FLINK FORWARD

Flink Forward Virtual Conference 2020

Stream Processing | Event Driven | Real Time

April 23: 10:10 am - 10:50 am PDT

Apache Flink Worst Practices

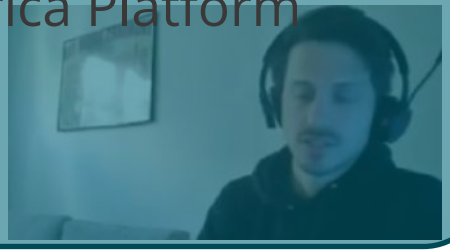
Speaker: Konstantin Knauf

Company: Ververica

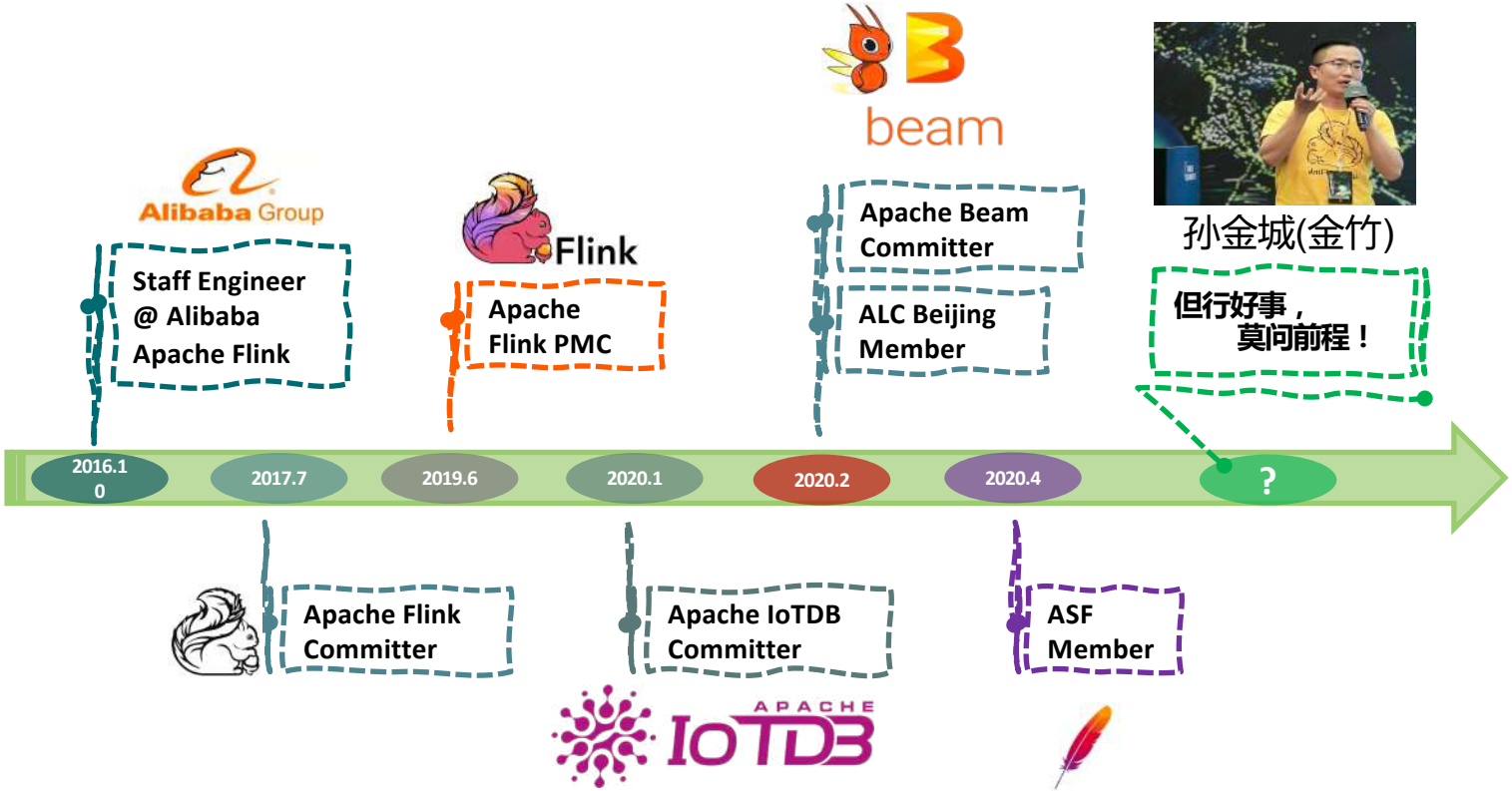
Distributed stream processing is evolving from a technology in the sidelines of Big Data to a key enabler... View More

Konstantin Knauf
Head of Product - Ververica Platform
Apache Flink Committer

@snntrable



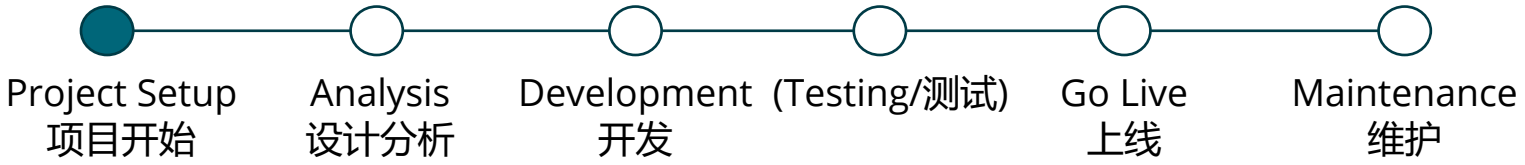
About Me



= Worst Practice

Agenda/目录

Don’t use an iterative development process! 😈
不要使用迭代开发过程



😈 = Worst Practice

Before you start...

...choose the right setup! 选择正确的切入...

- 😈 start with one of you most challenging uses cases
  从一个最具挑战性的用例开始

😈 = Worst Practice

Before you start...

...choose the right setup!

- 😈 start with one of you most challenging uses cases
- 😈 no prior stream processing knowledge
  之前没有流处理经验

😈 = Worst Practice

Before you start...

...choose the right setup!

- 😈 start with one of you most challenging uses cases
- 😈 no prior stream processing knowledge
- 😈 no training
  没有培训

😈 = Worst Practice

Before you start...

...choose the right setup!

- 😈 start with one of you most challenging uses cases
- 😈 no prior stream processing knowledge
- 😈 no training
- 😈 don’t use the community
  不利用社区

😈 = Worst Practice

Training & Community

Getting Help!/求助

- **Training/培训**
  - @FlinkForward
  - https://www.ververica.com/training

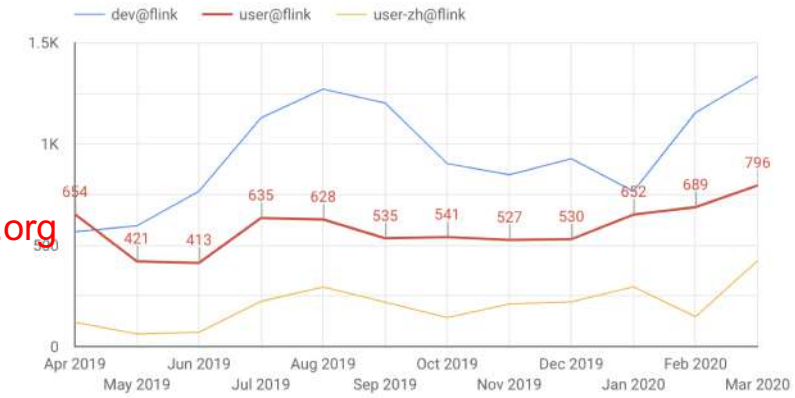
= Worst Practice

Training & Community

Getting Help!

- **Training/培训**
  - @FlinkForward
  - https://www.ververica.com/training

- **Community/社区**
  - user@flink.apache.org/user-zh@flink.apache.org
    - ~ 600 threads per month
    - Ø30h until first response



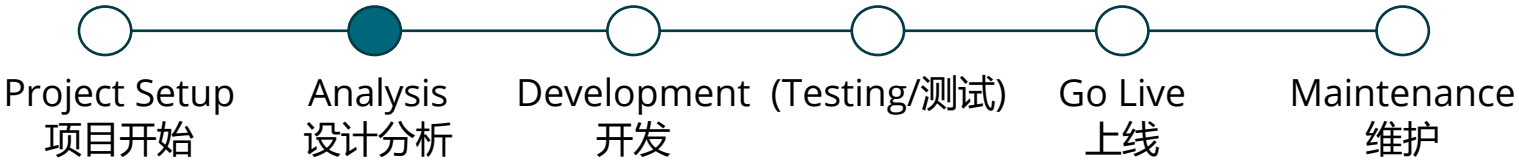
  - www.stackoverflow.com
    - How not to ask questions?/不要提低级问题
      - 😈 https://data.stackexchange.com/stackoverflow/query/1115371/most-down-voted-apache-flink-questions

  - (ASF Slack #flink)

😈 = Worst Practice

Agenda

Don’t use an iterative development process! 😈
不要使用迭代开发过程



 |

😈 = Worst Practice

Before you start...

...don’t think too much about your requirements. 😈
...不要过多思考你的需求。

- 😈 don’t think about consistency & delivery guarantees

  不考虑一致性和交付保证

😈 = Worst Practice

Before you start...

...don’t think too much about your requirements. 😈

- 😈 don’t think about consistency & delivery guarantees

- 😈 don’t think about upgrades and application evolution
  不考虑升级和应用程序改进

😈 = Worst Practice

Before you start…

…don’t think too much about your requirements. 😈

- 😈 don’t think about consistency & delivery guarantees
- 😈 don’t think about upgrades and application evolution
- 😈 don’t think about the scale of your problem
  不考虑业务问题的规模

😈 = Worst Practice

Before you start...

...don't think too much about your requirements. 😈

- don't think about consistency & delivery guarantees
- don't think about upgrades and application evolution
- don't think about the scale of your problem
- don't think too much about your actual business requirements
  不深入思考实际业务需求

😈 = Worst Practice

Consistency & Delivery Guarantees
一致性和交付保证

Do I care about losing records?

在乎丢失记录吗？

= Worst Practice

Consistency & Delivery Guarantees
一致性和交付保证

Do I care about losing records?

