徐振中 28 分钟阅读

我的名字是徐振中，
我于2015年加入Netflix，担任实时数据基础设施团队的创始工程师，后来领导了流处理引擎团队。我在2010年代初就对实时数据产生了兴趣，并从那时起就认为有很多价值有待发掘。

Netflix是一个奇妙的地方，被许多了不起的同事所包围。我为参与这个将共同信念变为现实的旅程的每个人感到无比自豪。我想用简短的时间来分享团队的主要成就。

- 我们把Netflix所有组织的流数据使用案例从0增加到2000多个。
- 我们建立并发展了成功的产品，如 Keystone, managed Flink Platform, Mantis和管理的Kafka平台。这些产品在数据生态系统的许多方面提供解决方案：包括数据摄取、移动、分析和操作处理以及机器学习用例。
- 我们是行业内第一批扩大开源Kafka和Flink部署的公司，在2017年左右每天处理1万亿个事件，后来到2021年又扩大了20倍的规模

几个月前，我离开了Netflix，去追求一个类似但更大的愿景。 实时机器学习，我认为现在是总结我在Netflix建立实时数据基础设施的经验的最佳时机。我希望这篇文章能帮助平台工程师开发他们的云原生、自助服务的流数据平台，并在许多业务功能中扩展用例（不一定是来自我们的成功，可能更多的是来自我们的失败）。我也相信，了解数据平台的构建方式可以帮助平台用户（如数据科学家和ML工程师）充分利用他们的平台。

我将分享实时数据基础设施在Netflix的四个阶段的迭代历程（2015-2021）。

Phase 1: 从批处理管道中拯救Netflix的日志数据(2015)

在Netflix的全球超增长期间，业务和运营决策比以往任何时候都更依赖于更快的记录数据。在2015年，基于Chukwa/Hadoop/Hive的批处理管道难以扩展。在这个阶段，我们从头开始建立了一个流数据优先的平台，以取代失效的管道。

Phase 2: 扩展100个数据移动的使用案例 (2016)

在最初的产品发布后，内部对数据移动的需求稳步上升。我们不得不把重点放在常见的用例上。在这个阶段，我们通过建立一个具有简单而强大的构件设计的自我服务、完全管理的平台，扩展到支持100多个用例。

Phase 3:
支持客户需求，扩展到1000个用户案例(2017–2019)

随着流处理在Netflix的发展势头，许多团队要求在数据工程、可观察性和机器学习领域有更低的延迟和更多的处理灵活性。在这个阶段，我们建立了一个新的流处理开发经验，以实现定制的用例，我们还解决了新的工程和运营挑战。

Phase 4: 扩展流处理的责任，挑战和机会并存 (2020 — Present)

随着行业内数据平台技术的快速发展，出现了很多新的挑战：协调困难、学习曲线陡峭、流与批的界限分化等。在这个阶段，我们探讨了流处理在连接技术和提高抽象性方面发挥了更突出的作用，使数据平台更容易使用。摆在我们面前的机会还有很多。

对于每个阶段，我都会去了解不断变化的商业动机、团队的独特挑战、战略赌注，以及我们一路走来发现的用例模式。

许多人帮助我审查这个帖子，如果没有他们的反馈，我永远不会挖掘出许多不偏不倚的细节。特别感谢 Chip Huyen, Steven Wu, Prashanth Ramdas, Guanhua Jiang, Scott Shi, Goku Mohandas, David Sun, Astasia Myers, , 和 Matt Willian!

