DTCC

2016中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2016

数据定义未来

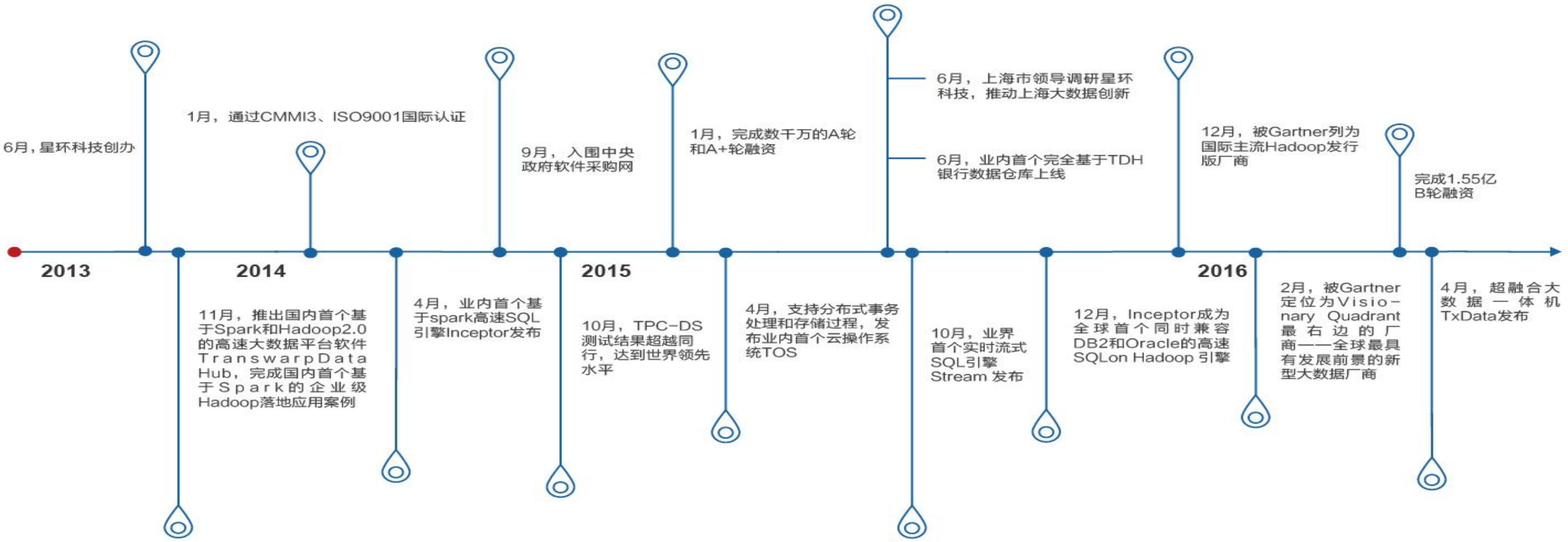
SequeMedia 盛拓传媒　IT168.com　ChinaUnix　ITPUB

Transwarp StreamSQL: A SQL/PLSQL Stream engine on Spark

季钱飞
星环科技
www.transwarp.io

公司介绍

TRANSWARP 星环科技



2013

6月，星环科技创办

11月，推出国内首个基于Spark和Hadoop2.0的高速大数据平台软件TranswarpDataHub，完成国内首个基于Spark的企业级Hadoop落地应用案例

