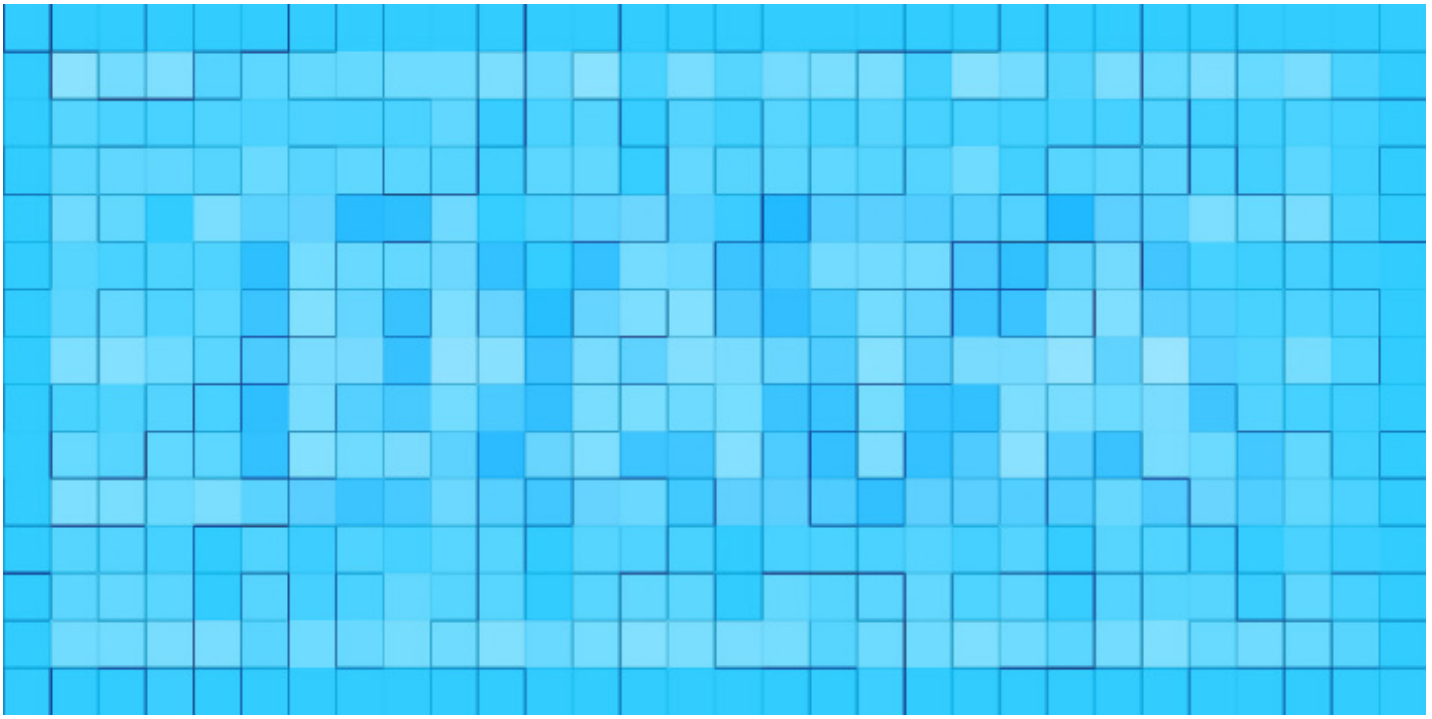
揭秘 IFTTT 每天处理几十亿事件数据的基础结构

2015/11/21 • IT技术 • ifttt, kafka

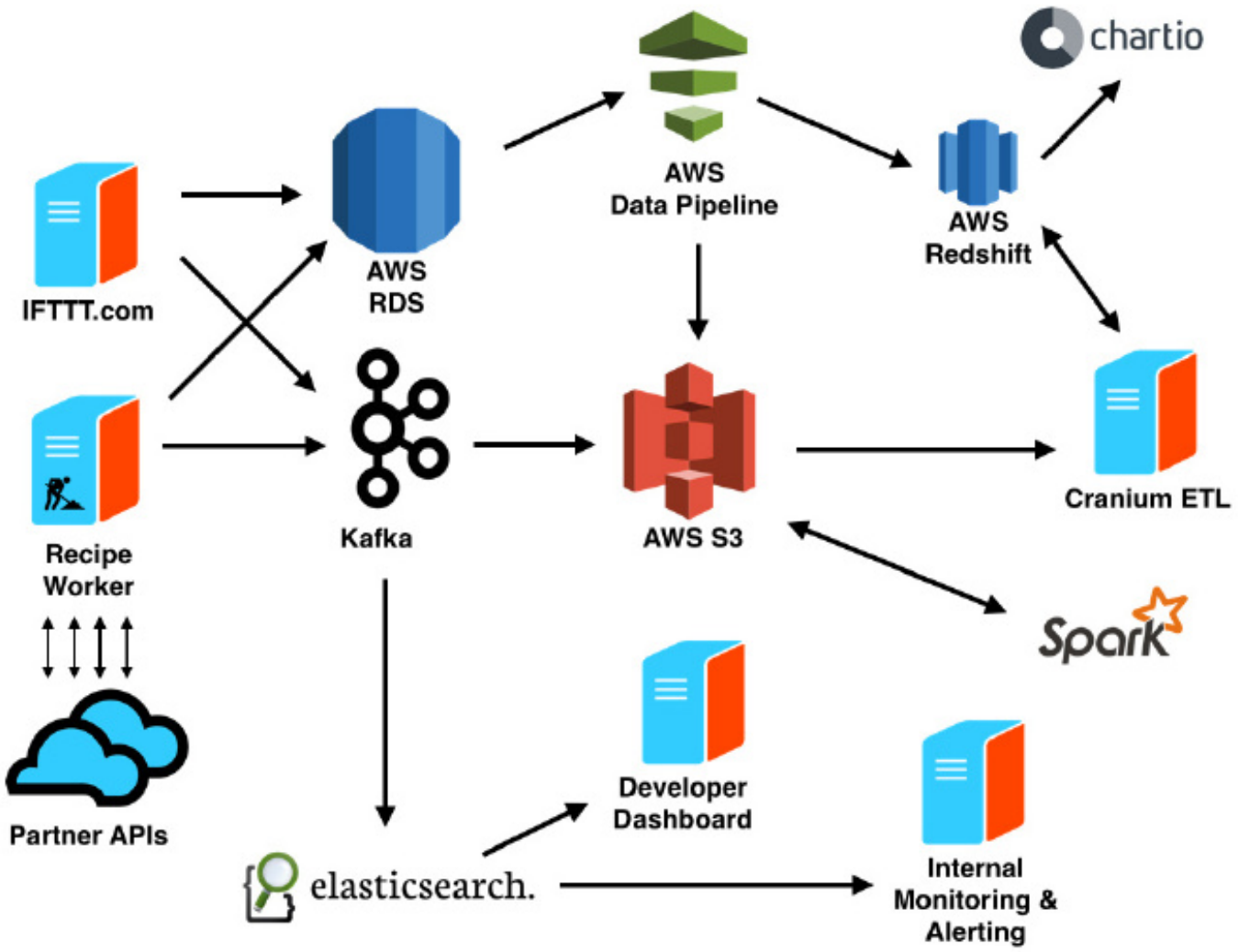
分享到： 4

本文由 伯乐在线 - mtunique 翻译，黄利民 校稿。未经许可，禁止转载！
英文出处：IFTTT。欢迎加入翻译组。



数据对于IFTTT至关重要。我们的 BD 和营销团队依靠数据来做出关键的业务决策。产品团队依靠数据来测试来了解用户是如何时候我的产品的，以做产品决策。数据团队本身依靠数据建立产品，比如Recipe推荐系统和垃圾邮件检测工具。此外，我们的合作伙伴依靠数据来实时获取Channels的性能。

