|  |  |
| --- | --- |
| 交底书名称 | 一种基于kubernetes的容器驱逐策略的优化方法 |
| 技术联系人姓名 | 马殿军 |
| 技术联系人电话 | 13269325833 |
| 技术联系人Email | madianjun@jd.com |

（技术联系人信息用于与外部代理沟通，发明人信息在ERP专利申请系统中填写）

注意事项：

1、代理人并不是技术专家，交底书要使代理人能看懂，尤其是完整技术方案，一定要写得全面、清楚。

2、在后续与专利代理人进行沟通时，对于代理人的疑问应认真讲解，要求补充的材料应及时补充（禁止通过私人邮箱与代理人沟通）。

3、常用检索网站：www.soopat.com（SOOPAT），http://so.baiten.cn/（佰腾），patents.google.com（谷歌专利）。

# 1. 现有技术

/\* 应记载某个应用场景或者解决某个技术问题当前所采用的技术，可以概述该技术，也可以仅给出参考文献的链接或相关专利号。

Kubernetes是一种容器集群编排和管理的分布式系统，它将容器调度并运行在集群的多个节点上。当容器由于某些原因（物理机问题或容器内部程序问题）导致无法正常运行时，kubernetes会反复重启容器，直到容器可以正常运行为止。

# 2. 现有技术的缺点

/\* 需要指出现有技术存在的缺点，本发明也不能克服的缺点无需提供。

/\* 应根据现有技术的实现过程，有针对性地说明缺点产生的原因。

当容器无法成功运行时，kubernetes会在同一个节点上反复重启容器。如果由于物理机自身问题导致容器无法运行，那么在同一个节点上反复重启容器是没有作用的，因为容器运行所依赖的外部环境并没有改变。现有的技术是针对反复重启的容器作驱逐，从而使容器被重新调度到其他节点。但是该方法无法判断容器运行失败的原因是外部环境问题导致，还是应用程序自身问题导致；如果原因是后者，那么即使重新调度容器，也还是无法运行。这时就需要一种比较准确的检测方法来检查容器无法运行的原因。

# 3. 本发明技术方案

3.1 本发明所要解决的技术问题（即发明目的）

/\* 描述本发明所要解决的技术问题，与“2.现有技术的缺点”部分指出的缺点相对应。

本发明主要解决的是检测容器无法运行的原因，区分出是外部环境问题还是应用程序自身问题。

3.2 本发明的完整技术方案的详细阐述

/\* 这是本文档最重要的部分，需要详细完整的阐述，不能光有原理，也不能仅有功能性介绍或操作说明。

/\* 在描述具体的技术方案时，必须结合附图（方法型专利按照数据流向或实现步骤抽象框图，装置型专利按照组成部件抽象框图）进行说明，每个附图都应当有对应的文字描述。如果本发明方案包含多个主题，方法与装置等，则需要分别进行描述。

1. 架构

本发明在基于原有的驱逐策略的基础上，增加了基于容器状态的检测方法。系统架构描述如下： descheduler定期从apiserver上获取所有pod的信息，并过滤出已经被调度到node上但没有正常运行的pod，并将其驱逐，进而触发scheduler的对新pod的重新调度，新创建的pod将以一定概率被调度其他node上，因此可实现使pod不在同一个节点上反复重启。架构图如下：

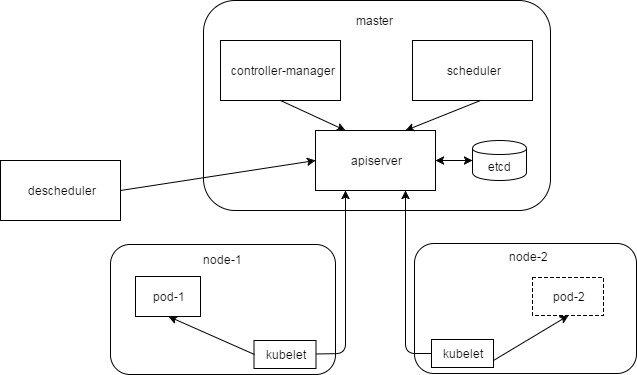


图1 descheduler与kubernetes交互架构图

详细流程描述如下：

1. pod-1在node-1上启动，但并没有成功运行，kubelet将其状态上报至apiserver。
2. descheduler定期从apiserver上获取pod列表，并检查pod状态。当发现pod-1处于“已经被调度但未成功运行”的状态时，向apiserver发出evict驱逐请求。
3. node-1上的kubelet接收到evict请求，将pod-1从node-1上删除，并将状态同步到apiserver。
4. controller-manager监听apiserver时，发现pod-1被删除，就创建一个pod-1的替代副本，称为pod-2，并将pod-2同步到apiserver。
5. scheduler监听apiserver时，发现pod-2虽然被创建了，但是没有被调度，则执行调度算法选择一个合适的node（该node可能是node-1或node-2）与pod-2绑定。
6. 如果pod-2与node-2绑定，则node-2上的kubelet从apiserver上监听到此状态变更，就在node-2上创建pod-2。

以上过程即实现了将pod-1从node-1驱逐并重新调度的过程。需要注意的是：descheduler无法区分pod-1未成功运行的原因是由于node-1的物理机环境问题导致，还是由于pod-1自身的程序问题导致。descheduler是在不严格区分这两种原因的情况下触发的pod的驱逐。本发明即是通过检测容器状态，区分出容器无法运行的真正原因。

