Flink 资源动态调整及其实践

Self Regulating Stream Processing in Flink

马庆祥
Qingxiang Ma

数据开发资深工程师
Senior data development engineer

FLINK FORWARD # ASIA
实时即未来 # Real-time Is The Future

FLINK FORWARD

Contents
目录

01 背景
Background

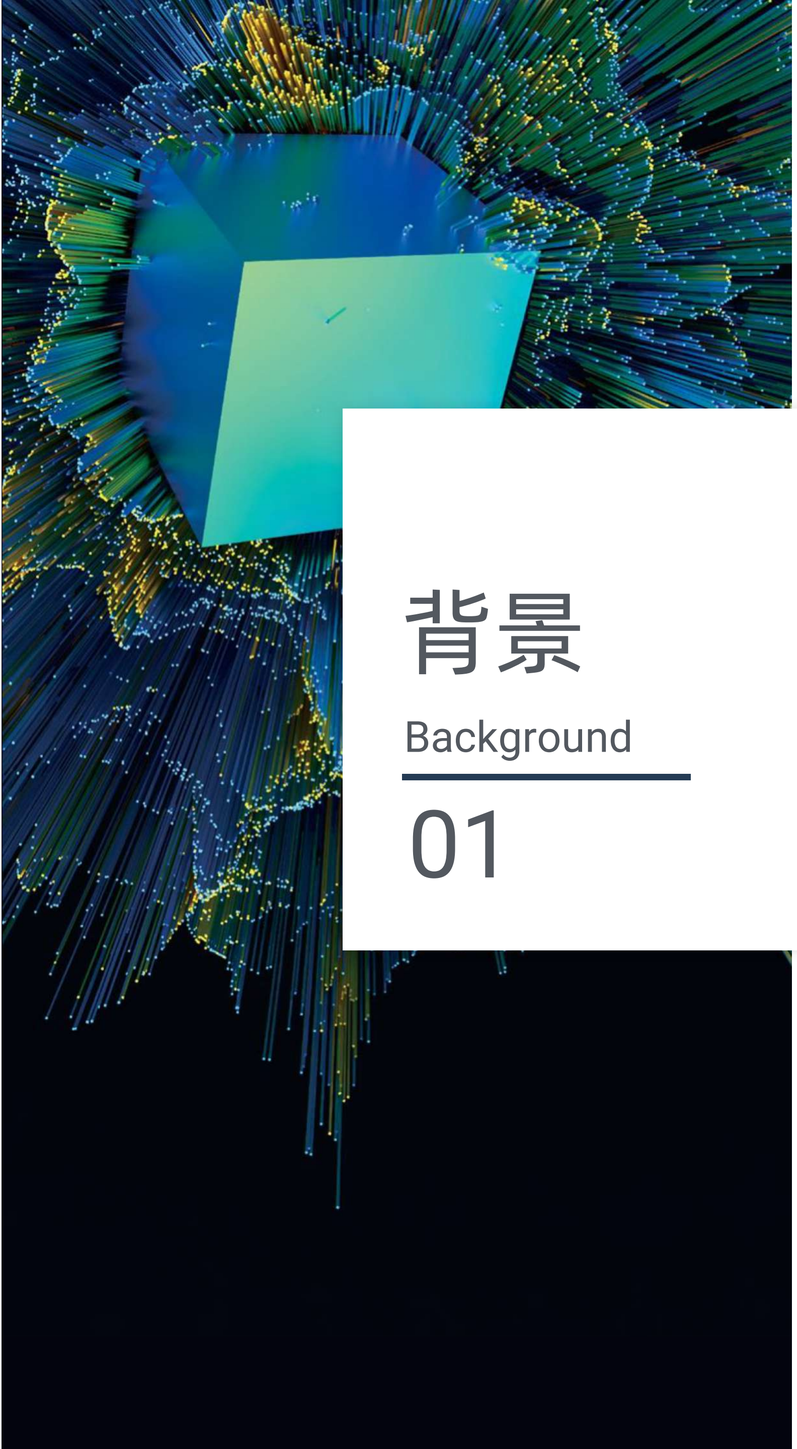
02 QDS 模型
The QDS model

03 生产实践
Production Practices

04 展望
Future Work

关于我们

About us



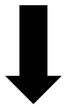
背景

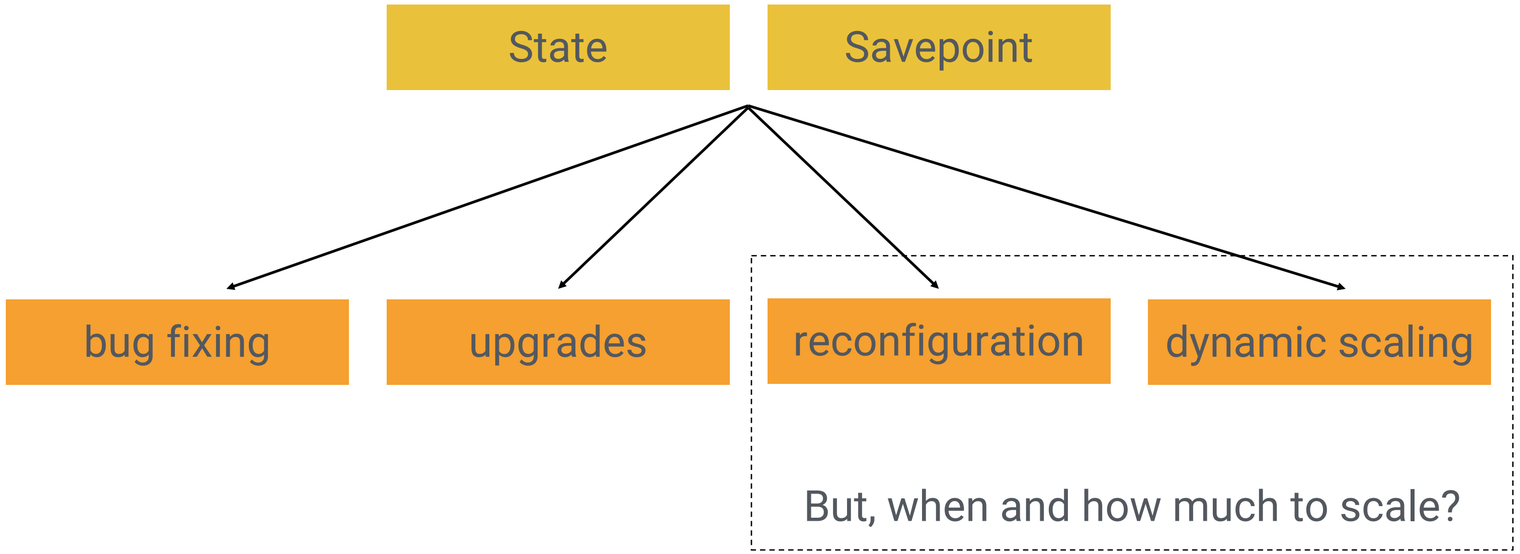
Background

01

背景
Background

Stateful Computations over Data Streams





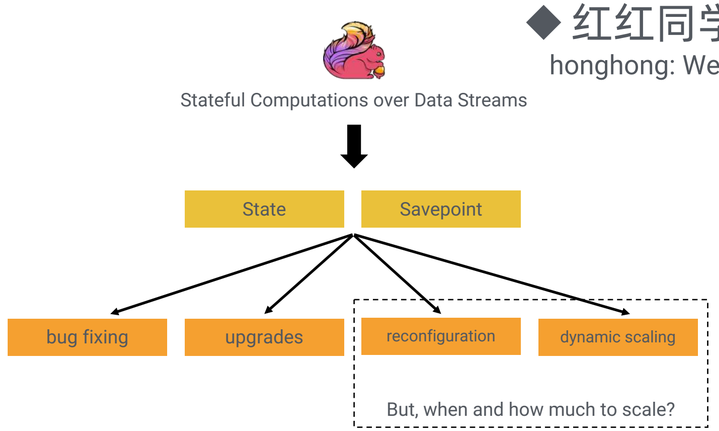
背景
Background

问题：
Problems:

◆ 大白同学：第一次用 Flink 进行业务开发，终于要上线了，如何配置并行度啊？
dabai: I am a newer of Flink, the job is coming online, how to configure parallelism?

