流处理基准测试

Introduction to a New Streaming Benchmark

演讲人名： 金浩
公司职位： 英特尔软件工程师

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

FLINK FORWARD

Contents
目录

基准测试的介绍

The introduction of benchmark

01

要点概述

key points

01 数据集

Data set

02 工作负载

Workload

03 度量指标

Metrics

04 经典的基准测试

Classical benchmark

数据集
Dataset

FLINK FORWARD

I 结构化数据 Structured data

传统的关系数据模型，可用二维表结构表示。典型场景有电商交易、财务系统、医疗 HIS 数据库、政务信息化系统等。

Traditional relational data mode, can be represented by a two-dimensional table structure. Typical scenarios include e-commerce transactions, financial systems, medical HIS databases, government information systems, etc.

II 半结构化数据 Semi-structured data

类似 XML、HTML 之类，自描述，数据结构和内容混杂在一起。典型应用场景有邮件系统、Web 搜索引擎存储、教学资源库、档案系统等等。

Similar to XML, HTML, etc., self-describing, data structure and content mixed together Typical application scenarios are mail systems, web search engine storage, teaching resource libraries, file systems, etc.

III 非结构化数据 Unstructured Data

各种文档、图片、视频和音频等。典型的应用有视频网站、图片相册、交通视频监控等等。

Various documents, pictures, videos, and audio. Typical applications include video sites, photo albums, traffic video surveillance, and more.

工作负载 Workload

FLINK FORWARD

1）密集计算类型 Intensive Computing type

CPU 密集型计算、IO 密集型计算、网络密集型计算
CPU-intensive computing, IO-intensive computing, network-intensive computing

2）计算范式 Computing paradigm

SQL、批处理、流计算、图计算、机器学习
SQL, batch processing, stream computing, graph computing, machine learning

3）计算延迟 Computing delay

在线计算、离线计算、实时计算
Online calculation, offline calculation, real-time calculation

4）应用领域 Application field

搜索引擎、社交网络、电子商务、地理位置、媒体、游戏
Search engine, social network, e-commerce, geography, media, games

度量指标
Metrics

FLINK FORWARD

1）架构角度
Architectural perspective

浮点型操作密度、整数型操作密度、指令中断、cache 命中率、TLB 命中
Floating point operation density, integer operation density, instruction interrupt, cache hit ratio, TLB hit

2）系统执行时间、吞吐和延迟性的角度
System execution time, throughput, and latency perspective

以 Spark 为例，Job 作业执行时间、 Job 吞吐量、Stage执行时间、Stage 吞吐量、Task 执行时间、Task 吞吐量，以及实时计算数据处理的延迟性
Take Spark as an example, Job job execution time, Job throughput, Stage execution time, Stage throughput, Task execution time, Task throughput, and delay in real-time calculation data processing

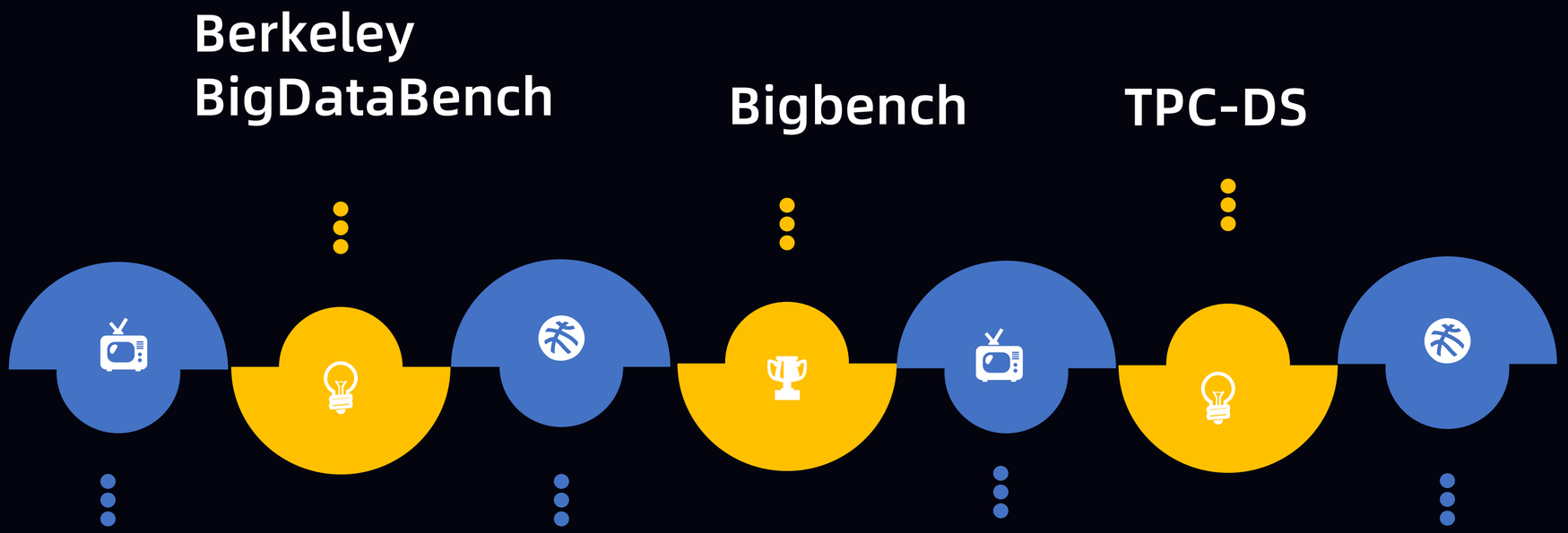
3）系统资源利用率的角度
System resource utilization

CPU 在指定时间段的利用率、内存在指定时间段的利用率、磁盘在指定时间段的利用率、网络带宽在指定时间段的利用率
The utilization of the CPU in the specified time period, the utilization of the memory in the specified time period, the utilization of the disk in the specified time period, and the utilization of the network bandwidth in the specified time period

4）扩展性的角度
Scalability angle

数据量扩展、集群节点数据扩展（scale out）、单机性能扩展（scale up）
Data volume expansion, cluster node data out (scale out), stand-alone performance scale (scale up)