1. 检测pod状态的方法

pod中的容器包含sandbox容器，init容器，regular容器。其中sandbox容器先创建，它为init容器和regular容器构建了基础运行环境；然后init容器运行，它负责初始化应用程序需要的配置信息和数据等；regular容器最后运行，它负责执行应用程序的业务逻辑。

pod中的容器的运行流程包括以下步骤：

1. sandbox容器运行
2. init容器拉镜像
3. init容器创建
4. init容器启动
5. init容器运行
6. init容器成功退出
7. regular容器拉镜像
8. regular容器创建
9. regular容器启动
10. regular容器运行

descheduler在判断pod是否应当被驱逐时，需要检测pod的状态和容器的状态。pod的状态字段包括以下三个（值为true或false）：

1. PodScheduled：如果pod被调度成功，则置为true；否则置为false。
2. Initialized：当init容器运行成功并退出后，regular容器再启动。如果init容器成功退出，则initialized被置为true。
3. Ready：如果regular容器运行成功，则被置为true；如果运行失败，则被置为false。

容器的状态包括：

1. Waiting：容器运行之前的状态，包括拉镜像、创建、启动。
2. Running：容器运行起来。
3. Terminated：容器成功退出。

对于容器状态，descheduler只关心Waiting状态，因为该状态才表示容器未运行起来，而Waiting又细分为以下几种原因：

1. PodInitializing：对于init容器来说，该原因表示sandbox正在创建中；对于regular容器来说，该原因表示init容器未成功退出。
2. ImagePullBackOff：表示拉镜像失败。
3. ContainerCreating：表示容器正在创建。
4. RunContainerError：表示容器执行启动命令时失败。
5. CrashLoopBackOff：表示容器启动成功，但是运行失败。

在下表中，总结了以上运行流程，以及pod状态和容器状态的变化。

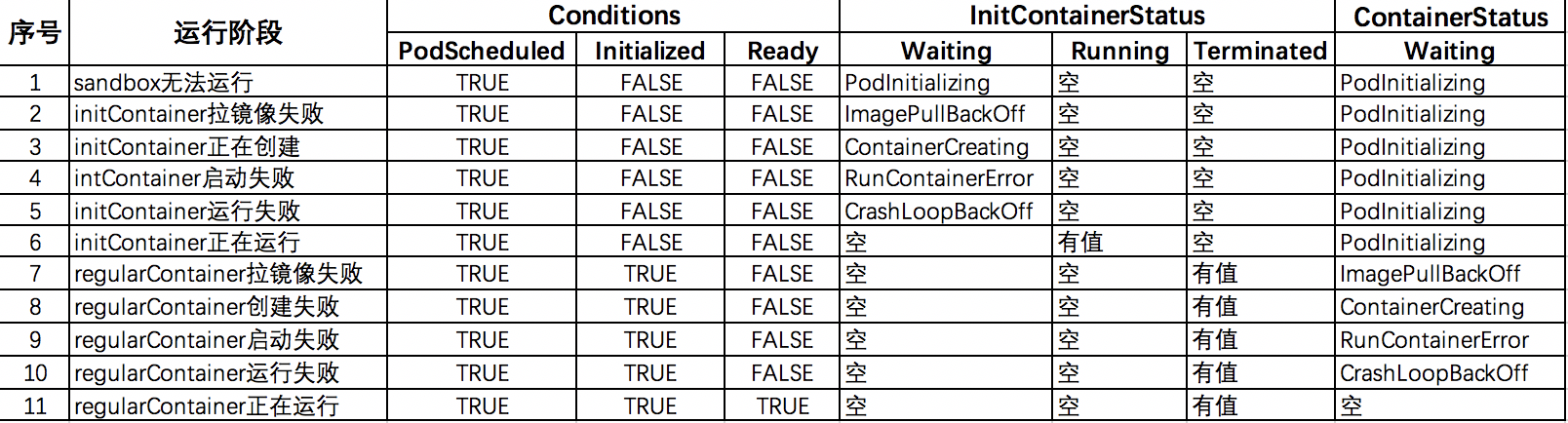


表1 pod运行流程及状态变化

如果pod长时间处于表中的第1、3、8阶段，则表示容器无法运行的原因是外部环境问题导致，这三个阶段分别为：

1. sandbox无法运行：PodScheduled=true, Initialized=false, init容器的Waiting是PodInitializing
2. init容器正在创建：PodScheduled=true, Initialized=false, init容器的Waiting是ContainerCreating
3. regular容器正在创建：PodScheduled=true, Initialized=true, regular容器的Waiting是ContainerCreating

descheduler只需要判断以上三种情况的状态字段的取值：当pod处于这三种状态之一，并且持续时间超过一定的阈值时，即可认为pod无法运行的原因是由外部环境问题导致。

3.3 本发明希望保护的技术创新点

/\* 指出技术方案中希望保护的技术关键点，并概括说明该关键点的技术原理。

本发明的技术创新点是根据容器状态字段的取值检测出由于外部环境问题导致无法成功运行的pod，优化了descheduler的驱逐策略，提高了驱逐的准确性。

3.4 针对3.3中的技术方案，是否还有别的替代方案同样能完成发明目的？

/\* 替代方案可以是完整技术方案的替代，也可以是部分结构或者步骤的替代。

3.5交底书中技术术语的名词解释

/\* 记载交底书中出现的专业技术术语、缩写、外文的解释。

node：集群中的节点，对应一台物理机。

kubernetes：由google开源的容器集群编排和管理系统。

master：kubernetes的控制节点，包含apiserver, controller-manager, scheduler三个组件。

etcd：一种分布kv数据库，为kubernetes提供数据持久化服务。

apiserver：为kubernetes其他组件提供api访问，它直接读写etcd。

controller-manager：维护pod副本数量，使之与用户期望的数量一致。

scheduler：调度pod，使pod与合适的node绑定。

kubelet：管理和监控本节点上的pod。

pod：是容器的一种抽象表示，它可以包含一个或多个容器实例。Kubernetes以pod作为最小的管理和调度单位。

evict：驱逐，即将pod从node上删除。