2014

1月，通过CMMI3、ISO9001国际认证

4月，业内首个基于spark高速SQL引擎Inceptor发布

9月，入围中央政府软件采购网

10月，TPC-DS测试结果超越同行，达到世界领先水平

2015

1月，完成数千万的A轮和A+轮融资

4月，支持分布式事务处理和存储过程，发布业内首个云操作系统TOS

6月，上海市领导调研星环科技，推动上海大数据创新

6月，业内首个完全基于TDH银行数据仓库上线

10月，业界首个实时流式SQL引擎Stream 发布

12月，Inceptor成为全球首个同时兼容DB2和Oracle的高速SQLon Hadoop 引擎

12月，被Gartner列为国际主流Hadoop发行版厂商

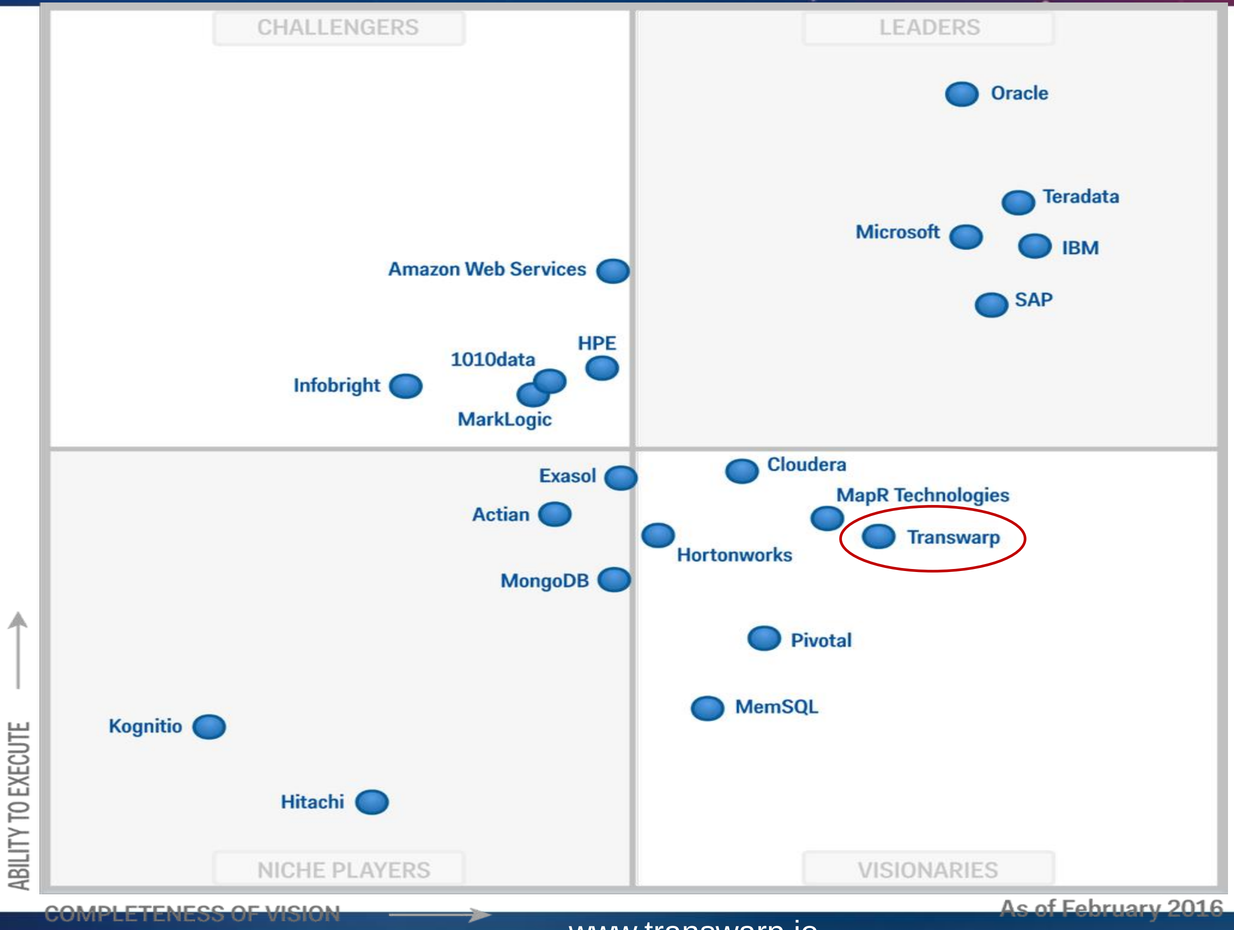
2016

2月，被Gartner定位为Visionary Quadrant最右边的厂商——全球最具有发展前景的新型大数据厂商

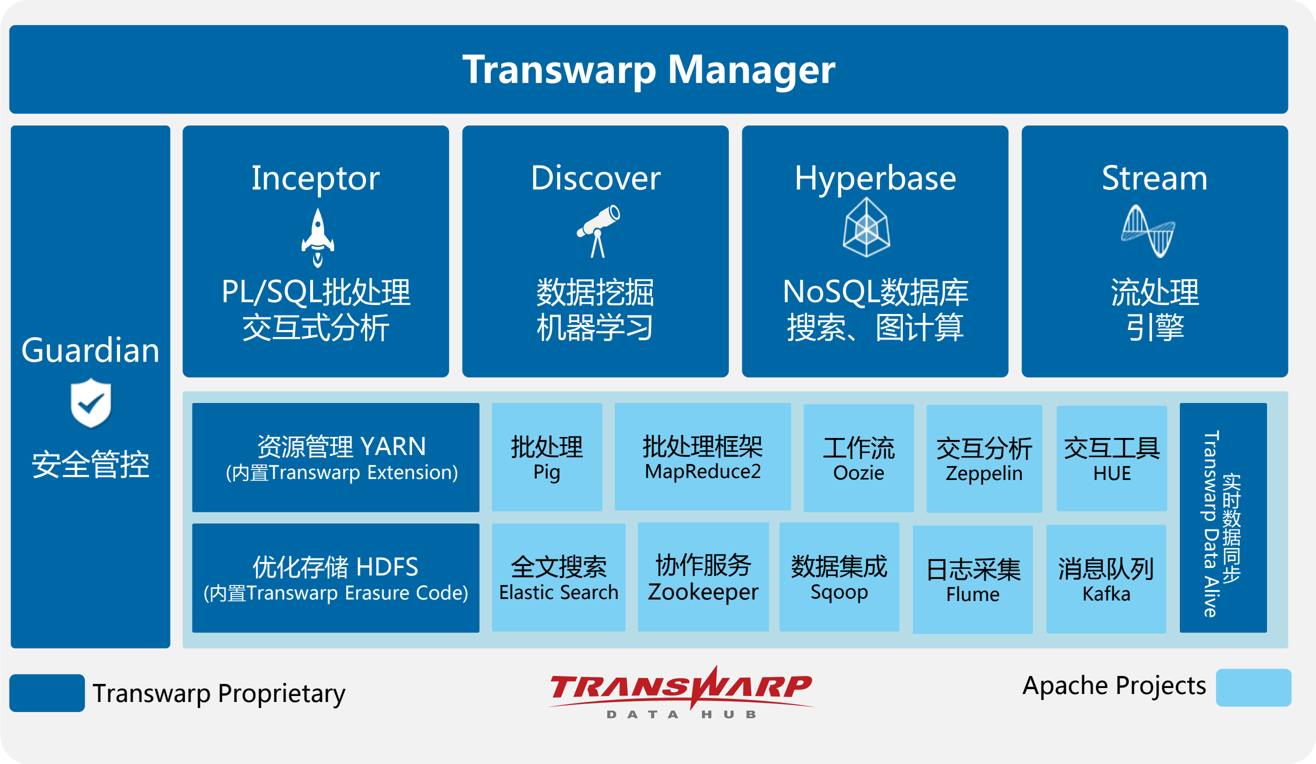
完成1.55亿B轮融资

4月，超融合大数据一体机TxData发布

Gartner魔力象限



Transwarp Data Hub架构图



Transwarp Manager

Guardian
安全管控

Inceptor
PL/SQL批处理
交互式分析

Discover
数据挖掘
机器学习

Hyperbase
NoSQL数据库
搜索、图计算

Stream
流处理
引擎

资源管理 YARN
(内置Transwarp Extension)

批处理
Pig

批处理框架
MapReduce2

工作流
Oozie

交互分析
Zeppelin

交互工具
HUE

优化存储 HDFS
