# 组合式弹性伸缩工具软件使用说明书

**开发单位：同济大学**

**1. 系统概述**

**1.1 开发背景**

弹性伸缩是云原生的一大特性，使得用户可以随时按需地获取计算资源，避免了计算资源的浪费。但是现有的弹性伸缩工具仅支持单一的伸缩方式，即仅水平伸缩与垂直伸缩，于是我们面向Kubernetes自主开发了“组合式弹性伸缩工具软件”来进一步提升云原生应用的服务质量，并降低资源使用成本。

**1.2 功能简介**

本工具主要有微服务的性能数据采集、微服务的请求模拟、微服务的组合式伸缩三大功能。

1.2.1 微服务的性能数据采集

本软件提供了微服务的性能数据收集功能。对一个微服务实例，在不同的资源量下，对其进行压力测试，采集其在不同资源量的吞吐量数据。

1.2.2 微服务的请求模拟

本软件通过调用第三方的压力测试工具，模拟不同类型的工作负载，如平稳型负载，周期型负载，激增型负载等。

1.2.3 微服务的组合式伸缩

在进行请求模拟的同时，启用本软件的组合式伸缩功能。根据实时的请求量、用户期望的请求响应时间、当前正在使用的微服务实例方案，本工具同时考虑资源成本和违约成本，定期地推荐一个最优的组合式伸缩方案。

此外，为了解决频繁伸缩带来的抖动问题，本工具在最优的伸缩方案和当前的微服务实例配置方案中做一个决策，以此减少伸缩的次数，保证了一定的稳定性。

**1.3 适用范围**

本软件适用于了解Kubernetes和弹性伸缩相关知识的人员，可用于学术研究和工业应用。

**1.4 运行环境**

本软件支持Linux平台。

**2. 使用说明**

**2.1 部署微服务应用**

本工具的功能是实现微服务的弹性伸缩，在使用该工具前，需要在Kubernetes集群中部署待伸缩的微服务。本工具针对的是无状态微服务的伸缩场景，因此微服务以deployment的形式进行部署，deployment是最常用的controller。能够管理pod的多个副本，并确保pod按照期望的状态运行。同时需要注册相应的service。service定义了外界访问一组特定pod的方式。service有自己的IP和端口，service为pod提供了负载均衡。

如图1、2，部署了一组用php实现的MD5加密任务，同时以NodePort的方式注册了相应的service；副本的初始个数为1个；CPU的初始资源量为1950m，内存的初始资源量为500Mi。

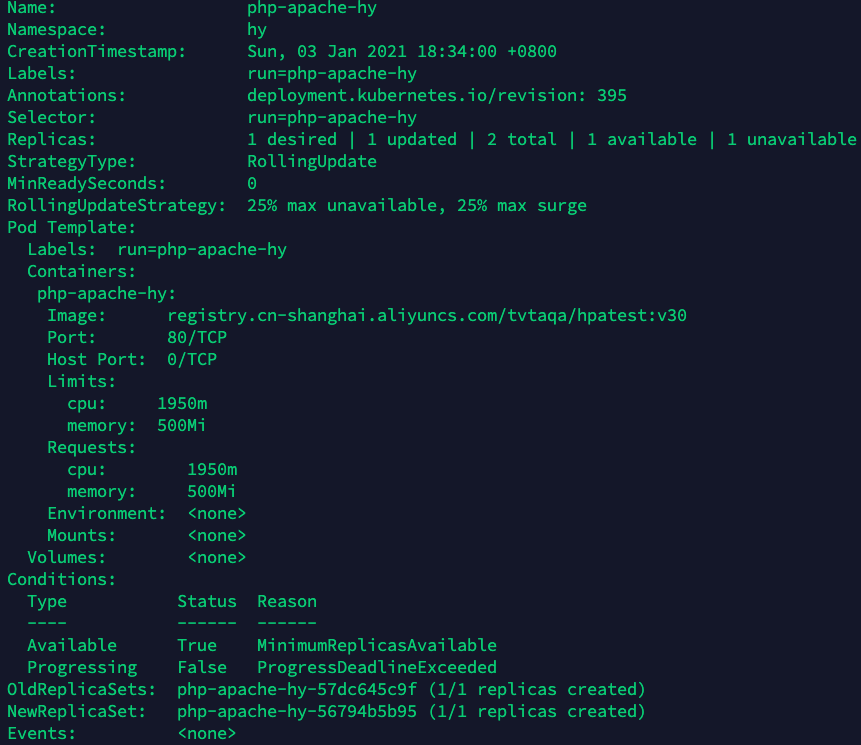


图1



图2

**2.2** 微服务的性能数据采集

采集微服务性能数据的方法是利用垂直伸缩改变实例的资源量，并对该资源量下的实例进行压力测试。对于微服务应用，一般而言，随着资源量的增加，其处理能力也会逐步上升。本软件的数据采集采用了shell脚本的方式，在shell脚本中调用第三方的压力测试工具hey。

