干货 | Rheos SQL: 高效易用的实时流式SQL处理平台

原创 刘鲁滨 eBay技术荟 4月3日

供稿 | eBay Rheos Team

作者 | 刘鲁滨

编辑 | 顾欣怡

本文3404字，预计阅读时间11分钟

更多干货请关注“eBay技术荟”公众号

导读

本文介绍了**Rheos SQL**这一实时流式SQL处理平台在设计和实现中的细节。用户可以使用SQL语言，在Rheos SQL平台方便地编写、提交和维护实时流处理业务，节省开发和运维成本。

一、动机

01 实时即未来

自2015年“streaming 101: the world beyond batch”[1] 发表以来，已经过去了将近5年。现在，越来越多的人相信，**批处理是流式处理的特殊情况，基于流式处理的实时数据分析才是未来发展的方向。**

与传统批处理相比，**流式处理的优势**有以下几点：

1. **输出结果的延时更低**：从之前的**小时级**和**天级**，降为**分钟级**和**秒级**。
2. **模型更契合**：很多大规模数据，比如访问网站的用户行为数据，生成的模式是源源不断的，流式处理模型更符合数据本身生成的模式。

流处理的框架，在**开源和商用领域**也取得了很好的发展[2]，如今更加成熟和稳定，足够在生产环境广泛使用。其中**Apache Flink**[3]的模型定义和机制设计比较符合公司里的业务场景，很多用户都在其上建立自己的应用程序。

02 简单强大的SQL语言

SQL是数据分析人员最熟悉的语言。相对于传统运行在数据库和批处理中的SQL，**流式SQL有以下特点：**

1. **面相无界的流式数据源。**与流式处理一样，流式SQL所针对的数据源也是无界流，没有起始，也没有终点。
2. **依照流式处理的定义，增加相应的关键字和语义扩展。**流式处理常见的概念有：窗口（Window），触发器（Trigger），水位线（Watermark）等。

一些开源SQL引擎已经增加了对流式SQL的支持。比如，**Calcite**[4]添加了Window等关键字。流处理框架产品，也几乎都增加了原生对于SQL的支持，比如**Flink SQL**[5]和**KSQL**[6]。使用SQL而不是更底层的API来描述业务逻辑的好处有：

1. **节省开发成本**：作为简单易用的高级语言，SQL的描述能力更强，同样的逻辑开发起来更快速。
2. **节省维护成本**：修改清晰易懂的SQL脚本，比修改复杂的代码逻辑更容易。
3. **更好的透明性**：对底层细节的隐藏，使得用SQL语言表达的逻辑可以规避掉一部分底层实现变动带来的迁移。

当然，由于抽象层次更高，SQL可能只能覆盖**80%**的用户场景，所以，流处理框架通过支持User Defined Function(UDF)和Complex Event Processing(CEP)来满足更高级别用户的需求。

03 拥抱开源社区的同时弥补缺陷

开源社区在流处理和流式SQL处理领域都已经做了大量的工作，没有必要从头开始重新造轮子。**但是，与其他开源社区产品一样，在公司内部做到开箱即用，还是会存在一些差距**，主要表现为以下几个方面：

1. **SQL语法和语义本身不够全面**：真正的线上业务场景复杂，使用到的外围系统众多，开源产品不能直接全部满足。举个例子，公司内部有一种将RESTful服务作为维表，与源数据进行JOIN的需求，但在当前的开源产品中，并不能找到原生的支持。

2. **易用性**：开源的SQL处理引擎是作为框架形式提供的，用户需要自己管理SQL脚本、UDF，并考虑如何与底层的执行引擎做集成。

3. **稳定性**：开源产品的接口变动是不受单个公司控制的，业务团队都希望只花力气在业务逻辑的开发，而不是不断地因平台层的接口变动而修改逻辑。

4. **监控与报警**：开源产品通常是提供基础监控的，但是要在生产环境广泛使用，且往往需要与公司内部的监控报警系统集成。在开源基础上，也需要依照实际情况添加更多的监控指标。