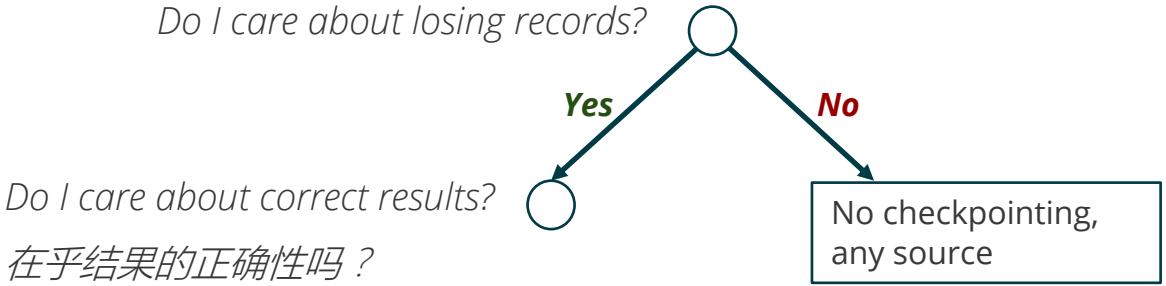
在乎丢失记录吗？

No

No checkpointing,
any source
不需要CP/任何数据

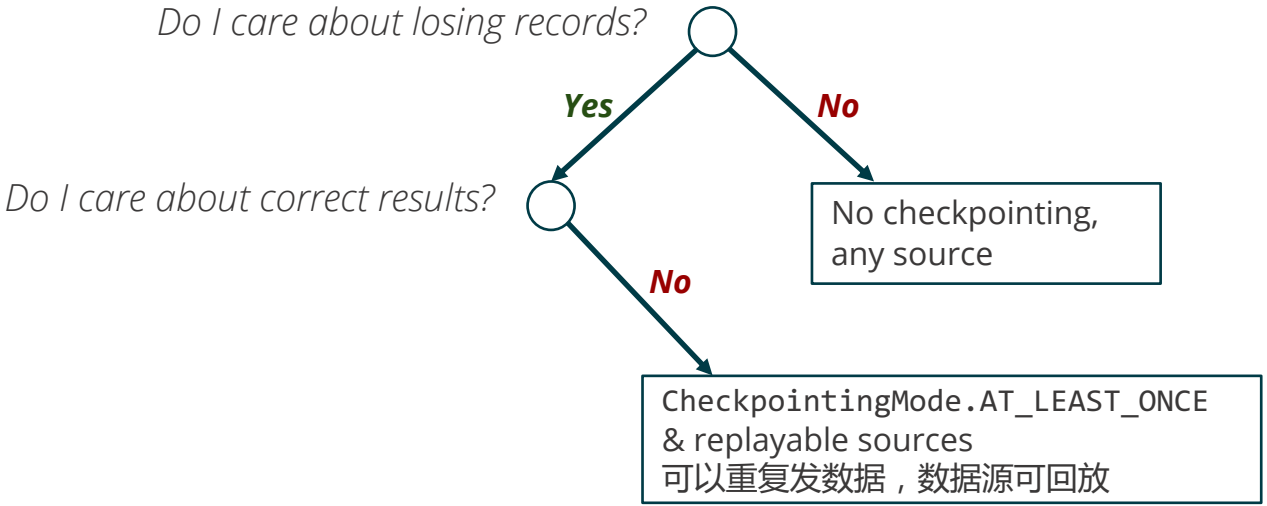
= Worst Practice

Consistency & Delivery Guarantees
一致性和交付保证



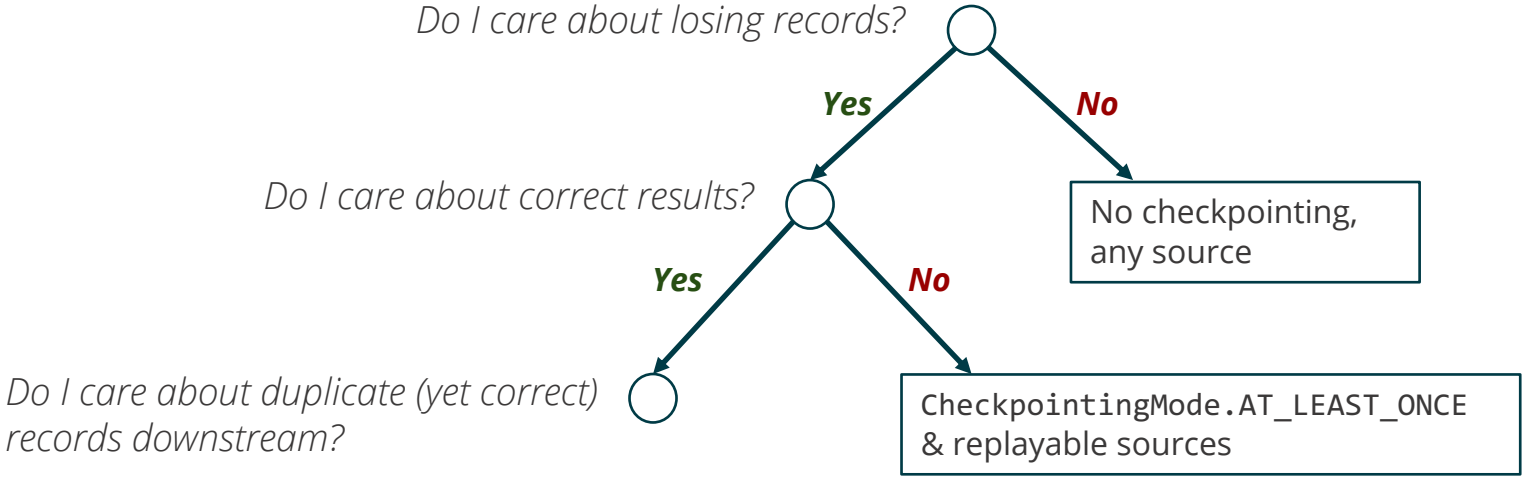
= Worst Practice

Consistency & Delivery Guarantees
一致性和交付保证



= Worst Practice

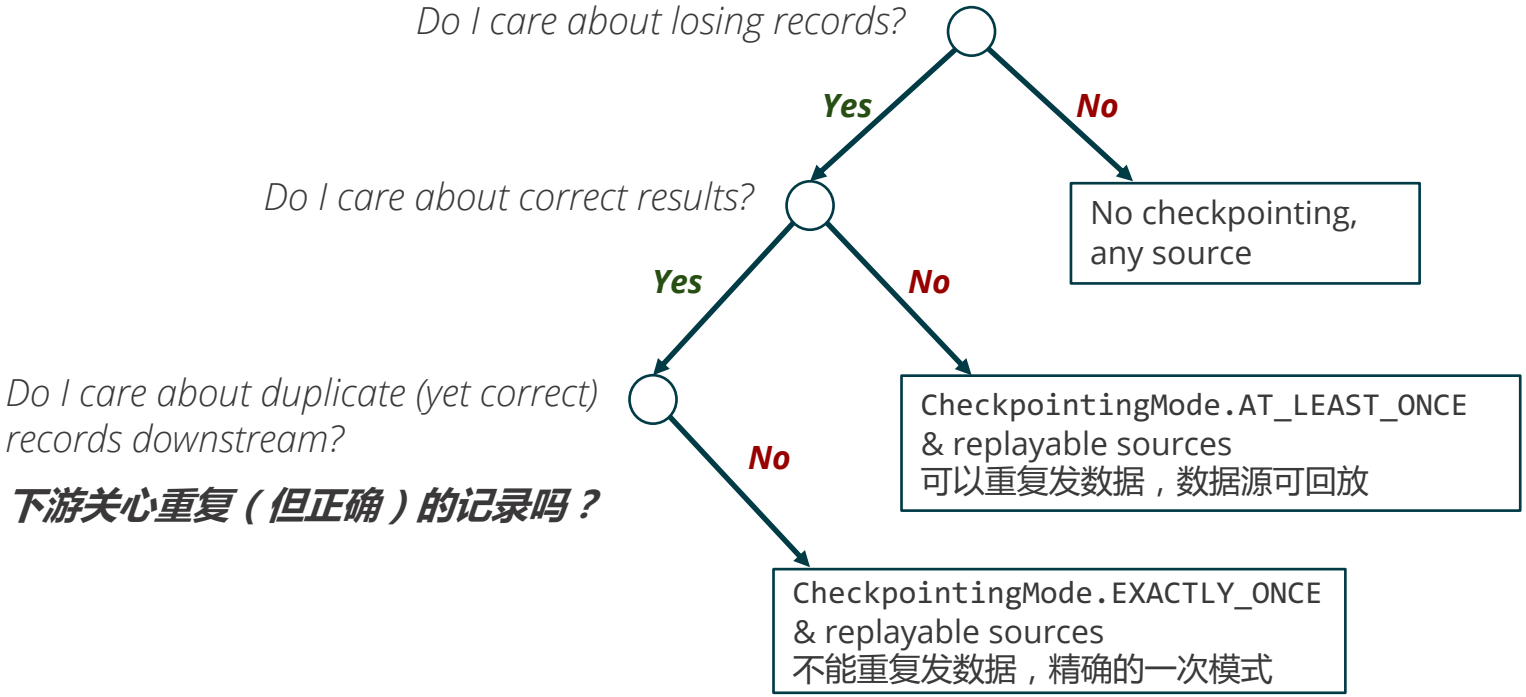
Consistency & Delivery Guarantees
一致性和交付保证