2.2.1 catch\_data.sh的输入

在Kubernetes集群的master节点上，使用脚本文件catch\_data.sh采集微服务实例的性能数据。执行该脚本时需要输入的信息如下：

微服务的service ip，如图3所示

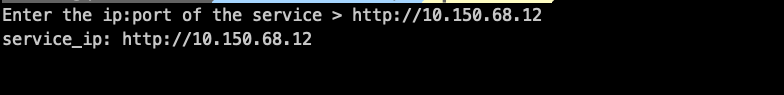


图3

微服务的namespace和deployment，如图4所示

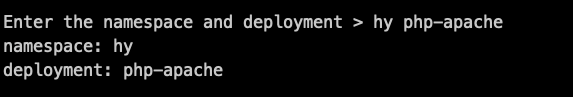


图4

微服务的CPU资源量的下限、上限以及增量值，如图5所示

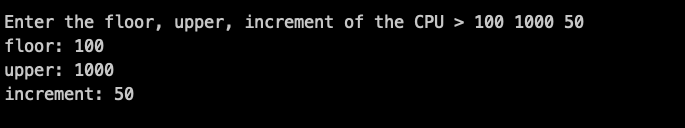


图5

每一种资源量下，重复采集的轮数，如图6所示

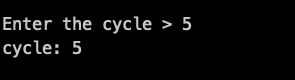


图6

2.2.1 catch\_data.sh的输出

脚本采集到的原始数据包含CPU的资源量，以及当前采集的轮次，采集数据的系统时间，以及hey压测工具的日志信息，如图7所示。

图X中展示的示例数据为：在CPU资源量为100m时，第5轮的压测数据，压测的时间为26.89s秒，每秒处理的请求个数约为3.3个请求。原始数据的下方还展示了具体的相应时间分布图：

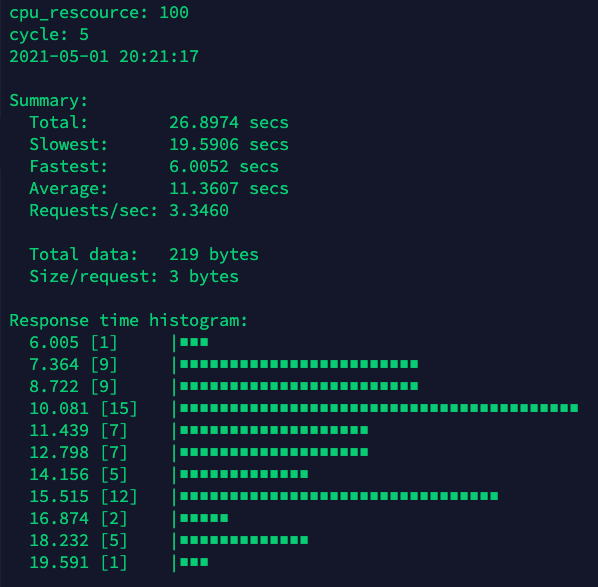


图7

为了便于工具的使用，需要对原始数据进行一定的处理，处理后的数据格式为key-value的键值对，其中key对应的是cpu的资源量，value对应的是该资源量下微服务每秒处理的请求个数，如图8

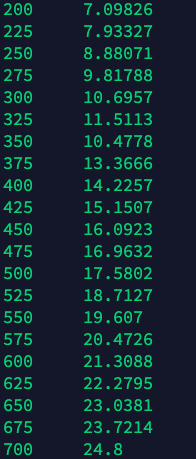


图8

**2.3** 微服务的请求模拟

实际生产环境中的负载类型是变化多端的，有上升型的负载、周期型的负载、激增型的负载等。如图9，是激增类型的负载。请求的模拟，本软件使用shell脚本的方式，在脚本中调用第三方压测工具模拟终端用户的请求，设置不同的发送速率以及不同的并发量以此模拟真实的负载。