Netflix的实时数据基础设施的四个阶段

第一阶段：从失败的批处理管道中拯救Netflix日志（2015年）。

背景介绍

2015年，Netflix已经拥有约60MM的用户，并且正在积极地扩展其国际业务。我们都知道，迅速扩大平台的杠杆作用将是维持用户飞速增长的关键。

题外话：对于那些不熟悉的人来说，平台团队通过集中管理基本的基础设施来提供优势，这样产品团队就可以专注于业务逻辑。

我们的团队必须弄清楚如何帮助Netflix扩展日志实践。当时，Netflix有~500个微

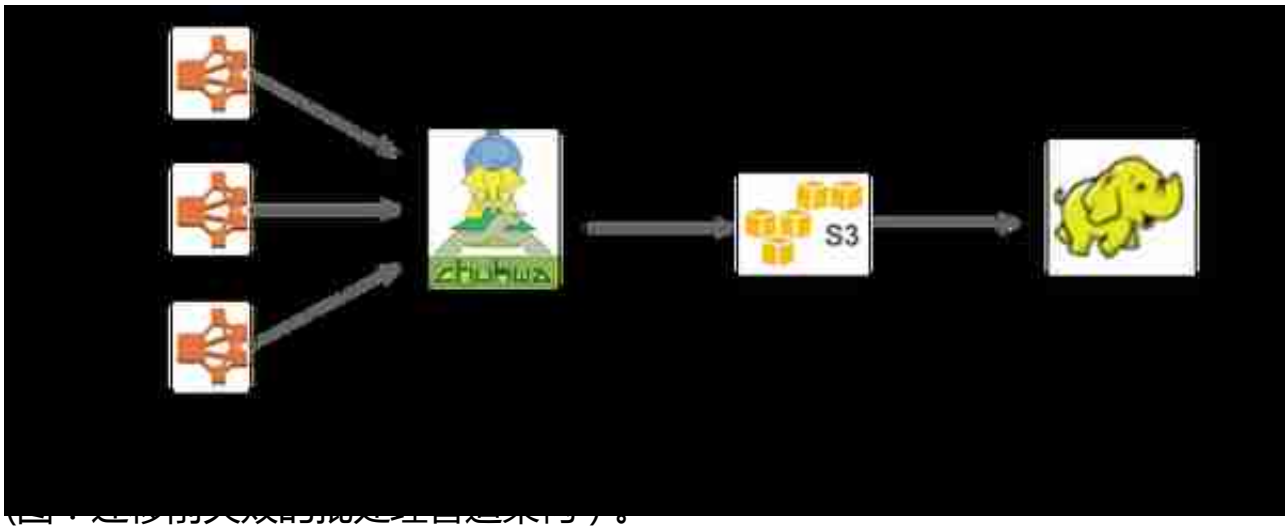
服务，每天在生态系统中产生超过10PB的数据。

收集这些数据对Netflix来说有两个主要目的。

- 获得商业分析洞察力
  （例如，用户保留率、平均会话长度、什么是趋势等）。
- 获得运营洞察力（例如，测量 streaming plays per second来快速、轻松地了解Netflix系统的健康状况），因此开发人员可以发出警报或执行缓解措施。