(内置Transwarp Erasure Code)

全文搜索
Elastic Search

协作服务
Zookeeper

数据集成
Sqoop

日志采集
Flume

消息队列
Kafka

Transwarp Data Alive
实时数据同步

🟦 Transwarp Proprietary

TRANSWARP DATA HUB

Apache Projects 🟦

最完整的SQL支持

99%的SQL 2003支持，唯一支持PL/SQL的引擎（98%），唯一支持ACID分布式事务的SQL引擎；定位数据仓库和数据集市市场，可用于补充或替代Oracle、DB2等分析用数据库。

高效内存/SSD计算

第一个支持SSD的基于Hadoop的高效计算引擎，可比硬盘快一个数量级；可用于建立各种数据集市，对接多种主流报表工具。

最完整的分布式机器学习算法库

支持最全（超过50余种）的分布式统计算法和机器学习算法，同时整合超过5000个R语言算法包。适合金融业风险控制、反欺诈、文本分析、精准营销等应用。

支持最完整SQL和索引的NoSQL数据库

支持SQL2003、索引、全文索引，支持图数据库和图算法，支持非结构化数据存储
支持高并发查询

最健壮和功能丰富的流处理框架

支持所有组件的高可用(HA)
支持流式SQL和流式机器学习

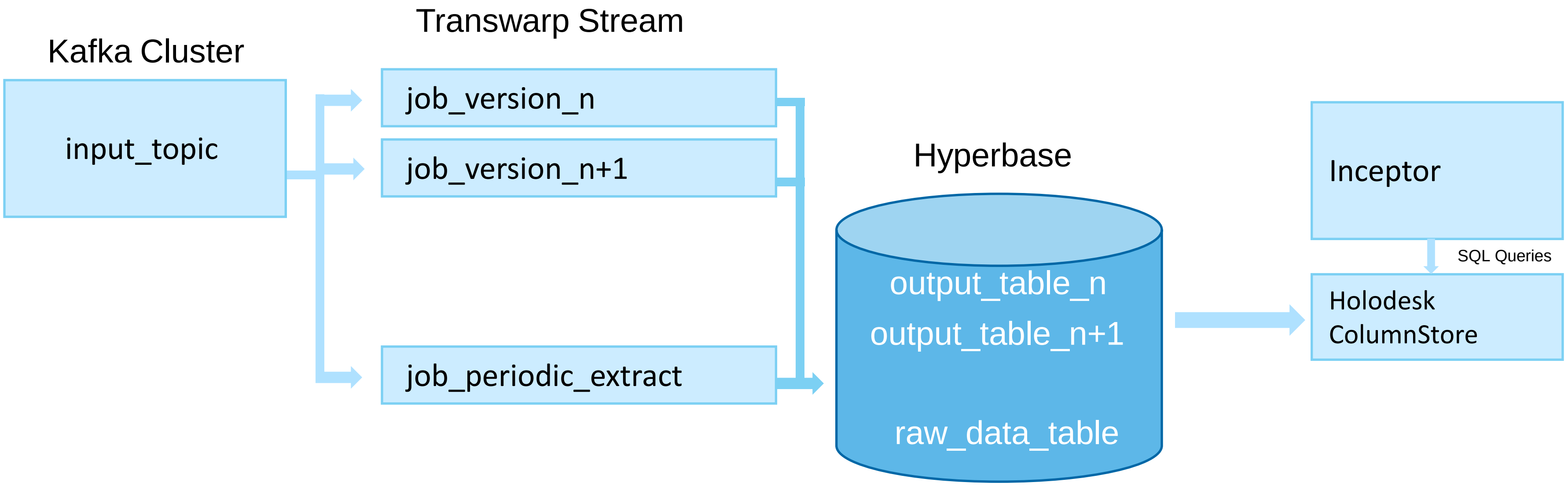
为什么使用流处理

- 从批处理转向流处理逐渐成为一种趋势
- 对于大多数批处理应用，可以完全转化为流处理逐步处理完：
  - 投行在每天交易结束时都需要计算整个公司的资产价值和可能存在的风险
- 实时监控系统运作，及时发现异常：
  - 风电行业需要实时监控风电运行状况，及时给出告警信息
- 彻底改变业务模式，提升业务价值：
  - 交通行业实现“秒”抓套牌车