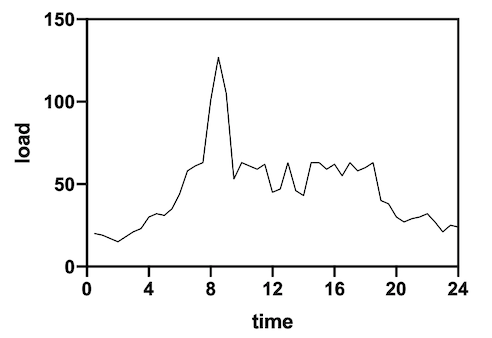


图9

2.3.1 压测工具命令参数

图10是hey压测工具的命令选项，关键参数为：

-n：发送的请求个数

-q：请求发送的速率

-c：并发的客户端的个数

-z：持续的发送时间

通常-q 与 -c一起使用，q\*c的值表示有c个客户端，每个客户端的发送速率为q。如果使用了-z选项，则-n选项会被忽略。

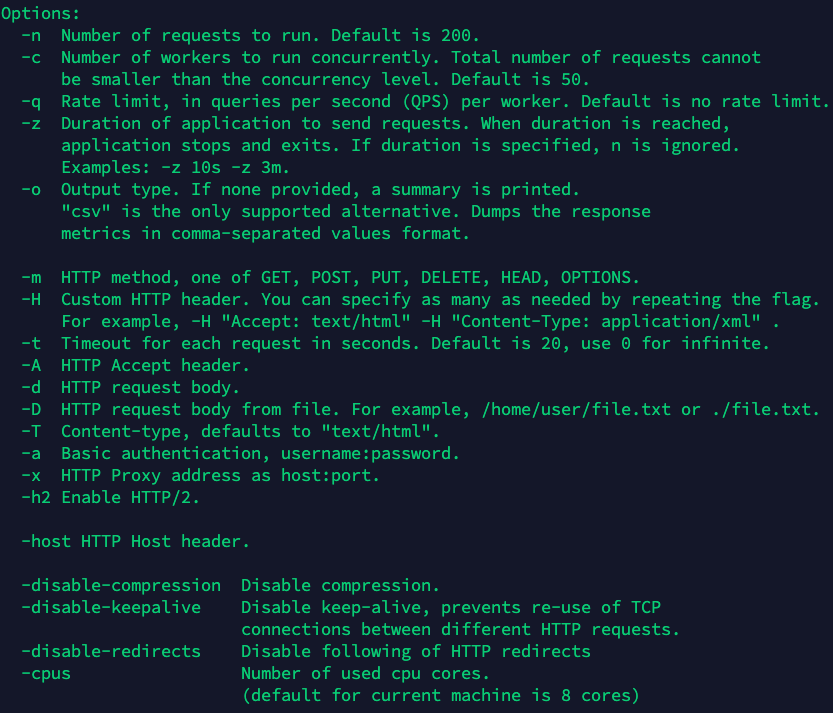


图10

2.3.2 压测工具日志

压测工具的具体日志如图11、12。

Total表示总运行时长；Slowest表示最慢的请求完成的时间；Average表示平均相应时间；Request/sec表示qps；柱状图0.102[152]表示相应时间在0.055～0.102之间的请求数量有152个；图12中Latency distribution为pct分布；最后的状态码分布表示有848个请求成功相应。