下游关心重复（但正确）的记录吗？

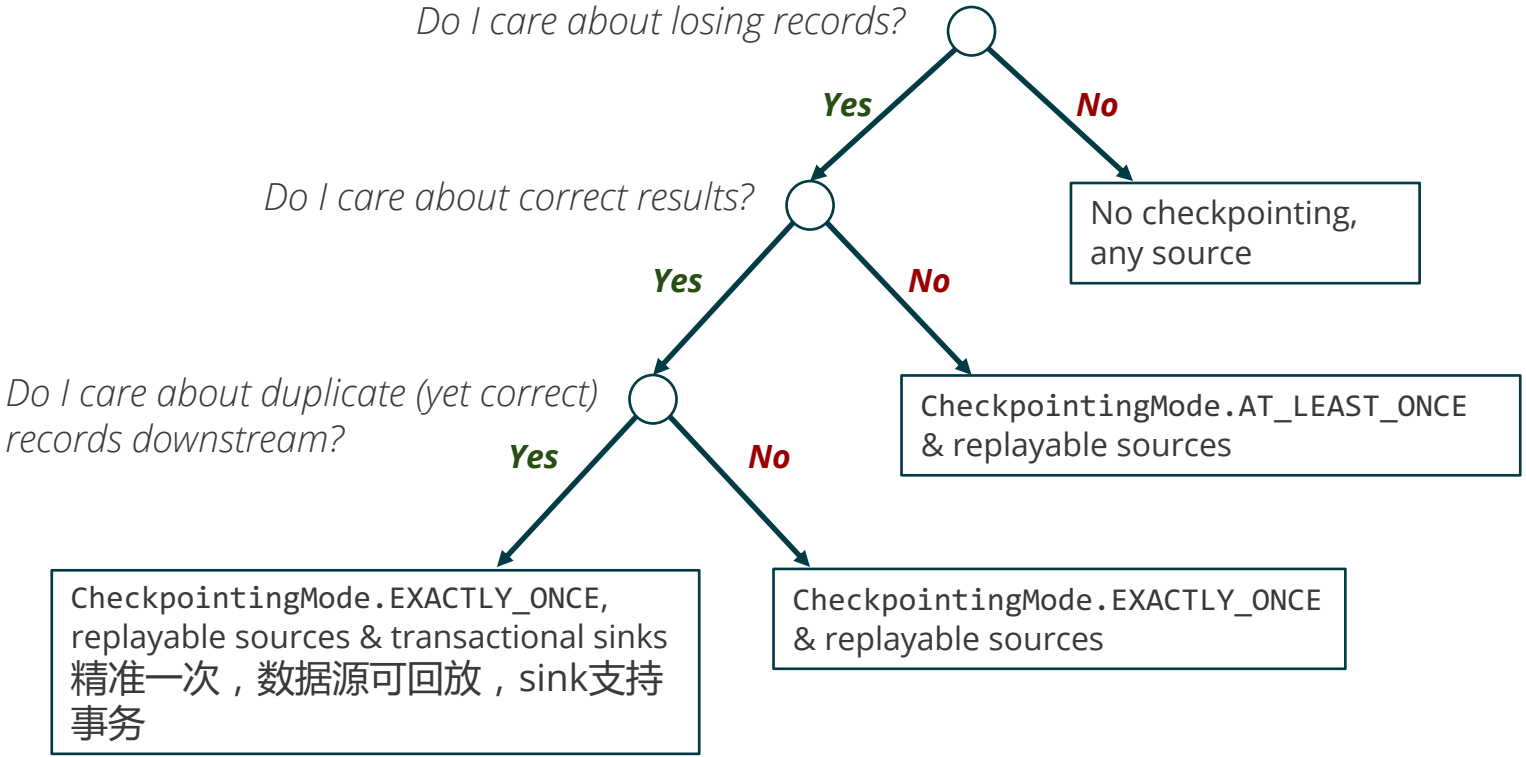
= Worst Practice

Consistency & Delivery Guarantees
一致性和交付保证



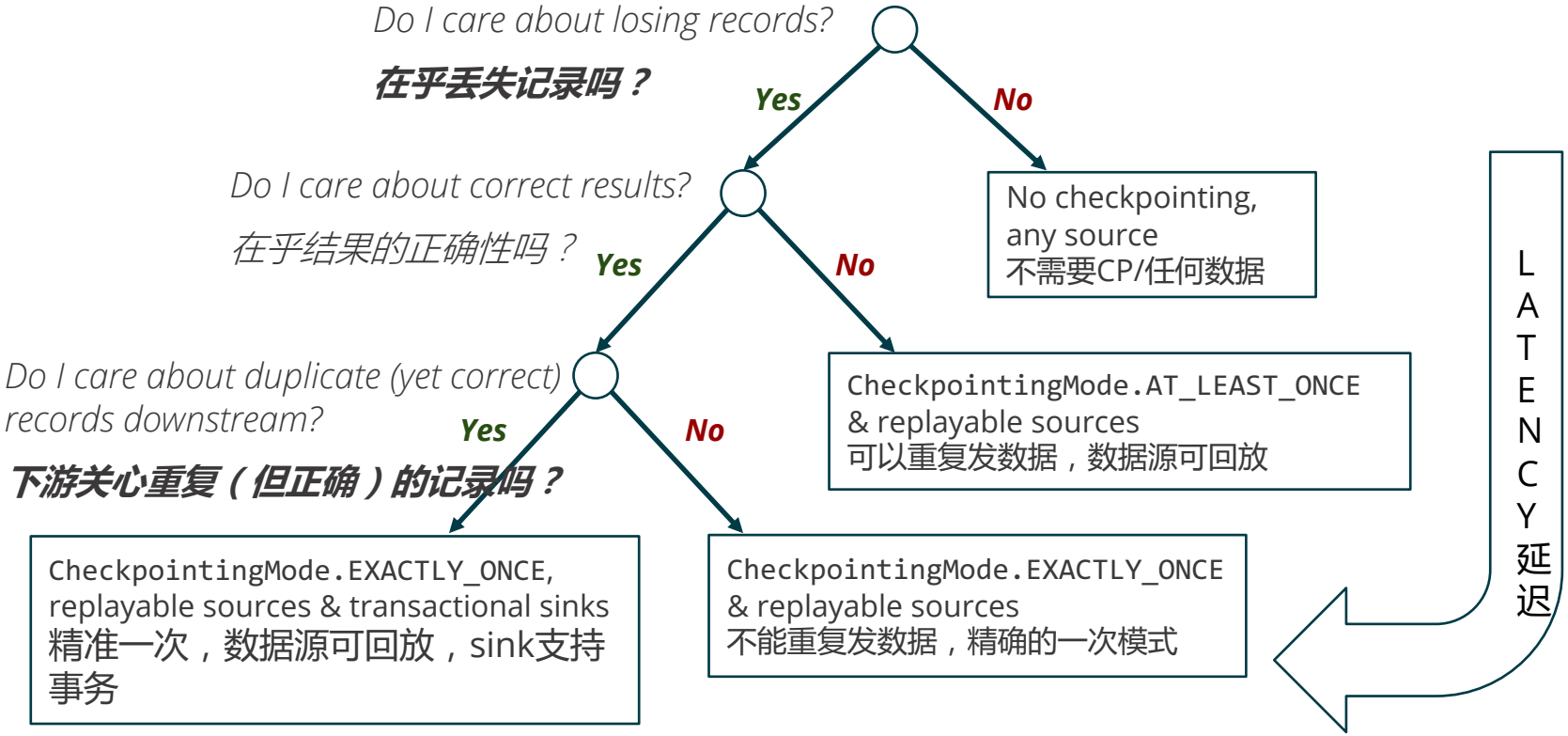
= Worst Practice

Consistency & Delivery Guarantees
一致性和交付保证



= Worst Practice

Consistency & Delivery Guarantees
一致性和交付保证



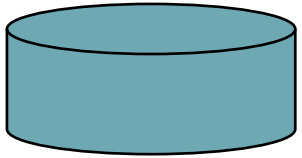
= Worst Practice

Application Evolution/应用升级

Basics/兼容升级

Flink job
user code

Local state
backends



local reads / writes that
manipulate state
本地读写操作状态

Persisted
savepoint

= Worst Practice

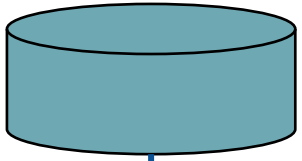
Application Evolution/应用升级

Basics/兼容升级

Flink job user code

Local state backends

Persisted savepoint



local reads / writes that manipulate state

persist to DFS on Savepoint SP时候状态持久化到DFS

= Worst Practice

Application Evolution/应用升级

Basics/兼容升级

Flink job user code

Upgrade Application

1. *Upgrade Flink cluster*
2. *Fix bugs*
3. *Pipeline topology changes*
4. *Job reconfigurations*
5. *Adapt state schema*
6. *...*

