干货 | Unicorn—近实时系统的自动化修复工具

eBay技术荟 2018-12-14

供稿 | 刘鲁滨

本文4900字，预计阅读时间15分钟

更多干货请关注“eBay技术荟”公众号

导读：Rheos是eBay的近实时数据平台，管理着eBay私有云中的数千台有状态机器。过去两年中，Rheos团队一直致力于构建和改进数据平台的自动化系统。通过大量的整合工作，团队现已成功创建了一个新型的自动修复系统 – Unicorn。

本文详细介绍了Unicorn的设计与实现，并且给出了其在eBay生产环境运行一段时间后的统计结果，希望可以给运维同业人员一些启发。

01 目标

管理云中数千台有状态机器并非易事。繁重的运维任务的来源分为两类：

- 每天发生的硬件故障
- 当有状态的应用规模大到一定程度时，随之而来的相关故障

Rheos需要一个可以集中管理运维任务的系统，来帮助团队更有效地处理报警，以及通过存储报警和修复历史记录，对运维任务进行进一步分析。

1.1 集中管理运维任务

之前Rheos就有修复集群的工具。但是这些工具都是脚本，且运行在不同的地方。我们发现分散的工具修复具有以下几个局限性：

- 难以开发和维护
- 工具之间会产生冲突
- 对后加入的技术支持人员来说，较难掌握和理解
- 不是真正的自动化，耗费人力

因此，构建系统整合这些离散的作业是很有必要的。

1.2 更有效地处理报警

Rheos的监控系统已经提供了报警，但是需要人工介入处理。如果一个系统能够接收报警，并为每类问题编写自动化处理的流程，那么可以带来以下优势：

- 减少人力投入
- 在更短时间内响应和处理报警

1.3 存储报警和修复历史记录以供进一步分析

自动化最初不能解决所有问题，如果算法不够好，也可能引起新问题。将历史记录集中存储可以帮助团队改进系统。

另一方面，通过分析报警历史记录，团队可以尝试找到新的自动化方向。

02 挑战

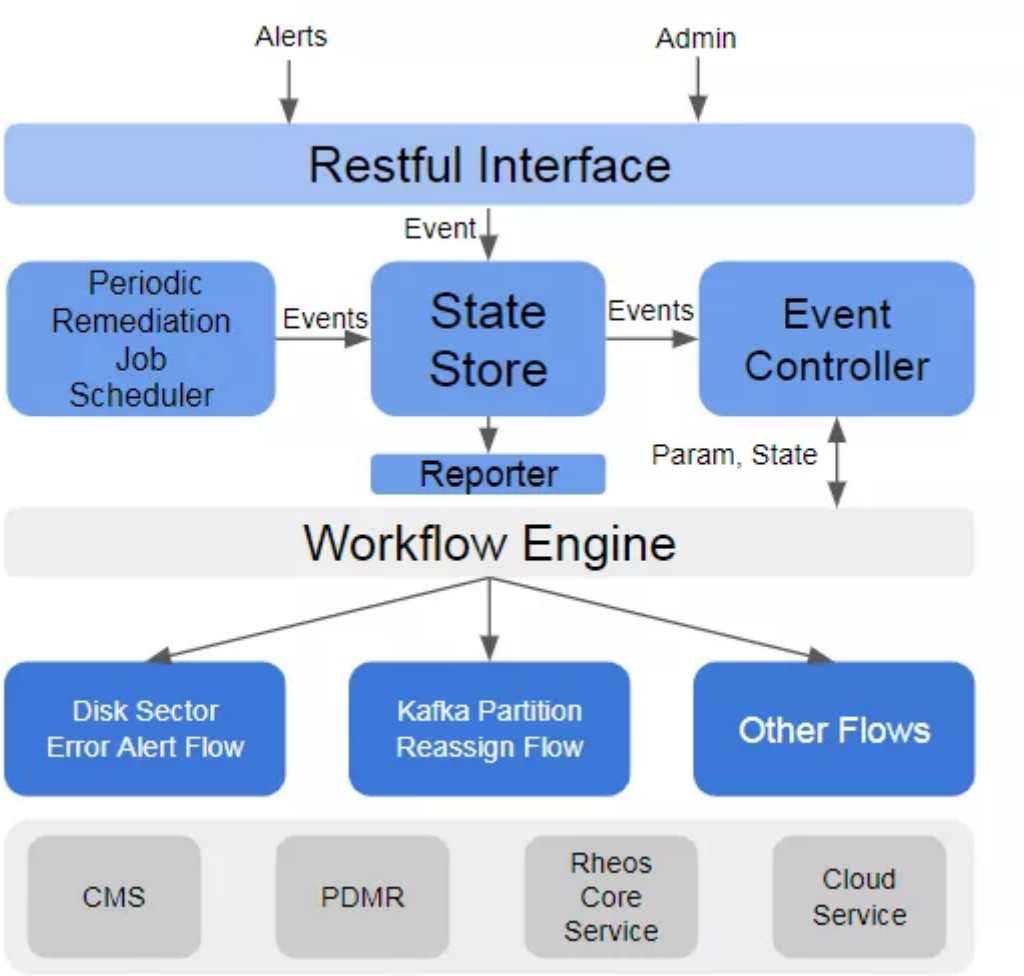
创建这样的一个集中修复系统有以下两个挑战：

- 需要创建一个易于拓展的通用模型，以覆盖所有已知运维经验。创建特定工具来解决特定问题是个很简单的实现方式。但是，找出适用于所有已知经验的通用模式和模型却困难的多。为了避免未来仍需创建另一组工具，我们设想的模型也需要具备可扩展性
- 定义准确的自动化策略。过于激进的自动化策略可能会带来危险的后果：

1. 1) 因进行一些不被允许的操作而损坏系统
2. 2) 因操作过快而造成的状态不同步: 在有状态集群的应用中，系统需要时间来同步状态
3.

过于保守的自动化策略可能会使很多紧急问题得不到及时处理，影响系统的效果。

03 设计



以上图示对Unicorn的整体设计进行了说明。

Unicorn 通过定义 “事件”这种关键资源，对所有需要解决的故障抽象化。事件有以下三种来源:

