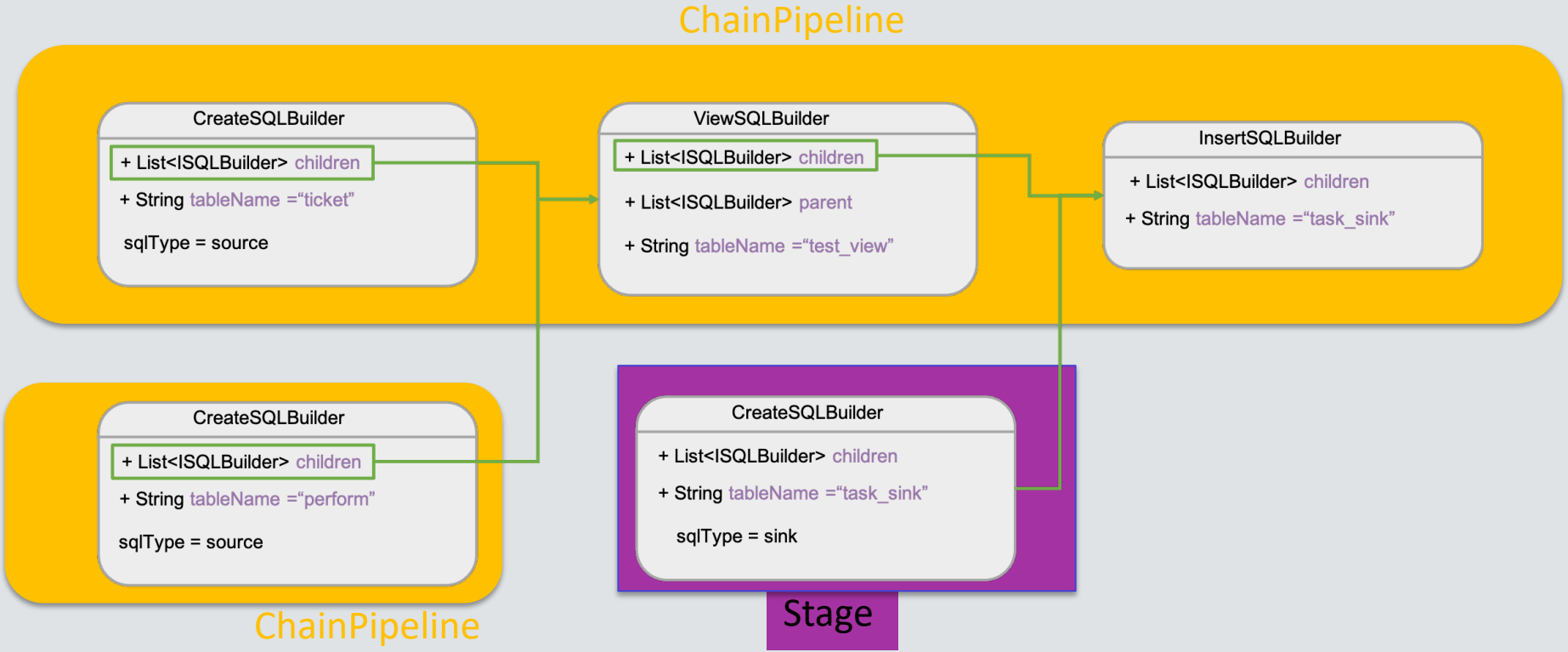
```
INSERT INTO result_table
SELECT *
FROM result_view;
```