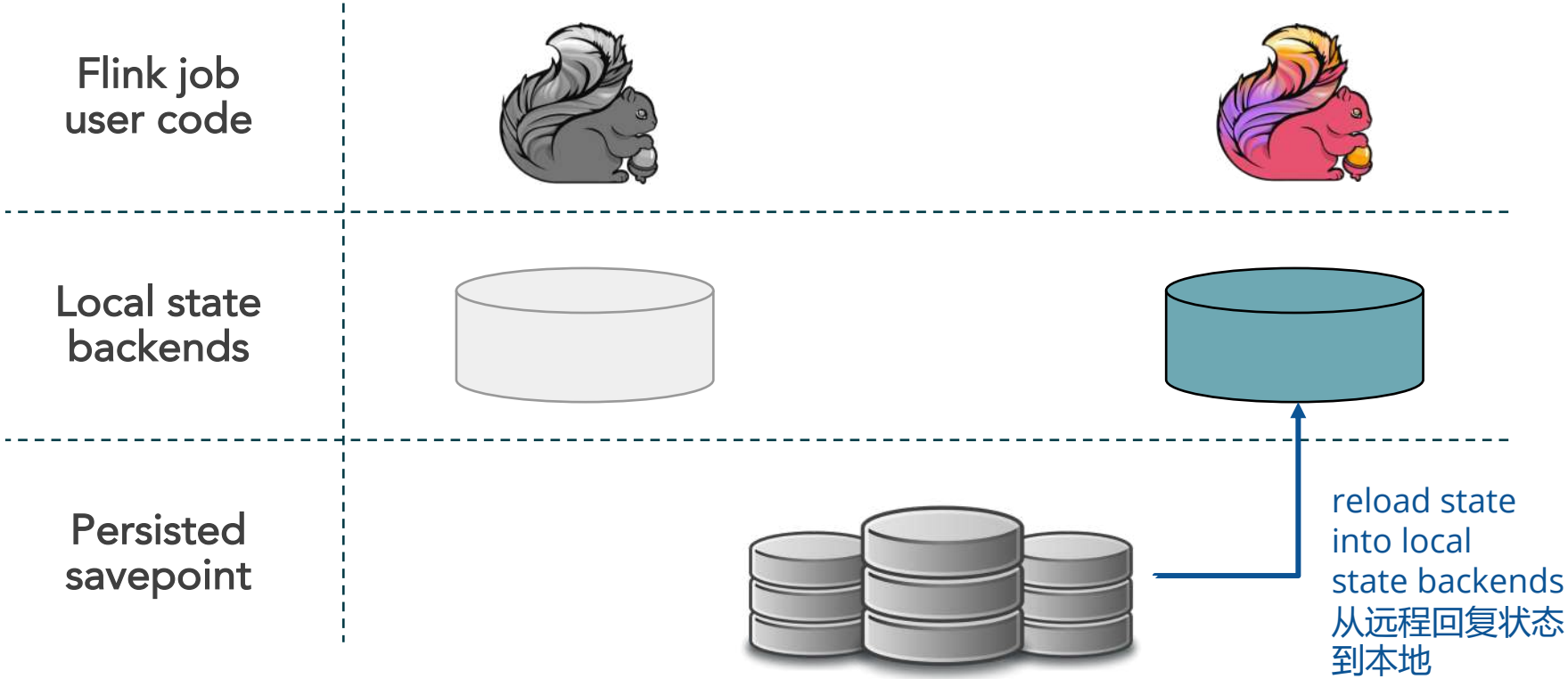
Local state backends

Persisted savepoint

= Worst Practice

Application Evolution/应用升级

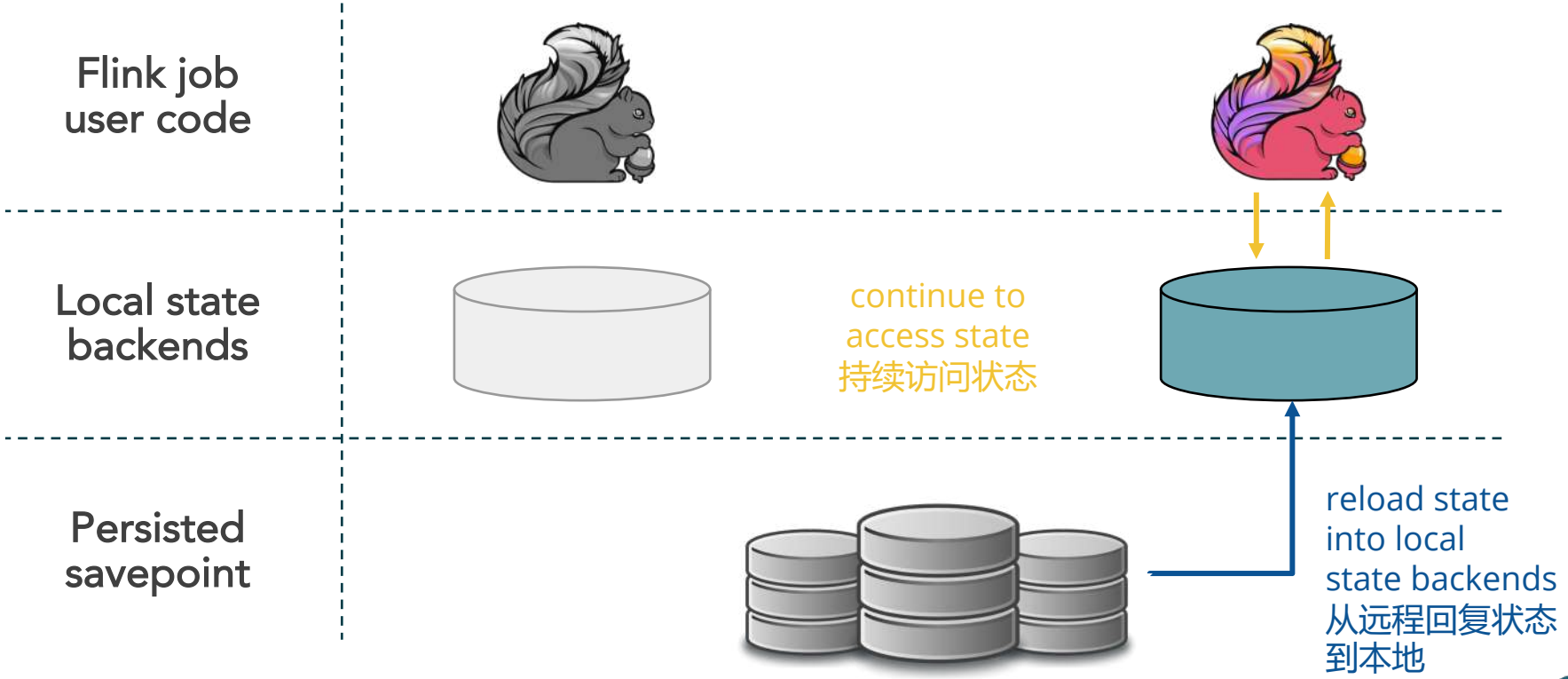
Basics/兼容升级



= Worst Practice

Application Evolution/应用升级

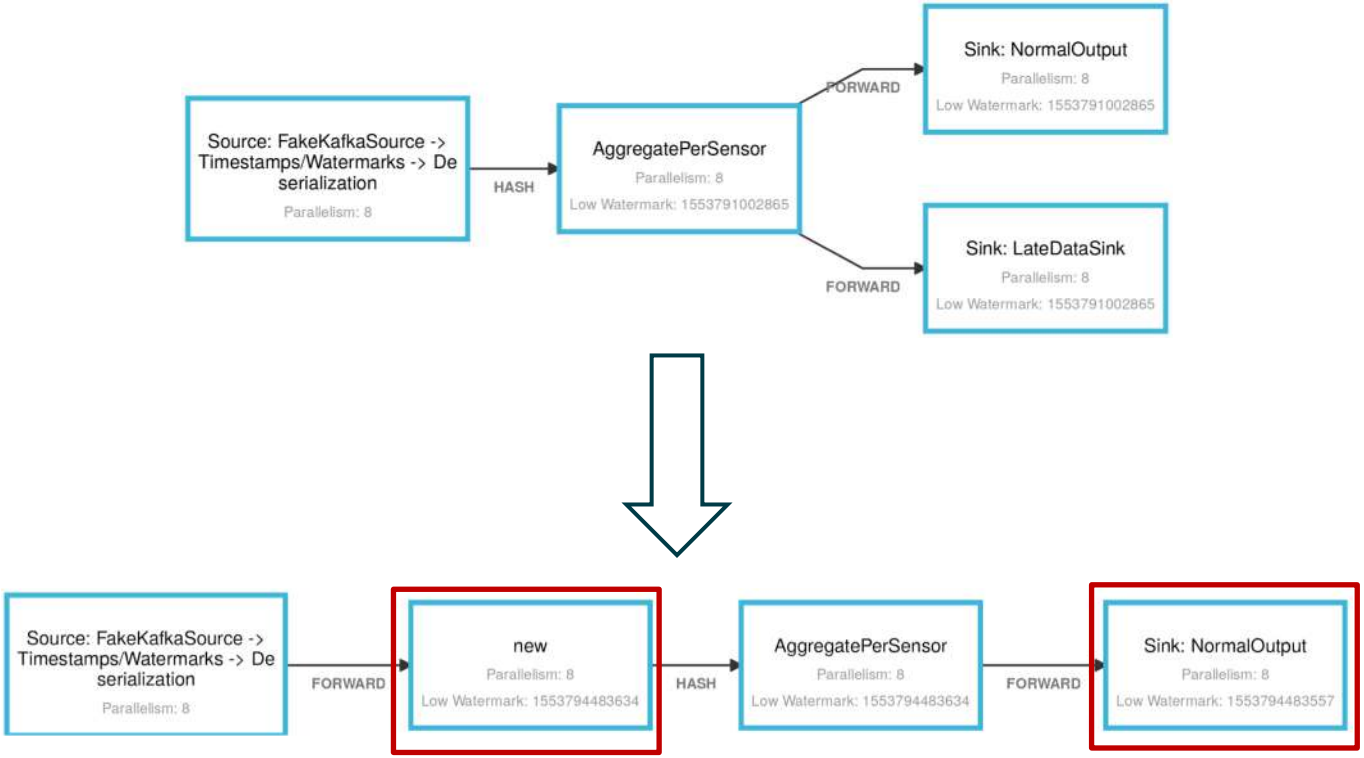
Basics/兼容升级



= Worst Practice

Application Evolution/应用升级

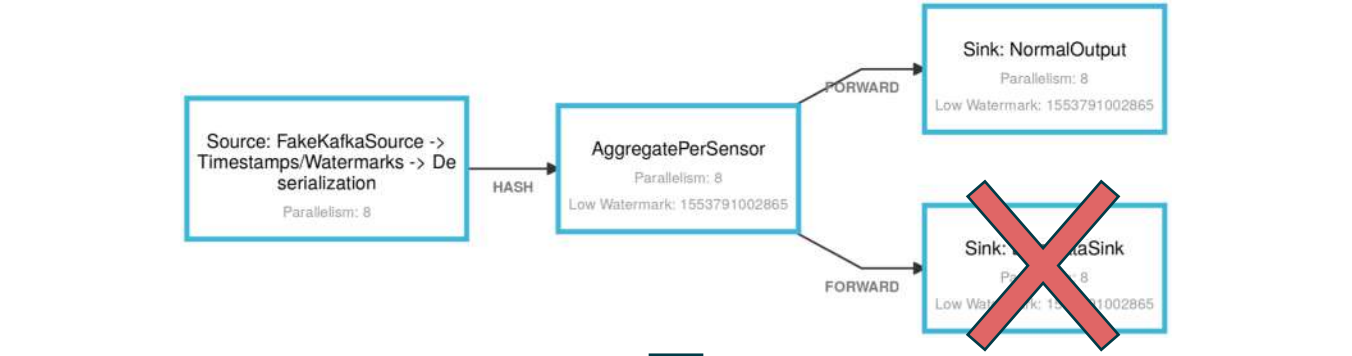
Topology Changes/拓扑改变



= Worst Practice

Application Evolution/应用升级

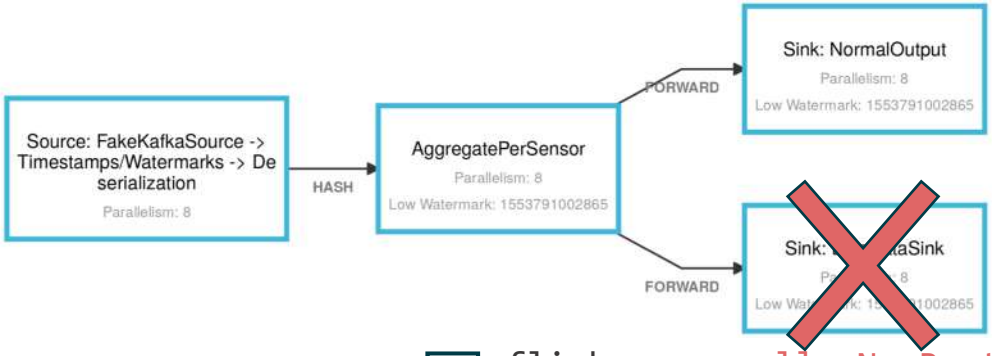
Topology Changes/拓扑改变



= Worst Practice

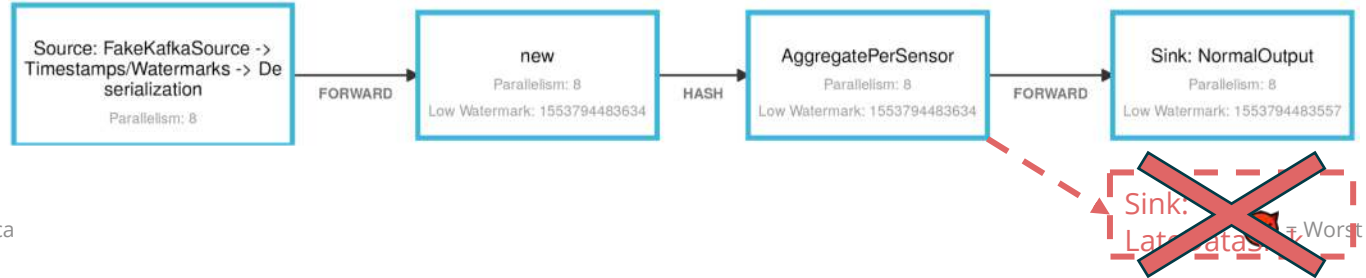
Application Evolution/应用升级

Topology Changes/拓扑改变




```
flink run --allowNonRestoredState <jar-file>
<arguments>
```