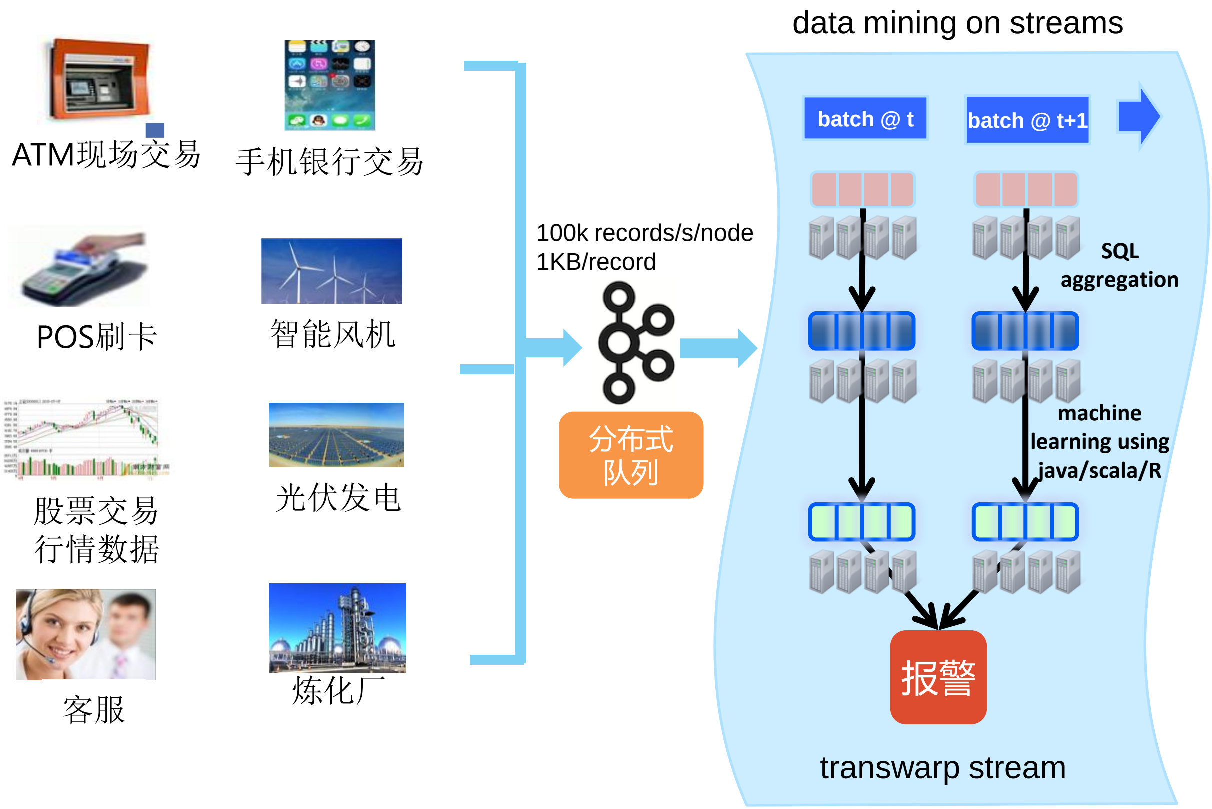
为什么采用SQL

- 星环2013年开始使用Spark，2014年初开始在客户现场部署Spark Streaming，至今已经有几十家流处理的客户
- 入门门槛极其高，有经验的程序员未必能胜任
- 迁移成本较高，原有业务基于SQL/PLSQL
- 产品化程度差，需要有Spark和Hadoop经验的专家进行运维

典型的流数据处理流程


Kafka Cluster
input_topic
Transwarp Stream
job_version_n
job_version_n+1
job_periodic_extract
Hyperbase
output_table_n
output_table_n+1
raw_data_table
Inceptor
SQL Queries
Holodesk
ColumnStore

Stream+SQL+ML对实时数据进行监测



ATM现场交易

手机银行交易

POS刷卡

智能风机

股票交易
行情数据

光伏发电

客服

炼化厂

100k records/s/node
1KB/record

分布式
队列

data mining on streams

batch @ t

batch @ t+1

SQL
aggregation

machine
learning using
java/scala/R

报警

transwarp stream

1. EMS实时消息订阅
2. 交易所实时风险研判
3. 券商实时交易行情监测
4. … …

为什么采用SQL

- 常见用户反馈：
- “你们的streaming程序怎么又丢数啦！”
  - 用户不会用BlockingQueue
- “怎么没有结果啊？”
  - 各种原因，茫茫代码海一行日志都没有…
- “你帮我看看我这条SQL写成streaming代码怎么写呢？”
  - 最终代码变成了我们实现
- “好像跑出问题了，能不能帮我们看看？”
  - 各种分析后发现是磁盘满了
- “Kafka取不出数据怎么回事啊？”
  - 最终发现是zk挂了

