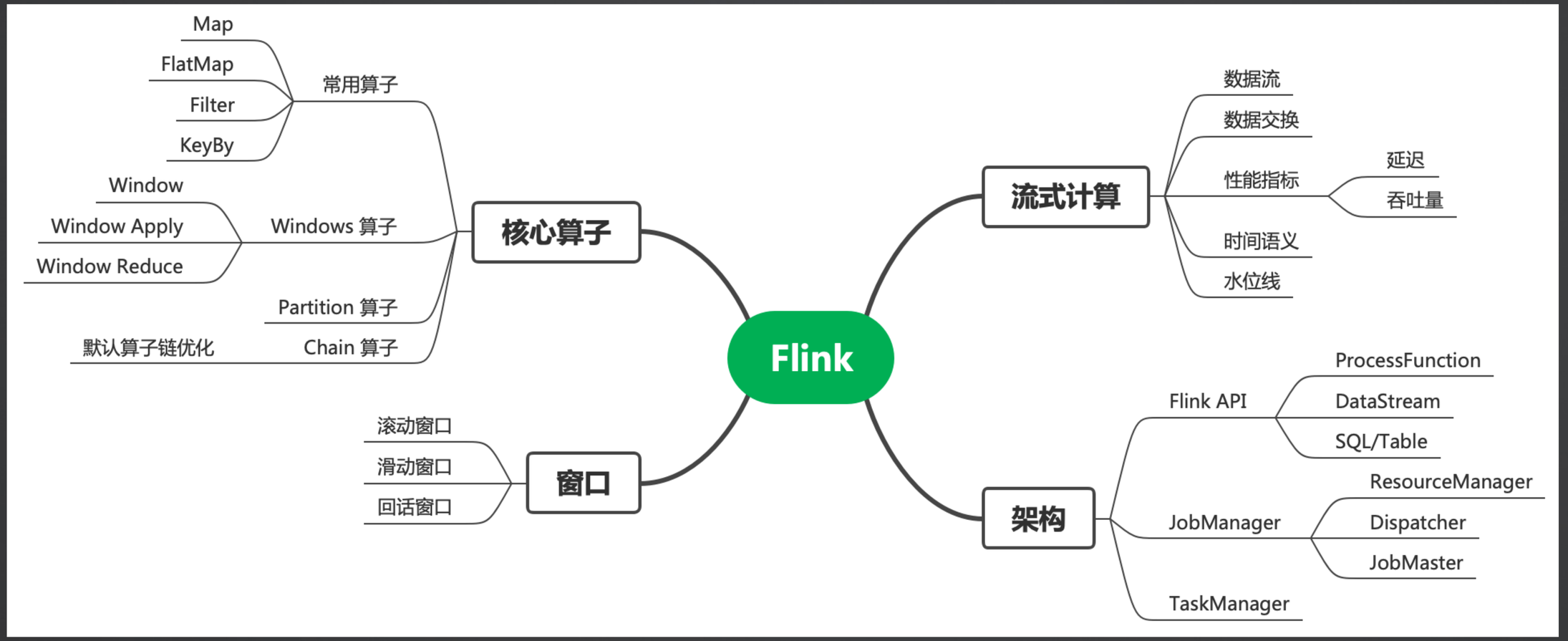
第十一周领教直播 · Flink（上篇）

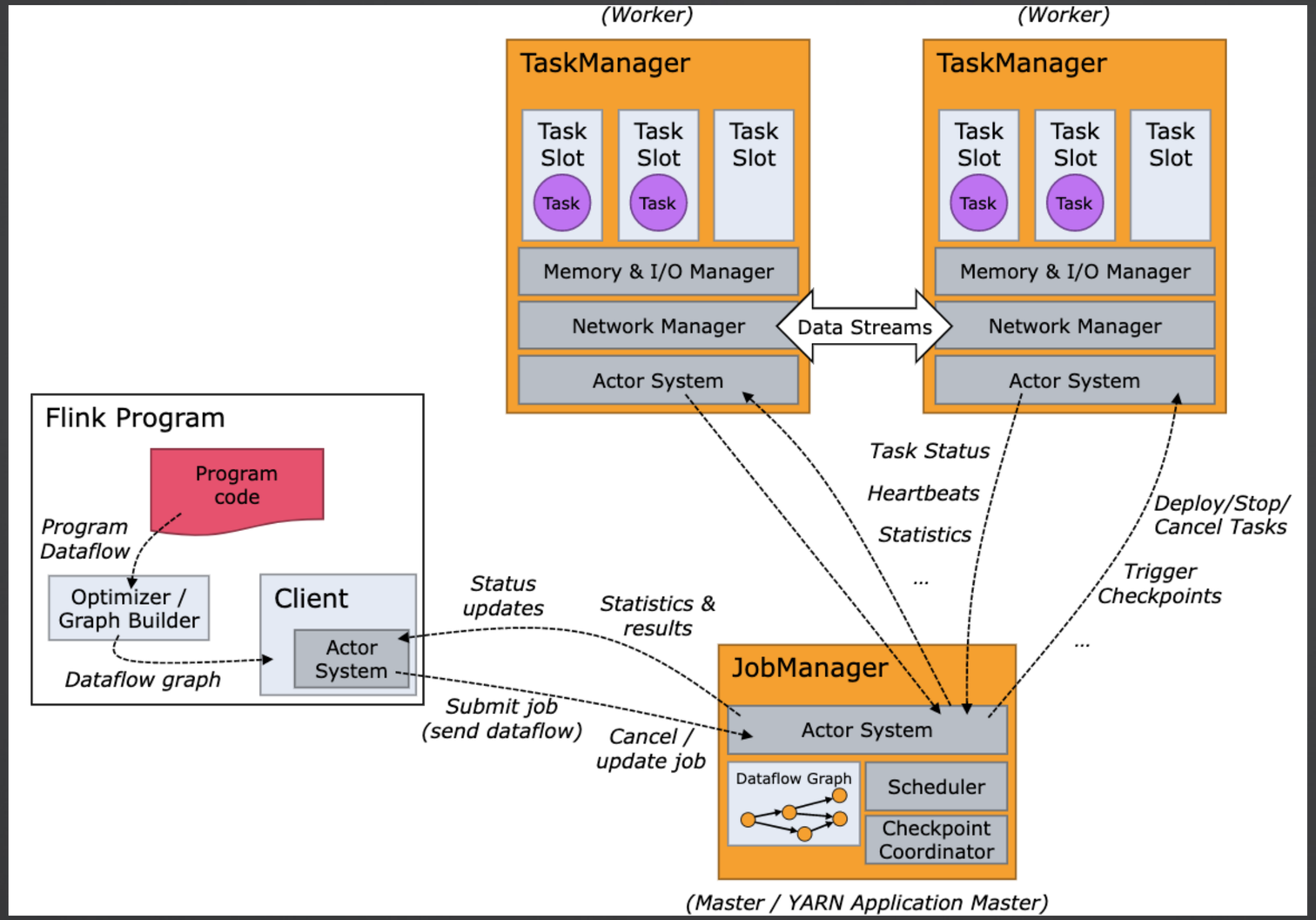
张语

目录

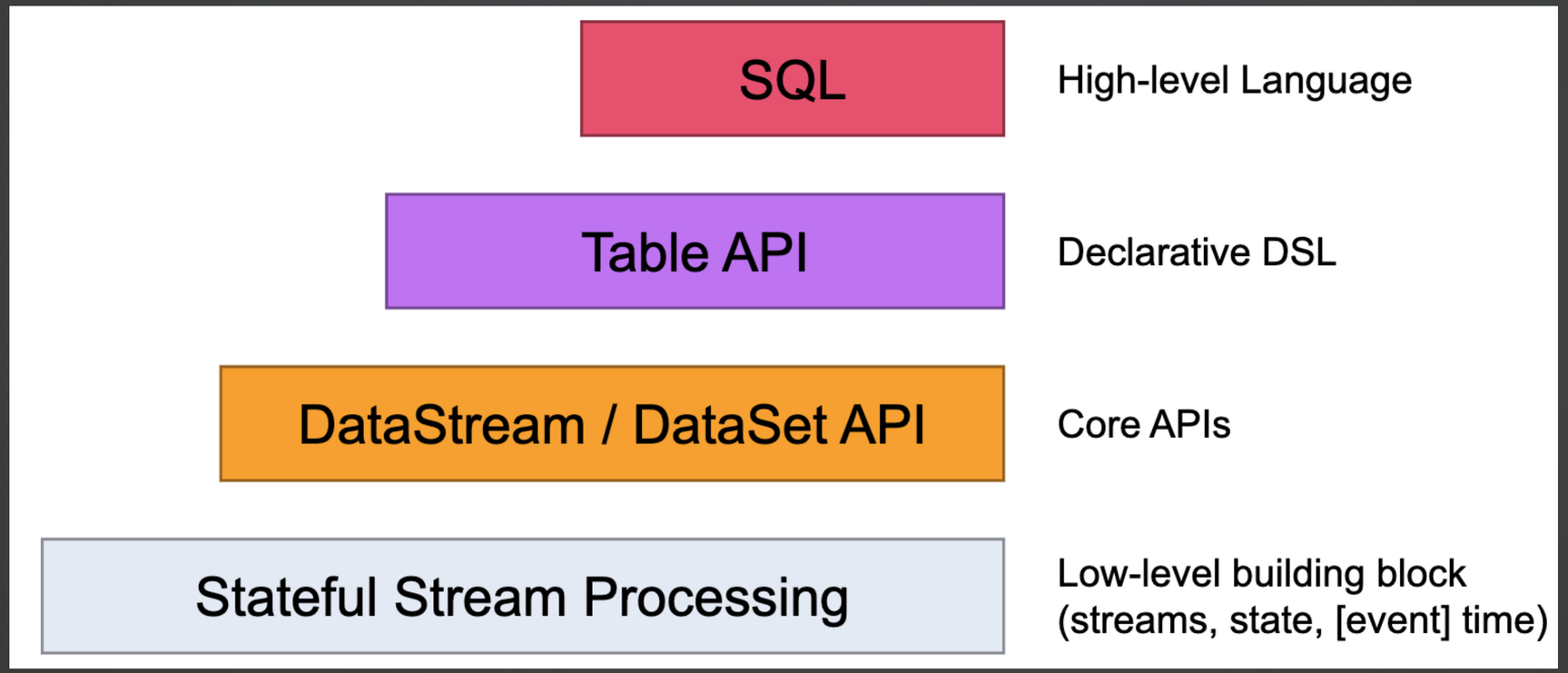
重点内容回顾


Flink
核心算子
常用算子
Map
FlatMap
Filter
KeyBy
Windows 算子
Window
Window Apply
Window Reduce
Partition 算子
Chain 算子
默认算子链优化
窗口
滚动窗口
滑动窗口
回话窗口
流式计算
数据流
数据交换
性能指标
延迟
吞吐量
时间语义
水位线
架构
Flink API
ProcessFunction
DataStream
SQL/Table
JobManager
ResourceManager
Dispatcher
JobMaster
TaskManager

Flink 架构图


(Worker)
(Worker)
TaskManager
TaskManager
Task Slot
Task
Memory & I/O Manager
Network Manager
Actor System
Data Streams
Flink Program
Program code
Program Dataflow
Optimizer / Graph Builder
Client
Actor System
Dataflow graph
Status updates
Statistics & results
Submit job (send dataflow)
Cancel / update job
Task Status
Heartbeats
Statistics
...
Deploy/Stop/ Cancel Tasks