经典的基准测试 Classic benchmark



流处理的基准测试 Streaming Benchmark

FLINK FORWARD

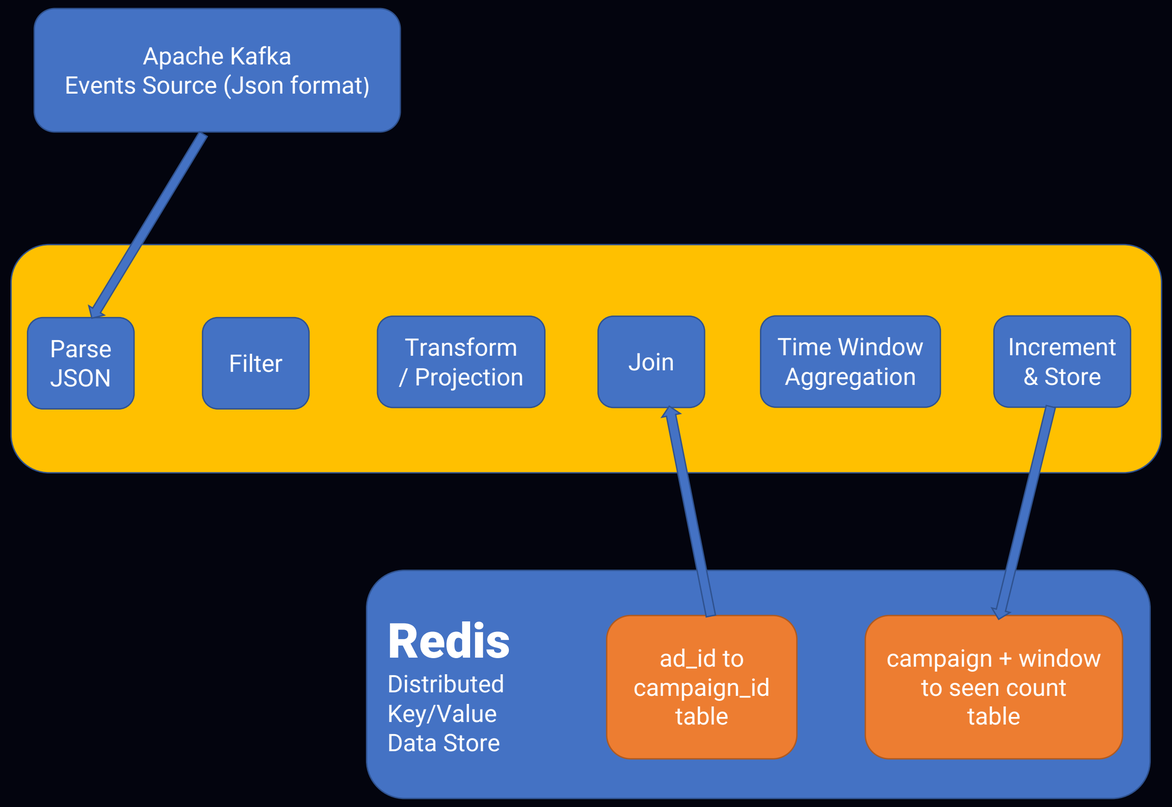
I 流处理框架

- Flink
- Spark Structured Streaming
- Storm

II 流数据基准测试工具

- Yahoo streaming benchmark
- Hibench

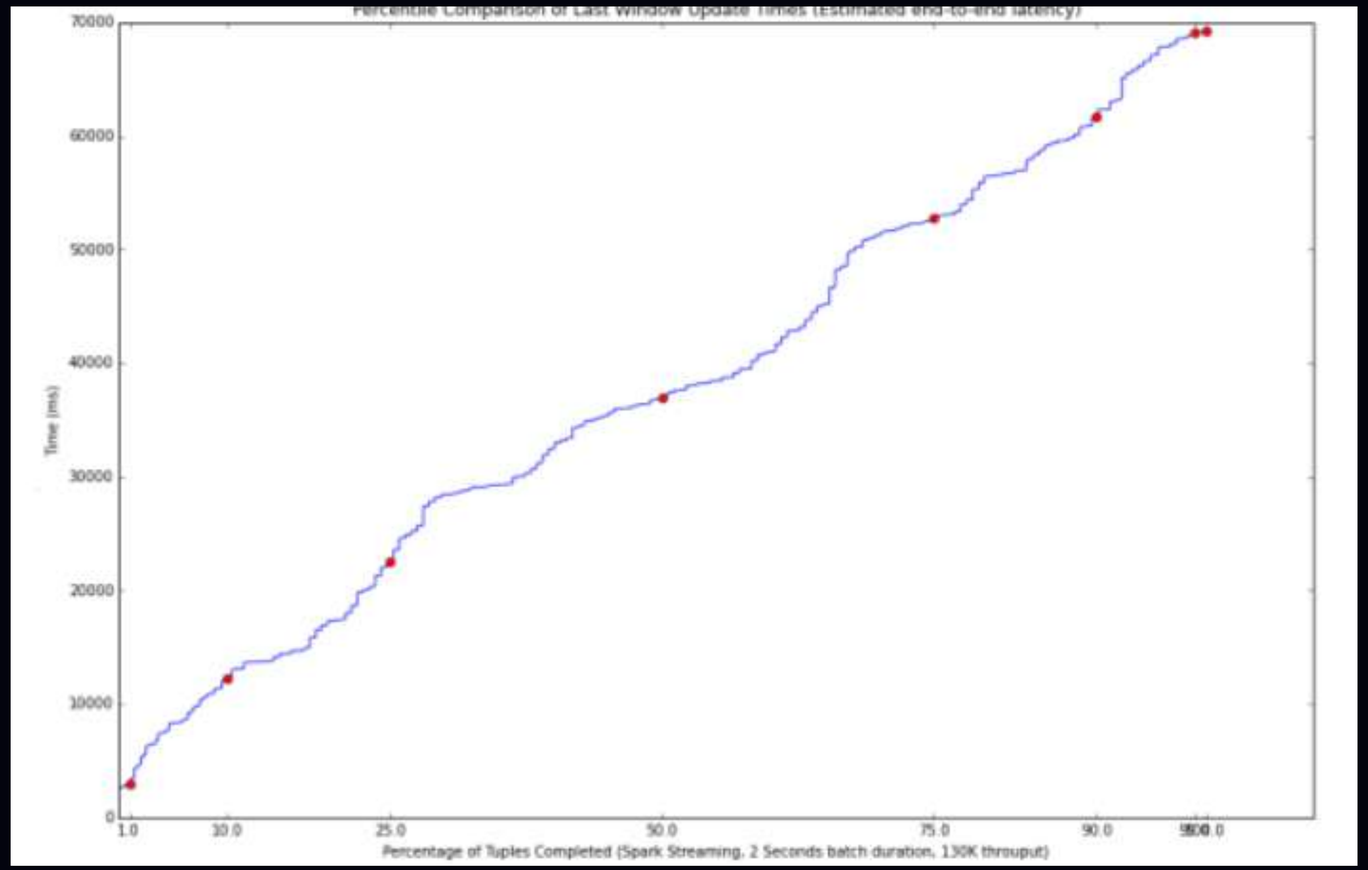
Yahoo Streaing benchmark



Flink Benchmark 数据处理部分

```
messageStream
        .rebalance()
        // Parse the String as JSON
        .flatMap(new DeserializeBolt())

        //Filter the records if event type is "view"
        .filter(new EventFilterBolt())

        // project the event
        .<Tuple2<String, String>>project(2, 5)

        // perform join with redis data
        .flatMap(new RedisJoinBolt())

        // process campaign
        .keyBy(0)
        .flatMap(new CampaignProcessor());
```

Yahoo Streaing benchmark



度量指标：延迟

Metrics: Lantency

延迟计算：窗口最后一条数据处理时间 - 窗口的起始时间（近似的处理）- 窗口长度

Delay calculation: the last data processing time of the window - the start time of the window (approximate processing) - the length of the window

缺点：

Disadvantages

1）测试集不够丰富，只涉及流处理的简单应用场景

The test set is not rich enough, and only involves a simple application scenario of stream processing.

2）数据生成速度过慢，测试集负载压力小

Data generation speed is too slow, test set load pressure is small

3）度量指标只能适用少部分场景

Metrics can only be applied to a small number of cases