- 上游发出的报警：这种是最为常见的来源
- 管理员手动提交的事件：紧急情况下手动操作的入口
- 定期修复的工作：某些常规检查任务

事件控制器根据自动化策略，定期选取事件，并启动相应的工作流进行处理。

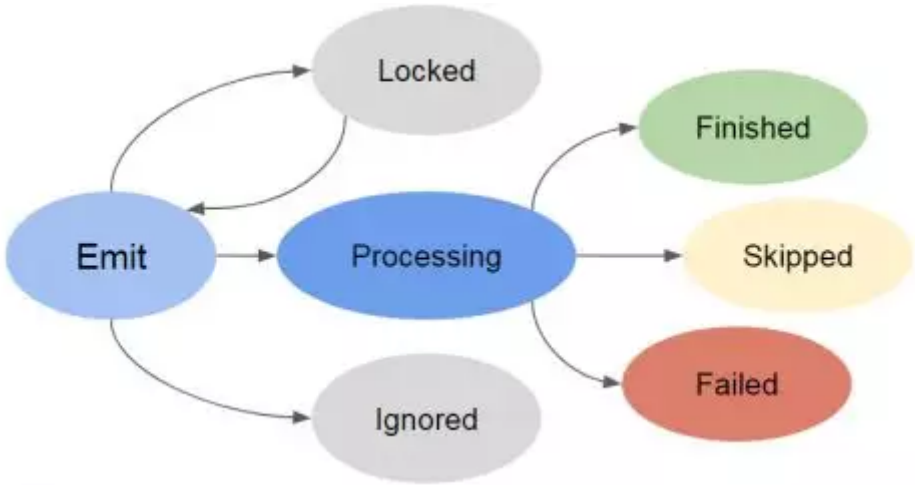
工作流引擎管理工作流的生命周期，同时会把事件状态发送回控制器。

每一个工作流都可能调用外部相关的服务系统，如配置管理系统和底层的造建，部署，监控，修复（PDMR）系统来进行修复。

报告模块会把总结或详细的报告直接发送给目标订阅者。

3.1 事件

名称	类型	描述
ID	Long	每个事件全局唯一的标识
TYPE	String	需要处理的事件类型
GROUP ID	String	需要依序处理的同组事件
STATUS	Enum	该事件在生命周期中的状态
LABELS	Map	描述问题需要的信息
PAYLOAD	String	处理过程中的信息输出
PRIORITY	int	事件的紧急程度；值越大代表越紧急
FLOW_ID	Long	处理此事件的工作流标识
TIMESTAMP	Long	事件生效的事件戳。对延迟事件来说，未来某个时间是时间戳。
TIME_TO_LIVE_MS	Long	事件生效后的存活时间
OWNER	String	这一事件的来源
RETRY_COUNT	int	此事件已尝试处理的次数
PROCESS_TIMESTAMP	Long	工作流开始处理事件的时间戳
REFERENCE_ID	String	用户用来定义的查询标识
LOG	String	处理日志



事件状态定义了事件的生命周期：

- 发出（Emit）：表示事件首次存储在状态存储区中
- 处理（Processing）：表示事件正在被工作流引擎处理
- 锁定(Locked)：表示同组的某个事件处于“处理”状态，当前事件暂时不会被选中
- 完成(Finished)：表示事件被成功处理
- 跳过(Skipped)：表示事件代表的故障已修复
- 失败(Failed)：表示事件处理失败，需要尽快采取相应措施
- 忽略(Ignored)：表示事件不在当前的考虑范围

3.2 自动化策略

3.2.1 组标识