Trigger Checkpoints
...
JobManager
Actor System
Dataflow Graph
Scheduler
Checkpoint Coordinator
(Master / YARN Application Master)

Flink API


SQL
High-level Language
Table API
Declarative DSL
DataStream / DataSet API
Core APIs
Stateful Stream Processing
Low-level building block
(streams, state, [event] time)

目录

作业

report(transactions).executeInsert("spend_report");

将 transactions 表经过 report 函数处理后写入到 spend_report 表。

每分钟（或小时）计算在五分钟（或小时）内每个账号的平均交易金额（滑动窗口）？使用分钟还是小时作为单位均可。

作业

```
public static Table report(Table transactions) {
    Table table = transactions.window(Slide.over(lit( v: 5).minutes()) SlideWithSize
                      .every(lit( v: 1).minutes()) SlideWithSizeAndSlide
                      .on($( name: "transaction_time")) SlideWithSizeAndSlideOnTime
                      .as( alias: "log_ts"))
              .groupBy($( name: "account_id"),$( name: "log_ts"))
              .select($( name: "account_id"),
                      $( name: "log_ts").start().as( name: "log_ts"),
                      $( name: "amount").avg().as( name: "amount"));
    return table ;
}
```

作业

演示环节

- 创建检查点和保存点目录

```
mkdir -p /tmp/flink-checkpoints-directory
mkdir -p /tmp/flink-savepoints-directory
```

- docker

```
docker-compose build
docker-compose up -d
docker-compose down -v
```

- Flink UI

http://localhost:8082/#/overview

- 查看结果

```
docker-compose exec mysql mysql -Dsql-demo -usql-demo -pdemo-sql
use sql-demo;
select count(*) from spend_report;
```

目录

Flink CEP

Flink CEP 是在 Flink 之上实现的复杂事件处理（CEP）库，实现在事件流中检测事件模式

通过分析事件间的关系，利用过滤、关联、聚合等技术，根据事件间的时序关系制定检测规则，持续地从事件流中查询出符合要求的事件序列，最终得到更复杂的复合事件.