4）代码方式不宜用户理解

The code method is not suitable for users to understand

Hibench

1) FixedWindow

2) Identity

3) Repartition

4) Wordcount

测试集只包括几个经典的案例，不能反映真实的应用场景。

The test set only includes a few classic cases and does not reflect the real application scenario.

流处理的基准测试存在的问题
The Problem of Streaming Benchmark

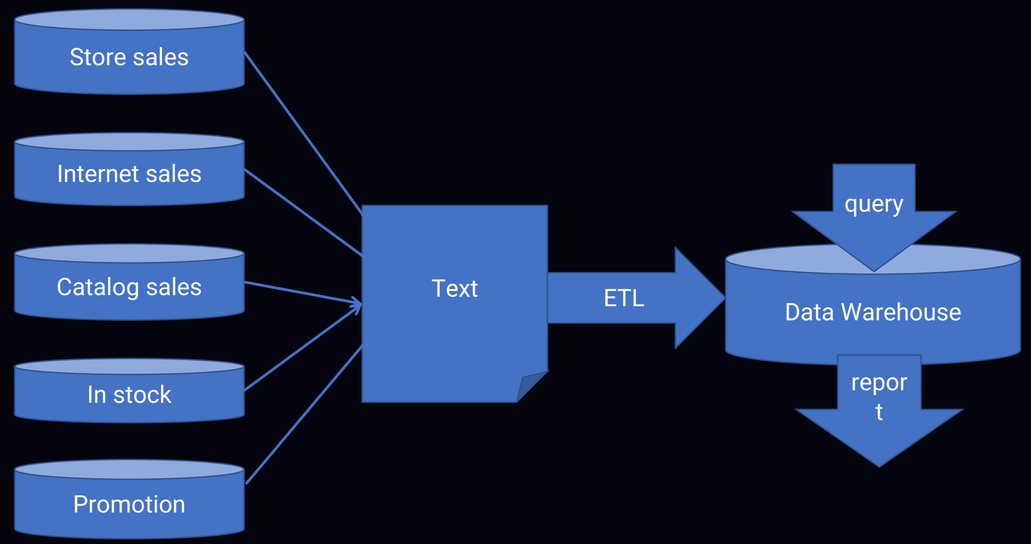
- 测试集不能反映真实的应用场景
  The test set does not reflect the real application scenario.
- 集群测试的负载不够大
  The load of the cluster test is not big enough
- 度量指标只适用少部分计算场景
  Metrics only apply to a small number of calculation cases
- 测试集基于代码的方式，用户学习成本较高
  Test set code-based approach, user learning costs are higher

TPC-DS：作为大数据批处理领域最流行基准测试之一，模拟了数据仓库，反应真实的应用场景

TPC-DS

数据模型：
Data model:

星型、雪花型等多维数据模式。包含 7 张事实表，17 张维度表
Multidimensional data modes such as star and snowflake. Contains 7 fact tables, 17 dimension tables



特点：
Features:

1）99 个测试案例，遵循 SQL 99 和 SQL 2013 的语法，案例复杂
The case is complicated and contains 99 test cases, following the syntax of SQL 99 and SQL2013

2）测试数据量大，并且测试案例回答的是真实的商业问题
The amount of test data is large, and the test case answers real business questions

3）测试案例中包含各种业务模型
Test cases contain various business models

4）几乎所有案例都有很高的 IO 负载和 CPU 计算需求
Almost all cases have high IO load and CPU computing requirements

流处理基准测试的设计

The design of streaming benchmark.