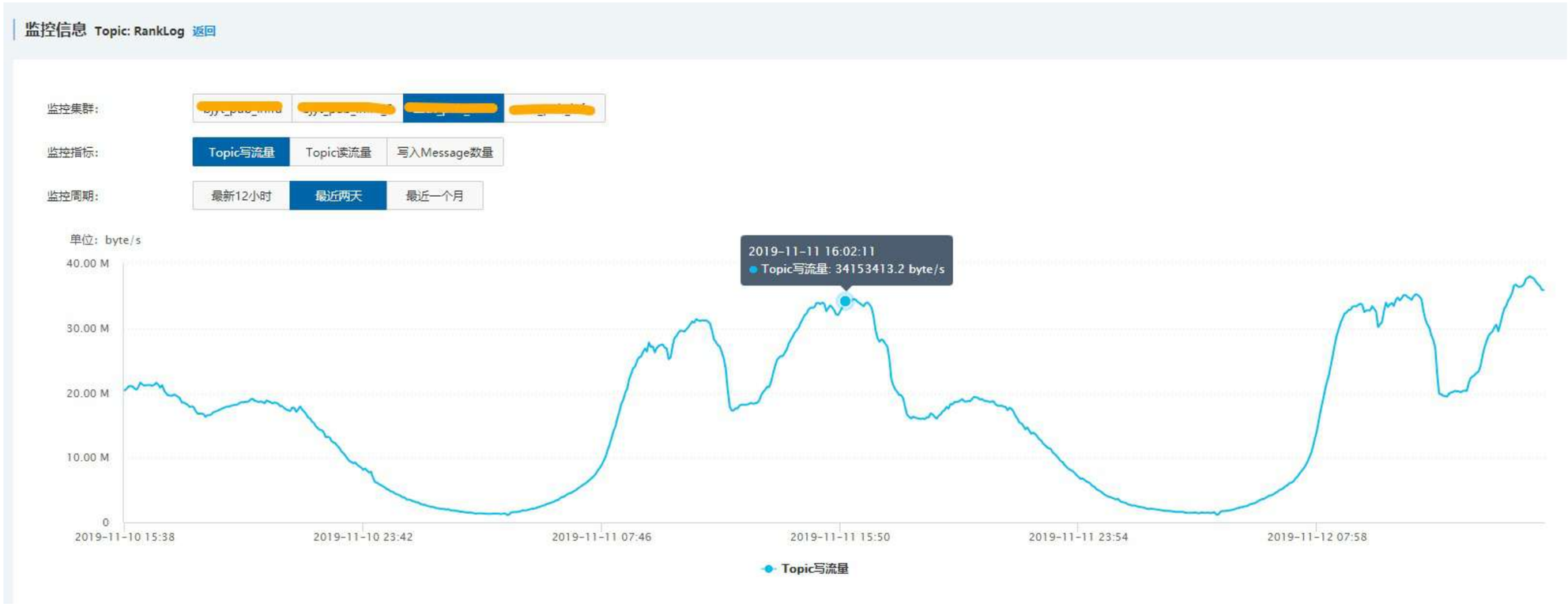
◆ 小明同学：江湖救急，业务线数据流量突增，现在积压了大量的数据，该咋办呀？
xiaoming: Help, business line data traffic suddenly increased, and a large amount of data has been accumulated, what should I do?

◆ 红红同学：通过调高并发总算是扛住了流量高峰，要不要把多余的资源还回去呢？
honghong: We managed to keep up with the traffic peak by increasing concurrency, do we need to return the extra resources?