任务解析器-表依赖

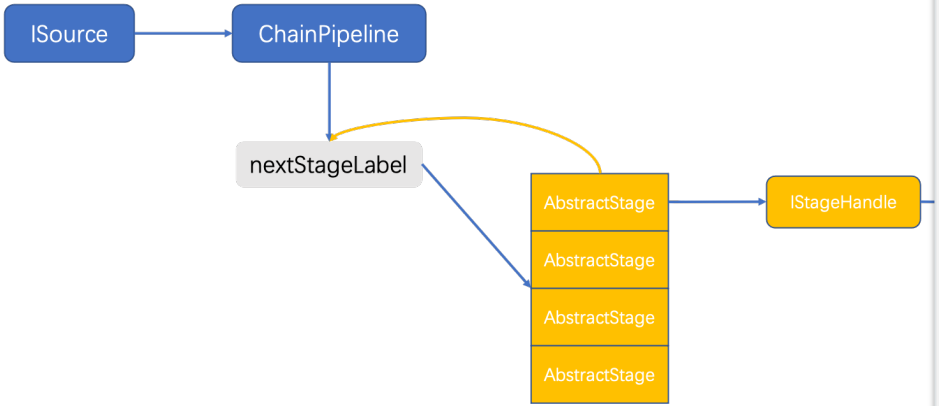

CreateSQLBuilder
+ List<ISQLBuilder> children
+ String tableName = "ticket"
sqlType = source
CreateSQLBuilder
+ List<ISQLBuilder> children
+ String tableName = "perform"
sqlType = source
CreateSQLBuilder
+ List<ISQLBuilder> children
+ String tableName = "task_sink"
sqlType = sink
ViewSQLBuilder
+ List<ISQLBuilder> children
+ List<ISQLBuilder> parent
+ String tableName = "test_view"
InsertSQLBuilder
+ List<ISQLBuilder> children
+ String tableName = "task_sink"

