## 一、开关机

### 1、调度

#### 功能：

1）把没有pod的节点关闭；

2）如果可用节点资源不够，再打开关闭的节点。

#### 实验方式：

部署在master节点上，在master节点上执行命令。

1）将node232上除了系统pod之外的pod清除掉，然后运行脚本就会关闭这台机器

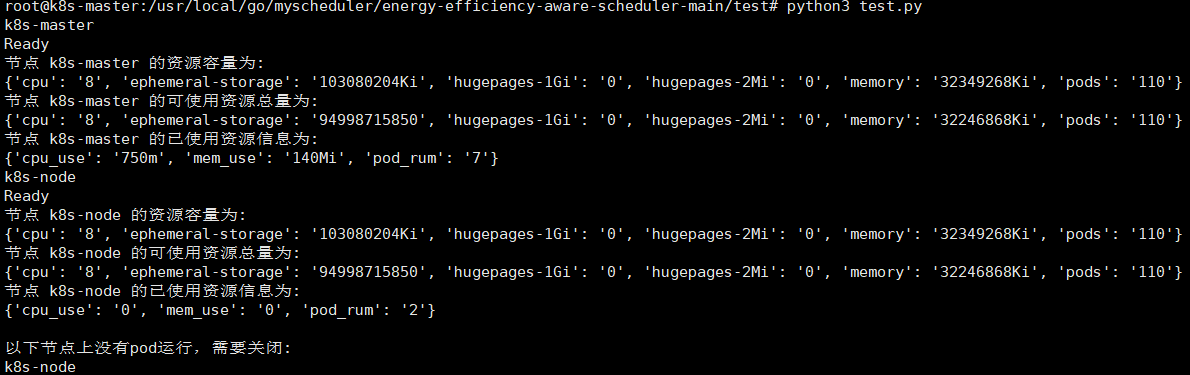
python ./open\_close\_code/dispatch/close\_machine.py

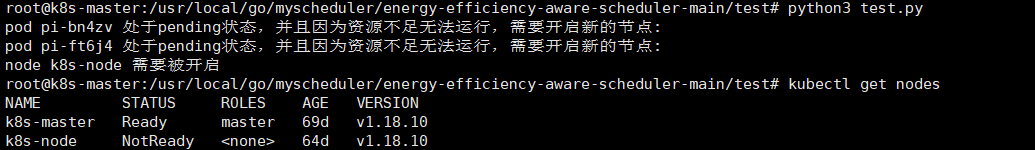
部署在master节点上，在master节点上执行命令。

2）调整任务的yaml文件，将任务的资源需求设置值增大，使其超过节点的资源总数，这样刚刚关闭的机器就会重新打开。

python ./open\_close\_code/dispatch/open\_machine.py

#### 自测结果：





### 2、迁移

#### 功能：

1) 遍历系统中的pod，找到运行Pod最少的num个节点，num目前指定为1

2) 把这些节点上的pod迁移到其它节点上。

3) 然后关闭选中节点。

#### 实验方式：

部署在master节点上，在master节点上执行命令。

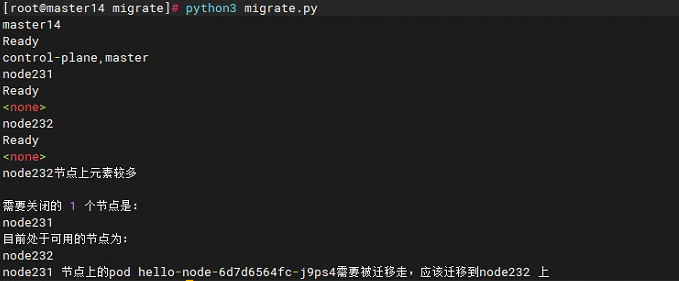
python ./open\_close\_code/migrate/migrate.py

1）node232上有足够的资源

2）node232上运行2个任务，node231上运行1个任务

3）将node231上的任务迁移到node232上，然后关闭node231

#### 自测结果：





## 二、DVFS

### 1、调度

#### 功能：

1）实时监控CPU利用率，当某个机器的利用率低于某个数值时，通过dvfs把该机的cpu频率降低。当CPU利用率上升时，再通过dvfs恢复cpu频率。

2) k8s在筛选节点时，只在没有其它可用节点的情况下，才考虑以最低频率运行的机器, 如果选中了某台机器，先把它的频率恢复正常。

#### 实验方式：

部署在node节点上，在node节点上执行。

1）部署监控cpu利用率的脚本，在node节点上运行多个任务，将cpu利用率跑至50%以上，将调整cpu运行状态为performance，然后将pod删除，cpu利用率降至50%以下，将调整cpu运行状态为powersave。

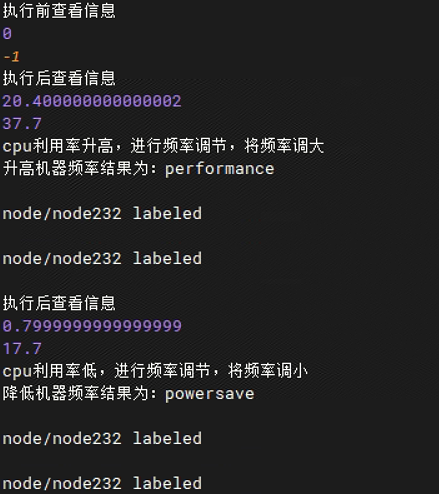
python ./dvfs\_code/dispatch/adjust\_freq.py