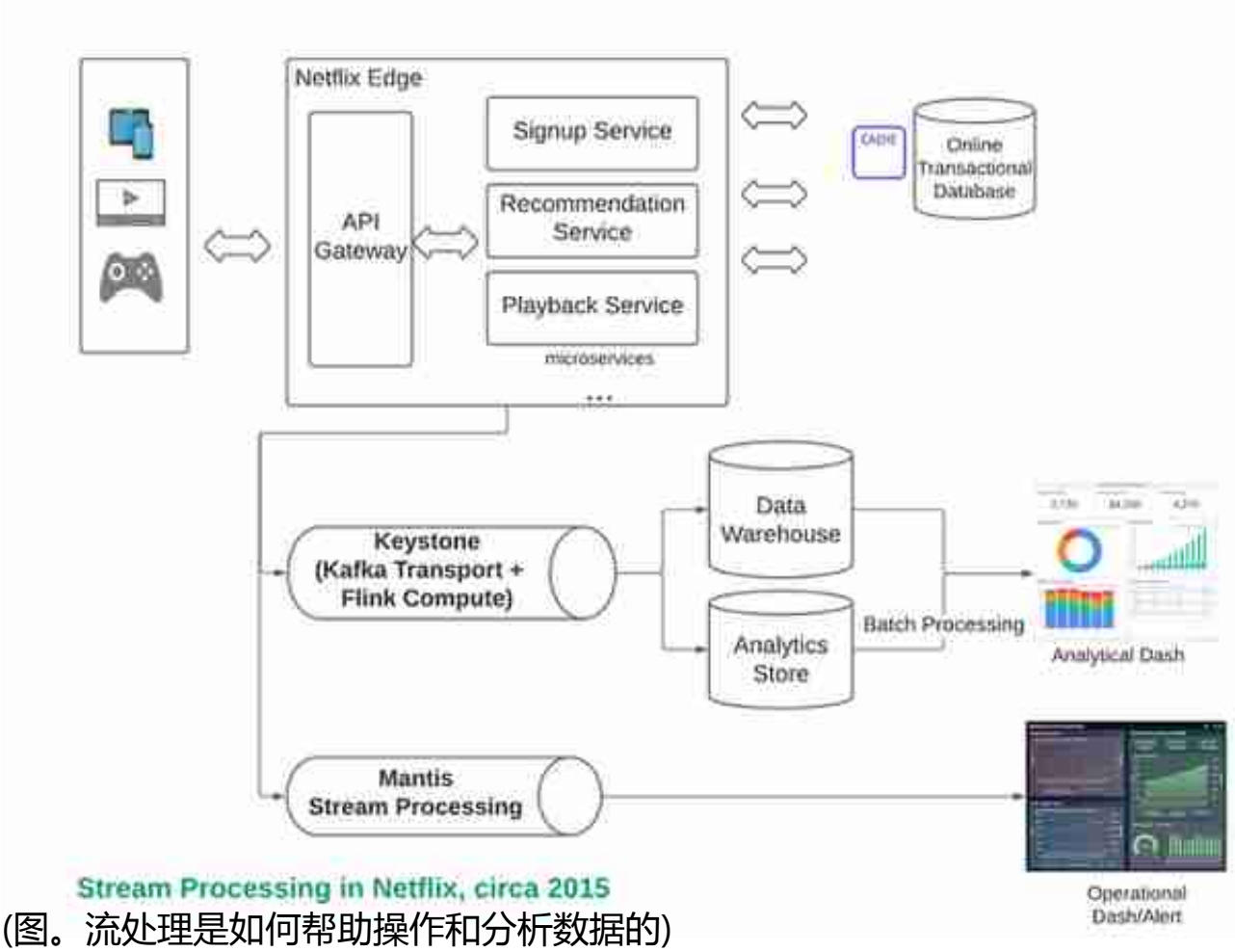
你可能会问，为什么我们首先需要将日志从边缘转移到数据仓库？由于数量庞大，在在线交易数据库上大规模地进行按需分析是不可行的。原因是，在线交易处理（OLTP）和在线分析处理（OLAP）是以不同的假设建立的--OLTP是为面向行的访问模式建立的，而OLAP是为面向列的访问模式建立的。在引擎盖下面，它们使用不同的数据结构进行优化。

例如，假设我们想知道数以亿计的Netflix用户的平均会话长度。假设我们把这个按需分析的查询放在一个面向行的OLTP系统上。它将导致行级粒度的全表扫描，并有可能锁定数据库，应用程序可能变得没有反应，导致不愉快的用户体验。这种类型的分析/报告工作负载最好在OLAP系统中完成，因此需要以低延迟的方式可靠地移动日志。



(图：迁移前失败的批处理日志架构)。

我们决定用以下方式取代这个失败的基础设施 Keystone.



(图。流处理是如何帮助操作和分析数据的)

策略赌注

赌注1：在建立MVP产
品时，要为最初的几个客户建立它。在探索最初的产品-市场契合度时，很容易被分散注意力。我们决定只帮助几个高优先级、高数量的内部客户，以后再担心扩大客户群的问题。这个决定不仅使我们能够专注于产品的核心。它还使我们意识到什么是不应该投资的（例如，我们用电子表格而不是一个成熟的控制平面来保存客户的名字、电子邮件地址和每个管道的元数据信息，以方便在MVP阶段的客户部署）。