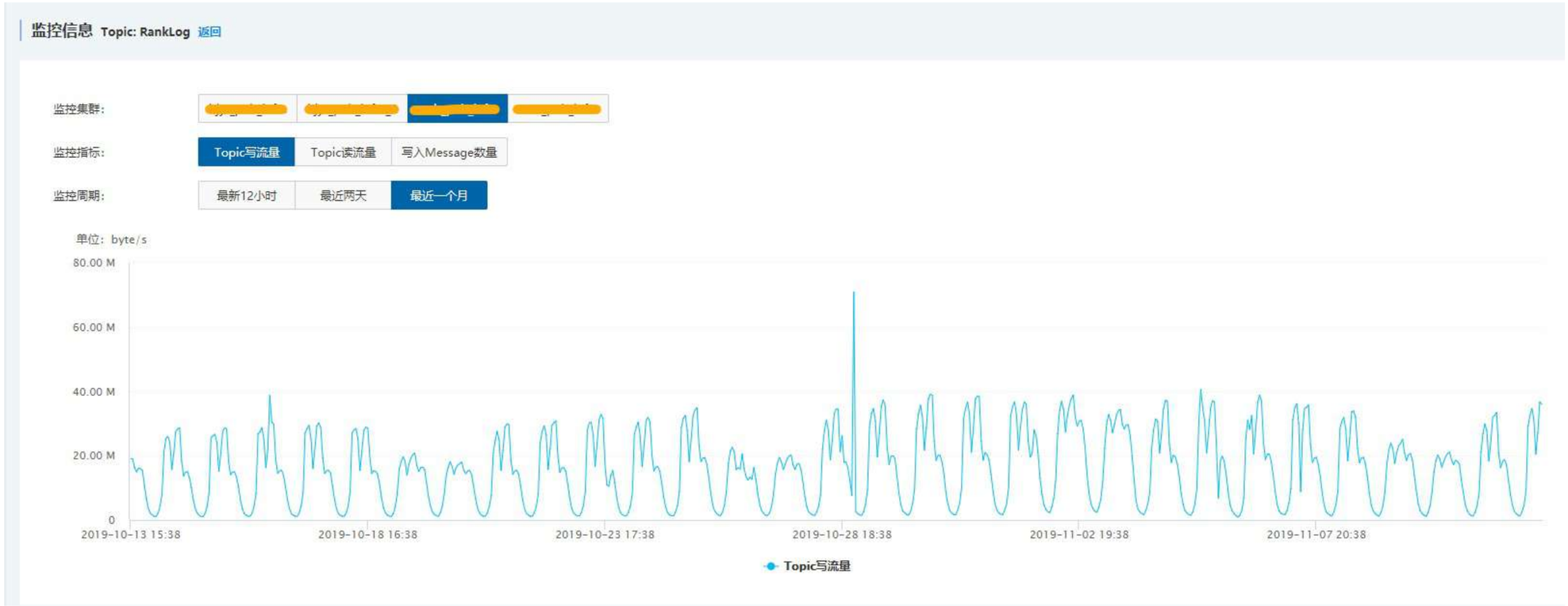
背景

Background



背景

Background



背景
Background

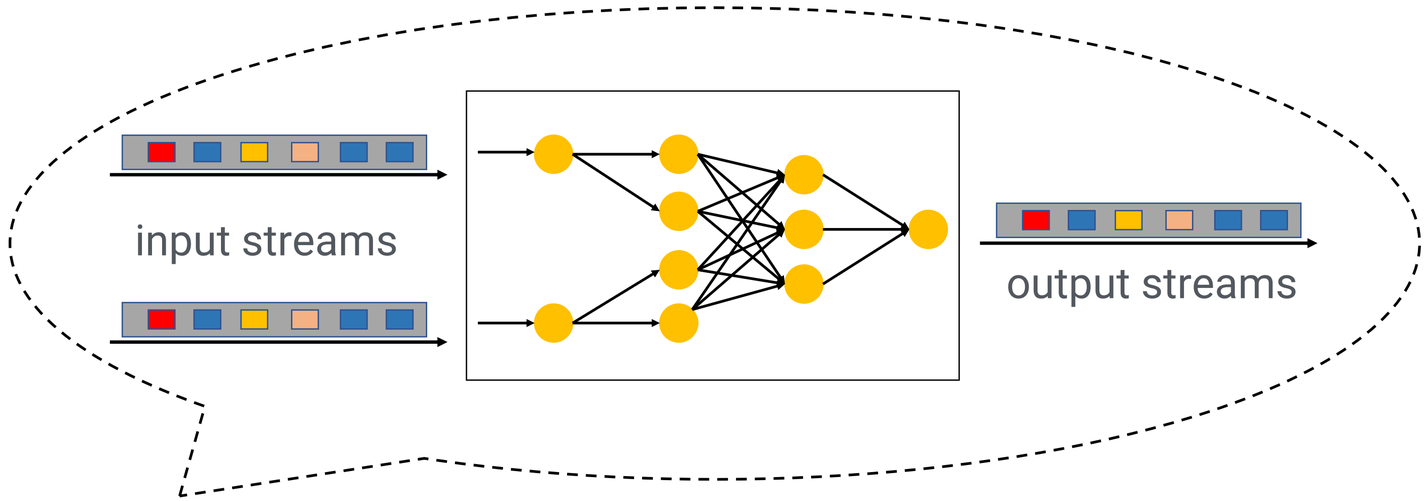
流式计算作业在未来都将不可避免的出现资源供应过剩或不足的情况！
Any streaming job will inevitably become over- or under-provisioned in the future.

何时调整，如何调整？
When and how much to scale?

背景
Background

手动调整
Manual scaling overview

1.监控源速率
monitor event rates

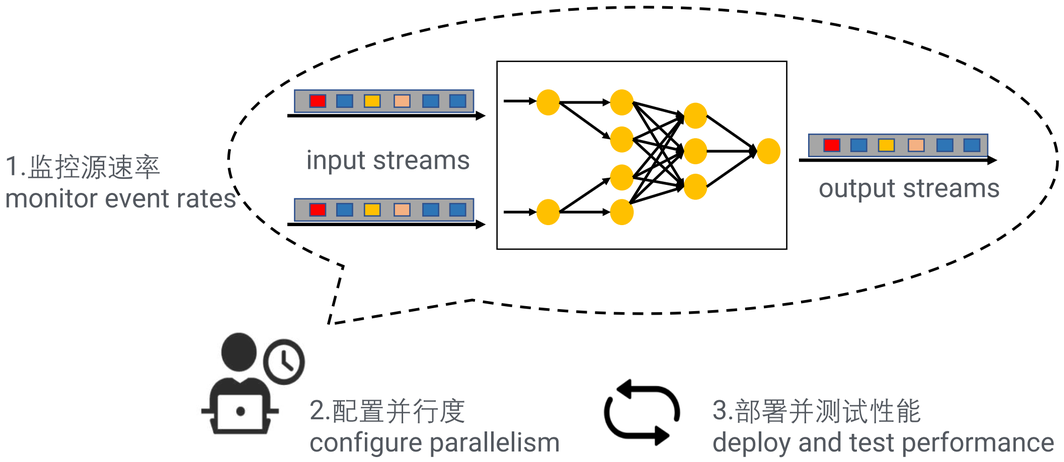


2.配置并行度
configure parallelism

3.部署并测试性能
deploy and test performance

背景
Background

手动调整
Manual scaling overview

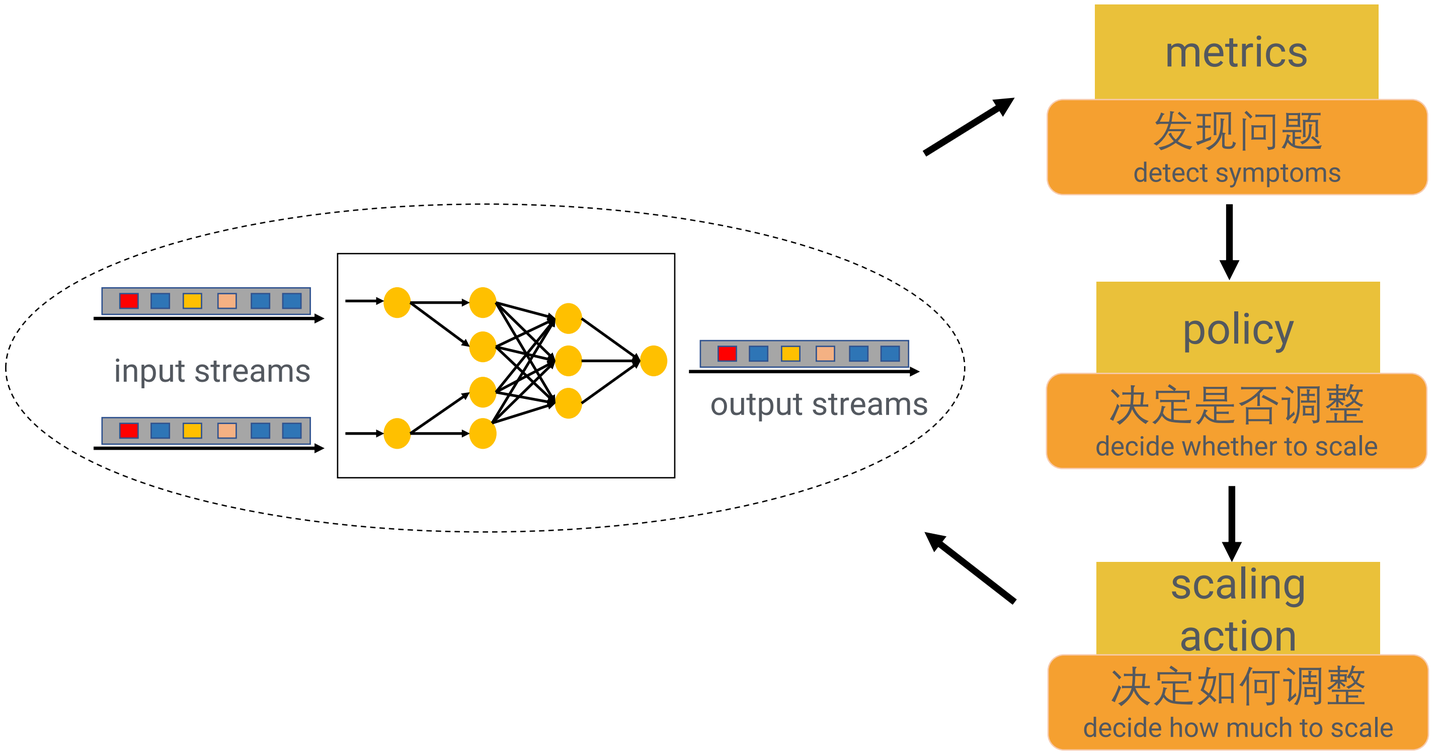


问题：
Problems:

- 额外的监控程序
  extra monitoring
- 凭经验，不准确
  depend on experience, inaccurate
- 影响作业稳定性
  affect job stability
- 手动配置难度大
  Manual configuration is difficult
- 手动调整成本高
  Manual scaling costs are high

