

1. 一 调度时的命令

1.1. 1 普通调度

1. 命令格式

在原生的K8S集群下，不修改任何的配置文件，直接运行容器请求命令

```
kubectl apply -f memory-demo.yaml
```

memory-demo.yaml文件格式内容

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo6
  namespace: mem-example
spec:
  containers:
  - name: memory-demo-ctr6
    image: polinux/stress
    resources:
      requests:
        memory: "100Mi"
      limits:
        memory: "200Mi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

2. 命令解释

新建一个deploy，这个容器请求100M内存。

3. 命令示例

原生调度策略，会按照节点打分，将容器请求分配到一个物理机上

1.2. 2 任务调度

1. 命令格式

运行调度算法：extenders

```
go run extenders
```

创建yaml文件，描述一下刚刚调度算法运行的地址

```
scheduler-extender-new.yaml 包含：
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
extenders:
- urlPrefix: "http://127.0.0.1:8080/"
  filterVerb: filter
```

```
prioritizeVerb: prioritize
weight: 1
enableHTTPS: false
nodeCacheCapable: false
ignorable: true
httpTimeout: 15s
leaderElection:
  leaderElect: true
clientConnection:
  kubeconfig: /etc/kubernetes/scheduler.conf
```

修改配置文件，告诉系统刚刚的yaml文件的位置

```
/etc/kubernetes/manifests/kube-scheduler.yaml
```

```
10 spec:
11   containers:
12   - command:
13     - kube-scheduler
14     - --authentication-kubeconfig=/etc/kubernetes/scheduler.conf
15     - --authorization-kubeconfig=/etc/kubernetes/scheduler.conf
16     - --bind-address=127.0.0.1
17     - --kubeconfig=/etc/kubernetes/scheduler.conf
18     - --leader-elect=true
19     - --config=/etc/kubernetes/scheduler-extender.yaml
20   image: registry.aliyuncs.com/google_containers/kube-scheduler:v1.27.2
21   imagePullPolicy: IfNotPresent
22   livenessProbe:
23     failureThreshold: 8
24     timeoutSeconds: 15
25   volumeMounts:
26   - mountPath: /etc/kubernetes/scheduler.conf
27     name: kubeconfig
28     readOnly: true
29   - mountPath: /etc/kubernetes/scheduler-extender.yaml
30     name: scheduler-extender
31     readOnly: true
32   hostNetwork: true
```

```
1 volumes:
2 - hostPath:
3   path: /etc/kubernetes/scheduler.conf
4   type: FileOrCreate
5   name: kubeconfig
6 - hostPath:
7   path: /etc/kubernetes/scheduler-extender.yaml
8   type: File
9   name: scheduler-extender
10 status: {}
```

请求一个新容器，与刚刚的请求相同

```
kubectl apply -f memory-demo.yaml
```

2. 命令解释
3. 命令示例

```
2023/06/19 17:05:45 start up sample-scheduler-extender!
2023/06/19 17:13:12 pod with name:memory-demo4 need schedule-scheme
2023/06/19 17:13:12 Node worker1 limit is:cpu→16000m,mem→60186MB.Now has requested is:cpu→100m,mem→450MB
2023/06/19 17:13:12 Node worker3 limit is:cpu→16000m,mem→60186MB.Now has requested is:cpu→100m,mem→50MB
2023/06/19 17:13:12 Best Node is worker1
2023/06/19 17:13:59 pod with name:memory-demo5 need schedule-scheme
2023/06/19 17:13:59 Node worker1 limit is:cpu→16000m,mem→60186MB.Now has requested is:cpu→100m,mem→550MB
2023/06/19 17:13:59 Node worker3 limit is:cpu→16000m,mem→60186MB.Now has requested is:cpu→100m,mem→50MB
2023/06/19 17:13:59 Best Node is worker1
2023/06/19 17:14:43 pod with name:memory-demo6 need schedule-scheme
2023/06/19 17:14:43 Node worker1 limit is:cpu→16000m,mem→60186MB.Now has requested is:cpu→100m,mem→650MB
2023/06/19 17:14:43 Node worker3 limit is:cpu→16000m,mem→60186MB.Now has requested is:cpu→100m,mem→50MB
2023/06/19 17:14:43 Best Node is worker1
```

按照我们的调度算法，请求三次，任务均被放置在worker1上。

2. 二 调度后返回的数据格式

内容为 json 格式，如下所示：

```
{
    # taskName 执行的整个任务的名称
    "taskName": "任务的名称",
    "time": "", # 从开始到结束的耗时
    "taskType": "", # 任务调度类型，普通调度：common； 任务调度：task；
    "tasks": [ # 任务拆分的子任务的数组，数组的元素是 子任务，为Object
        {
            "taskName": "", # 拆分的子任务的名称
            "nodeName": "", # 执行该子任务的节点名称
            "startTime": "", # 开始时间，格式为
            "endTime": "", # 结束时间
            "time": "" # 任务执行耗时
        }
    ]
}
```