赌注2：与技术伙伴共同
发展，即使不处于理想的成熟状态
，而不是自己重新发明车轮。这样一来，我们就可以共同发展生态系统了。我们在早期就选择了合作。

- 流媒体合作伙伴。在外部，我们与那些引领行业流处理工作的合作伙伴合作，例如LinkedIn（Kafka和Samza团队）、Confluent、Data

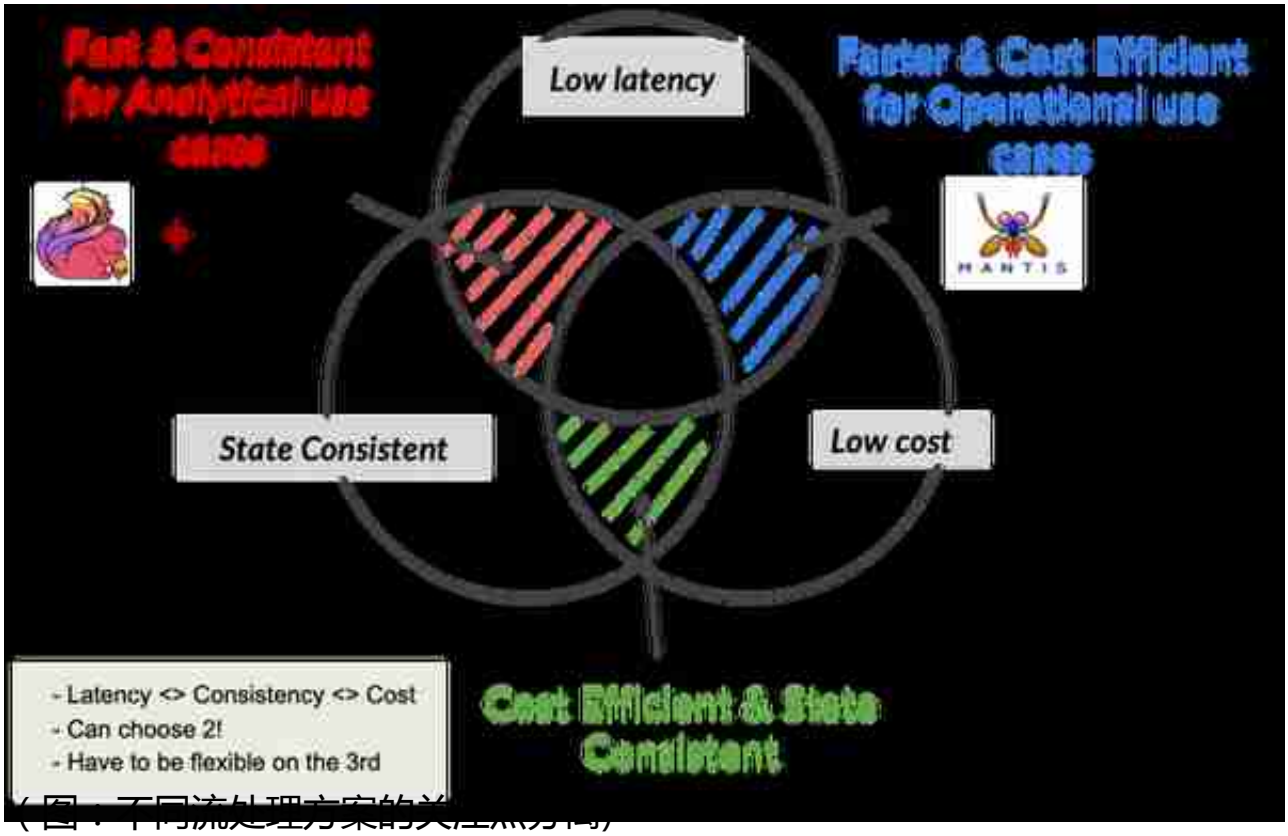
Artisan（Apache Flink背后的建设者，后来改名为Veverica）。这样做使我们能够为我们的需求贡献开放源码软件，同时利用社区的工作。

- 容器化合作伙伴：在2015年，这还是容器虚拟化技术的早期阶段。我们需要一个战略来快速部署。在
  内部，我们与新创建的容器基础设施建立了伙伴关系 Titus team.Titus建立在Apache Mesos之上，通过抽象的API，提供计算资源管理、调度和隔离部署。Titus后来在2020年初发展到利用K8S，他们的团队设法透明地迁移所有工作负载。由于这种合作关系，我们在建立数据平台时不必担心底层的计算基础设施。