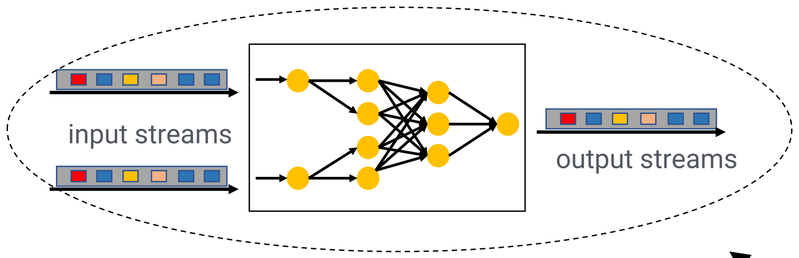
背景
Background

自动调整
Automatic scaling overview



背景
Background

自动调整
Automatic scaling overview



Dhalion

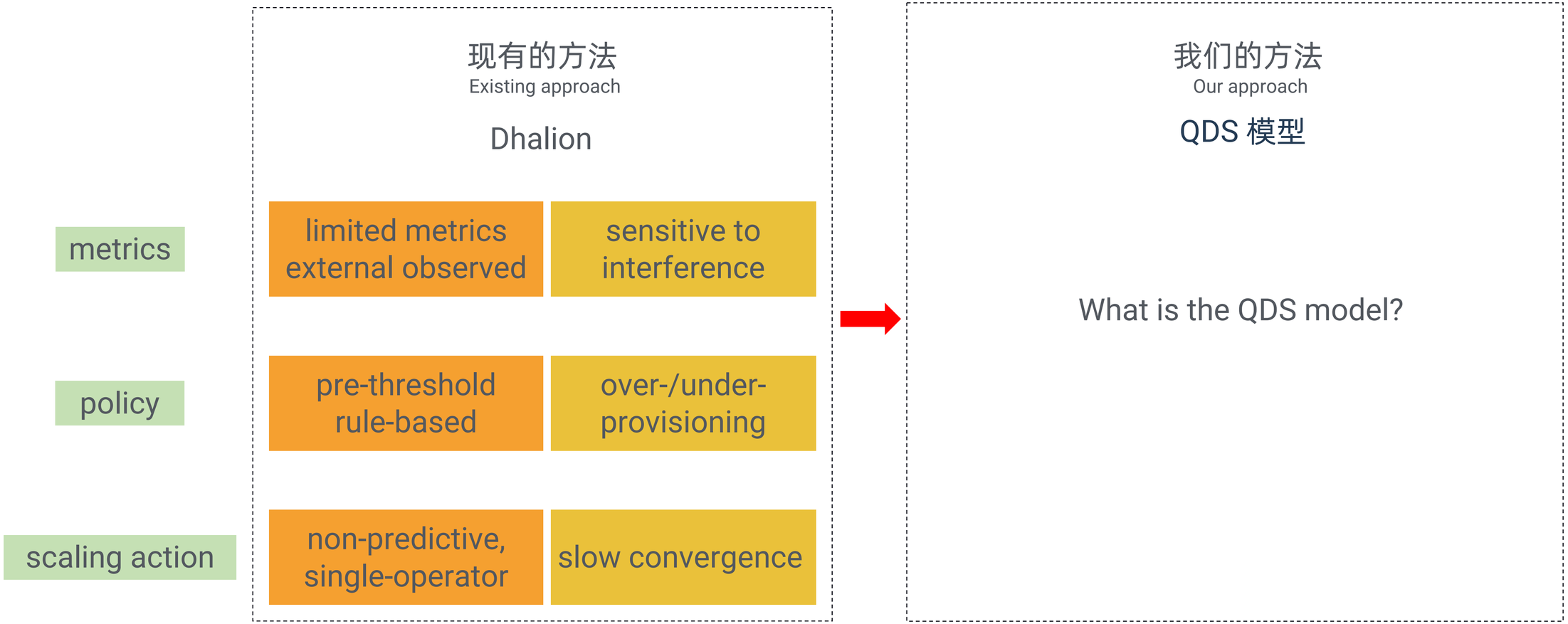
metrics
发现问题
detect symptoms

policy
决定是否调整
decide whether to scale

scaling action
决定如何调整
decide how much to scale

- Metrics
  CPU利用率，反压信号
  CPU utilization, backpressure signal
- Policy
  基于阈值和规则，比如：CPU利用率大于90%则调整
  threshold and rule-based, eg. CPU utilization > 90% => scale
- Action
  小范围调整，一次调整一个算子
  small changes, one operator at a time