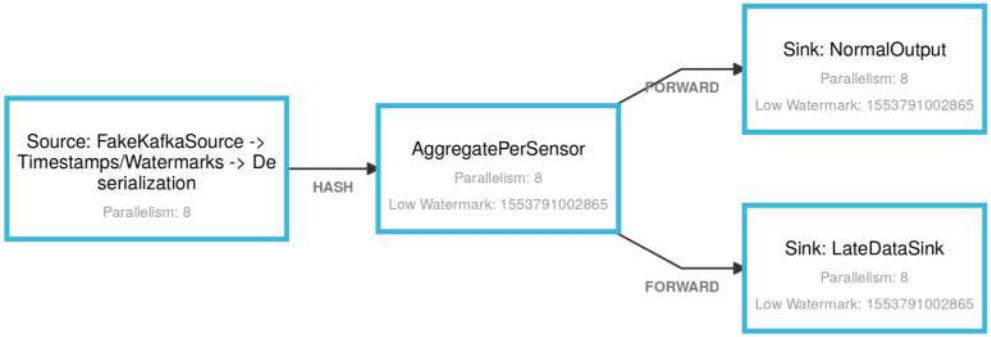
参数详解：https://ci.apache.org/projects/flink/flink-docs-stable/ops/state/savepoints.html#allowing-non-restored-state

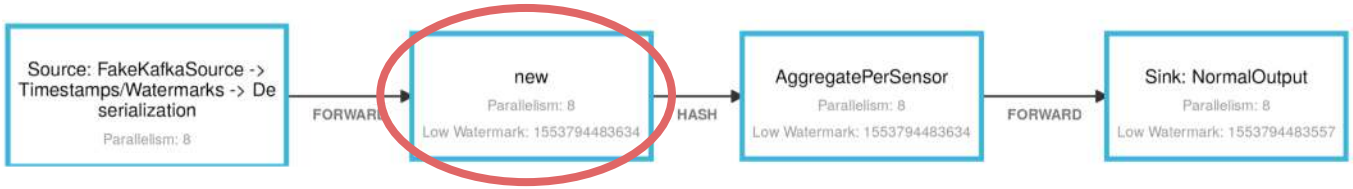


= Worst Practice

Application Evolution/应用升级

Topology Changes/拓扑改变





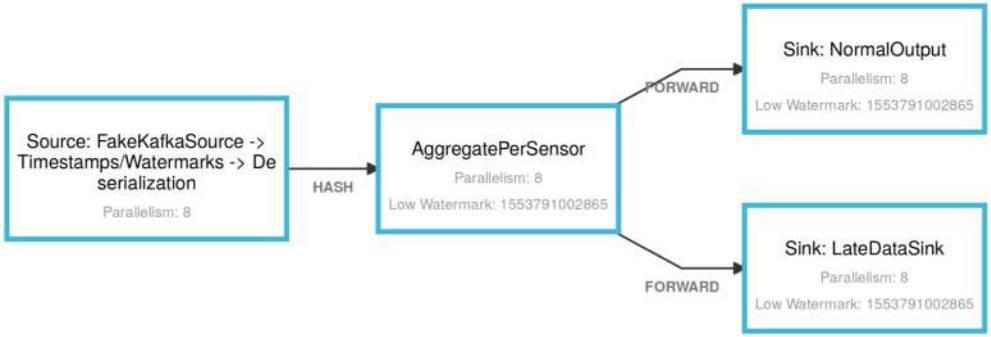
new operator starts → empty state

新增算子->初始状态是空

= Worst Practice

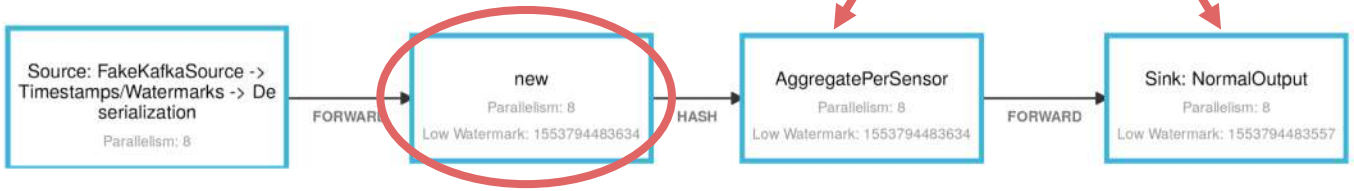
Application Evolution/应用升级

Topology Changes/拓扑改变



Still the same?

目前state文件编号是否有效？

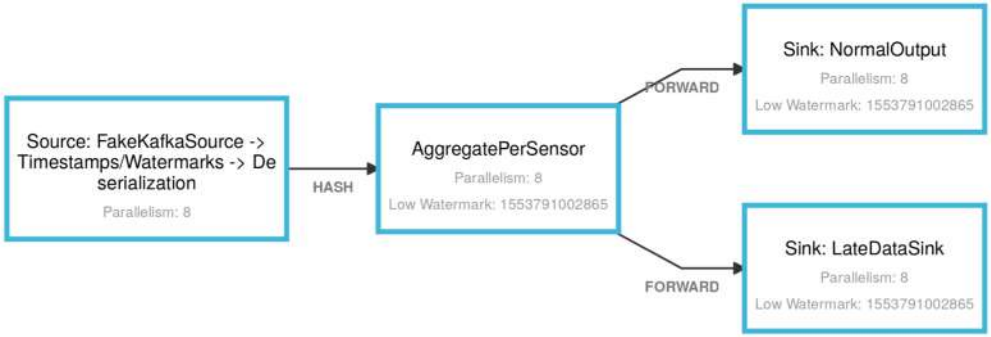


new operator starts → empty state

= Worst Practice

Application Evolution/应用升级

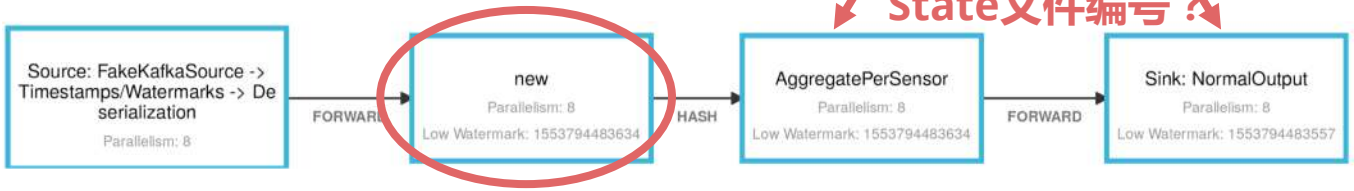
Topology Changes/拓扑改变




```
streamOperator
    .uid("AggregatePerSensor")
```

Still the same?

State文件编号 ?



new operator starts → empty state

= Worst Practice

Application Evolution/应用升级

State Schema Evolution/状态结构升级

- Avro Types ✓
  - https://avro.apache.org/docs/1.7.7/spec.html#Schema+Resolution
- Flink POJOs ✓
  - https://ci.apache.org/projects/flink/flink-docs-release-1.9/dev/stream/state/schema_evolution.html#pojo-types
- Kryo ✗
- Key data types can not be changed/ Key的数据类型是不能改变的

= Worst Practice

Application Evolution/应用升级

State Schema Evolution/状态结构升级

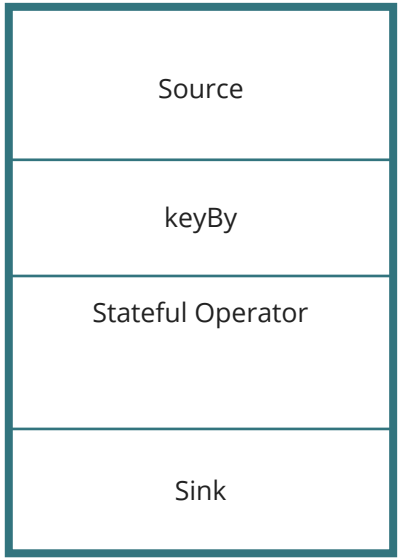
- Avro Types ✓
  - https://avro.apache.org/docs/1.7.7/spec.html#Schema+Resolution
- Flink POJOs ✓
  - https://ci.apache.org/projects/flink/flink-docs-release-1.9/dev/stream/state/schema_evolution.html#pojo-types
- Kryo ✗
- Key data types can not be changed
- State Processor API for the rescue