在合作过程中，我们一直保持沟通，分享学习成果并提供反馈。我们每两周与亲密的合作伙伴举行一次同步会议，以统一目标并讨论问题和解决方案。当出现阻碍性问题时，合适的人将会立即参与进来。

赌注3：将关注点脱钩，而不是忽略它们。

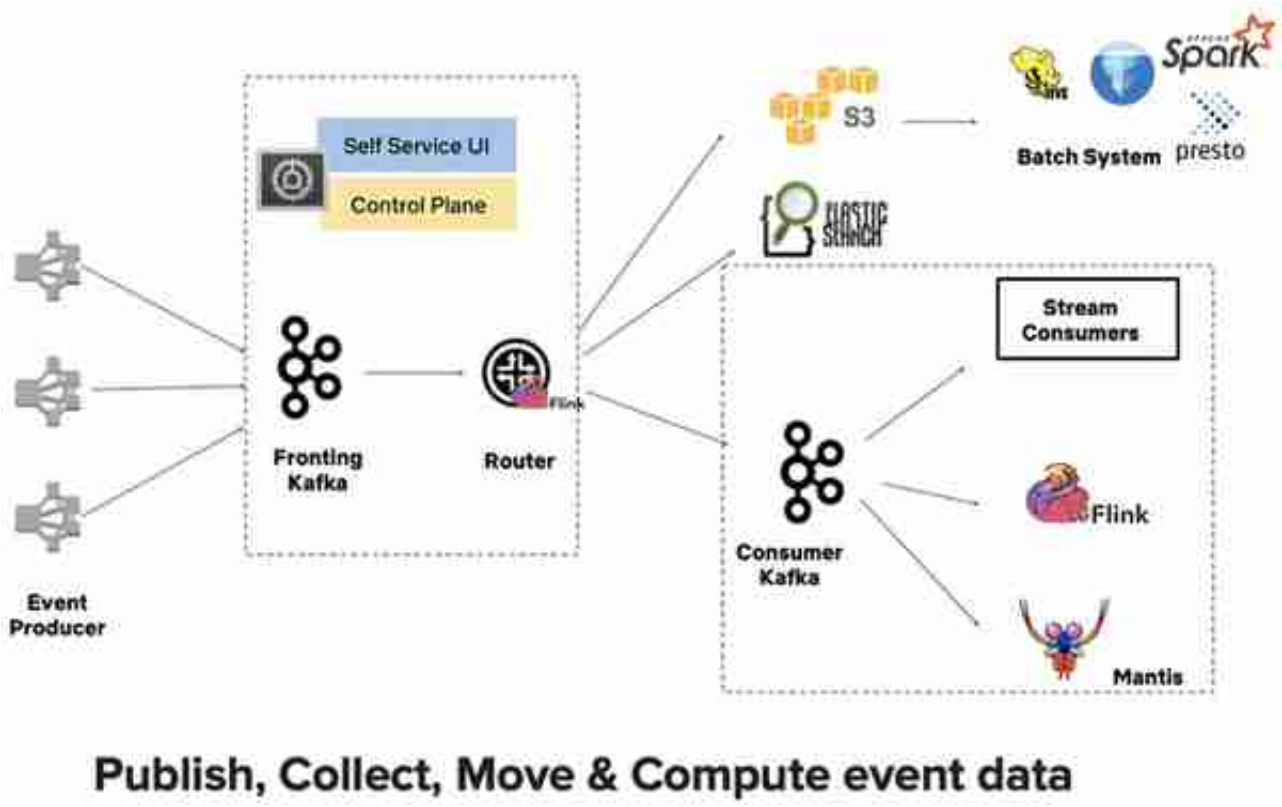
- 将运营和分析用例之间的问题分开。我们将Mantis（以运营为重点）和Keystone（以分析为重点）分开发展，但为这两个系统创造了接口空间。



（图：不同流处理方案的关注点分离）

- 分离生产者和消费者之间的关注。我们引入了生产者/消费者客户端，配备了标准化的电线协议和简
  单的模式管理，以帮助
  生产者和消费者的开发工作流程脱钩。它后来被证明是数据治理和数据质量控制的一个重要方面。
- 分离的组件责任。我们从面向微服务的单一责任原则开始，将整个基础设施分为消息（流传输）、处理（流处理）和控制平面（控制平面在这个阶段只是一个脚本，后来演变成一个成熟的系统）。组件责任的分离使团队在早期就能在接口上保持一致，同时通过同时关注不同的部分来
  释放团队的生产力。

