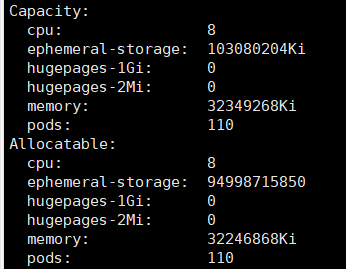
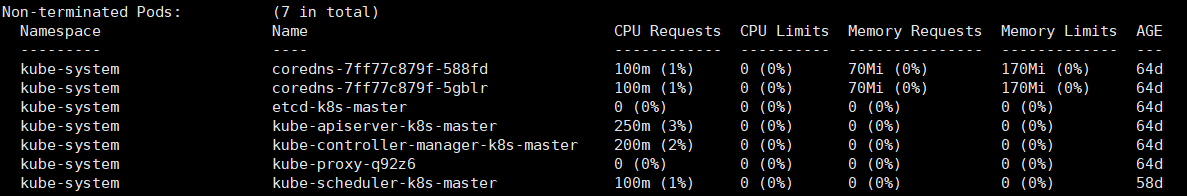
# 能效感知调度系统原型设计文档

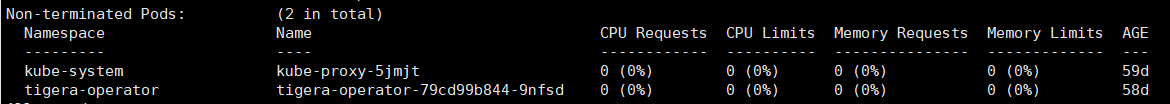
## 一、实验

kubectl describe node k8s-master查看节点信息：

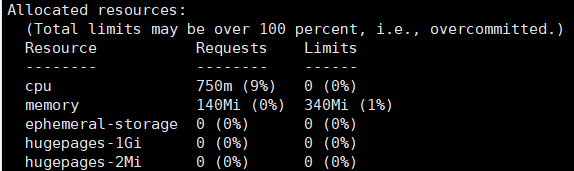


包含了节点的容量以及可分配的信息（需要白名单pod列表）



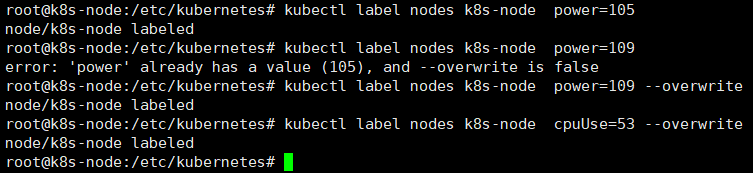


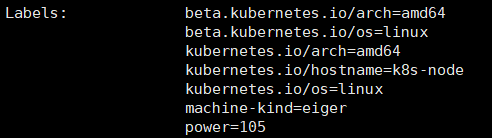
包含了运行的pod的信息

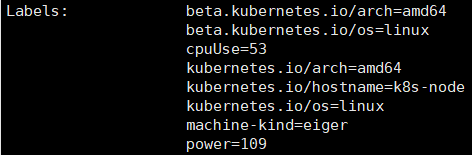


显示已经分配的资源的信息

在node节点上用 kubectl label nodes k8s-node power=109 --overwrite 命令设置power，在另一台节点上查看，发现设置成功。然后在该节点之上重新修改label，在另一台服务器上查看，发现进行了更新。