02

要点概述

key points

01 架构设计

Design of architecture

02 数据集

Data set

03 工作负载

Workload

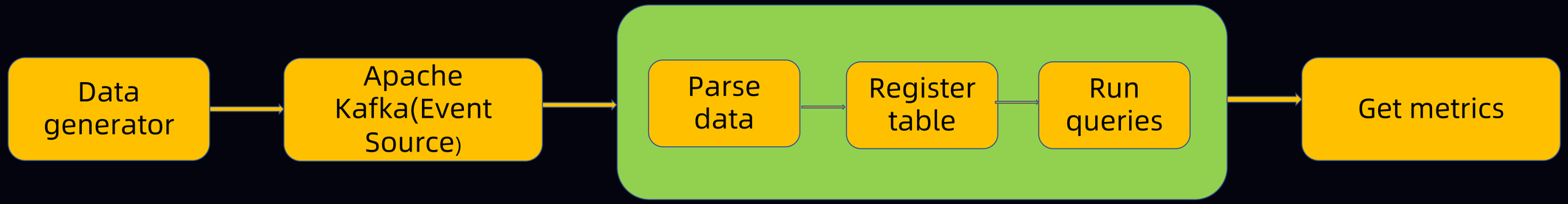
04 度量指标

Metrics

基准测试的改进 Benchmark improvement

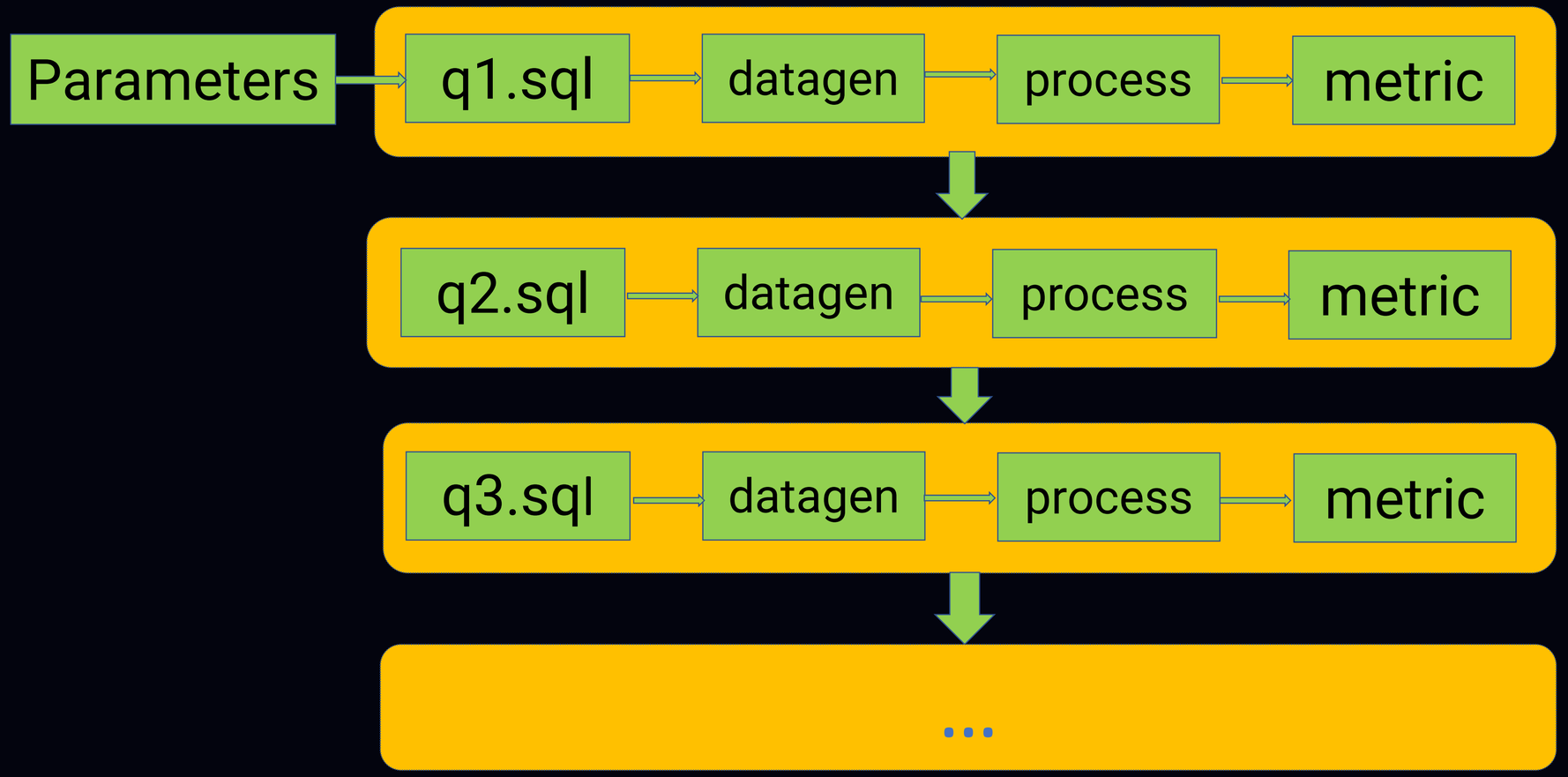
- 数据生成速率满足集群高负载要求
  Data generation rate meets cluster high load requirements
- 数据集应该模拟的是实际应用场景
  The data set should simulate the actual application scenario
- 测试集基于 SQL（SQL 应用广泛，语义精确，更易理解）
  The test set is based on SQL (SQL is widely used, semantically precise, and easier to understand)
- 对于不同的测试集可以使用统一的度量指标
  Uniform metrics can be used for different test sets

架构图 Architecture



- Data generator -> generate data -> Kafka topics -> Consumer -> streaming -> table -> streaming sql-> metrics
- For each topics, we define the schema and generate data as the format we need.

执行流程 Execution process



数据生成器 Data generator

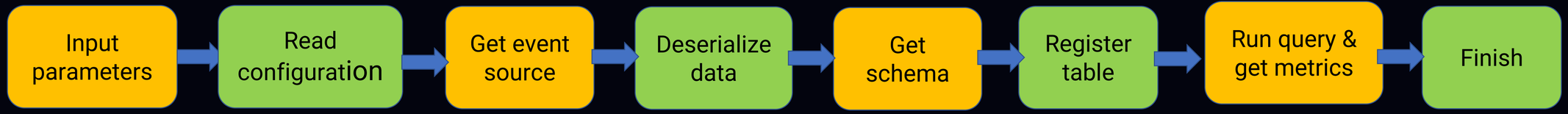
- 不同的查询可能需要不同的 topics
  Running Different queries may need different topics
- 一个查询可能会涉及一张或多个 topics
  Query may need one topic or multi topics
- 模块功能：为不同的查询生成原始数据
  function：create stream for certain query

Parameters	Description
DATAGEN_TIME	The running time of each query.
THREAD_PER_NODE	The number of thread to generate data.
QUERY	The name of query that will be run.