赌注4：投资建立一个
预期会发生故障并监控所有操作的系统
，而不是拖延投资。由于云的弹性、不断变化、故障概率较高的特点，我们需要设计系统来监测、检测和容忍各种故障。故障的范围包括网络突发事件、实例故障、区域故障、集群故障、服务间拥堵和背压、区域灾难故障等。我们在构建系统时，假设故障是持续的。我们在早期就接受了DevOps实践：如针对故障场景的设计、严格的自动化、持续部署、影子测试、自动化监控、警报等。这种DevOps的基础使我们拥有最终的工程敏捷性，可以每天多次发货。



(图：不断发展的Keystone架构图，约2016年。Keystone包括Kafka和Flink引擎作

为其核心组件。关于更多的技术设计细节，请参考博文，主要是关于 Kafka和 Flink)

挑战

挑战1：运营负担增加。
我们最初为新客户的入职提供白手套协助。然而，考虑到不断增长的需求，这很快变得不可持续。我们需要开始发展MVP，以支持超过十几位客户。因此，我们需要重建一些组件（例如，是时候把电子表格变成一个适当的数据库支持的控制平面了）。

挑战2：出现多样化的需求
。随着我们得到更多的客户要求，我们开始看到非常多样化的需求。有两个主要类别。

- 一组人倾向于使用简单的，完全管理的服务。
- 另一组人更喜欢灵活性，需要复杂的计算能力来解决更高级的业务问题，他们可以接受传呼，甚至管理一些基础设施。

我们不可能同时做好这两件事。

挑战3：我们打破了我们所接触的一切
。不开玩笑，由于规模太大，我们几乎在某些时候破坏了所有的依赖服务。我们破坏了S3。我们在Kafka和Flink中发现了许多bug。我们破坏了 Titus(管理容器基础设施）多次，并发现了奇怪的CPU隔离和网络问题。我们打破了 Spinnaker(持续部署平台），因为我们开始以编程方式管理数百个客户的部署。

幸运的是，这些团队也是最好的。他们与我们合作，逐一解决这些问题。这些努力对于整个流媒体生态系统的成熟至关重要。

策略赌注

赌注1：首先关注简单性，
而不是将基础设施的复杂性暴露给用户。我们决定首先关注高度抽象的、完全管理的服务，用于一般的流媒体使用案例，原因有二。

- 这将使我们能够解决大多数数据移动和简单的流式ETL（例如，投影、过滤）用例。为数据路由提供这种简单、高层次的抽象，将使所有Netflix组织的工程师能够将数据路由作为一个 "乐高

"积木，与其他平台服务一起使用。
- 这将使我们的用户能够专注于业务逻辑。

我们将在以后处理更高级的用例。

赌注2：投资于完全管理的多租户自助服
务
，而不是继续使用手动的白手套支持。我们必须专注于控制平面和工作负载部署的自动化。客户的工作负载需要完全隔离。
我们决定，一个客户的工作负载不应该以任何方式干扰另一个客户的工作负载。

通过A/B测试，选择最佳的艺术品来为用户提供个性化服务 Selecting the best artwork for videos through A/B testing

这些用例涉及更高级的流处理能力，如复杂的事件/处理时间和窗口语义，允许的延迟，大状态检查点管理。它们还需要围绕可观察性、故障排除和恢复的更多操作支持。一个全新的开发者体验是必要的，包括更灵活的编程接口和操作能力，如自定义可观察性堆栈、回填能力，以及管理10多个TB本地状态的适当的基础设施。我们在Keystone中没有这些，我们需要建立一个新的产品入口，但要尽量减少冗余的投资。

挑战2：在灵活性和简单性之间取得平衡
。
随着所有新的定制用例的出现，我们必须弄清楚适当的控制暴露水平。我们可以完全暴露最低级别的API，但同时也要牺牲更多具有挑战性的操作（因为我们永远无