背景
Background

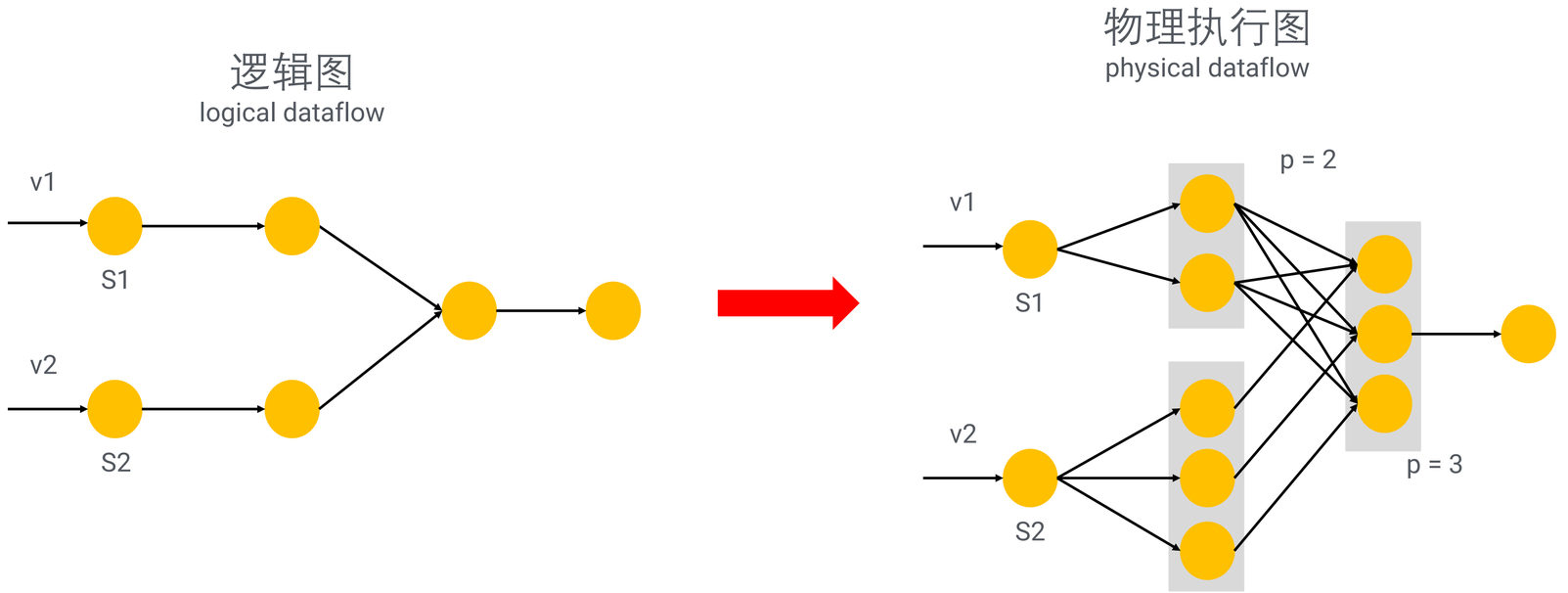


QDS 模型

The QDS model

02

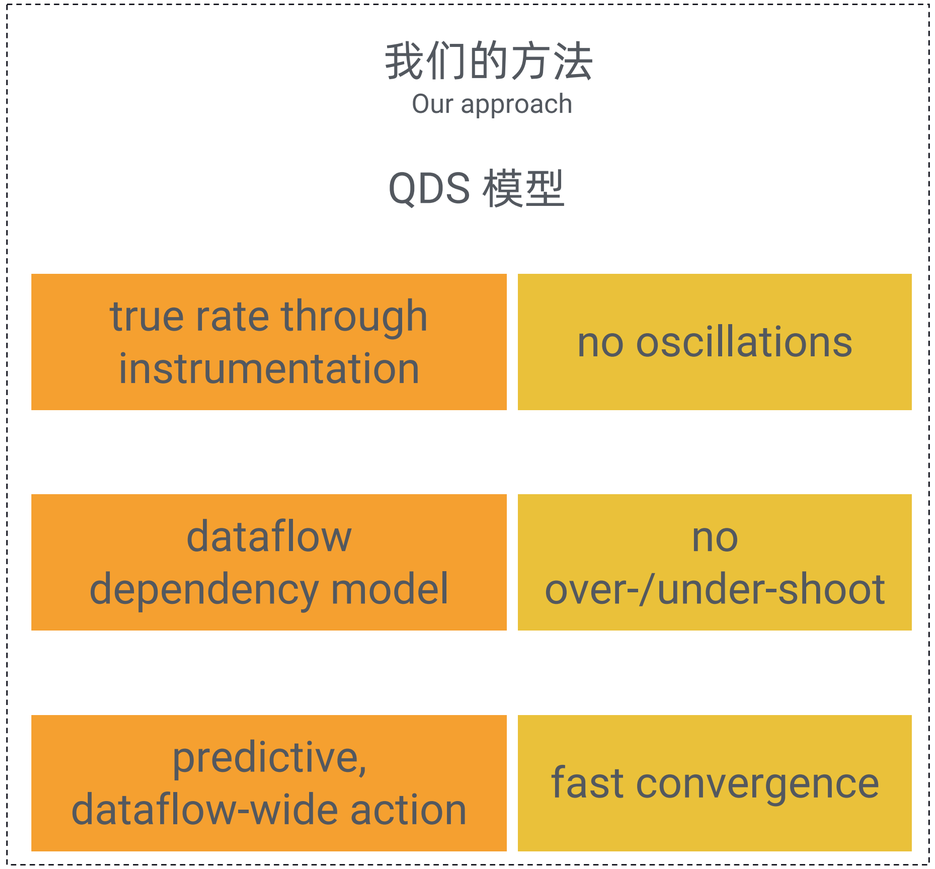
目标
Target



目标 1：在保证维持源速率的情况下确定各个算子的最小并行度！
Target 1: identify the minimum parallelism per operator such that the physical dataflow can sustain all source rates!

目标 2：在保证源没有堆积的情况下确定各个算子的最小并行度！
Target 2: identify the minimum parallelism per operator such that the physical dataflow have no data backlogs!

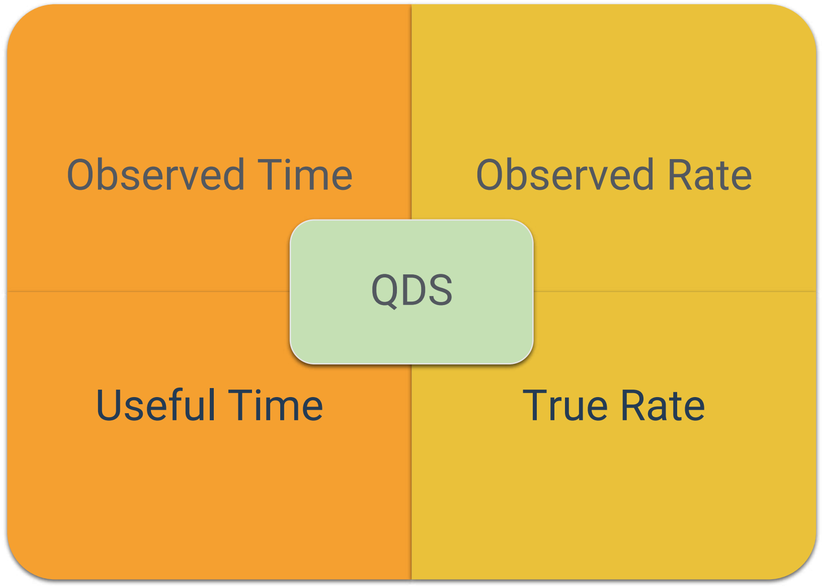
介绍
Introduction



- 关注数据源
  consider the sources
- 关注每个算子的真正的处理能力和输出能力
  consider each operator's true processing and output capabilities
- 关注数据流本身，各个算子间的计算依赖性
  consider the dataflow topology and computational dependencies among operators

核心概念

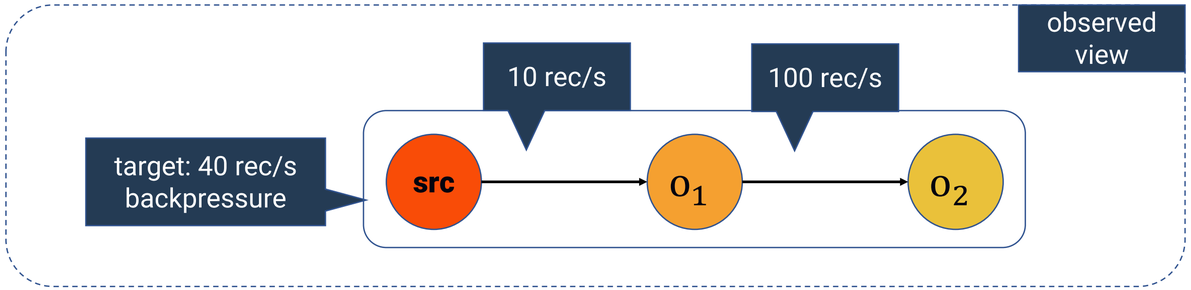
The core concept



$$\text{Optimal parallelism for } o_i = \frac{\text{aggregated true output rate of upstream ops}}{\text{average true processing rate of oi}}$$

举个例子

For example



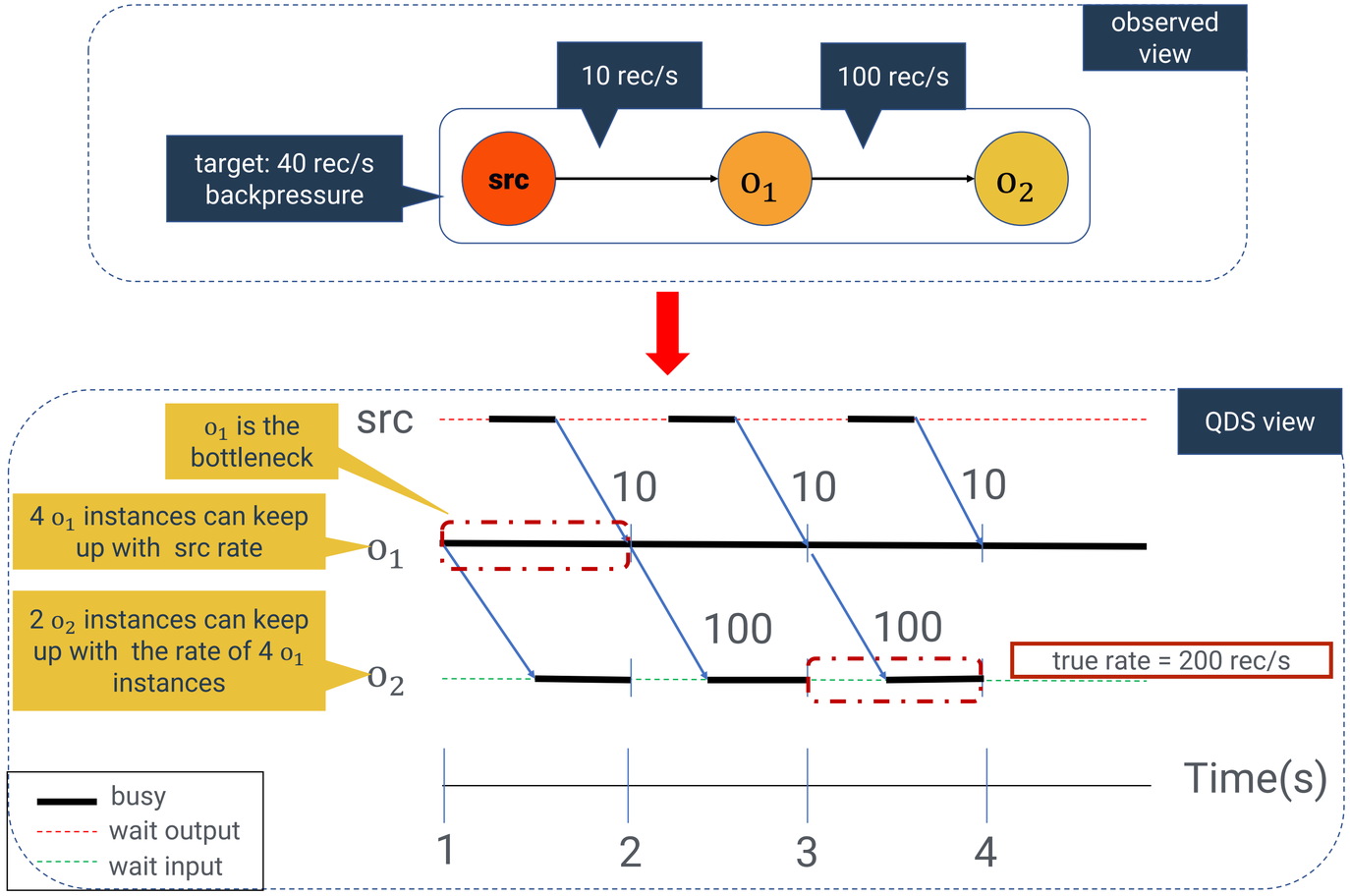
Which operator is the bottleneck?