为什么采用StreamSQL

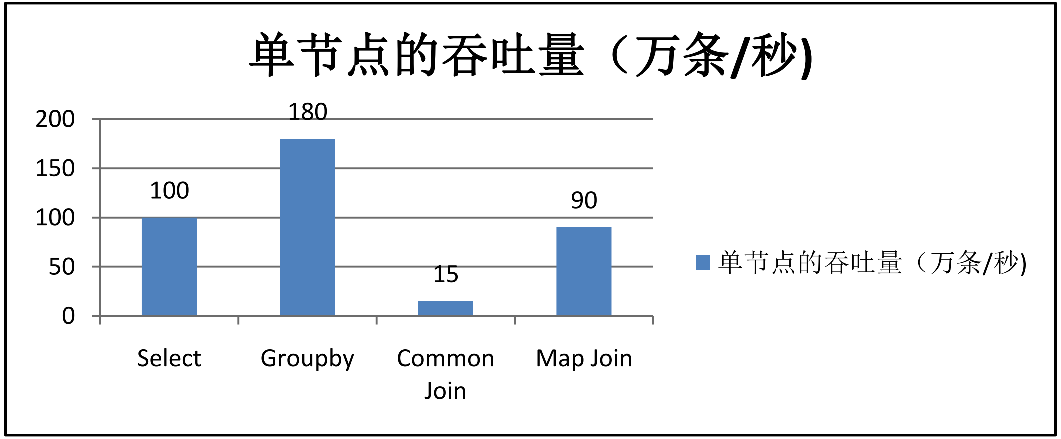
```
val name = "test"
val conf = new SparkConf()
  .setMaster("ngmr-yarn-client")
  .setAppName(name)
  .set("spark.streaming.blockInterval", "1000")
  .set("spark.streaming.receiver.maxRate","1000")
val ssc = new StreamingContext(conf, Milliseconds(1000L))

val KfkStreamNumbers = 18
val topic = Map(name -> 1)
val kafkaParams = Map(
  "zookeeper.connect" -> "localhost:2181",
  "group.id" ->  "mytest",
  "auto.commit.enable" ->  "true",
  "auto.commit.interval.ms" ->  "2000" )
val streams = for( i <- 1 to KfkStreamNumbers)
  yield

KafkaUtils.createStream[String, String, StringDecoder, StringDecoder](ssc, kafkaParams.toMap, topic, StorageLeve
l.MEMORY_ONLY)
val unionStream = ssc.union(streams)

val windowLength = Seconds(30L)
val slideLength = Seconds(10L)
val limit = 100
val buttomValue = 1
val tableName = "result"

val parts = unionStream.mapPartitions(iter=>
  iter.map(kv=> {
    val fields = kv._2.split(",")
    val id = fields(3).toLong
    val name = fields(4)
    (id, name)
  })
    .filter( idAndName  => idAndName._1 == 1187 || idAndName._2 == 8864 )
    .map(res => (res, 1L))
)

(new PairDStreamFunctions[(Long, String), Long](parts))
    .reduceByKeyAndWindow( _+_,  _-_, windowLength, slideLength)
    .transform(rdd=> rdd.sortBy(kv=> kv._2))
    .foreachRDD(rdd=> rdd.foreach(res =>
      TableFlusherManager.save(res._1._1, res._1._2, res._2)
    ))
ssc.start()
ssc.awaitTermination()
```

create stream test(id int, name string);
insert into result
select id, name, count(*)
from test
where id in (1187, 8864)
group by id, name
order by count(*)
window w1 as (length ’10s’)

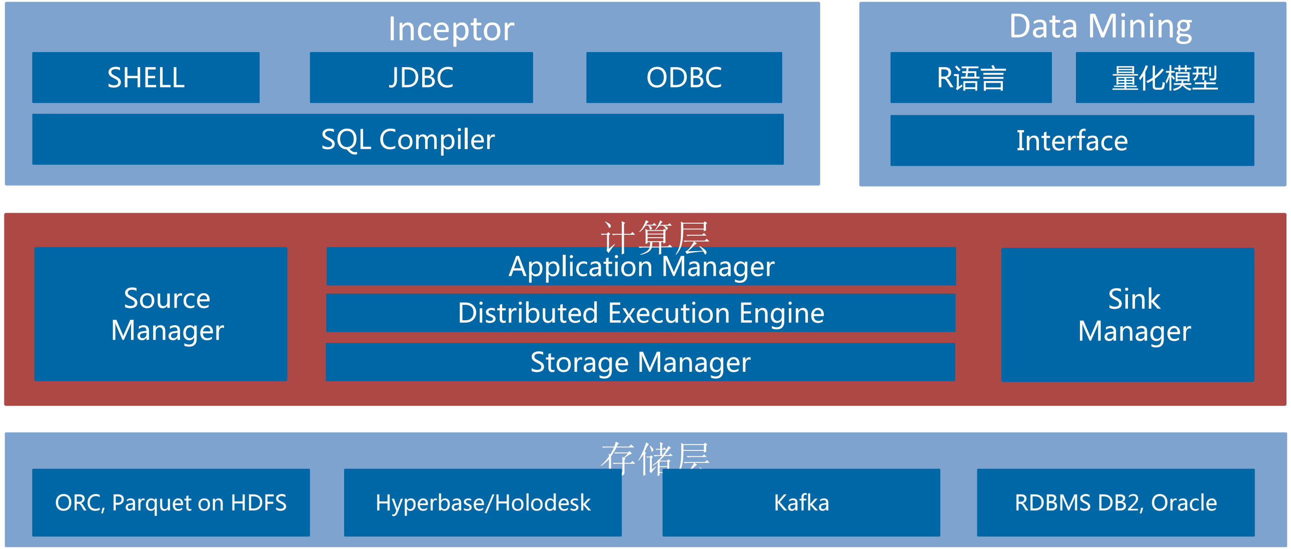
性能可能提升

性能

- 简单业务：
  - SQL与Spark Streaming比性能差别不大
  - Spark Streaming代码必须特殊优化，否则可能比SQL慢
  - 性能瓶颈往往在结果输出
- 复杂业务：
  - SQL比Spark Streaming稍慢
  - 数量级级别的性能差距取决于是否进行调优