Unicorn引进了“组标识”这个概念，对相关的事件进行隔离。“组标识”语义如下：

- 同组事件必须依次处理
- 不同组的事件可以同时处理

尽管这个域是开放给用户自己定义的，但是最常见的情况还是使用物理集群标识。

根据以往的操作经验，团队认为避免在一个有状态的集群处理多个结点，是比较安全的操作方法。

基于组标识的这个域，事件控制器每轮选择事件的流程如下：

- 列出所有组标识

- 迭代所有组标识，核查组群中是否存在处理中的事件
- 如果有事件在处理，跳过当前组标识。如果没有，则需选择一个事件，开始处理
- 等待下一轮

3.2.2 优先级

接下来回答的问题是如何判别哪个组群/事件应该优先处理。“优先级”语义如下：

- 拥有较高优先级事件的组标识总是会被首先扫描到
- 同组中有较高优先级的事件总是会被首先挑选出来

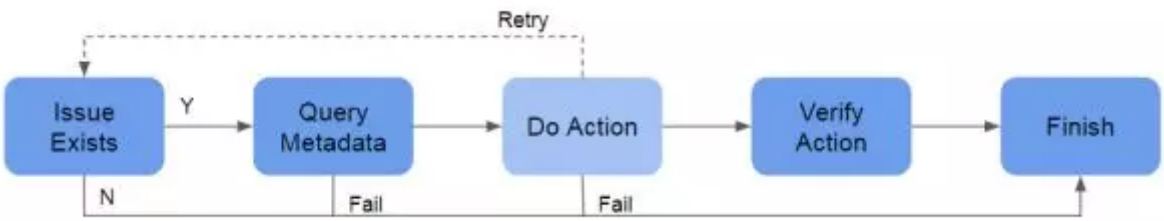
优先级和事件类型有关，也就是说，同类型的事件拥有同样的优先处理权。每次添加新的事件类型，用户需要自定义优先级。

3.2.3 速率控制

Unicorn定义了以下与速率控制相关的概念：

- 最大处理单元：处于“处理”状态的最多并发事件值。一旦处理事件值达到或超过处理器设置的上限，即使仍有群组等待扫描，扫描进程也会终止
- 窗口事件阈值：某个滑动窗口内，可处理特定类型事件的最大个数
- VIP优先级阈值：优先级超过相应阈值的组群可以打破“最大处理单元”的局限，也就是说, 即使并发的处理事件已达到最大处理器上限，具有较高优先级的事件仍可以正常处理

3.3 工作流



不同类型事件，具体工作流会有所差异。但是，Unicorn中大部分的工作流都会遵循以上的通用模式：

- 问题核查阶段：检查问题是否仍存在。该阶段能够帮忙我们避免做重复性工作

- 元数据查询阶段：基于输入信息，查询具体的元数据。大多数情况下，信息输入仅仅包含索引信息，如集群标识，节点标识。详细信息，需要在工作流中查询
- 操作阶段：针对不同的事件类型，该阶段会有不同的实现。如果事件需要重试，则直接返回到“问题核查阶段”
- 验证操作阶段：该阶段的操作也可能因事件类型而异。对工作流中已完成的操作进行验证，可以更好的保证操作的正确性。
- 完成阶段：更新并同步状态到事件中