What if we scale o1 x 4?

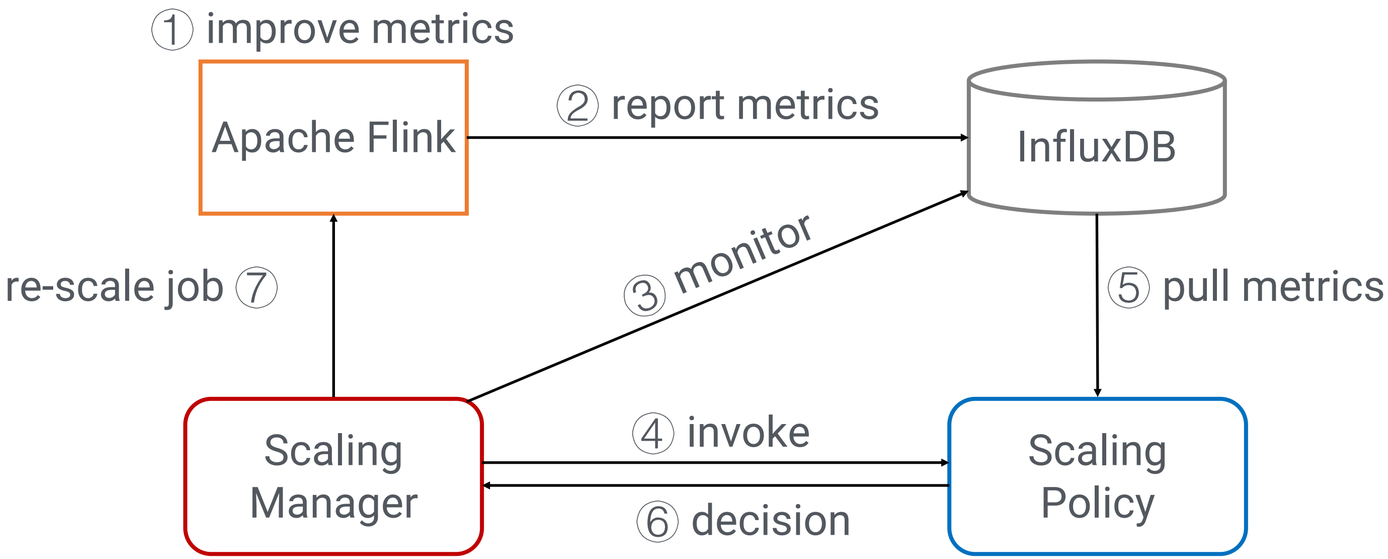
How much to scale o2?

举个例子

For example



QDS on Flink

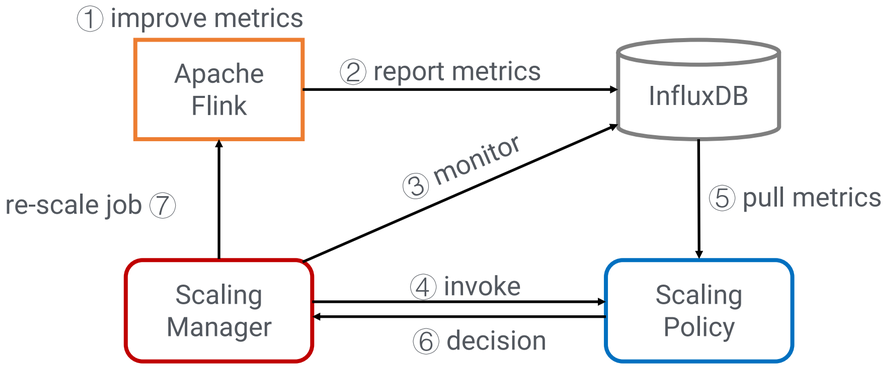

① improve metrics
Apache Flink
② report metrics
InfluxDB
re-scale job ⑦
③ monitor
⑤ pull metrics
Scaling Manager
④ invoke
⑥ decision
Scaling Policy

生产实践

Production Practice

03

生产实践
Production Practices



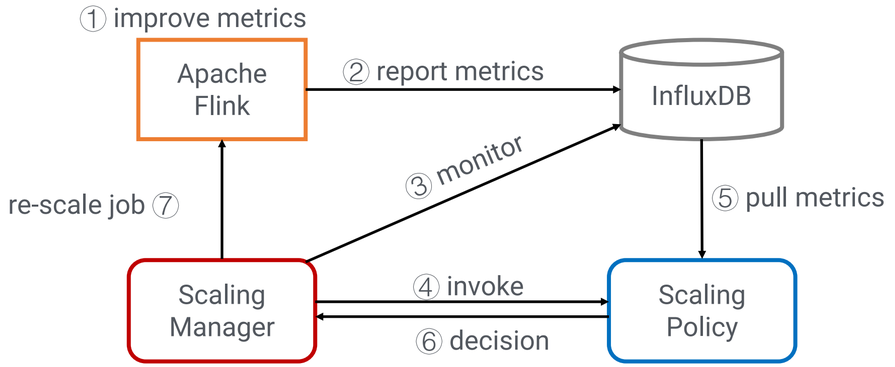
准备工作：
The preparatory work:

打开自动调整配置
Enable automatic scaling configuration

首次提交作业，各算子默认并行度为 1，不开启 chain 策略。
Each operator default parallelism is 1 in the first submission of the job, and disable chain.

生产实践

Production Practices

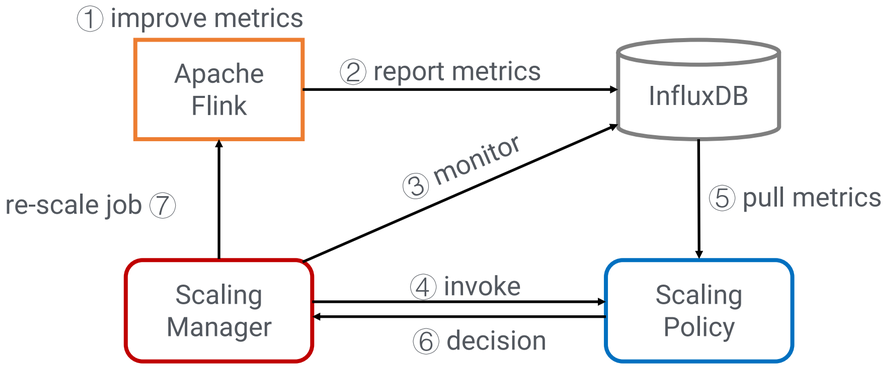


① improve metrics

- ✓ taskmanager_job_task_operator_KafkaConsumer_topic_partition_0_committedOffsets
- ✓ taskmanager_job_task_operator_KafkaConsumer_topic_partition_0_latestOffsets

- ✓ taskmanager_job_task_operator_deserializationDuration
- ✓ taskmanager_job_task_operator_processingDuration
- ✓ taskmanager_job_task_operator_serializationDuration
- ✓ taskmanager_job_task_operator_waitingDuration