StreamSQL架构图



StreamSQL设计思路

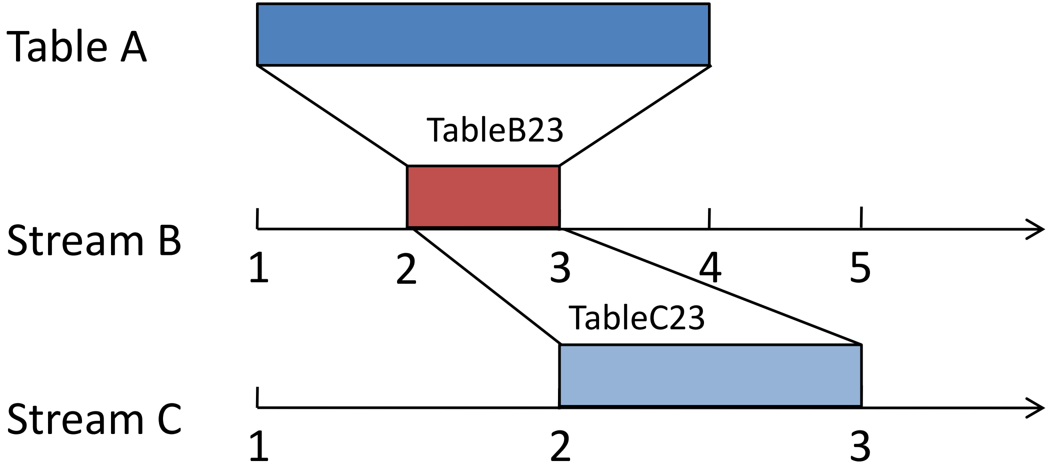
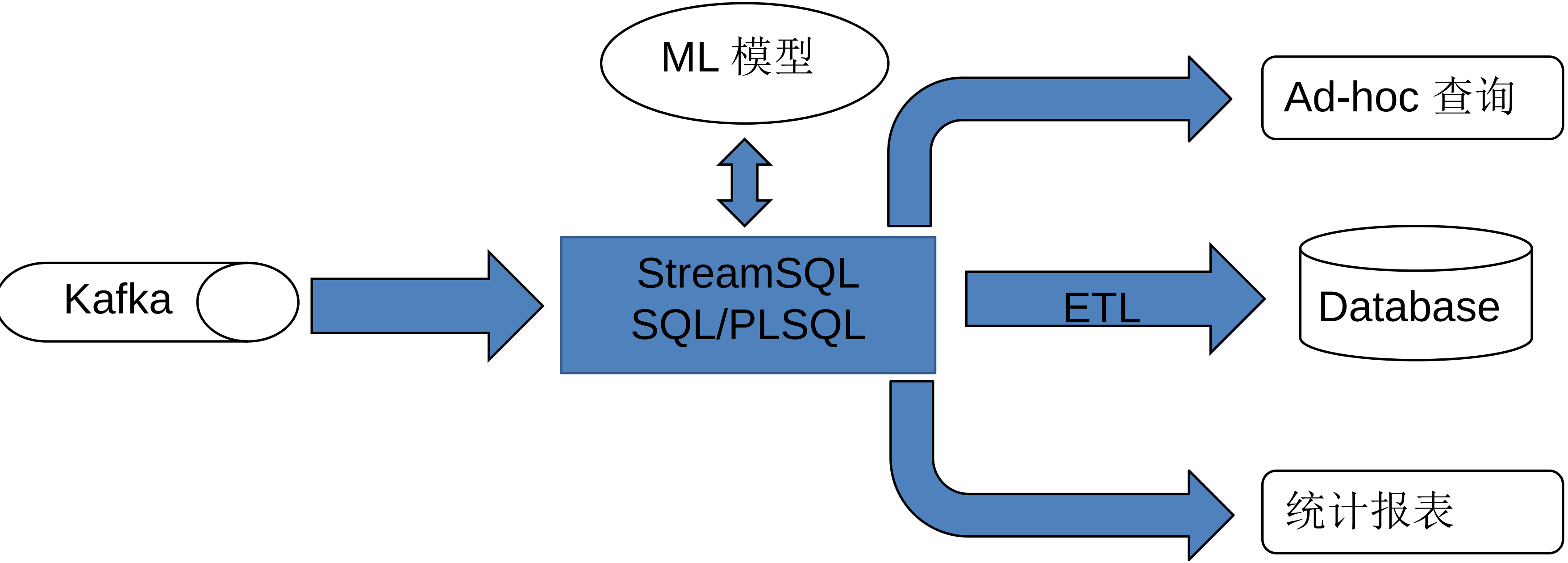


Table A join
Stream B Join
Stream C

StreamSQL主要功能



基本功能—语法支持

- SQL 2003标准
  - 除极少数不适合操作流的语法，如update、delete。
  - 增加少量流处理特有语法。
- Oracle PLSQL 11g
  - 基本语法都支持，包括游标
- DB2 PLSQL 最新版
  - 基本语法都支持，包括游标

原有SQL业务经过少量改写就可以上线！

StreamSQL HelloWorld

1. 创建流
   - create stream s1(id int, name string, value int) streamproperties(“topic” = “source” , “kafka.zookeeper” = “localhost:2181”) ;
2. 定义流的转换
   - create stream s2 as select name, case when value < 0 then 0 else value end from s1;
3. 创建结果表
   - create table t1(name string, value int);
4. 启动流应用
   - Insert into t1 select * from s2 ;
5. 查看流应用
   - list streamjobs;
6. 停止流应用
   - stop streamjobs;