State Unlocked
Today, 12:20pm - 1:00pm
Track 2
Seth Wiesman, Tzu-Li (Gordon) Tai

= Worst Practice

Feasibility Check/可行性检验

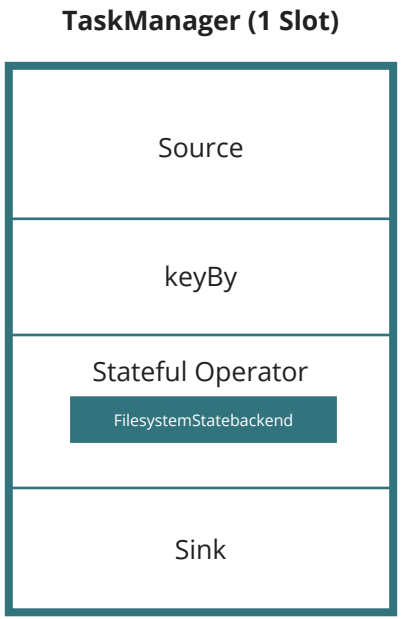
State Size & Network/状态大小和网络



= Worst Practice

Feasibility Check/可行性检验

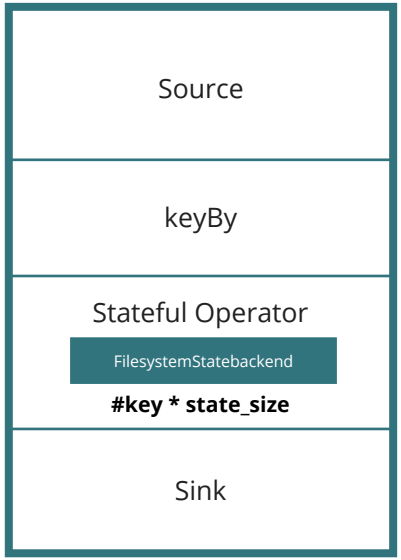
State Size & Network/状态大小和网络



= Worst Practice

Feasibility Check/可行性检验

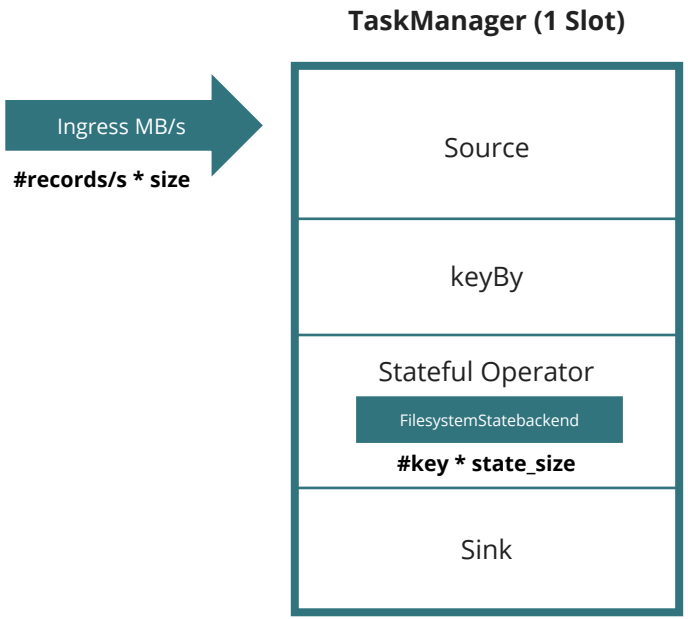
State Size & Network/状态大小和网络



= Worst Practice

Feasibility Check/可行性检验

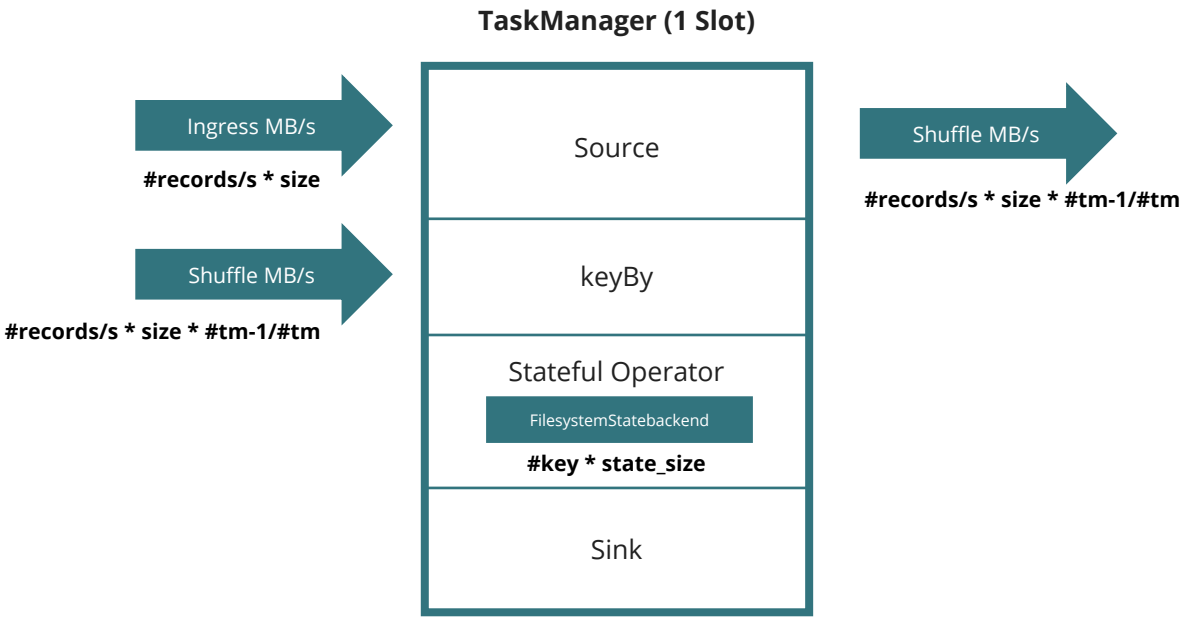
State Size & Network/状态大小和网络



= Worst Practice

Feasibility Check/可行性检验

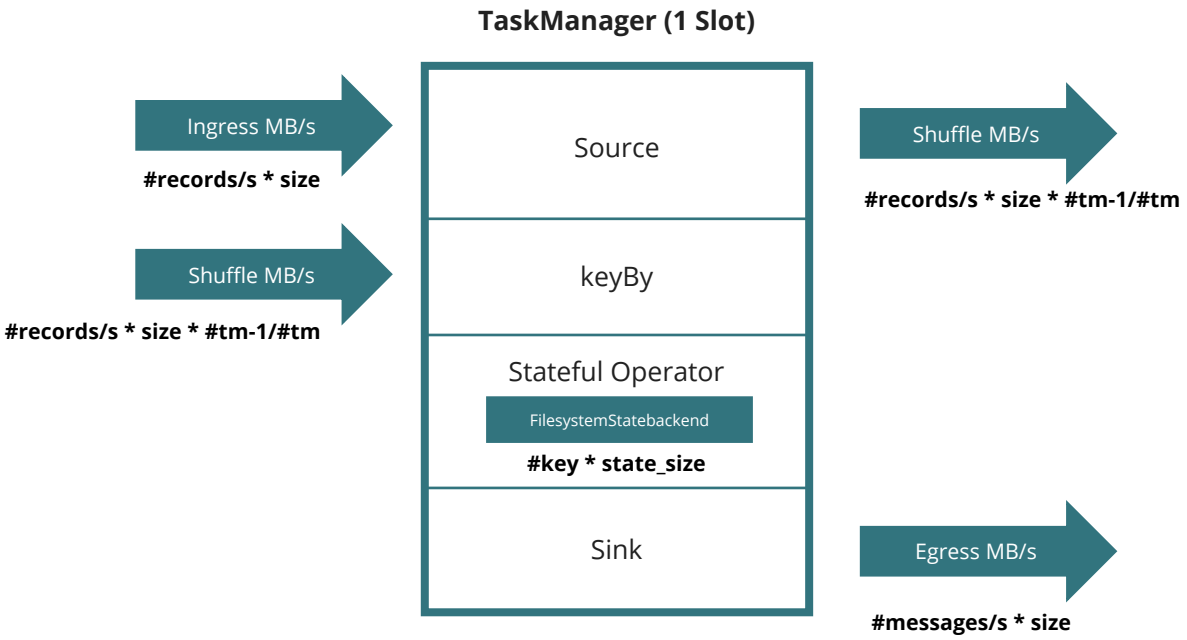
State Size & Network/状态大小和网络



= Worst Practice

Feasibility Check/可行性检验

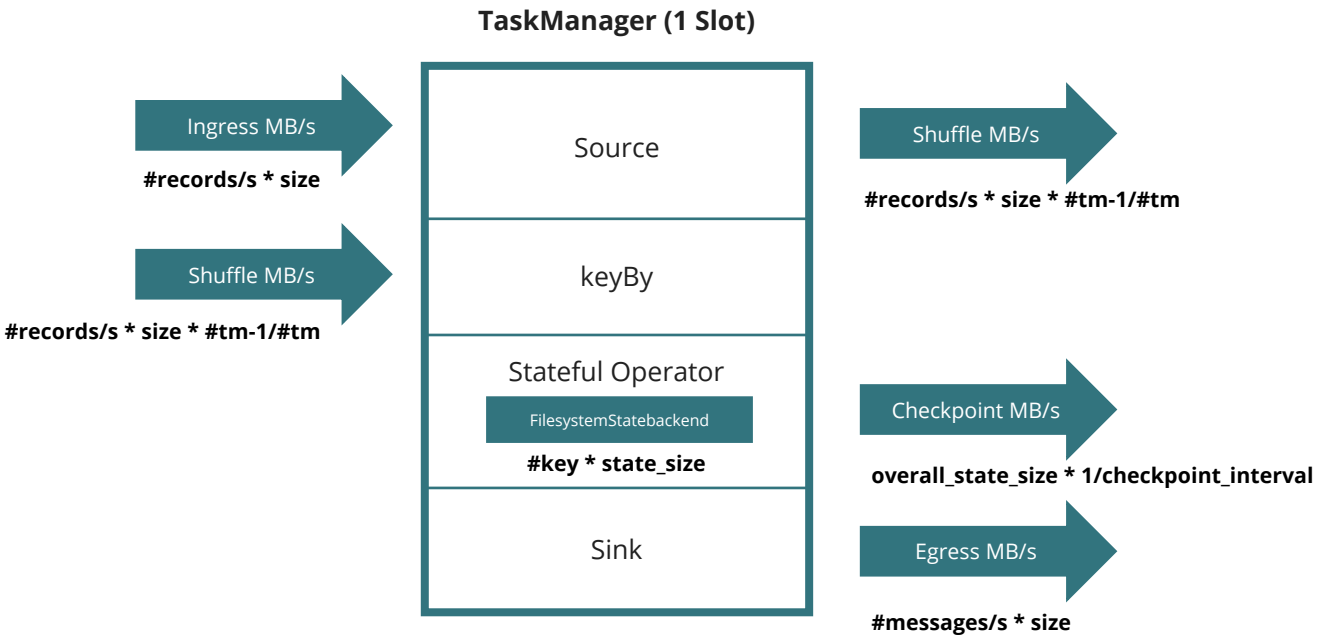
State Size & Network/状态大小和网络



= Worst Practice

Feasibility Check/可行性检验

State Size & Network/状态大小和网络

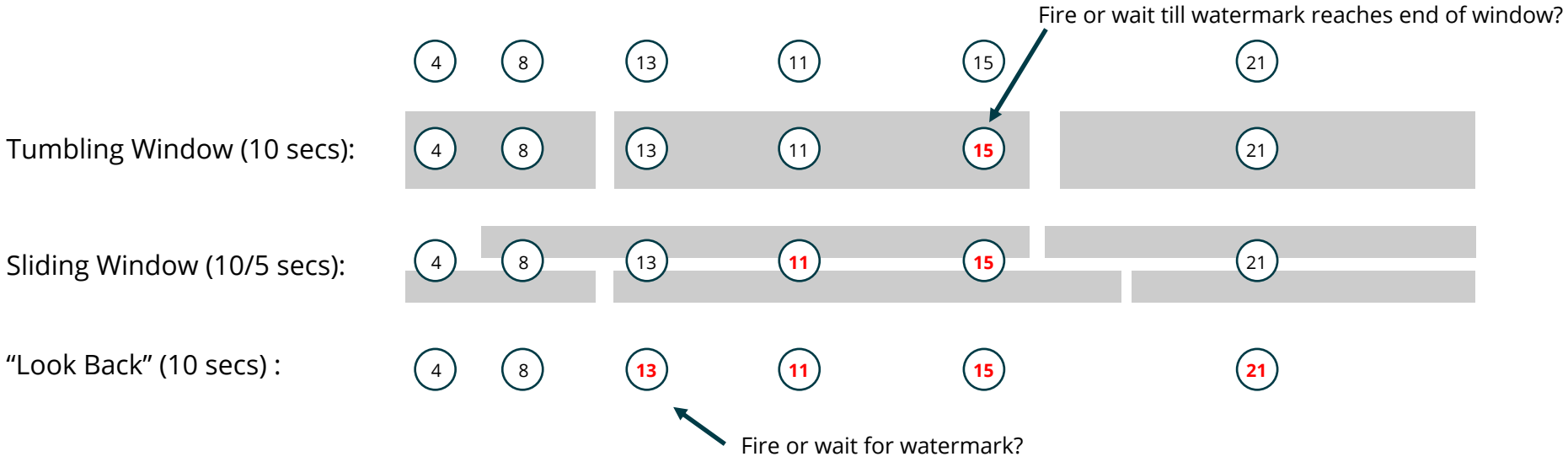


= Worst Practice

Business Requirements/业务需求

Event Time & Out-Of-Orderness/事件时间和乱序

I want to send an alarm when the number of transactions per customer exceeds three in ten seconds.
我想在每个客户的交易数在10秒内超过3笔时发出警报。



= Worst Practice

Business Requirements/业务需求

Batch Processing Requirements/批处理要求

😈 *I receive two files every ten minutes: transactions.txt & quotes.txt*

- *They need to be transformed & joined. / 需要被转换和连接*
- *The output must be exactly one file. / 必须只输出一个文件*
- *The operation needs to be atomic. / 必须原子操作*

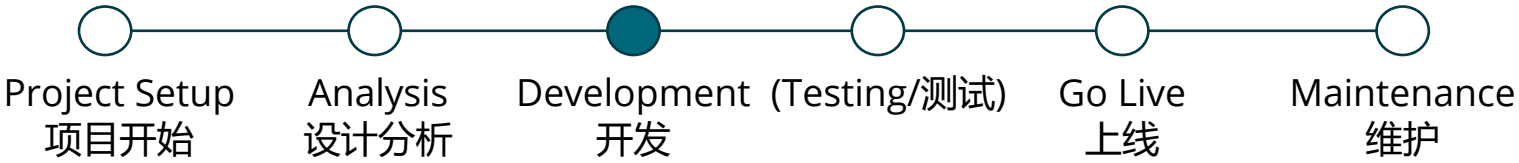
→ Don't try to re-implement your batch jobs as a stream processing jobs.

