103Kubernetes 系列（九十六）尝试新的就地 Pod 资源大小调整！

我多年来一直在等待就地 pod 资源调整大小的能力（自 2019 年以来！），我真的很高兴看到它终于作为 Kubernetes 1.27 中的 alpha 功能发布。这使您能够自动调整 Pod 的 CPU 和内存限制以及请求的大小，而无需重新启动它。我花了很多时间研究实时软件更新，并意识到任何“实时”的事情都是多么具有挑战性。我感谢贡献者为实现就地调整大小所做的所有工作，因为它需要跨多个 Kubernetes 组件进行大量更改。 （我没有参与这项工作，但我很高兴使用它！）

非常简化的版本是，此功能允许 Kubernetes 就地更新底层 c-group 分配，从而使 pod 规范资源可变。这在垂直扩展 pod 的情况下特别有用，例如使用 Kubernetes 内置的 Vertical Pod Autoscaler (VPA)，它允许应用程序在同一 pod 内向上/向下扩展资源（而不是通过更多 pod 进/出进行扩展）与传统的水平 Pod 缩放一样）。

在许多用例中，垂直扩展很有帮助，例如有状态数据库工作负载，其中流量可能会突然增加，但服务中断的成本高昂。在另一个示例中，KubeCon North America 2022 上有一场激动人心的演讲，展示了如何将这一就地功能与 eBPF 结合使用。

这里是一个简单的示例，介绍了如何尝试它，以及您将在 pod 规范中看到的新更改。在这里，我将详细介绍更新过程中的情况，并展示该过程中的所有步骤。

我通常使用 Kubernetes 的公有云版本，但由于这些托管版本中尚未提供 1.27 版（截至 2023 年 4 月），我们将使用 minikube 在本地启动一个版本。有很多方法可以做到这一点；这只是一个简单的例子。

这是在功能标志 `InPlacePodVerticalScaling` 下发布的。使用该标志启动一个 minikube 集群，并添加一个节点：

```
minikube start --kubernetes-version=v1.27.0 --feature-gates=InPlacePodVerticalScaling=true --cni calico
minikube node add
```

准备就绪后，启动测试容器。举例来说，对于我们的应用程序，无需重新启动即可安全地更改 CPU 数量，但更改内存数量则需要重新启动。例如，运行数据库的 pod 在运行时 CPU 计数变化不会出现问题，但减少内存量会导致意外行为。我们通过将“内存”的 restartPolicy 设置为 RestartContainer 来在 pod yaml 中显示这一点。否则，默认行为将尝试就地更新资源。

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: inplacedemo
spec:
  containers:
  - name: inplacedemo
    image: alpine
    command: ["tail", "-f", "/dev/null"]
    resizePolicy:
    - resourceName: "memory"
      restartPolicy: "RestartContainer"
    resources:
      limits:
        cpu: 2
        memory: "1Gi"
      requests:
        cpu: 1
        memory: "500Mi"
```

应用 yaml 并确保它正在运行，然后在准备就绪时通过获取 pod 信息来查看新字段：

```
$ kubectl apply -f yaml_above.yaml
pod/inplacedemo created
$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
inplacedemo   1/1     Running   0          53s
$ kubectl get pod inplacedemo -oyaml
```

在 pod 规范下首先要注意的是 `resizePolicy`：

```yaml
spec:
  containers:
    ...
    resizePolicy:
    - resourceName: memory
      restartPolicy: RestartContainer
    - resourceName: cpu
      restartPolicy: NotRequired
```

您还会注意到“状态”字段下有一些新字段。 `allocatedResources` 是分配给 Pod 容器的资源，还有一个附加的 `resources` 字段。起初我对为什么重复的资源字段感到困惑，但后来我意识到这反映了实际的当前资源，而不是所需/待更新的资源（接下来会详细介绍）。请注意，这全部位于 containerStatuses 下：

```
containerStatuses:
 - allocatedResources:
     cpu: "1"
     memory: 500Mi
   ...
   resources:
     limits:
       cpu: "2"
       memory: 1Gi
     requests:
       cpu: "1"
       memory: 500Mi
```

让我们首先修改 pod CPU 限制，将其从 2 增加到 3。我们将通过修补在命令行上执行此操作：

```
kubectl patch pod inplacedemo --patch '{"spec":{"containers":[{"name":"inplacedemo", "resources":{"limits":{"cpu":"3"}}}]
```

如果您现在查看 pod (`kubectl get pod inplacedemo -oyaml`)，您可能（但不一定）会看到 resize 字段出现，并且您还会看到带有以下内容的 pod 规格资源新值和旧值的 pod 状态资源：

```
spec:
  ...
    resources:
      limits:
        cpu: "3"
        memory: 1Gi
      requests:
        cpu: "1"
        memory: 500Mi

status:
  ...
  containerStatuses:
    ...
    resources:
      limits:
        cpu: "2"
        memory: 1Gi
      requests:
        cpu: "1"
        memory: 500Mi
    restartCount: 0
  ...
  resize: InProgress
```

您的进度可能会有所不同，具体取决于 Pod 从 resize: InProgress 到完成（标志消失）所需的时间。如果您看到其他标志，例如 resize: Infeasible ，请检查您的节点资源以确保它们足够。

继续讨论内存，让我们将限制从 1G 增加到 2G：

```
kubectl patch pod inplacedemo --patch '{"spec":{"containers":[{"name":"inplacedemo", "resources":{"limits":{"memory":"2Gi
```

现在，调整大小标签和 Pod 状态的过程与之前相同。通过检查该字段或资源状态来确保 resize 已完成。（在我的设置中，这大约需要 15 秒到 1 分钟。请注意，现在有一个错误，可能需要更长的时间。）之后，验证重新启动是否基于我们的参数：

```
$ kubectl get pods
NAME          READY   STATUS    RESTARTS      AGE
inplacedemo   1/1     Running   1 (88s ago)   25m
```

就是这样！我真的很期待在所有需要垂直 Pod 自动缩放的有状态应用程序中尝试这一点，并且我期待该功能集成到 VPA 中！