因为数据对于IFTTT如此关键，并且我们的服务每天处理几十亿事件，所以我们的数据基础设施必须具有高可扩展性、可用性的和弹性，以便跟上产品的快速迭代。在这篇文章中我们将带你浏览数据基础设施和架构，也会分享一些我们在构建和操作数据的收获。



数据来源

在IFTTT有三种数据来源，对于理解用户的行为和Channels的效率至关重要。

第一，在 AWS RDS 有一个 MySQL 集群，用来维护应用的基本内容的当前状态，比如用户、Channels和Recipes，包括他们的关系。IFTTT.com 和移动应用靠 Rails 应用支撑。获得数据并导出到S3，每天用AWS Data Pipline导入到Redshift。

接下来，像用户与 IFTTT 产品的交互数据，我们从Rails应用里收集事件数据到Kafka集群。

最后，为了帮助监控数以千计的合作伙伴 API 的行为（behavior），我们收集在运行 Recipes 时 workers 产生的 API 请求的信息，包括响应时间和HTTP状态码，都导入到Kafka集群。

IFTTT 的 Kafka

我们使用Kafka做数据传输层，实现数据生产者和消费者之间的解耦。这种架构中，Kafka在系统中作为生产者和消费者之间的一种抽象层，而不是数据直接推向消费者。生产者将数据推送到Kafka，消费者从Kafka中读数据。这使得添加新数据消费者更松散。

因为Kafka作为基于日志的事件流，因为消费者跟踪他们自己在事件流中的位置。这使消费者数据能在两种模式中切换：实时和批量。它还允许消费者数据进行重新处理他们以前所消耗的数据，如果在产生了错误的情况下数据需要重新处理，这将很有帮助。

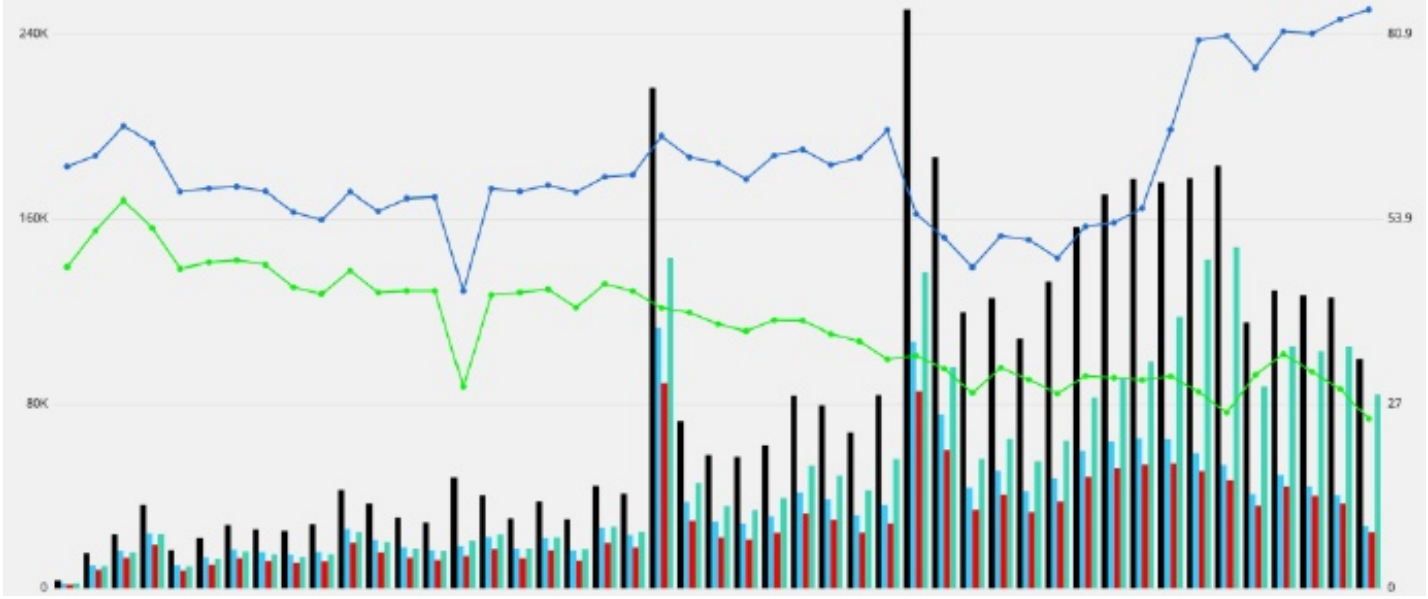
一旦数据在Kafka中，我们将它用在各种位置。批量数据的消费者每小时将数据的一个副本分批放到 S3。实时数据的消费者将数据发到 Elasticsearch 集群，用的 library 我们希望能尽快开