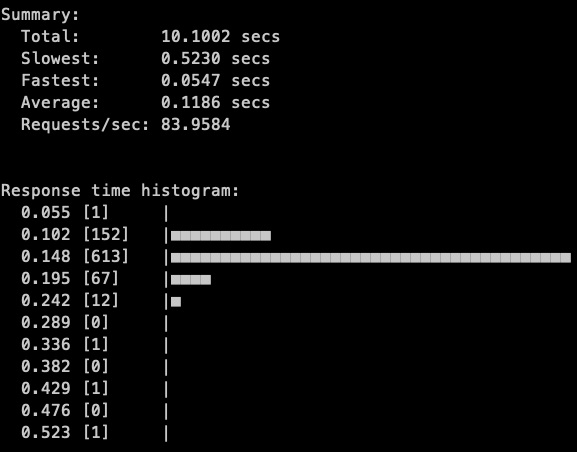


图11

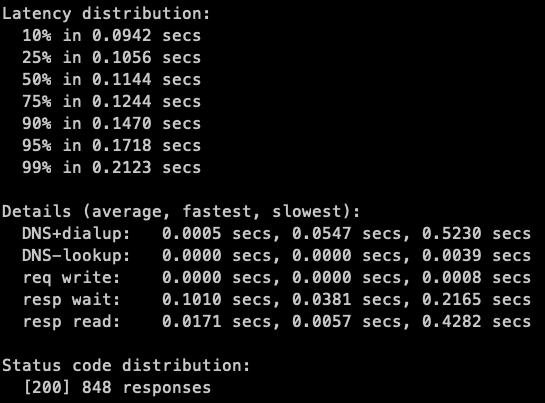


图12

2.3.3 hey\_load.sh脚本输入

微服务的请求模拟是通过shell脚本中调用hey压测工具实现的。为了让负载不断地发生变化，本工具支持定期地调整请求量，同时预设了4种负载类型的请求量，分别为平稳型、激增型、上升型、下降型以及，如图13。

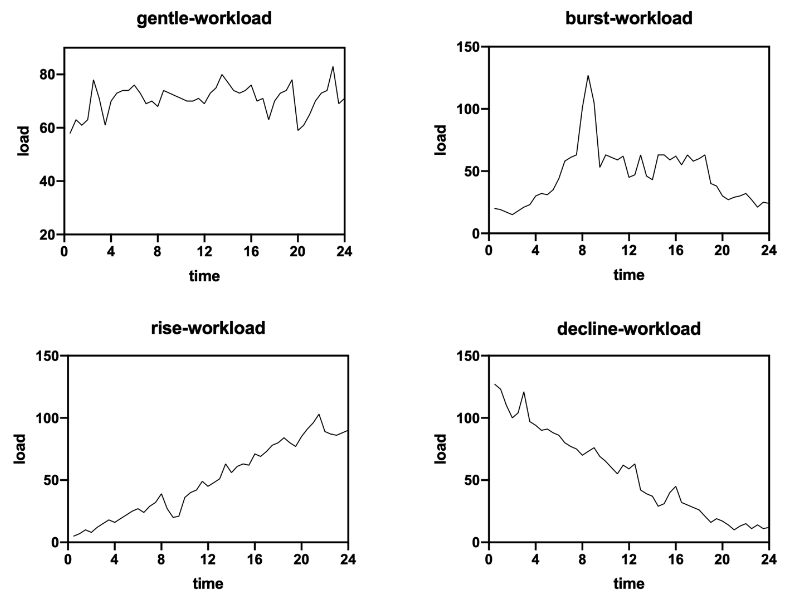


图13

待用户输入的参数如下：

微服务的cluster ip:port，如图14

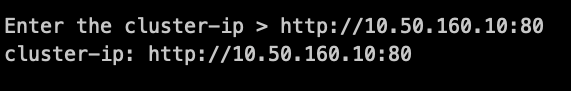


图14

请求量变换的时间，即一种请求量维持interval秒，如图15：



图15

输入负载类型，预设的负载类型有burst, gentle, rise, decline四种类型，如图16:



图16

2.3.4 hey\_load.sh脚本输出

第一行输出的信息为负载类型；第二行为hey压测工具的命令；第三行开始为hey的具体日志，如图17。

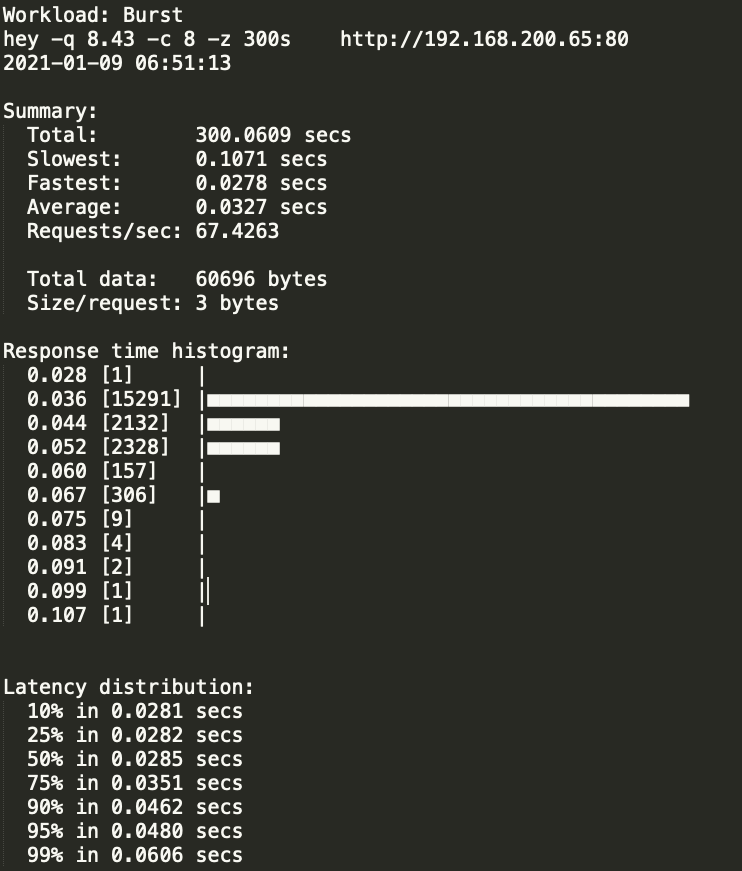


图17

**2.4** 微服务的组合式伸缩

2.4.1配置运行环境

组合式伸缩模块使用的是python kubernetes client库进行的开发，图18为运行所需的包。

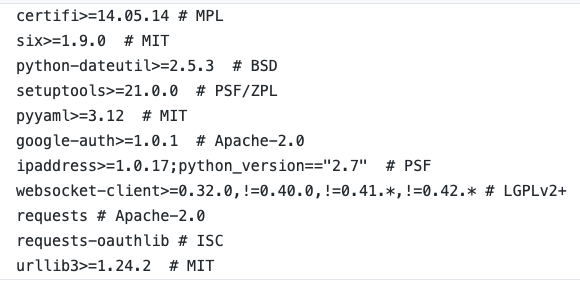


图18

2.4.2配置伸缩参数

arg.yaml文件中包含了待配置的参数。

ms和mu参数是Kubernetes滚动式更新的两个参数。ms参数的含义为：和期望ready的副本数比，超过期望副本数最大比例（或最大值），这个值调的越大，副本更新速度越快。mu参数的含义是为，和期望ready的副本数比，不可用副本数最大比例（或最大值），这个值越小，越能保证服务稳定，更新越平滑；

Wd Wr分别表示违约成本和资源成本的权重，Wd 的值越大，代表用户更注重服务质量，此时伸缩工具会偏向于使用更多的资源量。而的值越大，Wr代表用户对资源量更为敏感，此时伸缩工具会牺牲一定的服务质量，让微服务使用尽可能少的资源量。

threshold值是为了减少伸缩的频率。本伸缩工具定期地推荐一个最优伸缩方案，但每次根据最优的伸缩方案进行伸缩会造成抖动问题，为了降低伸缩的频率，本工具会重新评估当前正在使用的微服务实例方案，在最优的伸缩方案和正在使用的方案中，做一个决策。该决策是比较两个方案的得分，只有当qt_temp超过threshold时，才会根据推荐方案执行伸缩。

其余需要配置的参数如下表。

表1 伸缩算法参数

|  |  |
| --- | --- |
| p\_cpu | cpu每秒单价 |
| p\_mem | mem每秒单价 |
| ms | 滚动式更新 MaxSurge参数 |
| mu | 滚动式更新 MaxUnavailable参数 |
| rtt | 用户规定的响应时间 |
| interval | 每隔interval(s)获取当前负载 |
| redundancy | 冗余量，比如当前负载是100，计算推荐方案时，使用105计算 |
| pod\_num\_max | 最多允许多少个实例 |
| pod\_max\_limit | pod最大的资源量 |
| pod\_min\_limit | pod最大的资源量 |
| threshold | 伸缩的阈值 |
| Wd | 违约成本的权重 |
| Wr | 资源成本的权重 |

表1的配置参数示例如图19：

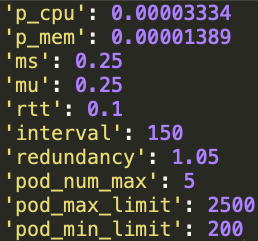


图19

2.4.3 SLA协议参数配置

arg.yaml文件中还需要用户配置SLA协议参数。当请求的响应时间小于等于期望的响应时间时，代表服务可用；反之则说明服务不可用。如表2，当服务可用性大于99%时，违约费用为10%；当服务可用性介于95%～99%之间时，需要支付30%的违约费用；当服务可用性低于95%时，需要支付100%的服务费用。