这是一个完整的ETL工具！

基本功能—多流应用共享输入

1. 创建流
   - create stream s1(int, name string);
2. 创建Hyperbase结果表
   - create table hyper(id int, name string) stored as hyperbase;
3. 创建Holodesk结果表
   - create table holo(id int,name string) stored as holodesk;
4. 启动流应用
   - insert into hyper select * from s1;
   - insert into holo select * from s1;

常见的ETL需求！

基本功能—输出到流

1. 创建流

    create stream s1(id int, name string, value int);

2. 创建结果流

    create stream s2(name string, value int);

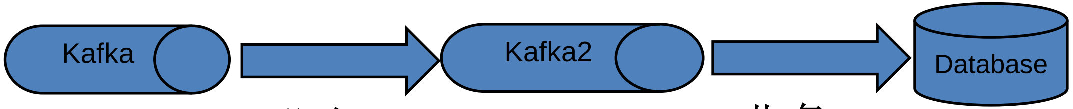
3. 创建视图

    create view v1 as select name, sum(value) from s1 group by name;

4. 启动流应用

    Insert into s2 select * from v1;

Kafka企业总线！



业务1

业务2

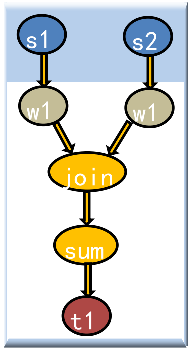
基本功能—数据字段切分与聚合

1. 创建流
   - create stream s1(id int, value int, ts timestamp);
   - create stream s2(id int, value int, ts timestamp);
2. 按字段切分与聚合
   - insert into result select sum(s1.value + s2.value) over w1 from s1 join s2 on s1.ts = s2.ts window w1 as(separated by ts length ’ 10’ seconds);

- 纯粹按机器时间导致结果不精确，在多表聚合的情况下，特别明显
- 系统时间与数据时间有较大差异，用户需要数据时间对应的精确结果

Window上的聚合示例

```
create stream s1(id string, v
int, ts timestamp);
create stream s2(id string, v
int, ts timestamp);
create table t1(id string, v int, ts
timestamp);
insert into t1 select
id,  sum(s1.v+s2.v), max(s1.ts)
from s1 join s2 on id window w1
as(separated by ts interval ’10’
seconds);
```



s1 id	v	ts
1	10	20151203 11:00:03
2	20	20151203 11:00:04
1	30	20151203 11:00:14
2	40	20151203 11:00:15
1	10	20151203 11:00:16

s2 id	v	ts
1	30	20151203 11:00:08
2	40	20151203 11:00:09
1	30	20151203 11:00:11
2	40	20151203 11:00:12
2	10	20151203 11:00:13

t1 id	v	ts
1	40	20151203 11:00:08
2	60	20151203 11:00:09
1	70	20151203 11:00:16
2	90	20151203 11:00:15

高级功能—运行抽象StreamJob

1. 创建StreamJob:
   - create streamjob job1 as ("insert into result_table select * from source") jobproperties("streamsql1"="transwarp1");
2. 查看/修改某个StreamJob:
   - desc streamjob job1;
   - alter streamjob job1 set jobproperties("streamsql1"="transwarp2");
3. 启动/停止StreamJob:
   - start streamjob job1;
   - stop streamjob job1;
4. 查看当前创建的StreamJob:
   - show streamjobs;
5. 删除StreamJob:
   - drop streamjob job1;

- **SQL持久化**
- 方便启动

高级功能—应用隔离Application

1. 创建Application:
   - create application app1 with appproperties("stream.batch.duration.ms"="1000");
2. 查看/修改Application的配置:
   - desc application app1;
   - alter application app1 set appproperties("stream.batch.duration.ms"="2000");
3. 使用Application，在其中创建StreamJob，并启动
   - use application app1;
   - create streamjob job1 as ("insert into result_table select * from source") jobproperties("streamsql1"="transwarp1");
   - start application app1;
   - stop application app1
4. 查看当前的Application:
   - show applications;
5. 删除Application:
   - drop application app1;

•隔离！

•权限！

使用PLSQL

```
CREATE STREAM transaction_stream(timestamp
      STRING, id STRING, transaction DOUBLE) ROW

FORMAT DELIMITED FIELDS TERMINATED BY ','
      TBLPROPERTIES ('topic'='transaction');

CREATE TABLE day_sum(id STRING , sd
      STRING, total DOUBLE) ROW FORMAT
      DELIMITED FIELDS TERMINATED BY ',';

CREATE TABLE warm_transaction (id
      STRING, timestamp STRING, total DOUBLE);

CREATE STREAM transaction_sum AS SELECT
      id, timestamp, sum(s.transaction) total
      FROM  transaction_stream s GROUP BY
      id, timestamp;

SET plsql.show.sqlresults=true;

SET stream.enabled=true;

SET
      stream.tables=qihuo2,xianhuo2,transaction_s
      tream;
```

```
DECLARE
threshold_count int := 0
BEGIN
INSERT INTO day_sum SELECT id, sd, CASE WHEN isnull(total2) THEN total1 ELSE
(total1
+ total2) END total FROM
(SELECT t1.id id, t1.timestamp sd , t1.total total1, t2.total
total2 FROM transaction_sum t1 LEFT JOIN day_sum t2 ON
t1.id=t2.id AND (to_unix_timestamp(t2.sd, 'yyyy-MM-dd HH:mm:ss')
+ 1)=unix_timestamp(t1.timestamp, 'yyyy-MM-dd HH:mm:ss'))
CREATE STREAM transaction_stream(timestamp STRING, id STRING, transaction
DOUBLE) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',' TBLPROPERTIES
('topic'='${db}.transaction');
SELECT count(*) INTO threshold_count FROM max_min_diff_window_stream WHERE
maxmindiff
>= 50
IF threshold_count > 0 THEN
INSERT INTO warm_transaction SELECT id, sd, total FROM day_sum ORDER BY total
DESC
LIMIT 20
END IF
END;
```