04 设计

NAP（NuData 自动化平台）

NAP 是NuData团队创建的框架。NAP汇聚了Nudata自动化的最佳实践，作为一个轻量级的自动化基础框架，它以模块化的方式提供了工作流引擎、REST、描述式UI构建、鉴权、审计等基础功能。NAP帮助用户专注于实际业务，通过按需组装模块，快速搭建出一个生产级的应用。

Unicorn的运行高度依赖NAP的功能。

4.1 Unicorn中的工作流

Unicorn的工作流基于NAP工作引擎。Unicorn通过配置文件来执行任务，创建工作流。

工作流中，每个阶段都有可能失败，这样会导致工作流图示中出现分支。Unicorn中所有的任务都会产生一个称作“流状态”的输出变量。“流状态”所描述的是当前任务的结果。Unicorn利用NAP工作流引擎中“SwitchBy”的语义，来决定下一项任务。

除了基本的工作流，Unicorn还有很多定时的工作流。NAP的调度器要比Java的ScheduleExecutorService强大的多，甚至能够支持crontab命令。

4.2 Unicorn的RESTful服务

Unicorn提供的RESTful服务全部都基于NAP的RESTful框架。Unicorn的服务可以分为以下两大类。

4.2.1管理资源的生命周期

Unicorn中存在两种主要资源：事件和报警。其中事件是Unicorn的核心资源，Unicorn管理着事件的创建，查询，更新和删除。

报警通过Webhook的方式插入到Unicorn。Unicorn中有一个处理器，可以将接收的报警转换为标准事件。同时，处理器还负责将这部分事件的组标识，定义为物理集群标识。