二、功能

本章罗列了Rheos SQL作为流式SQL处理引擎所支持的功能。

01 SQL语义与语法

当前支持的语法模块如下表所示：

模块	模块细节	细节介绍
DDL	Source	流式SQL的数据源表，表示从外部数据源读取数据
	Side (Lookup)	流式SQL的数据维表，用来做源表数据拓展
	View	临时性的视图，存放中间结果
	Sink	流式SQL的目标数据表，将计算结果导出到外部存储
DQL	Queries	普通的SQL查询语句，ANSI SQL语法
DML	Insert	数据插入到VIEW或者SINK，当Key相同时，可支持更新
Extensions	Join Side Table	数据源表与维表JOIN
	Window	流式计算概念，支持时间和事件窗口
	Trigger	流式计算概念，在窗口结束之前提前按照规则输出结果
	UDF	用户自定义函数

（点击可查看高清大图）

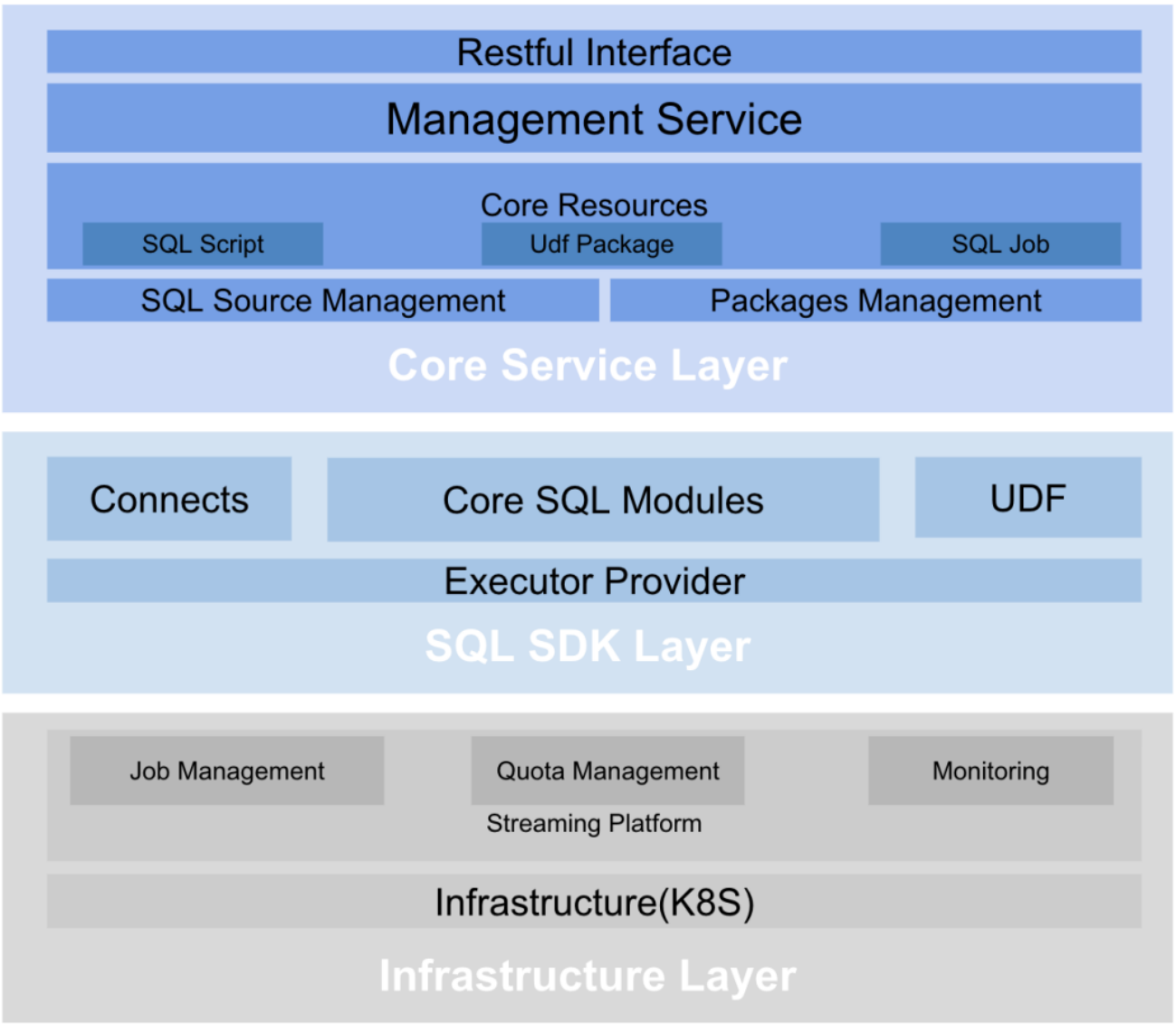
02 与外部资源对接模块

当前支持访问的外部模块如下表所示：

	Source	Side (Lookup)	Sink
Rheos Kafka	✓		✓
Cassandra		✓	✓
Couchbase		✓	✓
ElasticSearch			✓
MySQL		✓	
Restful		✓	
Oracle		✓	

（点击可查看高清大图）

三、架构



（点击可查看高清大图）

如上图所示，在总体架构上，**Rheos SQL分为三层：核心服务层**（Core Service Layer），**SQL开发工具包层**（SQL SDK Layer）以及**基础服务层**（Infrastructure Layer）。