数据生成器的吞吐量：多节点多任务灵活控制，保证测试的负载足够大
Throughput of date generator : multi-node multi-task flexible control to ensure that the test load is large enough

配置文件：数据生成节点 ip、每个节点的任务数
Configuration file: the ip of nodes to generate data and number of tasks per node

query 执行流程



根据配置文件路径读取公共的配置
Running Different queries may need different topics

读取对应计算引擎的配置参数
Query may need one topic or multi topics

根据不同的query，消费所需的主题数据，注 册流表，执行查询

According different queries, the data in Kafka will be consumed and registered as dynamic tables.

参数 parameters	描述 Description
CONF_FILE	The dislocation of benchmark config.
QUERY	The query will be run

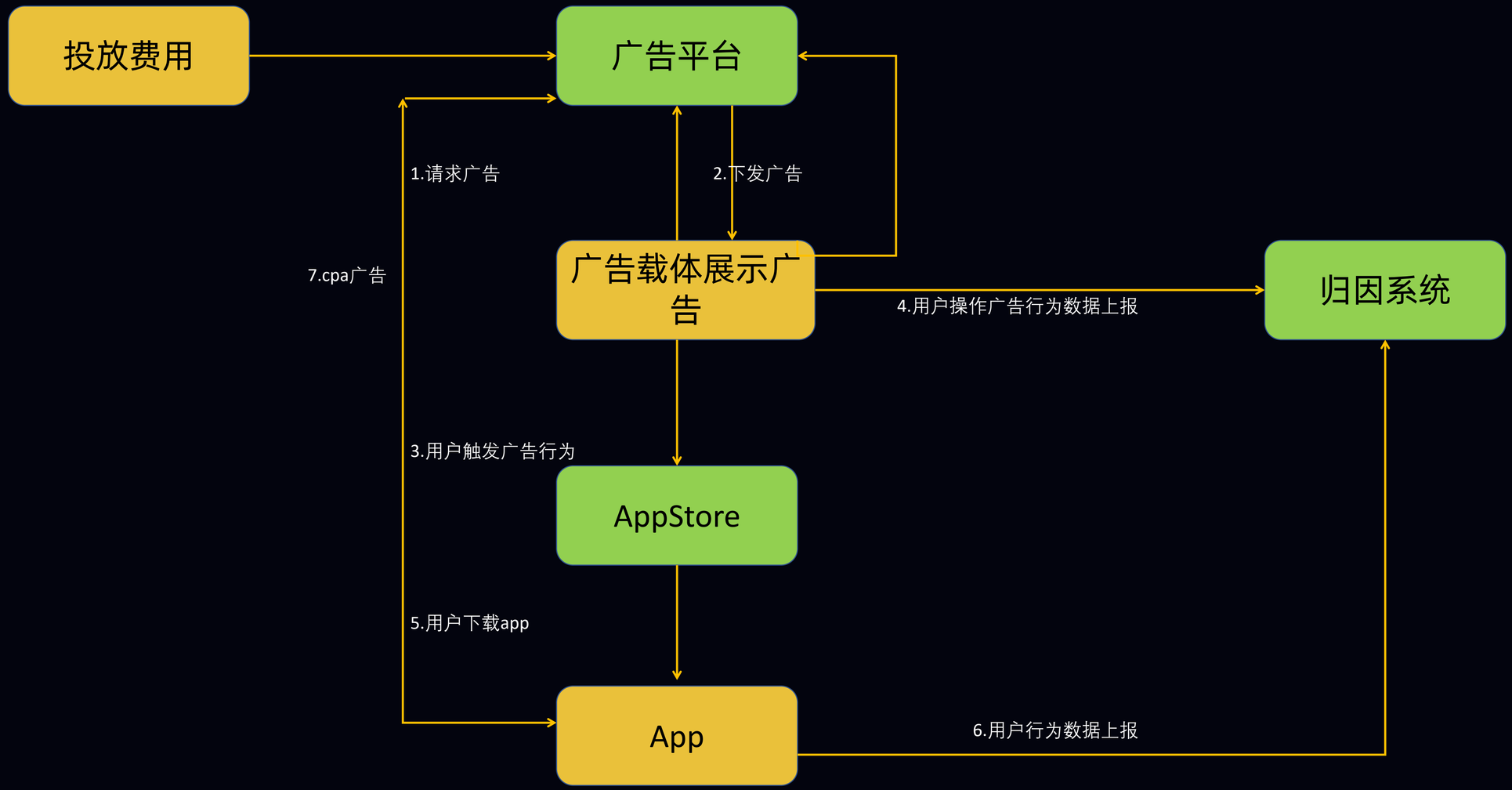
应用场景（数据集）
Application scenario (data set)

- 用户购物行为分析（测试案例）
  User shopping behavior analysis (test case)

- 广告投放效果分析
  Ad serving performance analysis

- 用户浏览会话分析
  User browsing session analysis

广告投放效果分析
Ad serving performance analysis



广告投放效果分析 Ad serving performance analysis

事实表: 曝光日志

Colume	Imp_time	strategy	site	pos_id	poi_id	cost	Device_id	Session_id
Type	Long	String	String	String	String	Double	String	String
Desc	Exposure time	Model strategy	Media providers (advertising, Baidu, headlines, Weibo, etc.)	Advertising space	Advertising material	Exposure cost	Device id	Session id