作为自动化系统，Unicorn暴露一个接口用于查询最新处理过的集群。这种功能不仅有助于生成报告，同时也能为做技术支持的人提供帮助。

4.2.2 管理员使用的服务

以下是NAP两个非常有用的原生服务，这组服务可以帮助管理员进行调试：

- 输出线程状态：返回当前服务的线程状态信息，可用于死锁分析和线程问题调试
- 显示日志：日志文本内容流，可用于具体调试

4.3 Unicorn门户

Unicorn门户完全基于NAP 配置驱动的 UI框架。Unicorn门户提供了资源查询和状态更新的功能。

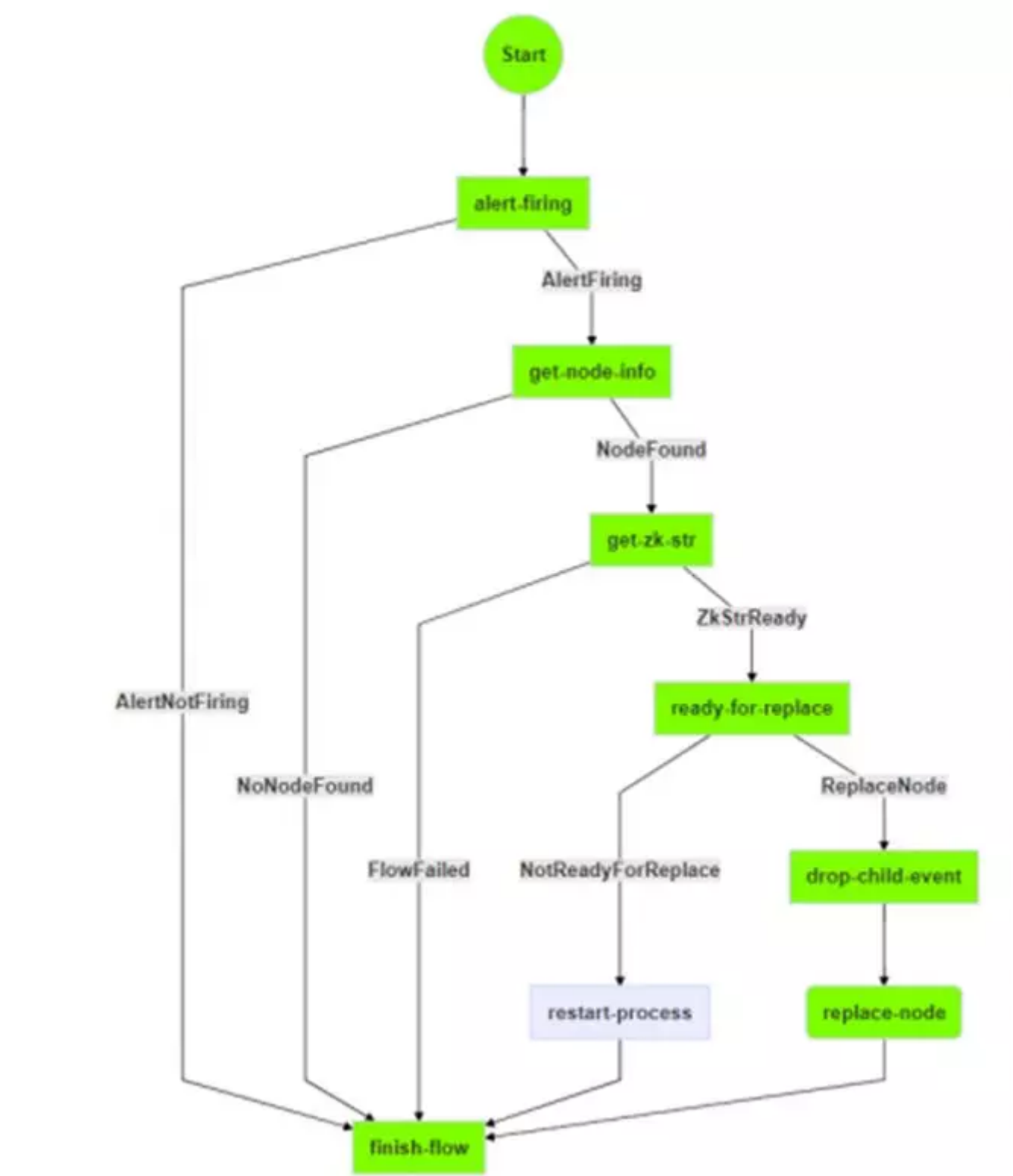
为了呈现系统的整体状态，Unicorn的主页面会提供一些统计资料。比如：

- 当前支持的所有事件类型
- 近期处理过的集群数量
- 近期处理成功的事件数量
- 近期处理失败的事件数量

4.4 工作流举例

4.4.1处理磁盘坏道的工作流

这是一种用来处理底层硬件故障的流程。



Kafka能否正常工作，很大程度上取决于磁盘的稳定性。如果节点的机械磁盘出现大量的磁盘坏道，会影响到Kafka的运行并导致以下两个主要问题：

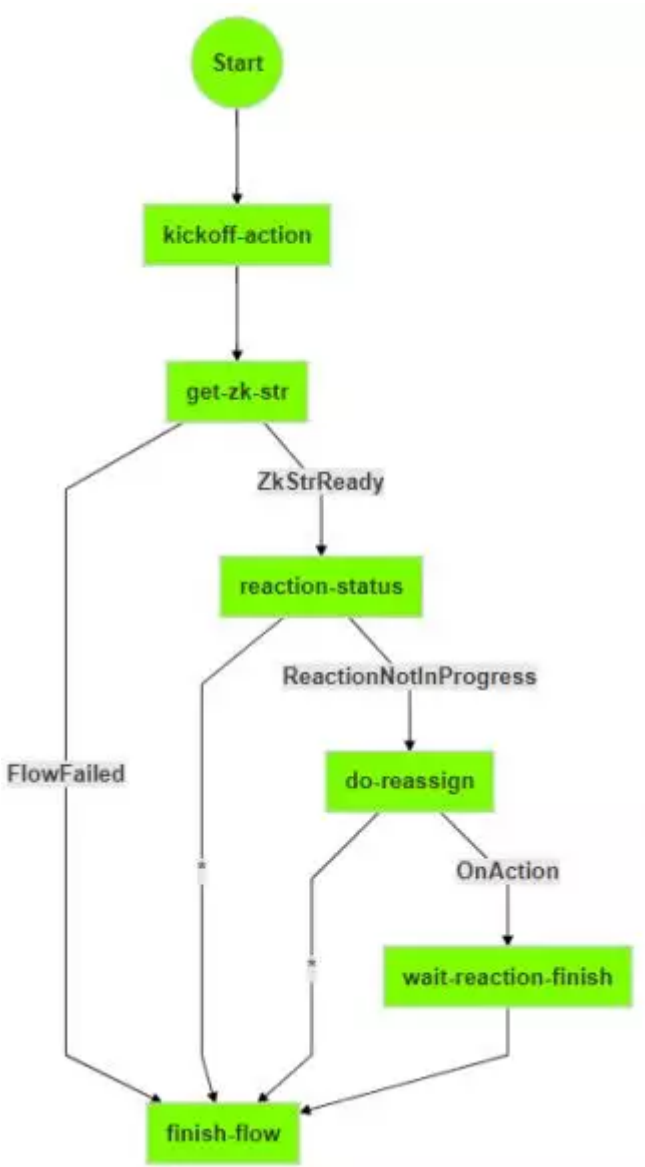
- 吞吐量急剧下降
- 端对端延迟急剧上升

一旦接收到磁盘坏道报警，Unicorn会进入上图所示的工作流程。以下两个步骤需着重强调：

- 警告检查阶段：回查监控系统，确认此问题是否仍然存在
- 准备替换阶段：移除旧节点，使用一个具有相同标识的新节点是解决此类问题的最佳方案。但是，如果同步副本（ISR）数量过少，直接替换节点可能会带来危害。在这种情况下，Unicorn选择重启Kafka进程，暂时减缓因磁盘坏道带来的影响.

4.4.2 Kafka分区再分配流程

该流程解决了应用程序特定设计引起的问题。



每个Kafka节点都有一些主题分区。当一个集群中的通信量急剧增加时，Rheos将增加该集群中的虚拟机数。然而，Kafka不会自动将现有的主题-分区移动到新的节点。该工作流将分区从繁忙节点搬运到空闲节点，保证流量的平衡，有助于集群的稳定，可以获得更好的吞吐量。