01 核心服务层（Core Service Layer）

该层主要负责管理面相用户的SQL资源，是用户接触的主要接口。

> 以RESTful服务的形式，将Rheos SQL的核心资源暴露给用户，**核心资源包括：**
>
> 1. **SQL脚本**（SQL Script）：用户所编写的SQL语句，用以描述业务逻辑，属性包括存贮的位置等。
> 2. **UDF包**（UDF Package）：用户的自定义函数包，属性包括存储的位置，包中支持的函数等。
> 3. **SQL作业**（SQL Job）：SQL运行时的抽象，是SQL脚本的实例化之后，在集群中运行的实例，属性包括关联的SQL脚本，启动参数等。
>
> **SQL源脚本的管理模块**：管理SQL脚本的生命周期，比如支持运行时拉取。
>
> **包管理模块**：管理用户上传包的生命周期，比如运行时加载。

02 SQL开发工具包层（SQL SDK Layer）

该层负责在开源流式SQL引擎上增强扩展，支持更丰富的语义和语法。

> **核心SQL模块**（Core SQL Modules）：负责SQL的解析，优化和执行计划生成。
>
> **与外部资源对接模块**（Connects）：加载和拉取存储在外部依赖中的数据，比如Kafa-Connect会维护Kafka的Consumer，从Kafka消费数据，并交给流处理引擎。
>
> **用户自定义函数模块**（UDF）：支持自定义函数和组件，比如对源数据的自定义解析函数。
>
> **执行提供模块**（Executor Provider）：抽象层，隐藏底层实现细节，保证对外编程接口稳定。