PLSQL迁移示例

Batch mode

```
create table t1(id int, value int);
create table t2 (id int, value int);
declare
   v_id int
   v_value int
   cursor cur(cur_arg int)
     is
       select * from (select * from t1 where id   <
cur_arg) order by id
begin
     open cur(5)
     loop
       fetch cur into v_id, v_value
       exit when cur%notfound
     end loop
end;
```

Stream mode

```
set stream.tables=t1;
create stream t1(id int, value int);
create table t2(id int,value int);
declare
   v_id int
   v_value int
   cursor cur(cur_arg int)
     is
       select * from (select/*+ adhoc*/ * from t1
where id < cur_arg) order by id
 begin
     open cur(5)
     loop
       fetch cur into v_id, v_value
       exit when cur%notfound
     end loop
end;
```

基本功能— 流控

- MaxRate
- Back-pressure
- FILO

基本功能— HA

- Check Point
- WAL
- Zookeeper Based Auto Failover
  - Active
  - Standby

高级功能— Adhoc查询

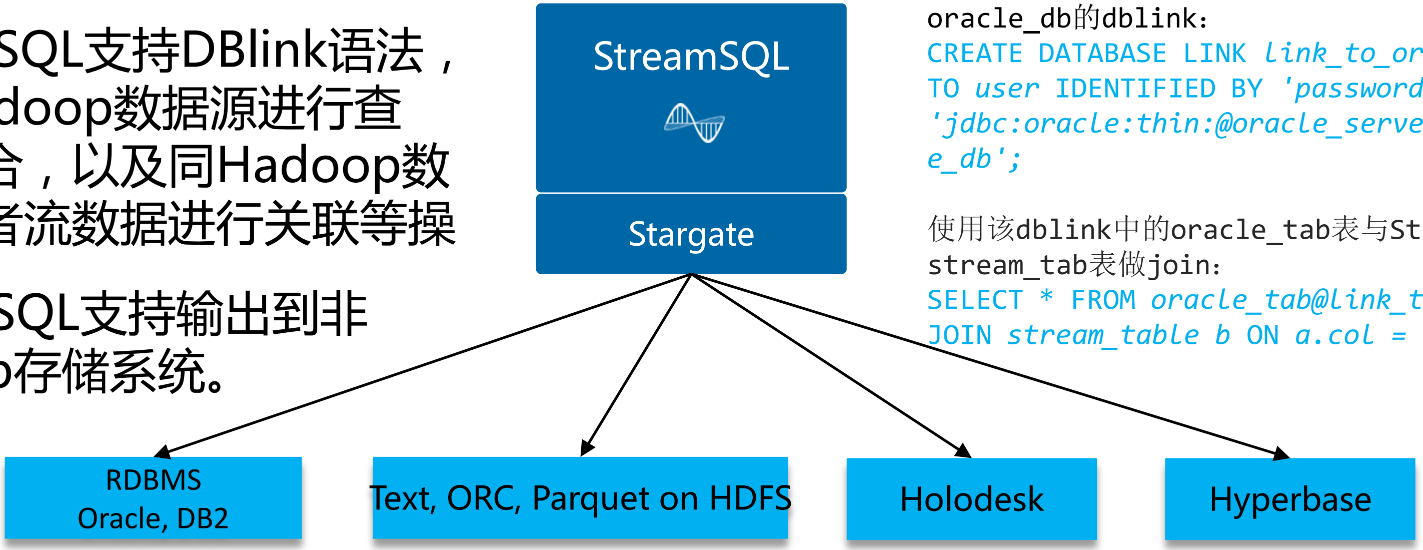
1. 创建流
   - create stream s1(id int, name string);
2. Adhoc查询 by hint
   - select /*+ adhoc*/ name from s1;

- 通过Adhoc查询，用户能基于当前的数据流尝试不同的操作，从而及时调整业务

高级功能—存储层适配器 Stargate

StreamSQL支持DBlink语法，对非Hadoop数据源进行查询，聚合，以及同Hadoop数据源或者流数据进行关联等操作。

StreamSQL支持输出到非Hadoop存储系统。



创建一个到oracle_server上Oracle数据库oracle_db的dblink：

```
CREATE DATABASE LINK link_to_oracle CONNECT TO user IDENTIFIED BY 'password' USING 'jdbc:oracle:thin:@oracle_server:1521:oracle_db';
```

使用该dblink中的oracle_tab表与StreamSQL中的stream_tab表做join：

```
SELECT * FROM oracle_tab@link_to_oracle a JOIN stream_table b ON a.col = b.col;
```

THANKS

SequeMedia 盛拓传媒 IT168.com ChinaUnix ITPUB