分区再分配事件是周期性产生的。Unicorn用贪婪算法来解决节点分区数目不平衡问题，此事件工作流的主要流程如下：

- 选择一个分区最多的节点
- 选择一个分区最少的节点
- 生成分区重新分配计划，并使用Kafka提供的分区再分配工具启动该过程
- 等待并检查，直到重新分配完成

4.5 报告系统

了解自动化系统完成的操作是很重要的，因此Unicorn会定期发出一份操作报告。

Unicorn通过NAP提供的定时工作流来发送报告。目前发送的大多数报告都与事件统计信息有关。开发人员和经理所接收的报告可能需要不同的粒度和不同的维度，Unicorn可通过以下配置使报告模块具备扩展性：

- 处理程序：实现报告逻辑的类
- 定时模式：以crontab样式发送报告的模式
- 报告窗口：此报告所包含的时间窗口
- 报告的事件类型：报表将包括的事件类型列表
- 接收者：接收此报告的接收者列表
- 类型：此报告的发送协议；目前，只支持电子邮件

05 在生产环境中运行

Unicorn已经在生产环境中运行了几个月。本节介绍的是基于生产数据的统计信息。

5.1 事件类型

事件类型	百分比	描述
mediaerrordisk	0.06%	修复高频率磁盘坏道区的节点
RollingRestart	0.02%	依次重启节点并确保同步状态

nodedown	66.89%	通过重启或更换节点把宕机的虚拟机带回来
Replace	0.37%	删除旧节点并建立具有相同ID和配置的新节点
behaviormmlaghighfor5min	20%	修复网络抖动造成的mirrormaker严重滞后
partitionreassign	4.79%	在一个集群中重新放置分区
leaderreelection	4.78%	为某些特定主题分区重选合适的领导者
readonlydisk	0.31%	磁盘处于只读状态时，修复节点
highdiskusagefor5min	0.09%	存储量过低时，清理磁盘
straasagentheal	1.76%	自动修复PRMR系统代理