当前，SQL工具包是在FLINK SQL的基础上，提供了功能扩展。

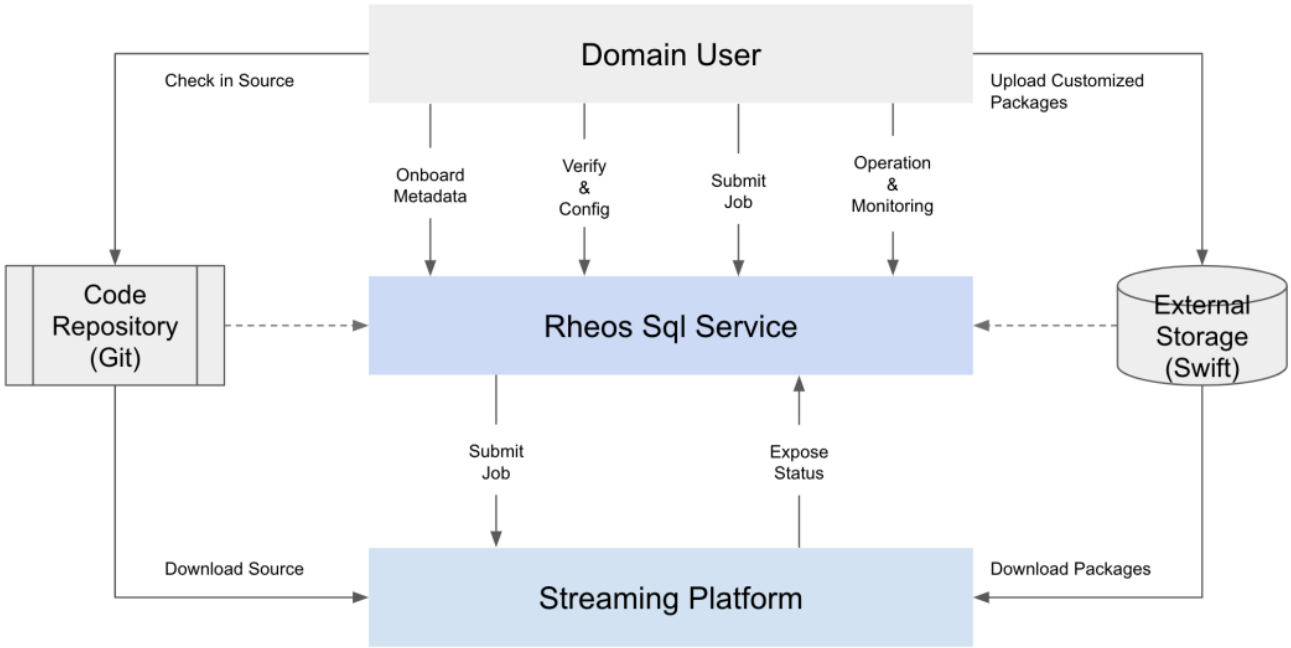
该层的设计可以帮助我们在不影响用户的同时，灵活切换底层基础服务的实现。比如，Rheos SQL可以内部升级FLINK的版本，或者使用其他流处理框架实现流处理的功能，而用户对接的则一直是Rheos SQL定义的语法与编程接口。

03 基础服务层（Infrastructure Layer）

该层负责运行时基础设施的提供与维护。在公司内部，所有集群都运行在**K8S**平台，并有流平台团队提供了流处理基础设施的管理。

当前，Rheos SQL选用的流处理框架是**Apache Flink**，流平台团队管理了Flink集群的创建与维护，并支持资源额度管理、基础监控暴露收集等。

四、用户体验



（点击可查看高清大图）

上图展示了Rheos SQL平台的使用流程，**其用户体验可归结如下：**

首先，像开发其他应用程序一样，编写SQL源代码，并提交到代码库。**在我们的实现中，提供对接两种代码库的实现：**

1. **Git**：作为公司广泛使用的代码库管理工具，Git本身就提供强大的版本控制和历史信息追踪，推荐部署到生产环境的Rheos SQL应用都采用这种源代码库。
2. **DB**：直接将SQL源脚本提交到Rheos SQL系统，落库。适用于跑DEMO，或者功能性验证阶段。

如果是比较高级的用户，需要编写UDF或者在SQL工具包的基础上定制功能，可以在本地编码结束后，上传到远程的存储。流平台团队提供了**MAVEN插件**，可以在IDE中方便地完成上传。在实现时，我们使用的对象存储是**Swift**[7]。

除此之外，用户需要将SQL脚本和自定义的包在Rheos SQL的系统中注册，完成源信息的提交（Metadata Onboard）。

为了尽快找到程序中的bug，验证逻辑和配置细节，Rheos SQL平台提供了**线上验证和配置**的功能。

一切就绪后，可以在Rheos SQL平台提交作业。Rheos SQL会将这个SQL作业，转化成**底层流平台**具体实现的作业。在当前实现中，Rheos SQL作业会被注册成Flink的作业，存放到流平台上。