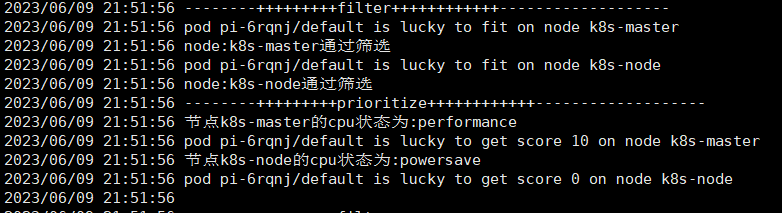
在master上部署了调度器，在node上部署调整cpu状态程序。

2）sudo ./dvfs\_code/dispatch/dvfsScheduler/dvfsScheduler（master）

执行以上命令后，将优先考虑处于performance的机器，如果机器都处于powersave，将会将选中的powersave的机器状态调整为performance（由脚本完成 sudo ./dvfs\_code/dispatch/ set\_freq.py node）

#### 自测结果：





### 2、迁移

#### 功能：

1) 监控CPU利用率，当某个机器的利用率高于某个数值时，把该节点上的pod迁移到利用率低的节点上。循环执行迁移，直到该节点的cpu利用率低于某个数值

2) 调用dvfs，把该节点的频率降低。

#### 实验方式：

部署在node节点上，在node节点上运行

1）将该节点的cpu利用率运行至50%以上，另一个节点cpu利用率保持较低水平，这样将会触发迁移

python ./dvfs\_code/migrate/dvfs\_migrate.py

调用完成后，如果需要迁移，将结果写入指定文件，完成迁移后，然后再次调用，如果有返回文件则继续迁移，直到调用脚本之后没有返回文件，则迁移结束。

#### 自测结果：



## 三、power capping

### 1、调度

#### 功能：

1) 实时监控服务器功率，当某个机器的功率高于最大功率的一个百分比时，通过power capping限制该机的功率。

2) k8s在筛选节点时，只在没有其它可用节点的情况下，才考虑被进行了power capping的机器，如果选中了某台机器，首先要解除限制。

#### 实验方式：

部署在node节点上，在node节点上执行。

1）部署监控功率的脚本，在node节点上运行多个任务，将功率运行至最大功率的50%以上，将进行power capping，然后将pod删除，功率下降至最大功率的50%以下，将解除power capping。

python ./power\_capping\_code/dispatch/adjust\_power.py

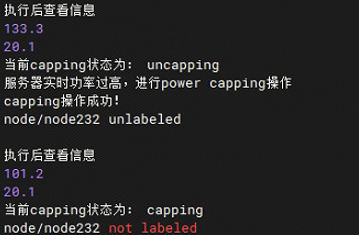
在master上部署了调度器，在node上部署进行power capping和uncapping的脚本。

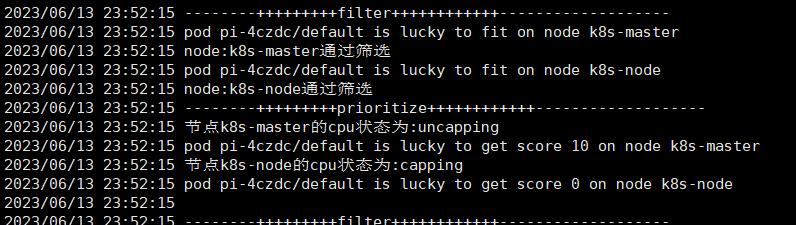
2）

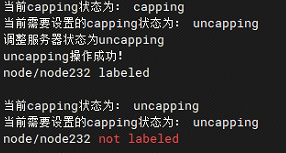
sudo ./power\_capping\_code/dispatch/powerCappingScheduler/powerCappingScheduler（master）

执行以上命令后，将优先考虑处于uncapping的机器，如果机器都处于capping，将会将选中的capping的机器解除功率限制（由脚本完成 sudo ./power\_capping\_code/dispatch/set\_power.py）

#### 自测结果：







### 2、迁移

#### 功能：

1）实时监控服务器功率，当某个机器的功率高于最大功率的一个百分比时，首先选择num个pod，执行迁移；

2）如果迁移之后服务器的功率仍然大于阈值，则调用rapl，限制该服务器的功率。

#### 实验方式：

部署在node节点上，在node节点上运行

1）将某节点的功率运行至最大功率的30%以上，然后将会触发迁移，将迁移动作写入指定文件，将节点上的一个pod迁移走，功率下降。20s的迁移时间，20s后查看功率，如果仍然高于30%，则继续迁移，最多进行num次迁移。num次之后，仍不能达到指定条件，则进行power capping。

python ./power\_capping\_code/migrate/capping\_migrate.py

调用完成后，如果需要迁移，将结果写入指定文件，完成迁移后（20S迁移时间），如果有返回文件则继续迁移，直到没有返回文件（最多num次迁移），则迁移结束。

#### 自测结果：