生产实践
Production Practices



② report metrics

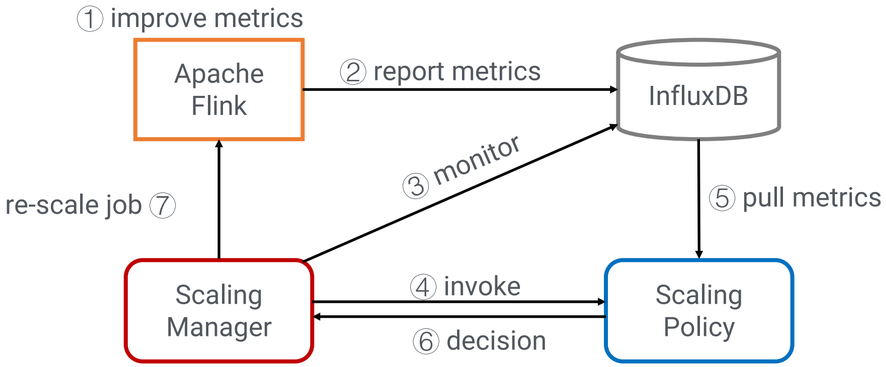
✓ org.apache.flink.metrics.influxdb.InfluxdbReporter

将 /opt/flink-metrics-influxdb-1.9.0.jar 拷贝到 /lib 目录下
Copy /opt/flink-metrics-influxdb-1.9.0.jar into the /lib folder of your Flink distribution

✓ Sampling rate

增加 job.metric.sample.rate 配置项，抽样率用于控制 metric 的计算频率
Add the configuration of job.metric.sample.rate, sampling rate is used to control how often certain metrics are computed

生产实践
Production Practices

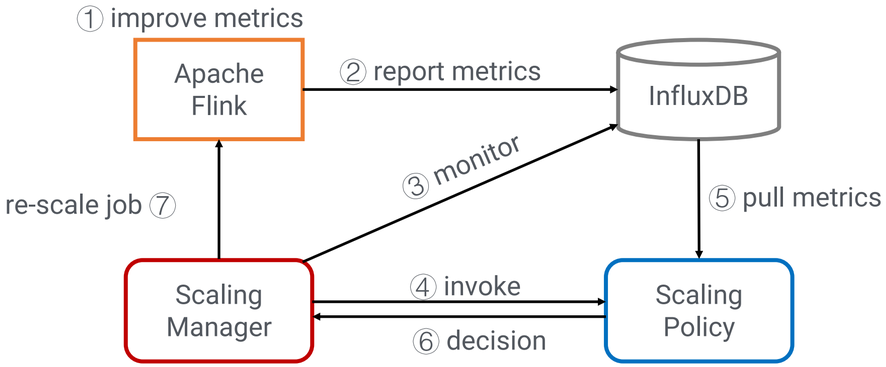


③ monitor

Scaling Manager 监控 InfluxDB

monitor the InfluxDB with scaling manager

生产实践
Production Practices



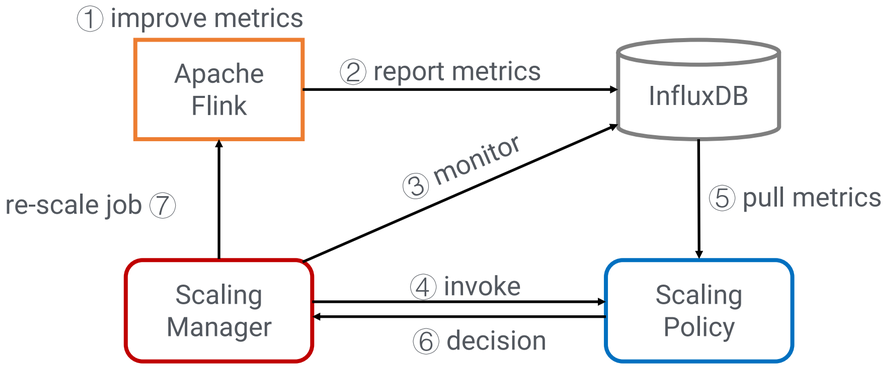
④ invoke

当 metrics 可用时，调用策略 Scaling Policy

invoke the scaling policy when metrics are available

生产实践

Production Practices



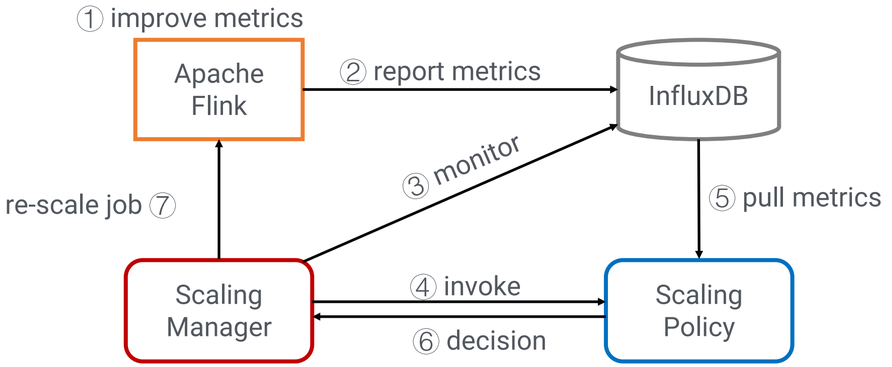
⑤ pull metrics

- ✓ taskmanager_job_task_operator_numRecordsIn
- ✓ taskmanager_job_task_operator_numRecordsInPerSecond

检查数据倾斜并告警

check data skew and alarm

生产实践
Production Practices



⑤ pull metrics

- ✓ taskmanager_job_task_operator_KafkaConsumer_topic_partition_0_committedOffsets
- ✓ taskmanager_job_task_operator_KafkaConsumer_topic_partition_0_latestOffsets

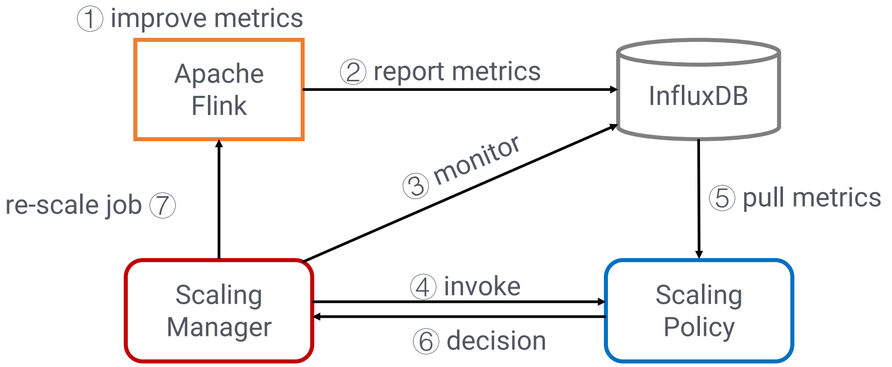
计算源速率
compute source rate

检查 kafka 堆积
check the kafka lag

自定义源速率
Custom source rate

生产实践

Production Practices



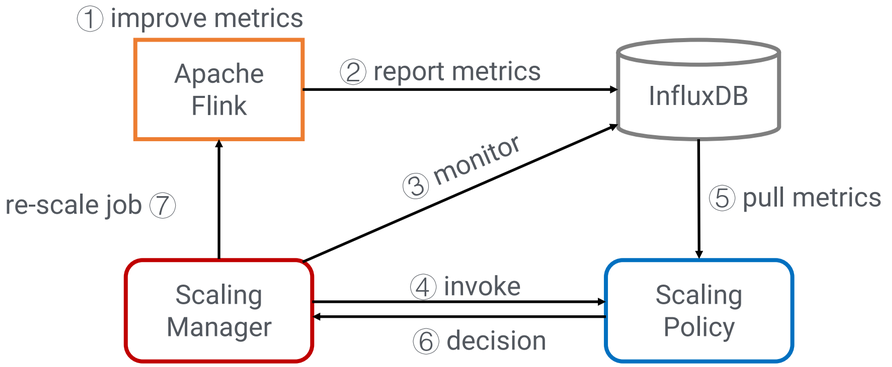
⑤ pull metrics

- ✓ taskmanager_job_task_operator_deserializationDuration
- ✓ taskmanager_job_task_operator_processingDuration
- ✓ taskmanager_job_task_operator_serializationDuration
- ✓ taskmanager_job_task_operator_waitingDuration