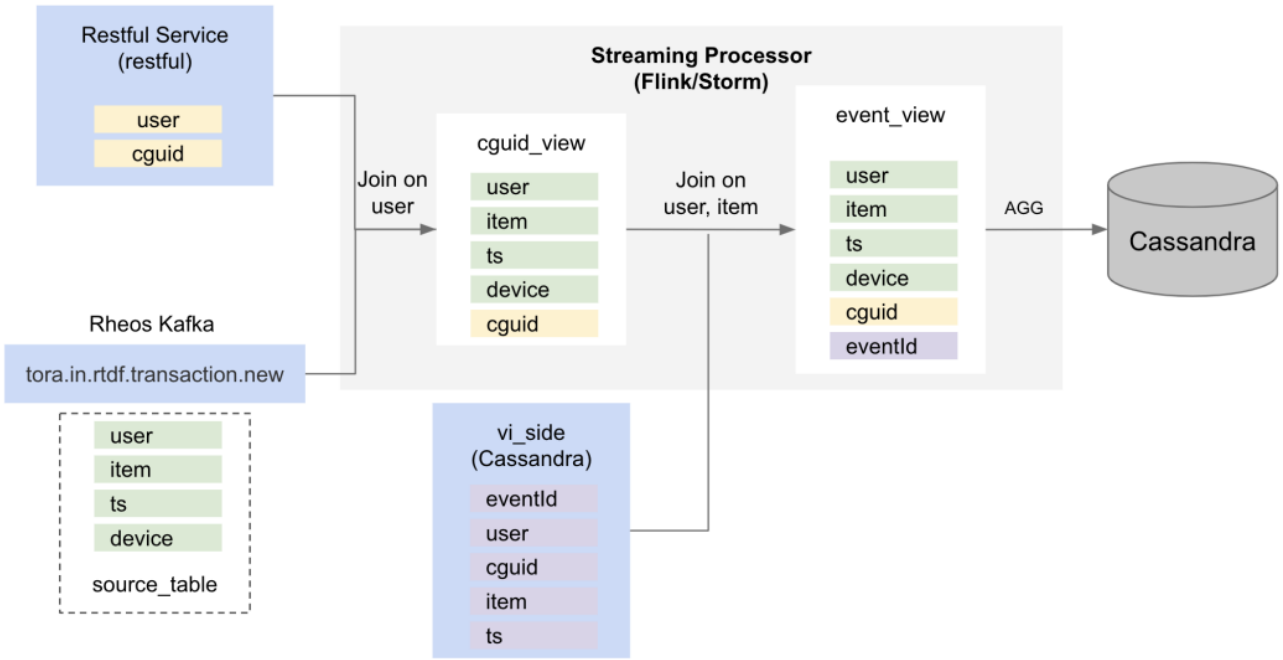
在运行时，Rheos SQL的工具包会根据用户注册的信息，动态拉取SQL脚本的源代码，并加载用户自定义的扩展包，进行解析、优化并在流框架上执行用户逻辑。

作业运行起来后，可以通过Rheos SQL的门户网站操作和监控。

五、与维表的JOIN

01 用例

在现有的业务场景中，有很多使用源表数据与外部存储JOIN的用例。



(点击可查看高清大图)

上图是实际应用的一个例子，输出与用户购买行为相关的统计信息。SQL作业的数据源是存放在**Kafka**中的用户交易数据。

处理的第一步，是将源表与一张RESTful形式的维表根据user域做JOIN，构造出cguid_view临时表。

处理的第二步，是将cguid_view临时表，与存放在Cassandra中的一张维表，根据user和itemId域做JOIN，输出event_view临时表。

处理的第三步，是在event_view临时表上做聚合，将最终的结果输出到Cassandra。

02 实现细节

为了提升性能，在实现维表JOIN时，Rheos SQL重点做了以下两方面的工作：

1. **LRU缓存**：为了减少与外部存储的频繁交互，在内存中会默认开启LRU的缓存。同样的KEY，在缓存失效前，会优先从缓存中返回。新的KEY在首次取回之后，也会加入到缓存中。

2. **异步查询**：为了在等待外部存储返回结果前，避免同步等待IO，Rheos SQL使用了FLINK的Async API实现异步查询。Rheos SQL中的所有维表实现，都继承自一个支持异步的抽象类，子类实现数据拿取的业务逻辑，并将结果以Future的形式返回给父类。父类负责对结果的最终处理。