源。

商业智能

用Cranium（内部ETL平台）将S3中数据变换和归一化后导入到AWS Redshift。Cranium使我们能够用SQL和Ruby编写ETL 任务（job），它定义了这些任务之间的依赖和安排他们执行。Cranium支持数据可视化报告（利用Ruby和D3），但是大多数数据可视化是用Chartio。我们发现对于SQL经验有限的人来说Chartio非常直观。

无论是工程中还是业务中甚至是社区中人们都可以利用这些数据来解决问题，发现新的点。



用 Chartio 做数据可视化

机器学习

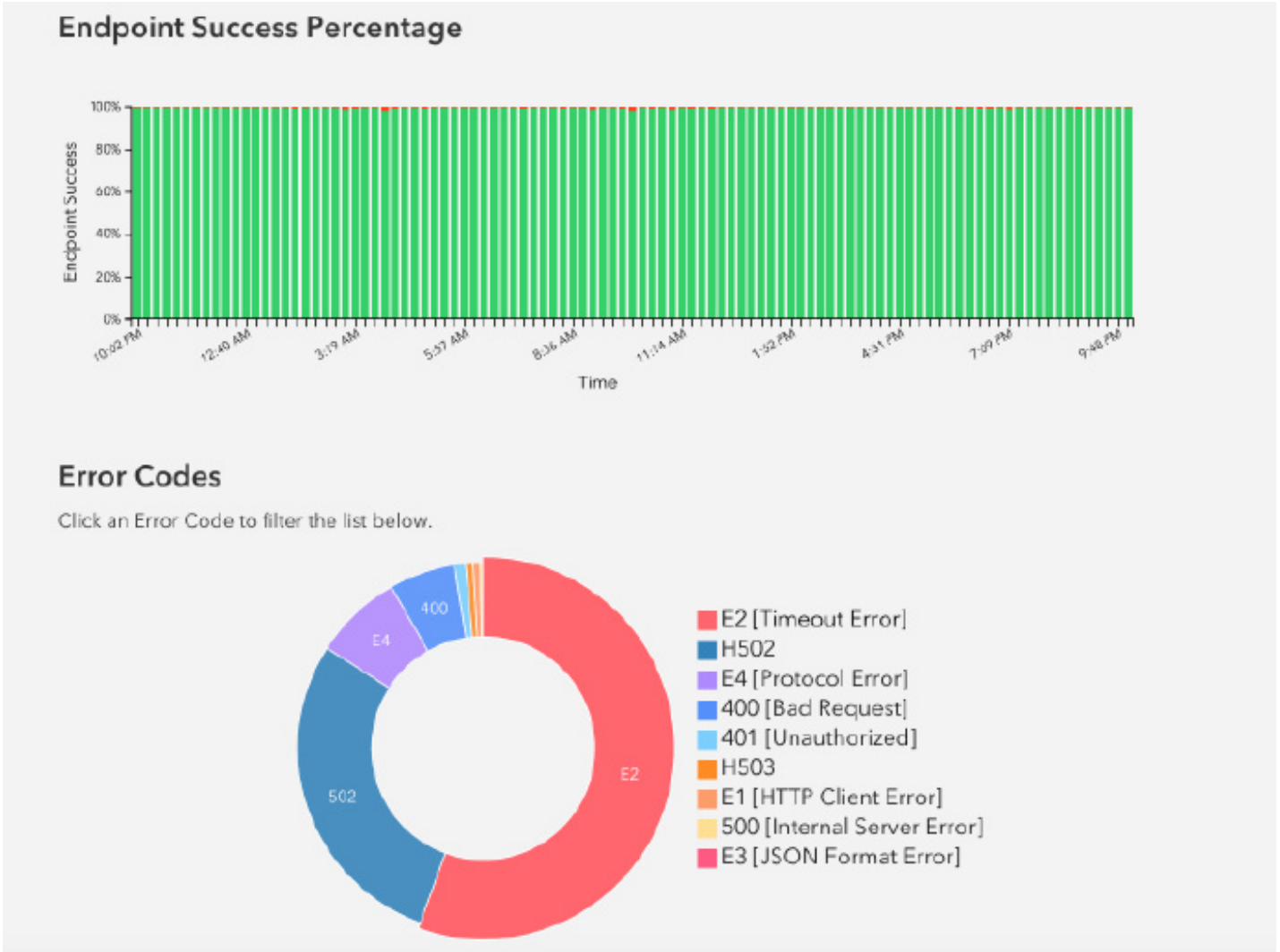
我们采用一些先进的机器学习技术来确保IFTTT用户有良好的体验。对于Recipe的推荐和滥用检测，我们用在EC2上的运行Apache Spark，用S3作数据存储。更多的内容我们将在之后的博文中解释。