不要尝试将批处理作业重新实现为流处理作业。

😈 = Worst Practice

Agenda

Don’t use an iterative development process! 😈
不要使用迭代开发过程



😈 = Worst Practice

SQL or DataStream API

Analytics or Application 分析性/应用型

Applications (physical)	**Analytics (declarative)**
Types are Java / Scala classes	Logical Schema for Tables
Transformation Functions	Declarative Language (SQL, Table DSL)
Executes as described	Automatic Optimization
Explicit control over State	State implicit in operations
Explicit control over Time	SLAs define when to trigger
DataStream API	**Table API/SQL**

= Worst Practice

SQL or DataStream API

Table API / SQL Red Flags/ Table API/SQL 红线

- *When upgrading Apache Flink or my application I want to migrate state. / 升级状态*
- *I can not loose late data. / 不能丢失迟到的数据。*
- *I want to change the behaviour of my application during runtime. / 在运行时更改应用程序的行为*

= Worst Practice

Data Types

Worst Practices

- 😈 make use of deeply-nested, complex data types / 使用深度嵌套的复杂数据类型
- 😈 `KeySelectors#getKey` can return any serializable type; use this freedom / 任意类型的Key（kyro）

😈 = Worst Practice

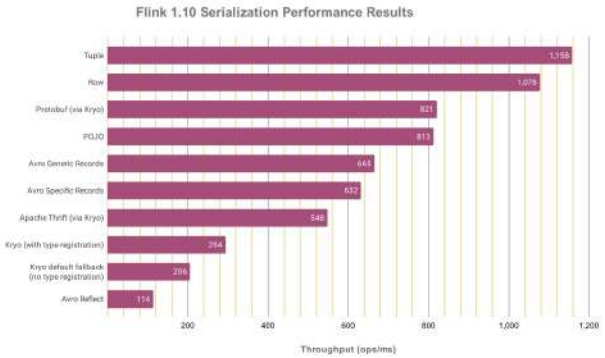
Serialization/序列化

- serialization is not to be underestimated
  不可小觑序列化
- the simpler your data types the better
  数据类型越简单越好
- use Flink POJOs or Avro `SpecificRecords`
  尽量使用POJO和Avro SpecificRecords

- key types matter most / key类型最重要的
  - part of every `KeyedState`
  - part of every `Timer`
- tune locally / 本地测试一下

Serializer	Ops/s
PojoSerializer	813
Kryo	294
Avro (Reflect API)	114
Avro (SpecificRecord API)	632

https://flink.apache.org/news/2020/04/15/flink-serialization-tuning-vol-1.html



Serialization ctd. /序列化-续

```
sourceStream.flatMap(new Deserializer())
            .keyBy("cities")
            .timeWindow()
            .count()
            .filter(new GeographyFilter("America"))
            .addSink(...)
```

don't process data you don't need

- project early
- filter early
- don't deserialize unused fields, e.g.

```
public class Record {
  private City city;
  private byte[] enclosedRecord;
}
```


= Worst Practice

Concurrency/并发性

- **`static`** variables to share state between Tasks

  在任务之间共享状态的静态变量

  - Bugs/问题
  - deadlocks & lock contention (interaction with framework code)
    死锁/锁竞争（与框架交互）
  - synchronization overhead
    同步开销

- spawning threads in user functions

  在用户函数中生成线程

  - complicated error prone (checkpointing)
    复杂易错（检查点）
  - use AsyncStreams to reduce wait times on external IO
    使用异步IO减少等待开销
  - use Timers to schedule tasks
    使用定时器和调度器
  - increase operator parallelism to increase parallelism
    增加运算符并行性度以增加并行性

= Worst Practice

Windowing

```
stream.keyBy("key")
        .window(GlobalWindows.create())
        .trigger(new CustomTrigger())
        .evictor(new CustomEvictor())
        .reduce/aggregate/fold/apply()
```

- Avoid custom windowing
  避免自定义Window
- KeyedProcessFunction usually
  - less error-prone 稳定
  - Simpler 简单

```
stream.keyBy("key")
        .timeWindow(Time.of(30, DAYS), Time.of(5, SECONDS))
        .apply(new MyWindowFunction())
```

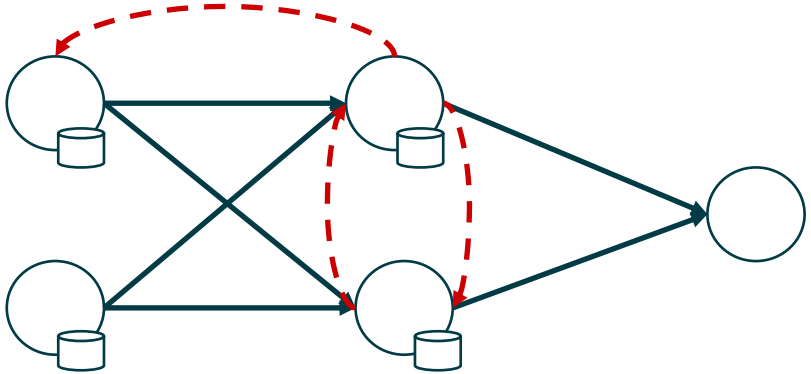
- each record is added to > 500k windows
  每个记录被50w个窗口计算，需要更多资源
- without pre-aggregation
  不能预聚合/增量计算

= Worst Practice

Queryable State

可查询状态

Queryable State for Inter-Task Communication
任务之间的状态查询



- Non-Thread Safe State Access（非线程安全访问）
  - `RocksDBStatebackend` ✓
  - `FilesystemStatebackend` ✗
- Performance?/性能
- Consistency Guarantees?/一致性保障？
- Use Queryable State for debugging and monitoring only /状态查询仅用于调试监控

= Worst Practice

Data Stream API Classics

```
stream.keyBy("key")
      .flatmap(..)
      .keyBy("key")
      .process(..)
      .keyBy("key")
      .timeWindow(..)
```

- `DataStreamUtils#reinterpretAsKeyedStream`（避免多次 `shuffle`）
- **Note:** Stream needs to be partitioned exactly as Flink would partition it.

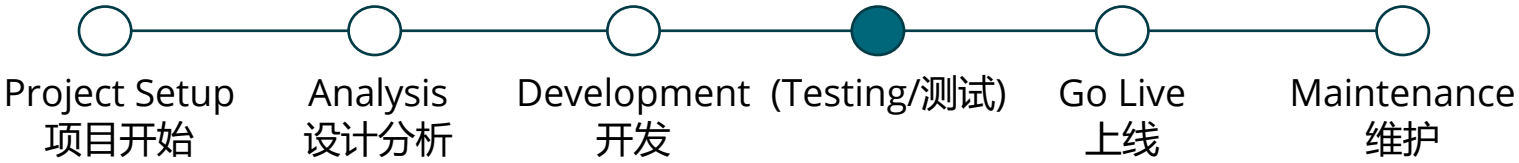
```
public void flatMap(Bar bar, Collector<Foo> out) throws Exception {
  MyParserFactory factory = MyParserFactory.newInstance();
  MyParser parser = factory.newParser();

  out.collect(parser.parse(bar));
}
```

- use `RichFunction#open` for initialization logic

= Worst Practice

Agenda

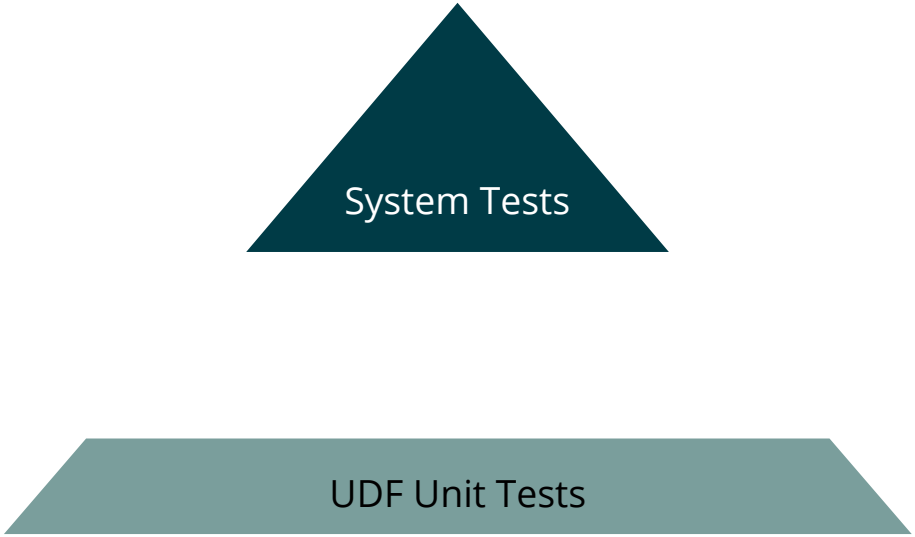
Don’t use an iterative development process! 😈
不要使用迭代开发过程



😈 = Worst Practice

Testing

That's a pyramid! 😈

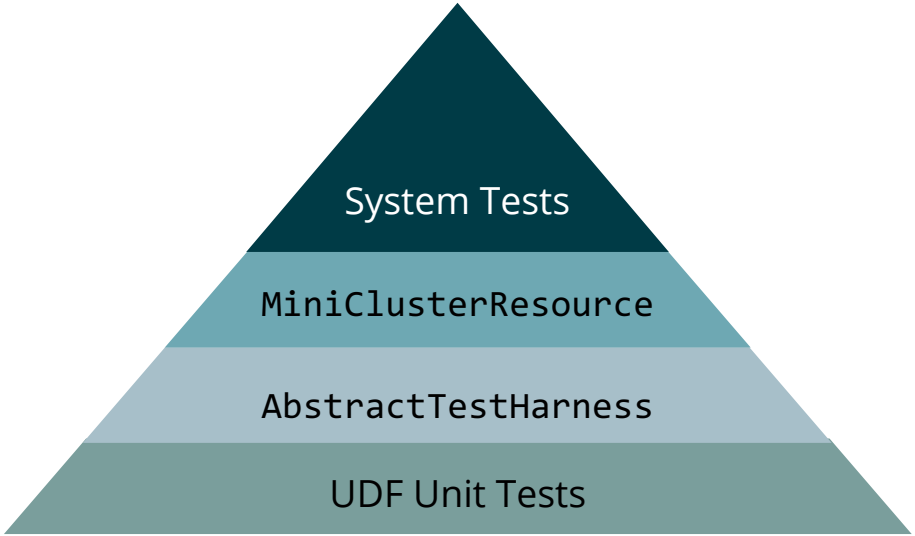


 |

😈 = Worst Practice

Testing

That's a pyramid!