处理事件的规则，被叫作模式（Pattern）。

Pattern API

- Pattern API 用于对输入流数据进行复杂事件规则定义，用来提取符合规则的事件序列。

Pattern 分类

- 个体模式（Individual Patterns）

包括单例模式和循环模式。单例模式只接收一个事件，而循环模式可以接收多个事件。

比如在模式“a b+ c? d”中，a，d是单例模式，b+，c？是循环模式

- 组合模式（Combining Patterns）

是多个个体模式组合

- 模式组（Groups of patterns）

可以定义一个模式序列作为begin, followedBy, followedByAny和next的条件。模式序列将被视为逻辑上的匹配条件，并将返回一个GroupPattern

模式序列

- 严格近邻

所有事件按照严格的顺序出现，中间没有任何不匹配的事件，由 .next() 指定。

- 宽松近邻

允许中间出现不匹配的事件，由 .followedBy() 指定。

- 非确定性宽松近邻

进一步放宽条件，之前已经匹配过的事件也可以再次使用，由 .followedByAny() 指定。

- 不希望出现某种近邻关系

.notNext()：不想让某个事件严格紧邻前一个事件发生。

.notFollowedBy()：不想让某个事件在两个事件之间发生。

模式的检测

指定要查找的模式序列后，可以将其应用于输入流以检测潜在匹配事件

调用 CEP.pattern()，给定输入流和模式，就能得到一个 PatternStream

```
DataStream<Event> input = ...;
Pattern<Event, ?> pattern = ...;
EventComparator<Event> comparator = ...; // optional

PatternStream<Event> patternStream = CEP.pattern(input, pattern, comparator);
```

匹配事件的提取

创建 PatternStream 之后，可以使用 PatternProcessFunction 从检测到的事件序列中提取事件

```
class MyPatternProcessFunction<IN, OUT> extends PatternProcessFunction<IN, OUT> {
    @Override
    public void processMatch(Map<String, List<IN>> match, Context ctx, Collector<OUT> out) throws Exception;
        IN startEvent = match.get("start").get(0);
        IN endEvent = match.get("end").get(0);
        out.collect(OUT(startEvent, endEvent));
    }
}
```

示例

```
DataStream<Event> input = ...;

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");
            }
         }
    );
```

```
PatternStream<Event> patternStream = CEP.pattern(input, pattern);

DataStream<Alert> result = patternStream.process(
    new PatternProcessFunction<Event, Alert>() {
        @Override
        public void processMatch(
                Map<String, List<Event>> pattern,
                Context ctx,
                Collector<Alert> out) throws Exception {
            out.collect(createAlertFrom(pattern));
        }
    });
```

目录

电商场景

1. 用户浏览收藏，加购物车，购买，商品
2. 平台推广 App，需要看市场营销效果
3. 失败恶意行为，比如：恶意登陆（灰产撞库）
4. 自动取消已超时未支付的订单，及时归还商品库存

实时热门商品统计

统计近1个小时内的热门商品，每5分钟更新一次，热门商品取 Top 100

热门度使用浏览（“PV”）来衡量

方案

1. 采用滚动窗口
2. 用商品 ID 分组，统计 PV
3. 计算 Top N

字段名	数据类型	说明
user_id	Long	用户 ID
item_id	Long	商品 ID
category_id	Int	商品所属类别 ID
behavior	String	用户行为类型，分为：PV, buy, cart, fav
timestamp	Long	行为发生的时间戳，单位：秒

实时热门商品统计

演示环节

使用 flink-quickstart-java 创建项目

```
mvn archetype:generate                \
  -DarchetypeGroupId=org.apache.flink   \
  -DarchetypeArtifactId=flink-quickstart-java \
  -DarchetypeVersion=1.13.6 \
  -DgroupId=com.jk.flink   \
  -DartifactId=mall-stat \
  -Dversion=1.0-SNAPSHOT \
  -DinteractiveMode=false
```

实时流量统计

统计 PV 和 UV

从埋点日志中，统计实时的 PV 和 UV

统计每小时的访问量（PV），访问人数（UV）

字段名	数据类型	说明
IP	String	访问 IP
user_id	Long	用户 ID
timestamp	Long	访问时间，单位：秒
method	String	访问方法，分为：GET, POST, PUT/DELETE
url	String	访问的 URL

方案：

1. 采用1小时的滑动窗口
2. PV：计数
3. UV：用户 ID 去重，可用用 Set、Bitmap、HyperLoglog（近似） 去重

市场营销分析

APP市场推广统计

从埋点日志中，统计APP市场推广的数据指标

按照不同的推广渠道，统计每小时各个渠道各行为的数据

1. 采用1小时滚动窗口
2. 渠道、行为分组（keyBy）
3. 统计次数

字段名	数据类型	说明
user_id	Long	用户 ID
channel	String	渠道，如：apple、oppo、wechat 等
behavior	String	用户行为类型，分为：click, download、install、uninstall
timestamp	Long	行为发生的时间戳，单位：秒

恶意登录监控

用户在短时间内频繁登录失败，有可能恶意攻击的可能

同一用户在两秒内连续登录失败，需要报警

方案：

1. 采用 Flink CEP 实现

演示环节

订单支付实时监控

用户下单之后，应设置订单失效事件，以提高用户支付的意愿，并降低系统风险

用户下单后15分钟未支付，则输出监控信息

方案

1. 采用 Flink CEP 的时间限制模式和匹配超时实现

演示环节

QA

极客时间 训练营

THANKS