根据上表可得出以下结论：

- 最多的警报来源是虚拟机宕机和网络抖动。
- 特定于应用程序的工作流在Unicorn中发挥着重要作用
- 磁盘故障问题很多

5.2 事件处理结果

每种事件类型最终状态的分布如下：

事件类型	已完成	失败	跳过
mediaerrordisk	50%.	9%	41%

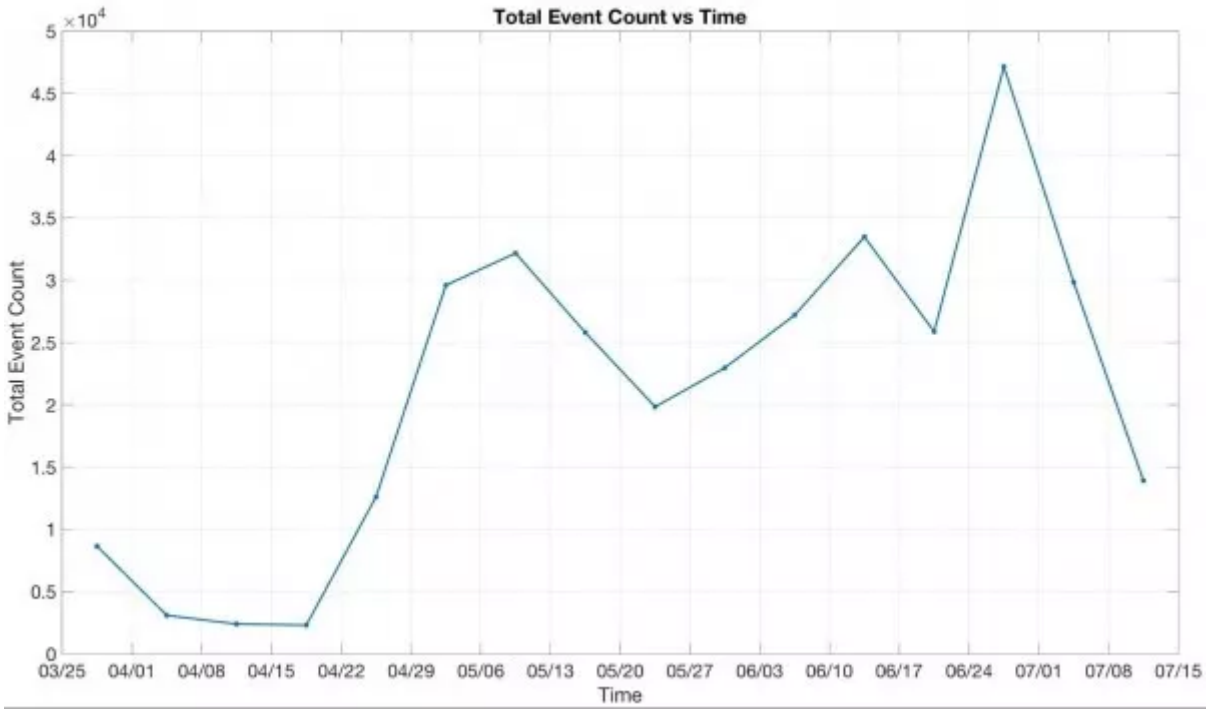
RollingRestart	36%	1%	63%.
nodedown	2%	3%	95%
Replace	20	79%	1%
behaviormmlaghighfor5min	0.04%	0.06%	99.90%
partitionreassign	11%	1%	78%
leaderreelection	70%	1%	29%
readonlydisk	81%	6%	13%
highdiskusagefor5min	18%	3%	79%
straasagentheal	76%	8%	16%
总计	6%	2%	92%

根据上表可得出以下观点：

- 大多数事件都被标记为“跳过”。当问题解决一次后，队列中的其他相关事件将不需要采取任何操作。处理误报警是Unicorn的一大优势
- 与磁盘有关的故障可以得到有效修复
- 替换节点工作流的失败率最高。作为很多流程的子流程，替换节点工作流需要处理底层的系统故障