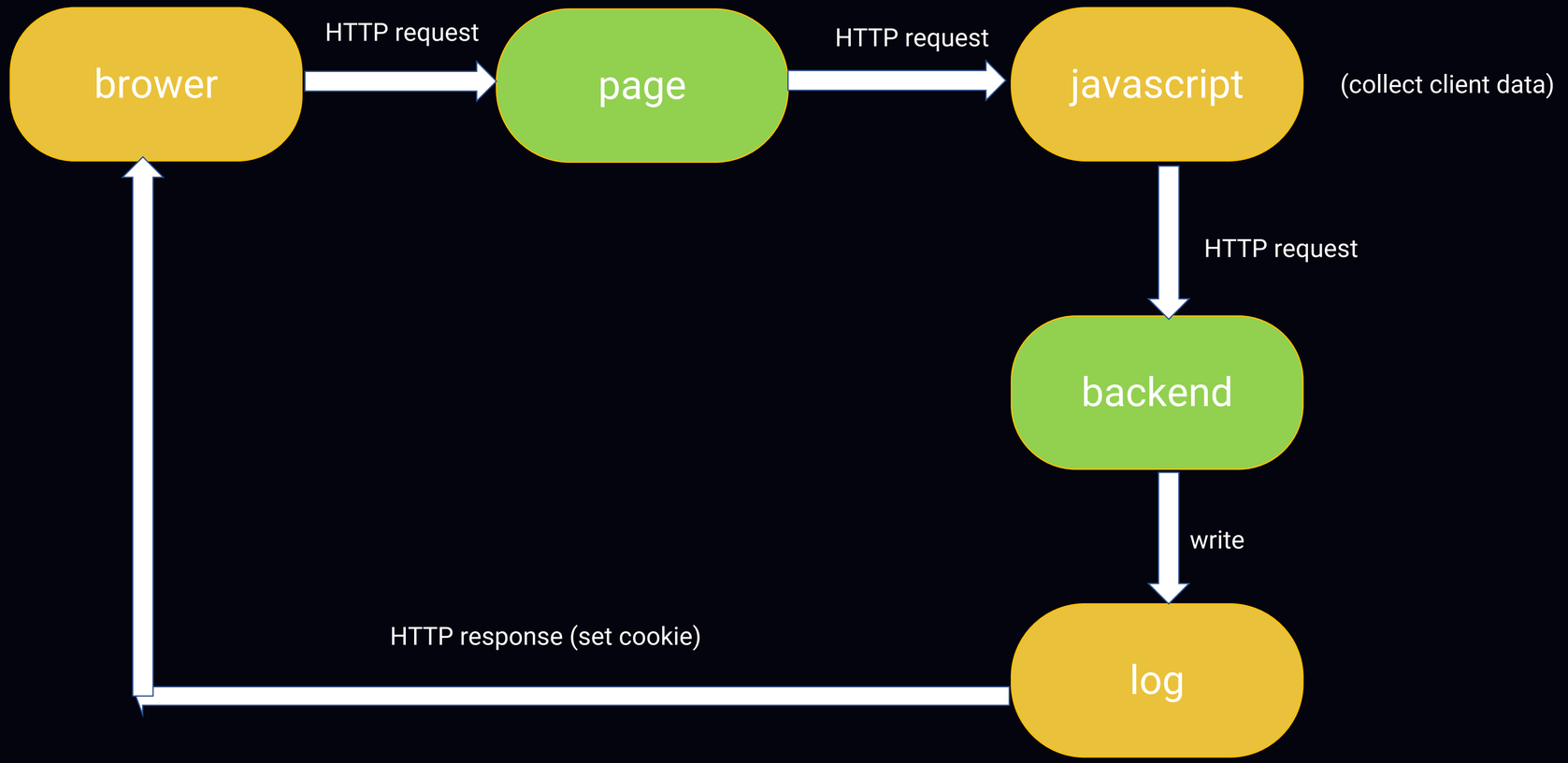
事实表: 点击日志

Colume	Click_time	strategy	site	pos_id	poi_id	Device_id	Session_id
Type	Long	String	String	String	String	String	String
Desc	Click time	Model strategy	Media (advertising provider)	Advertising space	Advertising material	Device id	Session id

事实表: 唤醒日志

Colume	Dau_time	Device_id	Session_id
Type	Long	String	String
Desc	Arouse time	Device id	Session id

用户浏览会话分析 User browsing session analysis



用户浏览会话分析
User browsing session analysis

事实表: UserVisitAction

colume	Date	userId	sessionId	pageId	actionTime	searchKeyword	clickCategoryId	clickProductId	orderCategoryIds	orderProductIds	payCategoryIds	payProductId	cityId
Type	String	Long	String	Long	String	String	String	String	String	String	String	String	String
Desc	The date of session	The id of user(user_id)	UUID	PageId(1-100)	time	the keyword for Search action	The prduct's type for Click Action	The productId for Click Action	The product's type for Order action	The productId for Order Action	The product's type for Pay action	The productId for Pay Action	The cityId of session

action type：pageView、Search、click、order、pay

* session ：
* 1. PageView-> Search -> Click -> Order -> Pay
* 2. Search -> Click -> Order -> Pay
* 3. PageView -> Click -> Order -> Pay
* Note: Browsing, searching, and clicking may appear multiple times in succession, but there will be no continuous occurrence of orders and payments.
* When a process is executed until an order or payment is made, the execution may be restarted again.
* Suppose the number of event triggers in a session is no more than 20 times.
* The interval between every two events is no more than 5 minutes, at least 1 second
* There may be three cases of searching, clicking, and continuing browsing after browsing.
* There may be three cases of clicking, browsing, and continuing to search after searching.
* There may be four situations: browsing, searching, ordering, and continuing to click after clicking.
* There may be three situations of search, browsing and payment after ordering.
* There may be two cases of search and browsing after payment
* Note： There may be an end operation after all events.

用户浏览会话分析
User browsing session analysis

维度表: productInfo

Colume	productId	productName	extendInfo
Type	Long	String	String
Desc	The id of product	The name of product(product_id)	Self-owned products or not

维度表: UserInfo

colume	userId	username	Name	Age	Profession	City	sex
Type	Long	String	String	Int	String	String	String
Desc	The id of user	The name of user(user_id)	The name of user(name_id)	The age of user(15-60)	Profession of user	City of user	Man, Woman,Unknown