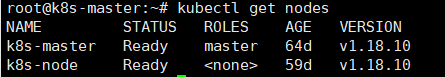




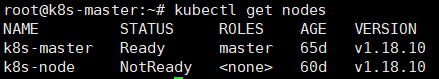


## 二、需要做的实验

2.1 两台服务器都开启的时候，我们使用命令行查看

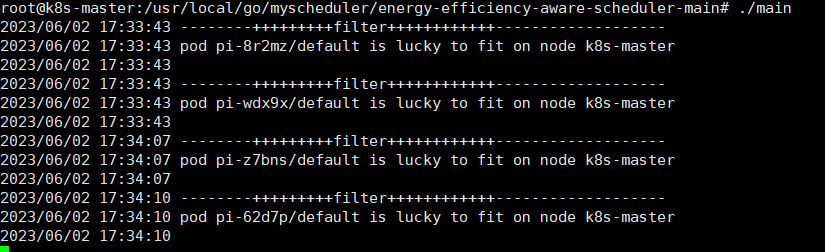


关闭其中一台机器（需要邮箱验证码），继续查看是什么情况？节点状态会是not ready嘛

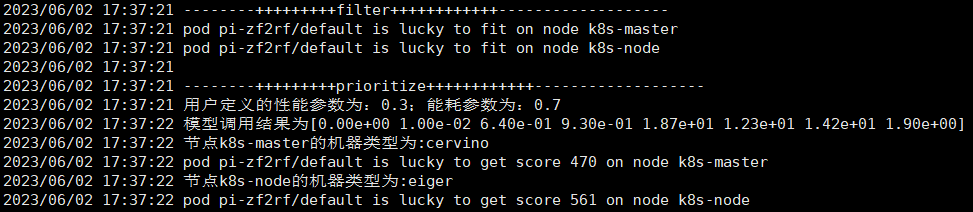


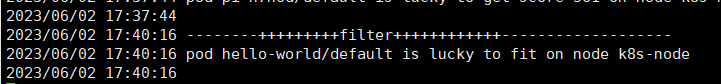
2.2 筛选阶段传递给scheduler extender的是经过k8s筛选之前的还是之后的，not ready的会传递过来嘛？

关闭结点之后，notready的不会传过来



节点都打开了，但是master没有传过来，是因为调度策略先做了筛选，才会传过来。只有一个经过筛选，就不会在进行优选。





## 三、代码设计

基本采集程序：

~~程序1：在每一个节点上部署该程序，获取该台机器的功耗和cpu利用率，然后使用kubectl命令设置标签的值~~。~~获取目前节点所处的cpu状态，并保存在label中。~~

开关机：

调度：

~~程序2：开关机的调度，获取所有的node，查看是否有node没有pod运行，有的话关闭（关机什么时候执行，还是只出接口就行：只出接口就行，他们调用）。~~

~~程序3：在我们的extender scheduler中，如果预选阶段传过来的node是空的话，我们直接调用程序4.~~

~~经过实验验证，如果没有节点满足情况，不会调用到这里，我们选择遍历pod，如果有对应的pod 处于pending状态，并且原因是因为Insufficient，我们就开启新结点。~~

~~程序4：获取当前所有的节点，找到为开启节点中的没有准备的节点，从中选择能效最高的。我们的实现随机选择了一台节点~~

迁移：

~~程序5：设置参数num，遍历节点，获取有任务运行的、并且运行pod个数最少的num个节点，将这些节点关闭，并且记录下在这些节点上运行的pod，输出这些pod（该如何去处理？是直接告诉他们pod名称，然后他们去重新创建pod，然后走k8s的调度流程；还是需要我获取pod的资源，然后模拟k8s的调度过程，自己完成调度，然后告诉他们pod和node映射关系，比较负载，是完全模拟还是部分模拟？自己写的话考虑不全，要实现整个k8s的过程）（自己模拟简单的，只考虑资源就行，别的不考虑，而且用轮询也可以）~~

DVFS：

调度：

~~程序1：获取每台机器列表，获取其标签中的cpu利用率，低于阈值时，像该台机器发送thrift远程调用命令，调节cpu频率，并记录标签，高于阈值，发送thrift远程调用命令恢复频率。在每一个节点上部署程序即可，~~

~~程序2：scheduler extender优选阶段优先选择label中没有在最低频率运行的节点，如果节点被选择，那么我恢复其频率。~~

迁移：

程序3：现在是在DVFS阶段，节点不会被关闭，因此会有空闲节点（没有的话人为创造几个空闲节点），获取集群中所有节点，将空闲节点通过thrift设置为power saving模式。

程序4：实时监测cpu利用率，高于某个数值是挑选pod完成迁移，迁移到空闲节点上直到低于某个阈值，然后通过thrift调节其频率。

power capping：

调度：

程序1：~~获取服务器的功率，高于某个阈值时，通过power capping限制功率，thrift接口。同时修改状态，不想dvfs有自己的一套更新逻辑，我们在进行功率封顶时同时修改状态。~~

~~程序2：scheduler extender优选阶段优先选择label中没有被限制功率的节点，如果节点被限制，那么我解除功率限制。~~

迁移：

程序3：获取服务器的功率，高于某个阈值时，先通过迁移num个pod来降低功率，num个pod迁移完还是高，则进行功率限制。（该执行哪个动作？是分开执行的）