5.3 事件计数

投入运行后，已处理事件的简单计数如下图所示：



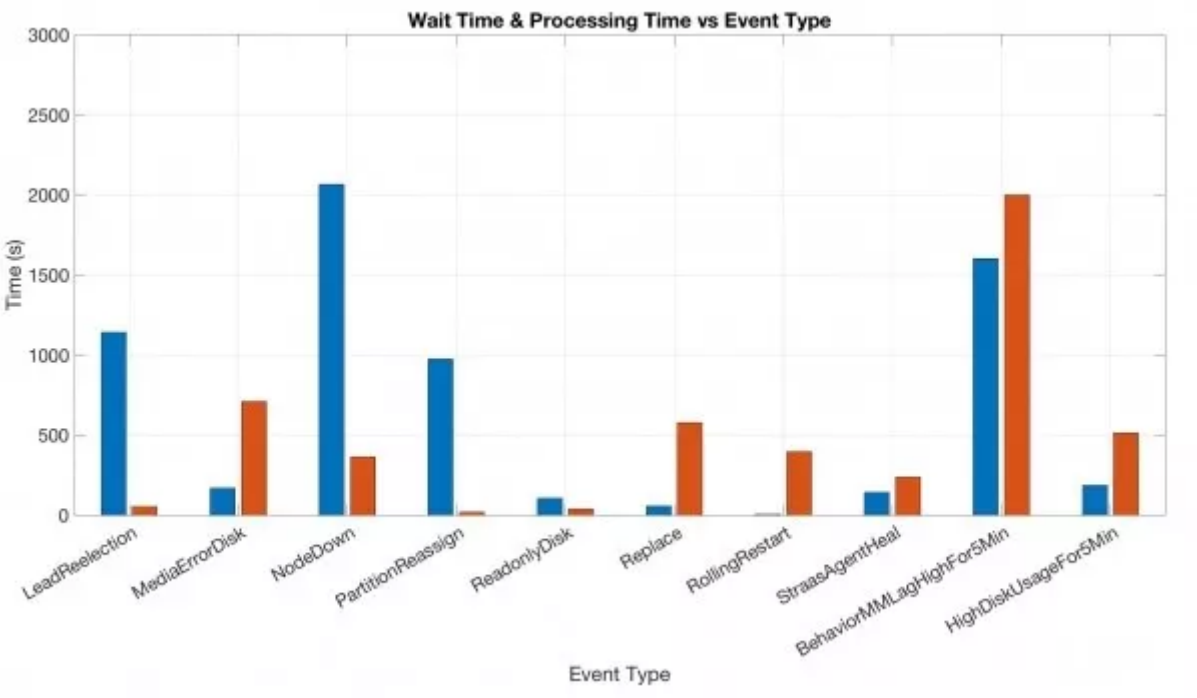
从这个图表中可以观察到以下几点：

- Unicorn每周可以扫描数万个事件
- 在启用新功能之后，已处理的事件计数明显增加
- 某些问题解决后，需要处理的事件减少了

5.4 事件时间戳

显示Unicorn效率的两个概念如下：

- 响应时间(图表上的蓝色)：处理时间戳-创建时间戳，从接收事件到开始处理之间的时延
- 处理时间(图表上的红色)：上次修改的时间戳-处理时间戳，Unicorn将事件标记为“已完成”或“失败”的时间，即事件处理所消耗的时间



从这张图表中观察到：

- 具有控制(Nodedown和BehaviorLagHighFor5Min)和低优先级(LeaderReelectio-nandPartitionReAsding)的事件需要更长的响应时间
- 紧急事件，包括与磁盘相关的事件和手动提交的事件，响应时间很短：不到5分钟
- 大部分事件的处理时间在10分钟左右
- BehaviorMMLagHighFor5Min需要一些采样时间，所以需要改进算法

06 总结

Unicorn减少了人力投入，为解决紧急问题提供了更好的SLA，并保证了Rheos中有状态集群的稳定性。

从技术角度出发，Unicorn的贡献可总结如下：提供事件驱动的解决方案来建模过去Rheos的运维经验，模型易拓展；定义组标识和优先级的概念，以隔离具有潜在冲突的操作，并确保了高效率；如果一个工具同样需要管理C3上的集群，可以参考基于NAP的最佳实践。

Unicorn仍需在日常工作中不断加强提升，未来团队可以考虑增加智能化模块，以实现自动处理新问题的目标。

主题：云计算，数据基础设施和服务

作者简介

刘鲁滨，上海交大硕士毕业, 2015年加入eBay, 担任软件开发工程师，专注研究云计算，消息中间件和流式计算等领域。