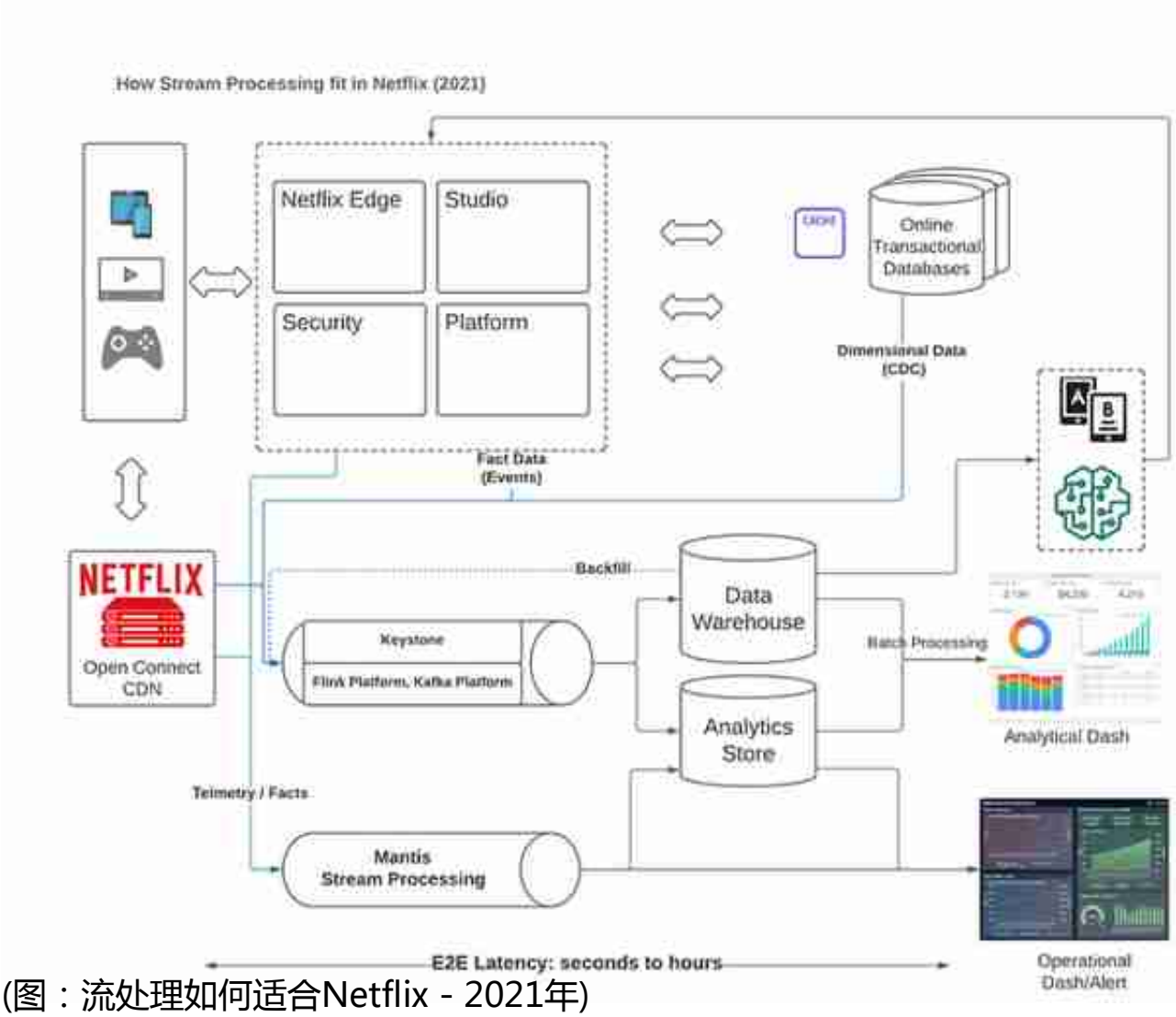
法完全预测用户将如何使用该引擎）。或者我们可以选择走中间路线（例如，暴露有限的功能），冒着让客户不满意的风险。