表2 SLA协议

|  |  |
| --- | --- |
| 服务可用性*p* | 违约金额 |
| 99%≤ *p* <99.9% | 10%的服务费用 |
| 95≤ *p* <99% | 30%的服务费用 |
| *p*<95% | 100%的服务费用 |

SLA协议参数配置示例如图20所示：

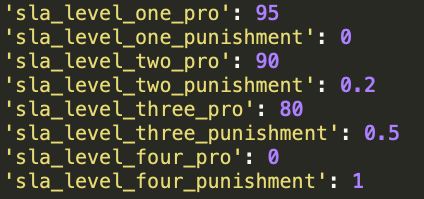


图20

2.4.4微服务信息配置

在arg.yaml文件的最后包含了微服务信息的配置，如表3

表3

|  |  |
| --- | --- |
| namespace | namespace的name |
| deployment | deployment的name |
| per\_pod\_start\_time | 每一个pod的理论启动时间 |

2.4.5执行组合式伸缩

python3 hybrid.py 或者 在后台运行 nohup python3 hybrid.py > log.txt 2>&1 &!

2.4.6 组合式伸缩输出

如果使用命令python3 hybrid.py，在命令行会有输出信息。current index代表当前的的时间戳，是第几次执行伸缩的推荐。load代表的是此时的请求量。pod\_num表示的副本的个数；pod\_res代表每一个pod的cpu资源量；pod\_total\_res代表总的cpu资源量；pod\_rps代表每一个pod的最大处理能力；pod\_total\_rps代表该微服务总的处理能力；svc\_ws为理论的请求平均响应时间。svc\_pro为服务可用性；res\_cost和sla\_cost分别是资源成本与违约成本。图21是输出示例。如果使用命令nohup python3 hybrid.py > log.txt 2>&1 &!，则该输出信息在log.txt文件中。

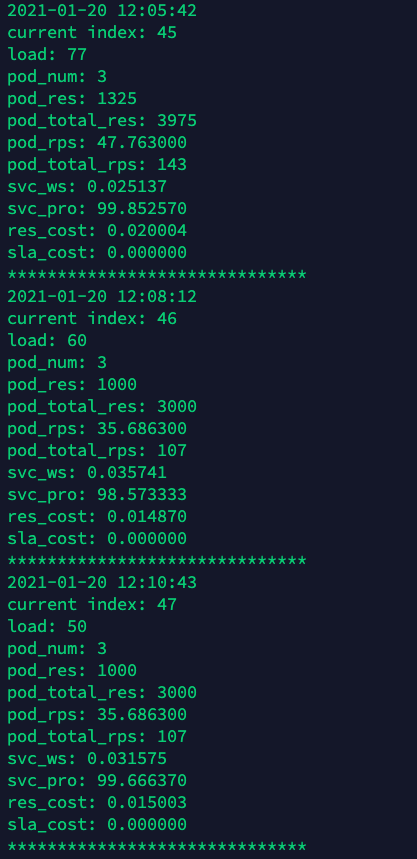


图21

2.4.6 组合式伸缩日志

hybrid\_log.txt中记录了更详细的日志，包含推荐方案和正在使用的方案的具体信息，如图22。

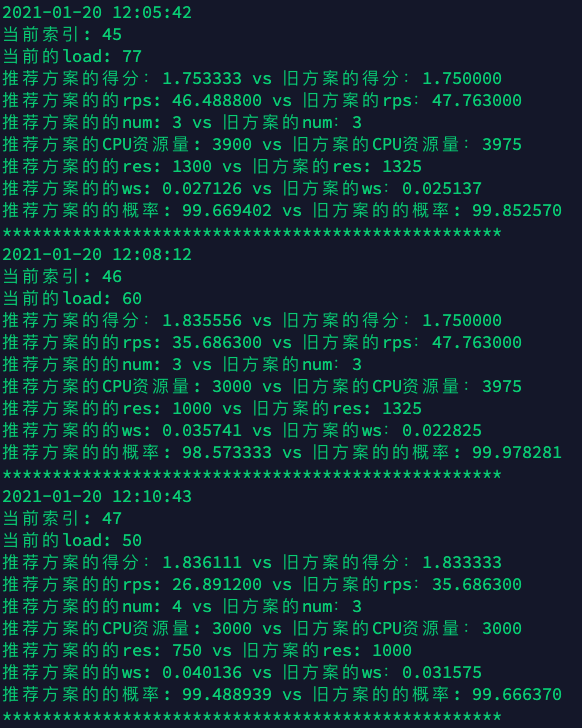


图22