工作负载 workload

FLINK FORWARD

- 基准测试支持 **Spark** & **Flink**
  Benchmark support Spark and Flink
- Flink benchmark 包含 q1.sql ~ q14.sql
  The Flink benchmark contains q1.sql ~ q14.sql
- Spark benchmark 包含 q1.sql ~ q9.sql
  The Spark benchmark contains q1.sql ~ q9.sql
- Flink & Spark benchmark 中的q1.sql ~ q9.sql查询逻辑相同 (如下两张图)
  The q1.sql ~ q9.sql query logic in the Flink & Spark benchmark is the same
- Flink SQL支持更多的语义，q10.sql ~ q14.sql包含 topN 以及动态表的多重聚合等操作
  Flink SQL supports more semantics, q10.sql ~ q14.sql contains topN and multi-aggregation of dynamic tables
- Query 文件路径所在路径：
  The location of queries
- $rootDir/spark/query & $rootDir/flink/query

```
SELECT
        strategy, site, pos_id, WINDOW(imp_time, '10 seconds').start, pos_id, WINDOW(imp_time, '10 seconds').end, SUM(cost)
FROM
        imp
GROUP BY
        strategy, site, pos_id, WINDOW(imp_time, '10 seconds')
```

```
SELECT
        strategy, site, pos_id, TUMBLE_START(rowtime, INTERVAL '10' SECOND), TUMBLE_END(rowtime, INTERVAL '10' SECOND), SUM(cost)
FROM
        imp
GROUP BY
        strategy, site, pos_id, TUMBLE(rowtime, INTERVAL '10' SECOND)
```

工作负载 workload

q4.sql: 以10秒为滑动窗口的宽度，实时统计不同的广告策略在每个时间段触发的用户下载操作，涉及多张事实表的连接聚合操作:

```
SELECT
    b.device_id, a.strategy, a.site, a.pos_id, count(b.device_id)
FROM
    click a
JOIN
    dau b
ON
    a.session_id = b.session_id AND a.rowtime BETWEEN b.rowtime - INTERVAL '1' second AND b.rowtime + INTERVAL '1' second
GROUP BY
    b.device_id, a.strategy, a.site, a.pos_id, TUMBLE(a.rowtime, INTERVAL '10' SECOND)
```

工作负载 workload

q9.sql: 以广告投放策略，广告位，时间等维度统计广告唤醒的次数，涉及两张事实表的连接操作：

```
SELECT
      a.device_id, a.strategy, a.site, a.pos_id, b.var2, b.var1, count(*)
FROM
      (SELECT device_id, strategy, site, pos_id FROM click) a
JOIN
      (SELECT device_id, FROM_UNIXTIME(CAST(dau_time/1000 AS BIGINT), 'yyyyMMdd') as var1,
FROM_UNIXTIME(CAST(dau_time/1000 AS BIGINT), 'HH') as var2 FROM dau) b
ON
       a.session_id = b.session_id
GROUP BY
       a.device_id, a.strategy, a.site, a.pos_id, b.var2, b.var1
```

工作负载 workload

q10.sql: 统计不同时间段的会话时间总长度，涉及单张事实表的多重聚合操作:

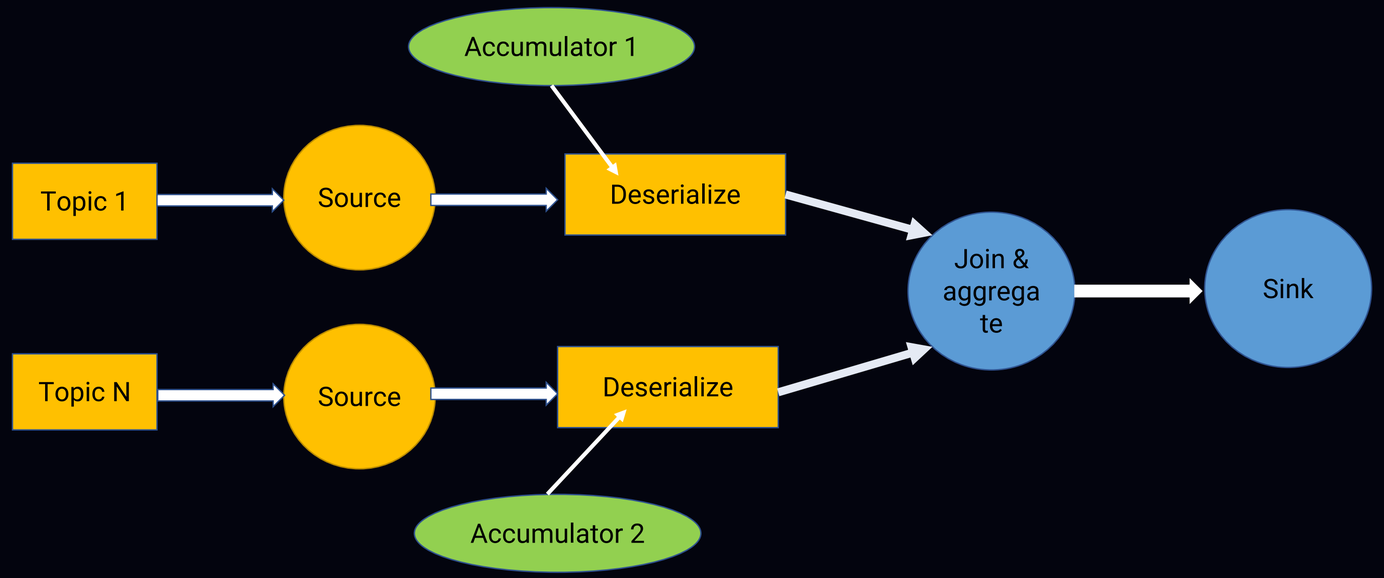
```
SELECT
      a.dt, a.h, SUM(a.len) total
FROM
      (SELECT
            sessionId, MAX(actionTime)-MIN(actionTime) as len, DAYOFMONTH(CAST(actionTime AS TIMESTAMP))
as dt, HOUR(CAST(actionTime AS TIMESTAMP)) as h
       FROM
            userVisit
       GROUP BY
            sessionId, DAYOFMONTH(CAST(actionTime AS TIMESTAMP)), HOUR(CAST(actionTime AS TIMESTAMP)))
a
WHERE
      a.len > 100
GROUP BY
      a.dt, a.h
```

工作负载 workload

q12.sql: 以 10 秒为窗口，实时统计不同城市热门商品的购买量（TopN）：

```
SELECT
      *
FROM
   (SELECT
         *, ROW_NUMBER() OVER (PARTITION BY w.cityId ORDER BY w.num DESC) as rownum
    FROM
         (SELECT
               TUMBLE_START(rowtime, INTERVAL '10' SECOND), TUMBLE_END(rowtime, INTERVAL '10'
SECOND), cityId, payProductIds, count(*) num
          FROM
               userVisit
          WHERE
               payProductIds IS NOT NULL
          GROUP BY
               cityId, payProductIds, TUMBLE(rowtime, INTERVAL '10' SECOND)
         ) w
   ) v
WHERE
   v.rownum <= 10
```