挑战3：增加操作的复杂性。
支持自定义用例要求我们增加平台的自由度。因此，我们需要改善许多复杂场景下的操作可观察性。同时，随着平台与许多其他数据产品的整合，我们系统的接触点增加，需要与其他团队进行业务协调，以更好地服务我们的集体客户。

挑战4：中央平台与本地平台。
我们团队的责任是提供一个集中的流处理平台。但是由于之前的策略是专注于简单性，一些团队已经在他们的本地流处理平台上投资，使用不支持的技术，例如Spark Streaming。我们必须说服他们重新回到铺设好的道路上，因为他们有可能失去平台的杠杆作用，并在多余的投资上浪费资源。现在是合适的时机，因为我们扩展到了自定义用例。

策略赌注

赌注1：建立一个新的产品入口，
但重构现有架构
，而不是孤立地建立一个新产品。在分析处理方面，我们决定从原来的架构中剥离出一个新的平台，以暴露出Apache Flink的流处理的全部能力。我们将从头开始创建一个新的内部客户群，但我们也决定现在是重构架构的正确时机，以尽量减少冗余投资（在Keystone和Flink平台之间）。在这个新架构中，较低的Flink平台同时支持Keystone和自定义用例。



(图：流处理如何适合Netflix - 2021年)

背景介绍

随着流处理的使用案例扩展到Netflix的所有组织，我们发现了新的模式，并且我们享受了早期的成功。但现在还不是自满的时候。

Netflix作为一个企业，继续探索新的领域，并在内容制作室和最近的游戏领域进行了大量投资。一系列新的挑战出现了，我们跳进了解决这些有趣的问题空间。

挑战

挑战1。不同的数据技术使协调变得困难
。由于团队被授权，Netflix的许多团队都在使用各种数据技术。例如，在事务性方面：有Cassandra、MySQL、Postgres、CockroachDB、Distributed Cache等。在分析方面：有Hive, Iceberg, Presto, Spark, Pig, Druid, Elasticsearc

h等。在Netflix的数据生态系统中，同一数据的许多副本往往被存储在不同的数据存储中。

由于有许多选择，把技术放在划分的桶里是人类的天性。批量与流。事务性存储与分析性存储。在线处理与离线处理。这些都是数据世界中经常争论的话题。重叠的分界线往往给终端用户带来更多的困惑。

今天，跨技术边界的
数据协调和工作是令人难以置信的挑战。前沿是很难用划分的边界来推动的。

挑战2：更陡峭的学习曲线。
随着可用数据工具的不断增加和专业化程度的持续加深，用户要学习并决定什么技术适合特定的使用情况，这对他们来说是一个挑战。

挑战3：ML实践没有利用数据平台的
全部力量
。之前提到的所有挑战都给ML实践带来了损失。数据科学家的反馈回路很长，数据工程师的生产力受到影响，产品工程师在分享有价值的数据方面遇到挑战。最终，许多企业失去了适应快速变化的市场的机会。

挑战4：中央平台模式的规模限制。
随着中央数据平台以超线性的速度扩展用例，单点支持的做法是不可持续的。现在是评估中央平台支持地方-中央平台以增加杠杆的模式的时候了（这意味着我们将优先支持建立在我们平台之上的地方平台）。

机会

我将对这部分内容进行相对简单的介绍，并在今后的博文中展开详细介绍。

用流来连接世界
。对于流处理，除了低延迟处理的优势外，它在现代数据平台中越来越显示出更关键的优势：连接各种技术，实现流畅的数据交换。诸如变化数据捕获（CDC）、流式物化视图和数据网格概念等技术正在得到普及。最后，马丁-
克莱普曼在2015年提出的""愿景开始实现其价值。Turning Database Inside Out”开始实现其价值。

通过结合简单性和灵活性的优点来提
高抽象性
。了解各种数据技术的深层内部结构有很多价值，但不是每个人都需要这样做。在

云优先的数据基础设施正在成为商品的时候，这种思路尤其正确。适当提高数据基础设施的抽象性，成为让更多人容易获得所有先进能力的直接机会。诸如流式SQL等技术将降低入门门槛，但这只是个开始。数据平台也应该提高抽象度，使最终用户看不到分界线（例如，流媒体与批处理）。