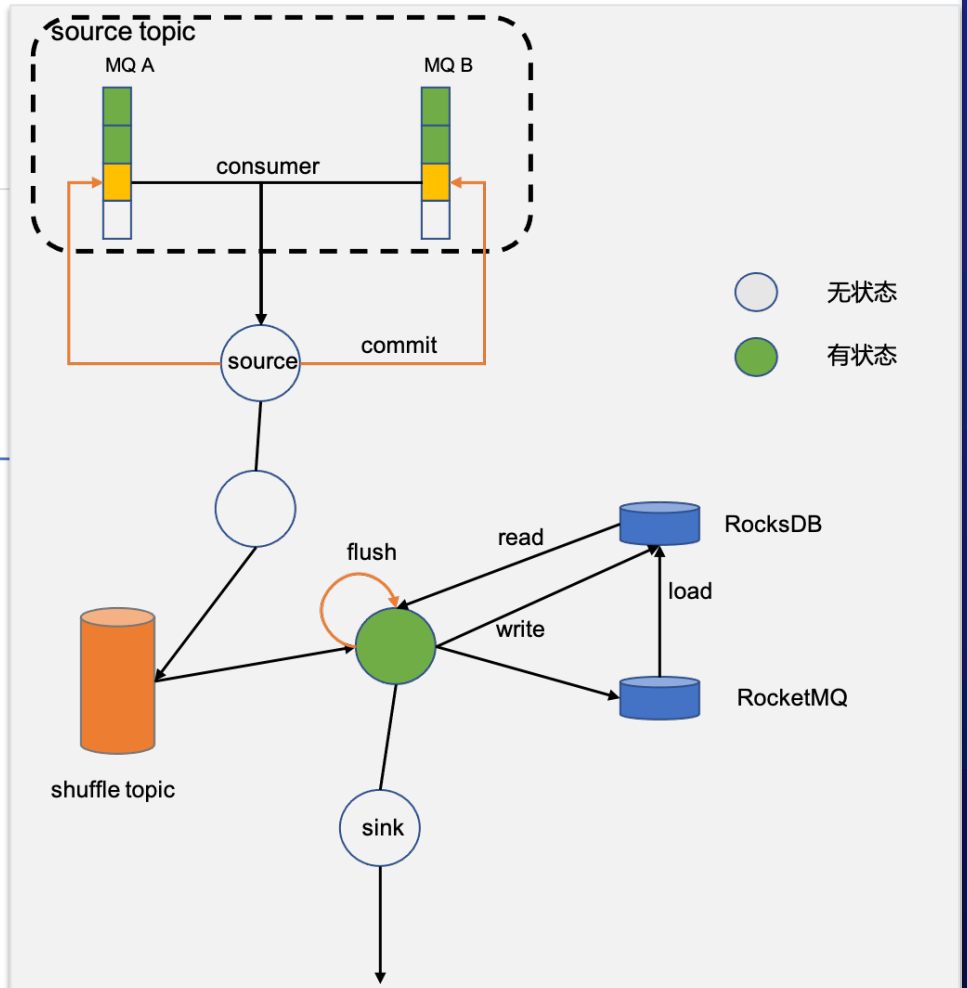
任务解析器-任务核心概念

Task

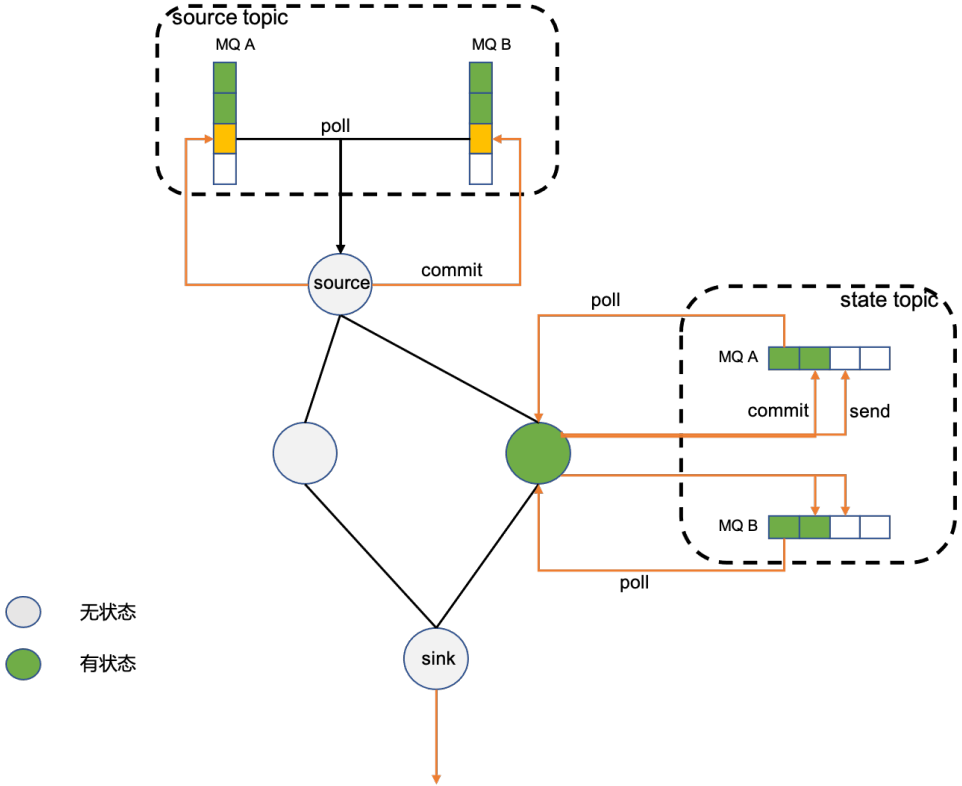


任务执行器-执行拓扑


ISource
ChainPipeline
nextStageLabel
AbstractStage
AbstractStage
AbstractStage
AbstractStage
IStageHandle