= Worst Practice

Flink’s Test Harnesses

Testing Stateful and Timely Operators and Functions

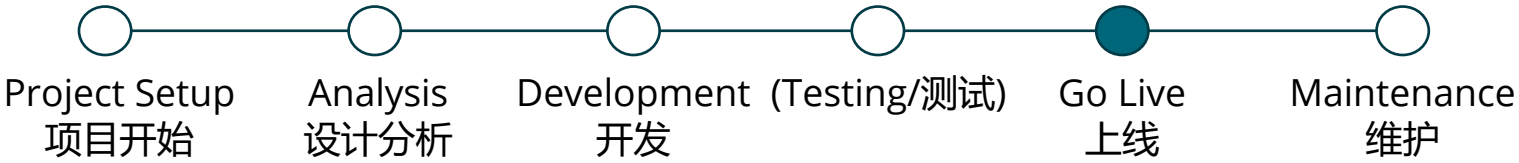
```
@Test
public void testingStatefulFlatMapFunction() throws Exception {

    //push (timestamped) elements into the operator (and hence user defined function)
    testHarness.processElement(2L, 100L);

    //trigger event time timers by advancing the event time of the operator with a watermark
    testHarness.processWatermark(100L);

    //trigger processing time timers by advancing the processing time of the operator directly
    testHarness.setProcessingTime(100L);

    //retrieve list of emitted records for assertions
    assertThat(testHarness.getOutput(), containsInExactlyThisOrder(3L))

    //retrieve list of records emitted to a specific side output for assertions (ProcessFunction only)
    assertThat(testHarness.getSideOutput(new OutputTag<>("invalidRecords")), hasSize(0))
}
```

Examples: https://github.com/knaufk/flink-testing-pyramid

= Worst Practice

Agenda

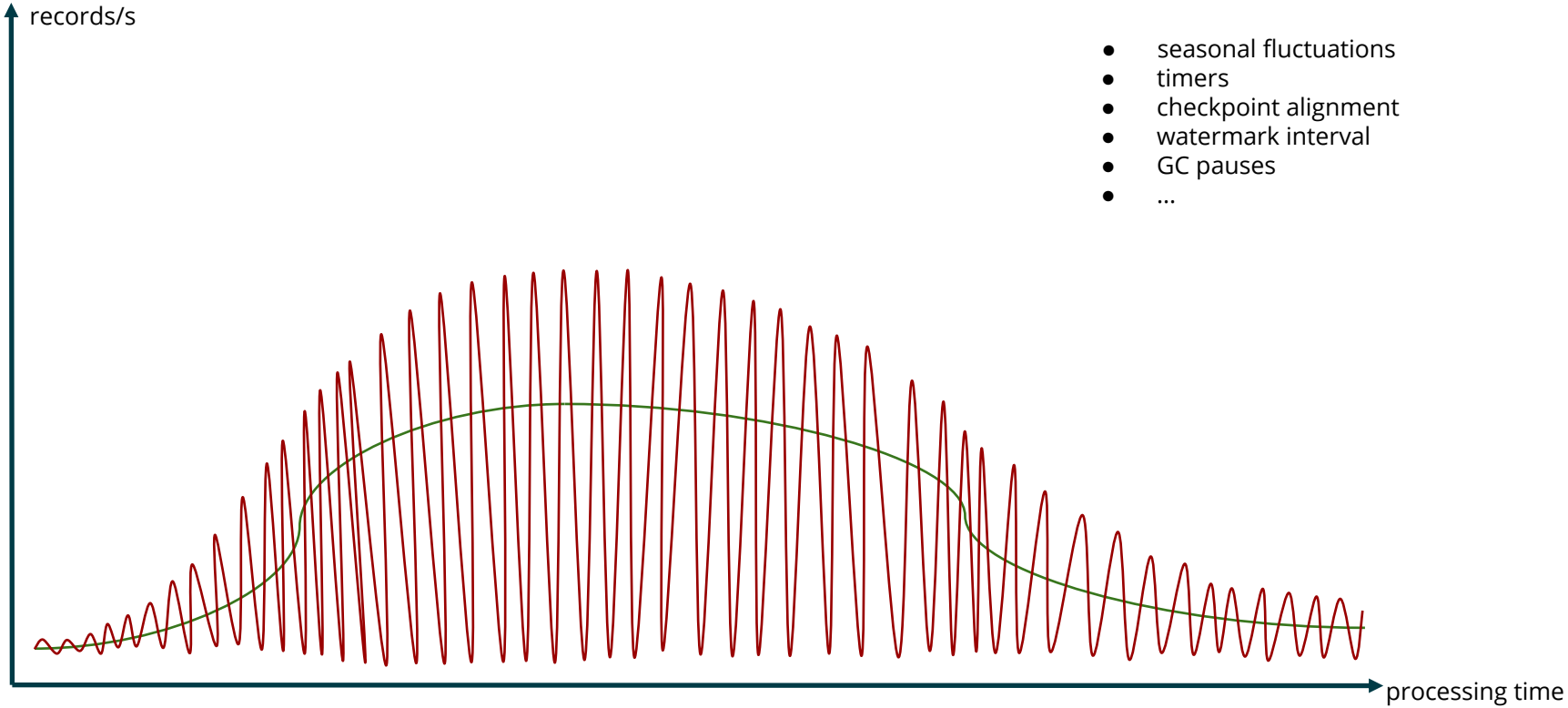
Don’t use an iterative development process! 😈
不要使用迭代开发过程



😈 = Worst Practice

Go Live

Ignore Spiky Loads!/忽略抖动 😈



- seasonal fluctuations
- timers
- checkpoint alignment
- watermark interval
- GC pauses
- ...

😈 = Worst Practice

Go Live

Ignore Spiky Loads! /忽略抖动 😈

catch-up scenario/追数据

- seasonal fluctuations
- timers
- checkpoint alignment
- watermark interval
- GC pauses
- ...

😈 = Worst Practice

Go Live

Monitoring & Metrics

😈 if at all start monitoring now

- don't miss the chance to learn about Flink's runtime during development
- not sure how to start → read [1]

😈 use the Flink Web Interface as monitoring system

- not the right tool for the job → `MetricsReporters` (e.g Prometheus, InfluxDB, Datadog)
- too many metrics can bring JobManagers down 太多Metrics会对JM很大压力

😈 using latency markers in production

- high overhead (in particular with `metrics.latency.granularity: subtask`) 使用 latency是很伤身的
- measure event time lag instead [2]

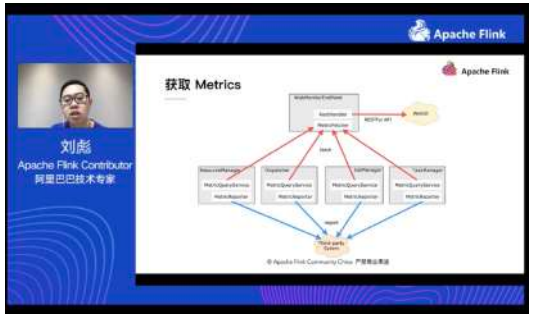
[1] https://flink.apache.org/news/2019/02/25/monitoring-best-practices.html

[2] https://flink.apache.org/2019/06/05/flink-network-stack.html

😈 = Worst Practice

86-91监控&指标

Metrics 和监控



基于 Apache Flink 的监控告警系统

Metric 指标、监控、报警

Flink 反压:延时监控和调参控制

= Worst Practice

Go Live

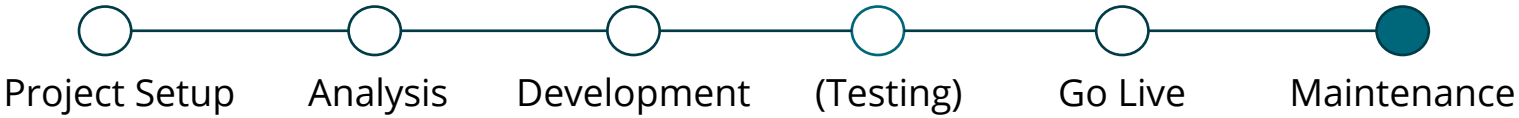
Configuration

- 😈 choosing `RocksDBStatebackend` by default
  - `FilesystemStatebackend` is faster & easier to configure
  - State Processor API can be used to switch later
- 😈 NFS/EBS/etc. as `state.backend.rocksdb.localdir`
  - disk IO ultimately determines how fast you can read/write state
    磁盘IO最终决定了读/写状态的速度
- 😈 playing around with `Slots` and `SlotSharingGroups` (too early)
  - usually not worth it（不要优先选择）
  - leads to less chaining and additional serialization, more network shuffles and lower resource utilization
    导致更少的Chain和更多的序列化，更多的 网络shuffle以及更低的资源利用率

😈 = Worst Practice

Agenda

Don’t use an iterative development process! 😈



😈 = Worst Practice

Maintenance

- with a fast-pace project like Apache Flink, don’t upgrade
  像Apache Flink这样的快节奏项目，请不要升级

= Worst Practice

一些PyFlink/SQL/TableAPI补充 ☺

- 使用TableEnvironment VS StreamTableEnvironment？推荐TableEnvironment 。（分段优化）
- State TTL 未设置，导致 State 无限增长，或者State TTL 设置不结合业务需求，导致数据正确性问题。
- 不支持作业升级，例如增加一个 COUNT SUM会导致作业 state 不兼容。
- 解析JSON时，重复调度UDF，严重影响性能，建议替换成UDTF.
- 多流JOIN的时候，先做小表JOIN ，再做大表JOIN . 目前，flink还没有表的meta信息，没法在plan优化时自动做join reorder。
- 其他？。。。跟多讨论 扫描 二维码 ➔

= Worst Practice

Flink Forward Virtual Conference 2020

Stream Processing I Event Driven I Real Time