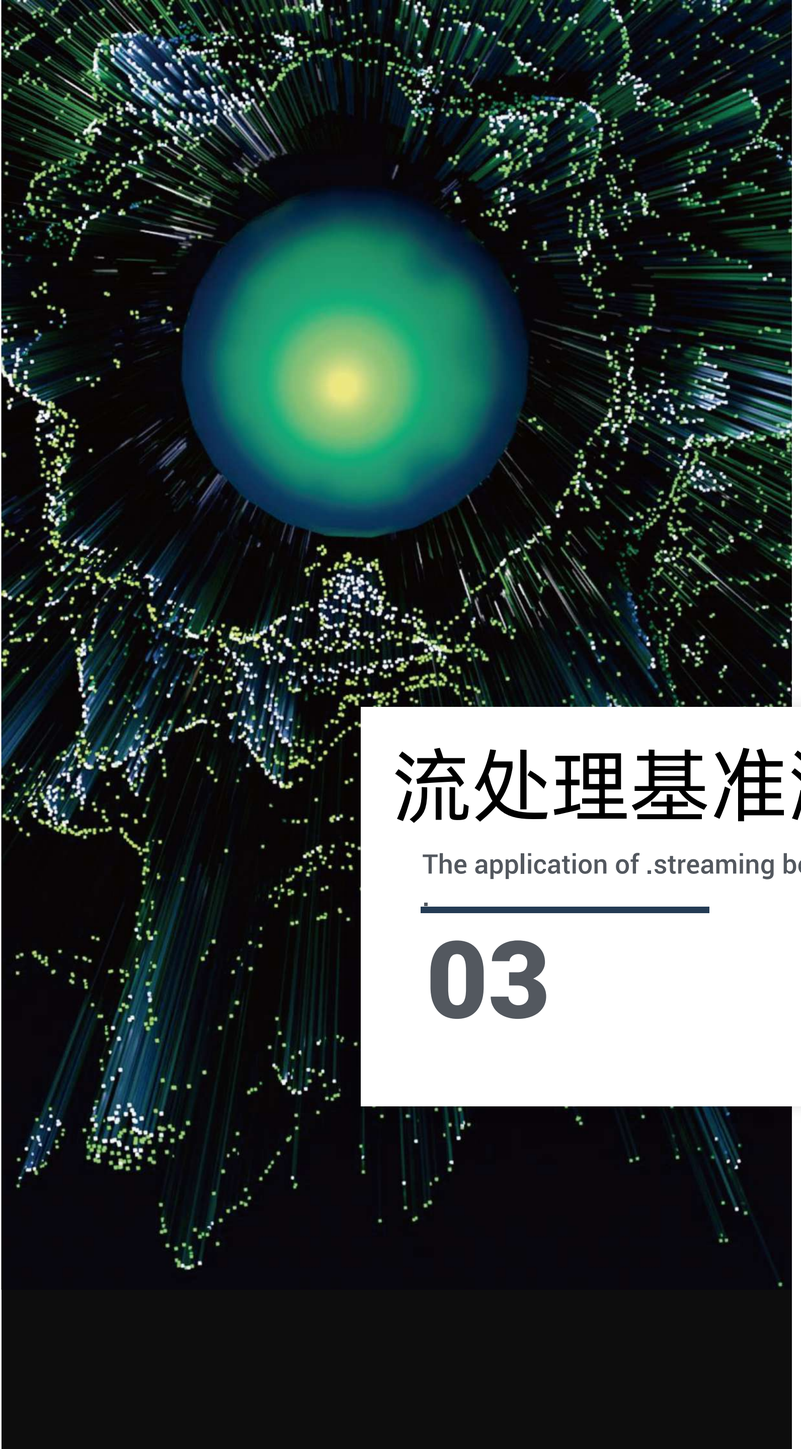
度量指标 metrics



- 吞吐量(TPS)：单位时间内系统处理的数据条数
  Throughput (TPS): The number of data processed by the system per unit time
- 统计方法：定义计数器，在数据解析阶段统计处理过的数据条数
  Statistical method: define the counter, count the number of processed data in the data analysis stage
- 性能评估结果：记录查询完成时间、运行时间、吞吐量
  Performance evaluation result: record query completion time, running time, throughput

Finished time: 2019-11-06 19:09:31; q1.sql Runtime: 298s TPS:5893449

- 延迟性统计的难点：SQL 不同类型的操作很那有统一的度量标准
  Difficulties in delay statistics: SQL different types of operations are very uniform metrics



流处理基准测试的应用

The application of .streaming benchmark

03

测试环境 Test Environment

Compute

Data

参数	配置
计算节点	1 master + 1 slave
存储节点	3 node
CPU核心数/节点	88 logic core
内存/节点	16*32GB
网络	10Gb
磁盘/节点	OS:1* 800GB SSD; Storage: 5* 1.1T SSD

执行细则 Executive rules

1.clone the project into your machine

2.mvn clean package

3.Apache Kafka, Apache zookeeper, Apache Spark-2.4.4 and Flink-1.9 have been installed in your cluster.

4.Update conf/benchmarkConf.yaml (The properties of Kafka, Zookeeper, benchmark...)

5.Update flink/conf/benchmarkConf.yaml (The properties of flink)

6.Update spark/conf/benchmarkConf.yaml (The properties of spark)

7.Update conf/dataGenHosts (The hosts where data will be generated; suggest to generate data on kafka node)

8.Update conf/queriesToRun (The queries will be run)

9.Update conf/env

10.Copy the project to each hosts in conf/dataGenHosts .

11.sh bin/runFlinkBenchmark.sh or sh bin/runSparkBenchmark.sh

12.Get results on flink/result/result.log & spark/result/result.log

未来工作 The future work

- 扩展数据集
- 丰富测试集
- 增加延迟度量指标

工程访问路径:
https://github.com/Intel-bigdata/StreamingBench

Welcome to get your suggestion!


THANKS