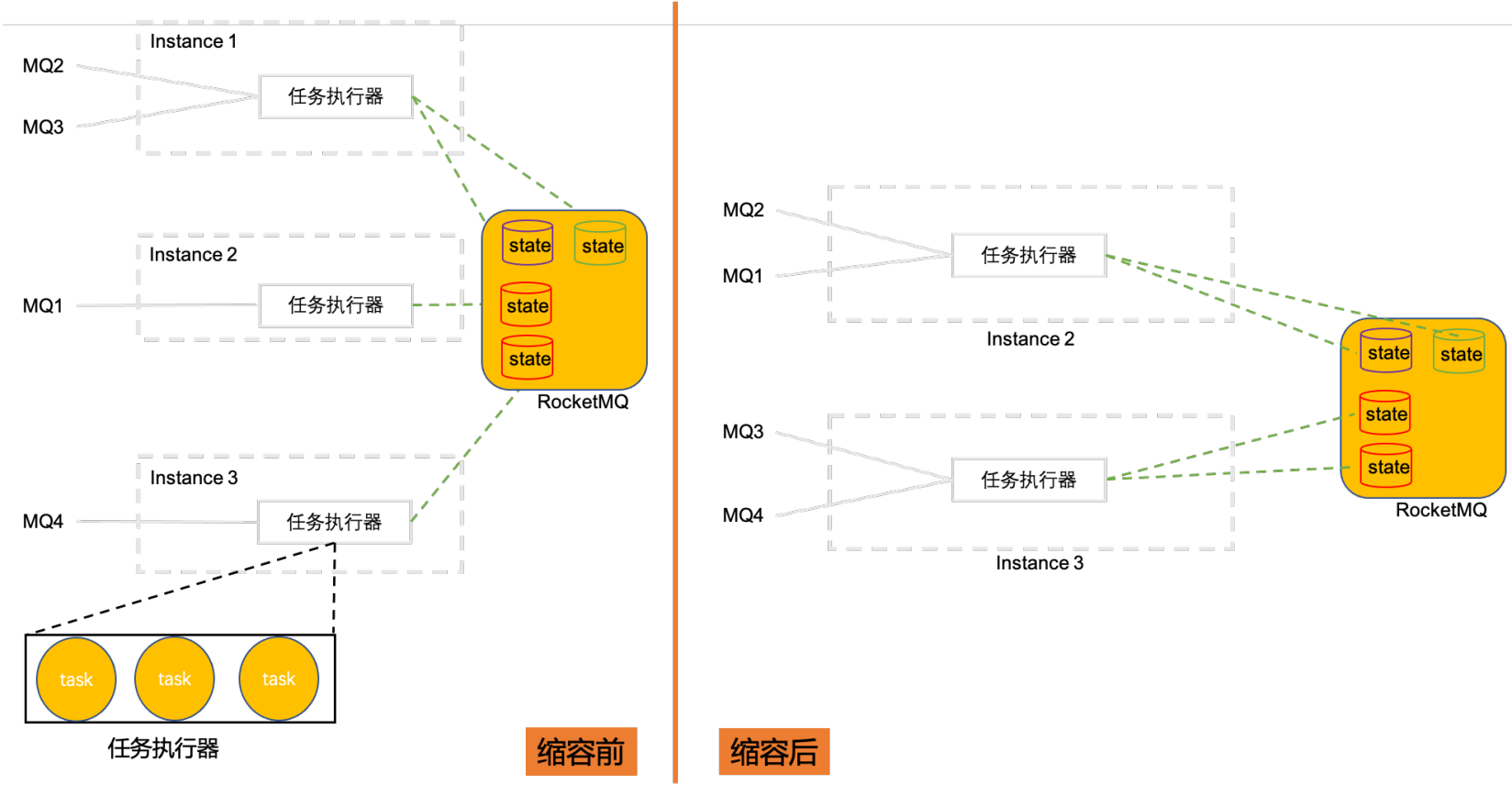

source topic
MQ A
MQ B
consumer
source
commit
无状态
有状态
flush
read
load
write
RocksDB
RocketMQ
shuffle topic
sink

任务执行器-状态存储



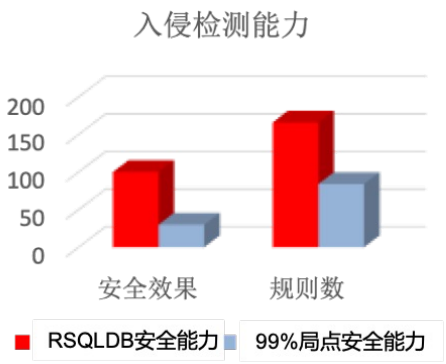
- compact topic作为存储
- pull-process-commit
- 分区发现-加载对应状态到内存

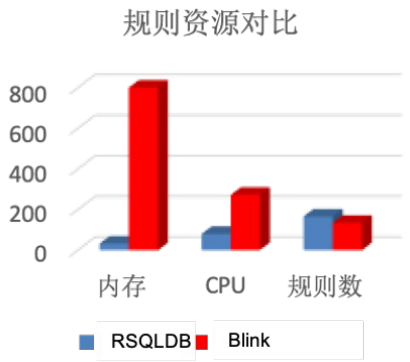
扩缩容


Instance 1
MQ2
MQ3
任务执行器
Instance 2
MQ1
任务执行器
state
state
state
state
RocketMQ
Instance 3
MQ4
任务执行器
task
task
task
任务执行器
缩容前
MQ2
MQ1
任务执行器
Instance 2
MQ3
MQ4
任务执行器
Instance 3
state
state
state
state
RocketMQ
缩容后

优化效果


负载持续增加
入侵规则增加
日志种类、日志量增加
新业务
Bilnk
物理机3台,768G,192C
53条规则
RSQLDB
虚拟机2台,32G,80C
166条规则
Blink
RSQLDB