六、Debug和监控

为了加快迭代速度，及时发现问题，Rheos SQL在不同的阶段提供了丰富的工具帮助用户调试和监控SQL作业。

01 本地测试框架

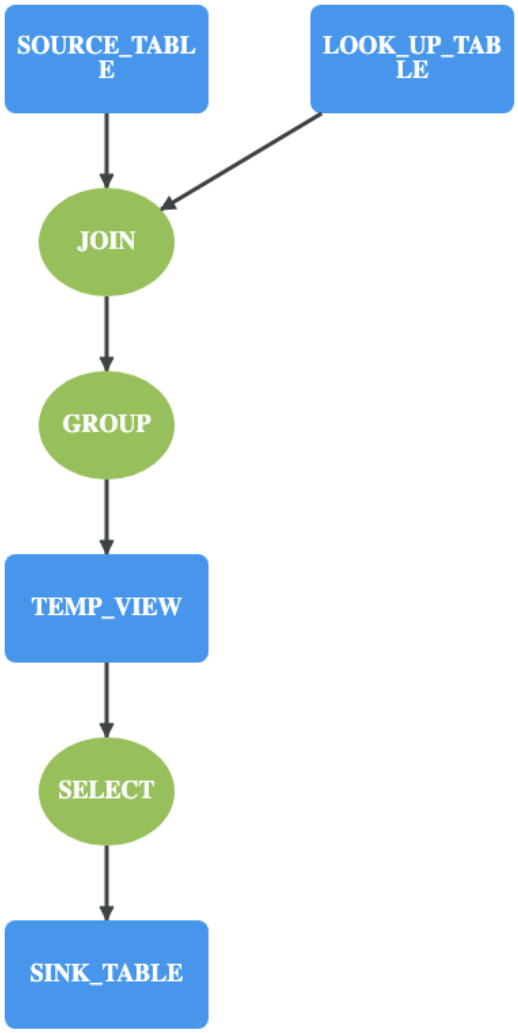
SQL开放工具集中，提供了一个**本地测试框架**，具体功能为：

1. 以文件的形式，定义SQL作业的输入和输出。
2. 本地启动所有的外部依赖模块，将输入写到源和对应的维表。
3. 运行SQL的处理逻辑，并将输出写入到目标存储和文件。
4. 比较目标输出是否与用户一开始定义的一致，并返回比较结果。

在这个测试框架的帮助下，用户可以本地完成SQL脚本的开发和测试工作，并初步验证逻辑的正确性，及时对bug做好修正。

02 SQL逻辑图

用户在Rheos SQL系统中注册好SQL之后，可以查看**SQL的逻辑执行图**。



(点击可查看高清大图)

上图是一个实际应用中的例子。源表和维表在第一次JOIN之后，将聚合的结果写入到了TEMP_VIEW，然后从临时表中选取了部分数据输出到目标表。

用户通过逻辑执行图，可以验证逻辑是否符合预期，在将SQL作业真正运行前，发现问题并修正。

03 线上SQL作业的监控

除去Flink本身提供的监控指标之外，Rheos SQL还提供了很多从SQL表级别暴露的信息。根据表类型的不同，部分指标详情如下：

1. **源表**：消费者尚未读取的数据量（Consumer Lag），读入数据条数，读入数据量等。
2. **维表**：接受数据条数，读取外部数据响应时间，与外部系统建立的连接数等。
3. **目标表**：输出的数据条数，写入外部系统失败的条数等。

用户通过对业务指标的监控，可以在作业运行时感知到异常情况的发生，及时采取应对措施。

总结与展望

Rheos SQL平台在开源流式SQL处理框架的基础上，提供了丰富的语义与扩展。用户可以在Rheos SQL平台上，方便地开发、调试、监控SQL作业，节省流式作业的开发，维护成本。接下来，Rheos SQL在进一步满足用户需求的同时，将会在**资源管理、动态扩容以及CEP语义支持**等方面，投入更多的努力。

参考文献

[1] https://www.oreilly.com/radar/the-world-beyond-batch-streaming-101/
[2] https://www.upsolver.com/blog/popular-stream-processing-frameworks-compared
[3] https://flink.apache.org/
[4] https://calcite.apache.org/
[5]https://ci.apache.org/projects/flink/flink-docs-release-1.10/dev/table/sql/
[6] https://www.confluent.io/product/ksql/
[7] https://wiki.openstack.org/wiki/Swift

您可能还感兴趣：

分享 | “三高”产品设计的这些坑，你是不是也踩过？（上）

分享 | “三高”产品设计的这些坑，你是不是也踩过？（下）

一探究竟 | eBay流量管理之看不见的手

解密 | 一桩由数据洁癖引发的DNS悬案

分享 | eBay流量管理之Kubernetes网络硬核排查案例

分享 | eBay流量管理之负载均衡及应用交付