计算每个算子速率

compute each operator's rate

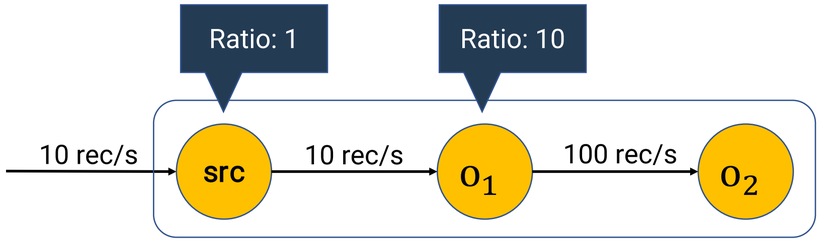
生产实践
Production Practices



⑤ pull metrics

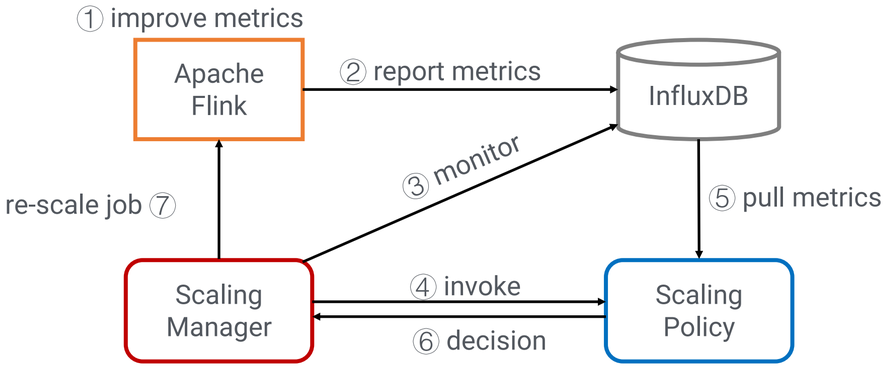
- ✓ taskmanager_job_task_operator_numRecordsIn
- ✓ taskmanager_job_task_operator_numRecordsOut

计算生产消费比
compute the pc ratio



$$\text{Optimal parallelism for } o_i = \frac{\text{aggregated account of upstream ops} * \text{the pc ratio}}{\text{average true processing rate of } o_i} \times (1 + \text{scaling factor})$$

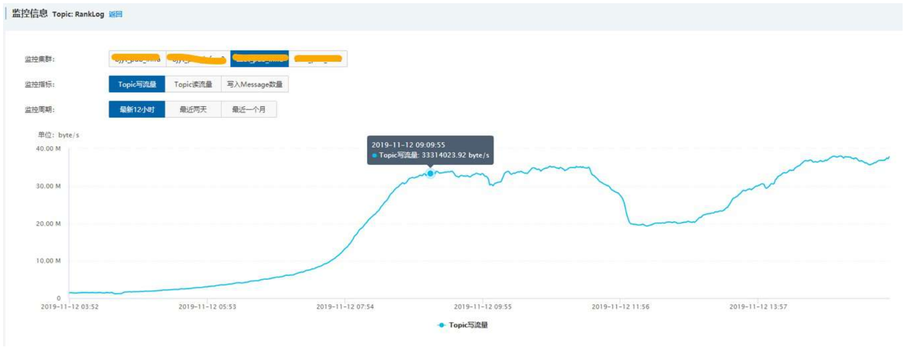
生产实践
Production Practices



⑥ decision

忽略微小改动
ignore minor changes

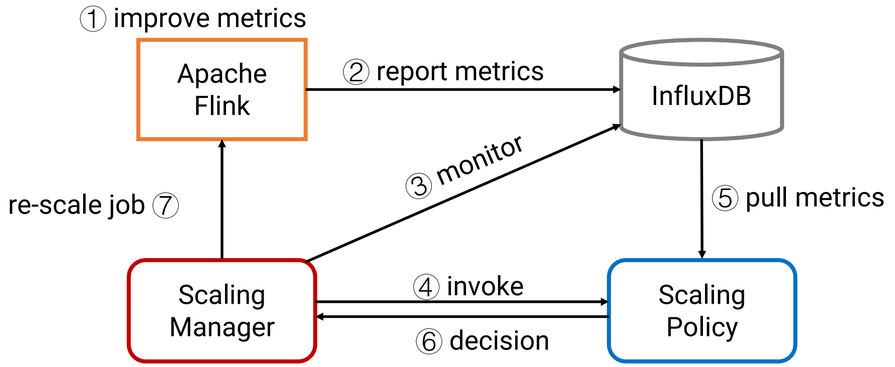
源速率趋势图 + 调整幅度




```
if(before < after && after * 1.0 / before > (1 + configuration))
        return true;

if(before > after && before * 1.0 / after > (1 + configuration))
        return true;
```

生产实践
Production Practices



⑦ re-scale job

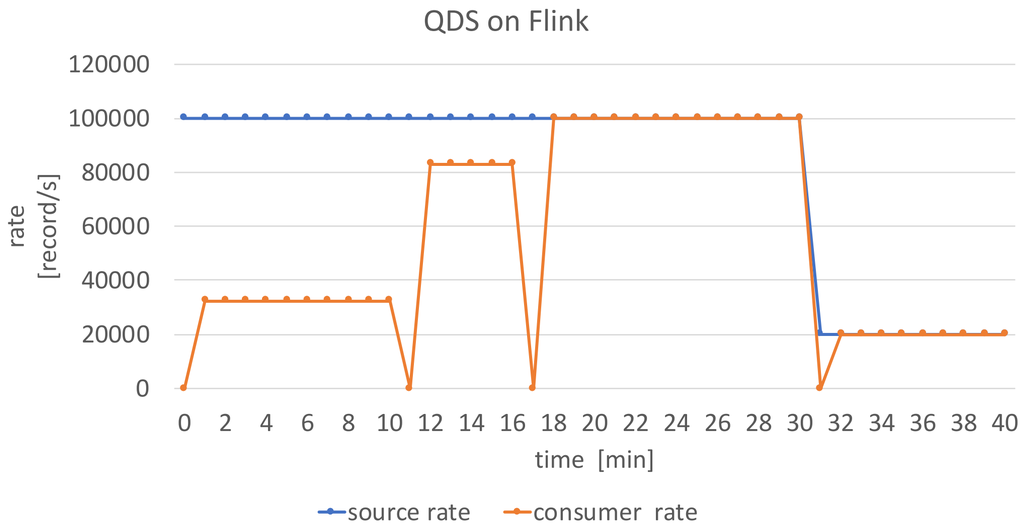
触发 Savepoint， 暂停作业，改造 JobGraph，并使用新的并行度重新部署。
Flink triggers a Savepoint, halts the computation, reconstructs jobgraph and redeploys it with the new parallelism.

不释放 container 资源，减少作业停止时间。
Do not release container resources, reduce the job stop time.

✓ yarn-session 模式：调整 resourcemanager.taskmanager-timeout 配置项
yarn-session mode: adjust the configuration of resourcemanager.taskmanager-timeout

✓ per-job 模式：增加 rescaling 命令
per-job mode: add the command of rescaling

总结
Summary



X 同学：可以更多的去关注业务逻辑本身了。

Student X: we can focus on the business logic itself.

实践表明：任务最多三次调整即可达到收敛，集群的资源利用率也提高了。

Practice shows that stream processing took up to three steps to converge, and the resource utilization of the yarn cluster is improved.

展望

Future Work

04

展望
Future Work

- 进一步提高准确性，“关键路径”分析模型
  Further improve accuracy, critical path analysis model
- 进一步提高稳定性，更快的重新配置机制
  Further improve stability, a faster, more dynamic reconfiguration mechanism
- 回馈社区
  Contribute back to the community

致谢

Acknowledgement

Kalavri V, Liagouris J, Hoffmann M, Dimitrova D, Forshaw M, Roscoe T.
Three steps is all you need: fast, accurate, automatic scaling decisions for distributed streaming dataflows.
OSDI ’18.


FLINK
FORWARD
THANKS