实时监控和报警

API 事件存储在 Elasticsearch 中，来进行实时监控和报警。我们使用Kibana来可视化worker进程的实时性能和合作伙伴的API性能。

IFTTT的合作伙伴访问Developer Channel（当他们的API有问题时触发这个Channel）。他们可以用Developer Channel 创建Recipes，可选择不同的方式（SMS，Email，Slack，等）来通知合作伙伴们。

在开发者dashboard，合作伙伴可以访问实时日志和可视化他们的Channel健康状况。 用Elasticsearch来实现比较好。此外，提供给开发者强大的分析能力帮开发者们了解谁在使用他们的Channel以及如何使用的。



经验

我们从开发数据基础设施主中总结出一些经验：

- 通过使用像Kafka这样的数据传输层将生产者和消费者分离，使得数据管道更有弹性。比如，几个慢点消费者不会影响其他消费者或生产者的性能。
- 从一天开始就以集群的模式启动，这使得你可以很容易地扩展。但是在扩展集群前要确定性能瓶颈是什么。比如，在 Elasticsearch 中如果shard非常大，添加更多节点并不会帮助提高查询速度。你不得不减少shard。
- 在复杂系统中，要在一些地方给适当的报警，确保一切正常。我们用Sematext来监控Kafka

集群和消费者们。我们也用Pingdom来监控，用Pagerduty来报警。

- 为了能充分信任你的数据，自动验证数据是非常重要的。比如我们开发了一个服务，来比较生产中的表和Redshift中的表，如果有异常将报警。
- 使用基于日期的文件夹结构 (YYYY/MM/DD)来将 event 数据持久化存储（我们例子中的S3）。这方式存储易于处理。比如如果你想读某一天的数据，你只需要从一个目录中读取数据。
- 与上面类似的是，在Elasticsearch中创建基于时间的索引（例如：每小时）。如果在Elasticsearch中查询最后一小时所有API错误，那么利用简单的索引就可以做到，提高效率。
- 要批量提交event（基于时间段或者大量事件）到Elasticsearch，而不是提交独立的event。这有助于限制IO。
- 根据运行的数据和查询的类型，来优化 Elasticsearch 的节点数、shard数、shard 和 replication factor 的最大值等参数。