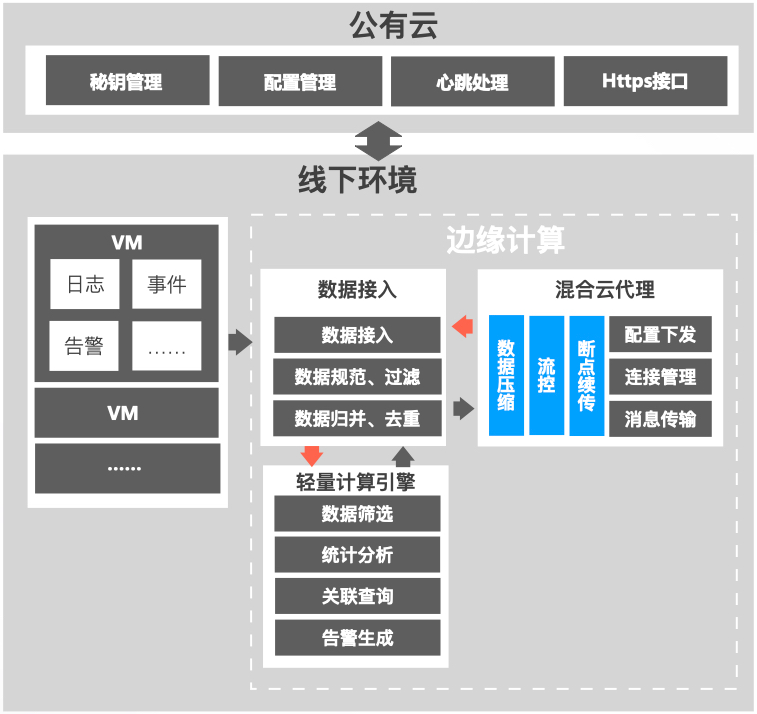

入侵检测能力
200
150
100
50
0
安全效果
规则数
RSQLDB安全能力
99%局点安全能力


规则资源对比
800
600
400
200
0
内存
CPU
规则数
RSQLDB
Blink

Agenda

1. 设计背景&目标
2. 架构设计&演进
3. 行业实践

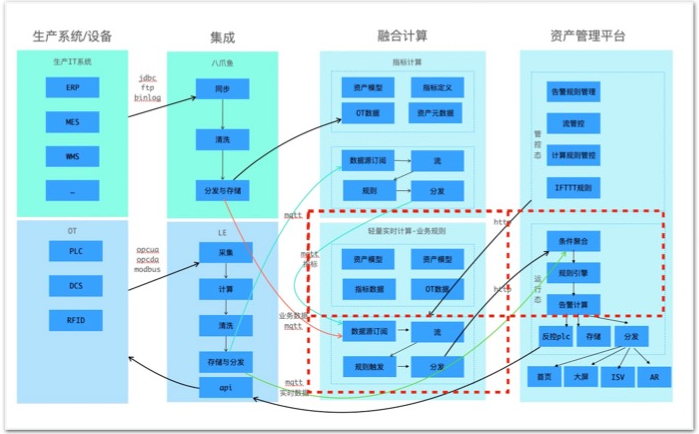
业务落地 | 混合云场景

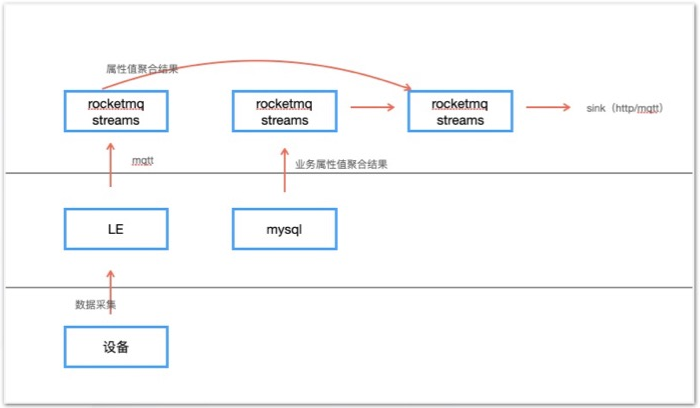


支持混合云部署

- 跟随混合云代理服务，安装在各云的线下环境，由代理服务进行安装部署和拉起，方便服务管理；
- 实时引擎内置的rocketmq，作为混合云代理的系统消息队列， 用于采集aegis上报的各类数据；
- 实时计算引擎负责对采集的各类数据进行处理、检测，最终生成告警，通过固定的topic上报给代理；
- 代理收到实时引擎上报上来的告警数据后，即可通过固定的域名上报到服务端，从而完成各类云告警的汇总；

业务落地 | Iot场景





支持工业智能Iot的健康监控

- 支持一汽厂房设备的数据汇总：目前一汽工厂有1.3w的设备，设置了6.8W个数据采集点， 一共有77w的属性点， 每天大概产生上TB的监控数据；
- 支持基于用户设定的指标告警： 用于可以基于Iot采集的数据，设置各种指标的监控阈值，以及阈值判断的逻辑组合，来生成最终的告警；
- 容器化部署，支持实时引擎在Iot的多种边缘环境上轻松部署，目前已经Iot测试环境部署并启动规则测试；