

# 复赛任务书-v1.0

发布日期

2021-03-30



华为技术有限公司



目 录

1 更新记录 ..... 1

2 背景信息 ..... 2

3 题目定义 ..... 3

3.1 服务器 ..... 3

3.2 虚拟机 ..... 4

3.3 资源规划和调度 ..... 4

4 交互协议 ..... 6

4.1 协议说明 ..... 6

4.2 交互示例 ..... 8

5 评分规则 ..... 9

# 1 更新记录

表1-1

版本	修改内容	发布时间
V1.0		2021-03-30

# 2 背景信息

---

- 云上资源的规划和调度是云计算场景中非常重要的一个优化问题。好的优化算法能够为云运营商节约上亿的运营成本，并为客户提供更稳定、更流畅的云端体验。
- 此次赛题来源于华为公司实际面对的一个生产场景，并作了相应的简化。我们期待您精彩的解决方案。

# 3 题目定义

- 3.1 服务器
- 3.2 虚拟机
- 3.3 资源规划和调度

## 3.1 服务器

- 服务器类型：**在公有云的运营场景中，我们的数据中心可以选购的服务器类型有多种，以下是目前公有云常用的一些服务器类型。

表3-1

服务器型号	规格（CPU 核数，内存大小）	硬件成本	每天能耗成本
NV603	92C， 324G	¥ 53800	¥ 500
NV604	128C， 512G	¥ 87800	¥ 800
...	...	...	...

- NUMA 架构：**目前主流的服务器都采用了非统一内存访问（NUMA）架构，你可以理解为每台服务器内部都存在两个 NUMA 节点：**A** 和 **B**（下文中提到的节点均指 NUMA 节点）。服务器拥有的资源（CPU 和内存）均匀分布在这两个节点上。以 NV603 为例，其 A、B 两个节点分别包含 46C 和 162G 的资源。保证服务器的 CPU 核数和内存大小均为偶数。
- 服务器成本：**数据中心使用每台服务器的成本由两部分构成：硬件成本和能耗成本。硬件成本是在采购服务器时的一次性支出，能耗成本是后续服务器使用过程中的持续支出。为了便于计算，我们以天为单位计算每台服务器的能耗成本。若一台服务器在一天内处于关机状态，则其不需要消耗任何能耗成本，否则我们需要支出其对应的能耗成本。

## 3.2 虚拟机

- **虚拟机类型：**我们面向用户提供了多种类型的虚拟机售卖服务，用户可以根据自己的需求来自由选购，以下是一些常用的虚拟机类型。

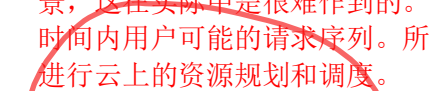
表3-2

虚拟机型号	规格（CPU 核数，内存大小）	单节点/双节点 部署
s3.small.1	1C, 1G	单节点
c3.large.4	2C, 8G	单节点
c3.8xlarge.2	32C, 64G	双节点
...	...	...

- **单节点/双节点部署：**由于服务器存在两个节点，对应的虚拟机也存在两种部署方式：单节点部署或双节点部署。单节点部署指的是一台虚拟机所需的资源（CPU 和内存）完全由主机上的一个节点提供；双节点部署指的是一台虚拟机所需的资源（CPU 和内存）必须由一台服务器的两个节点同时提供，并且每个节点提供总需求资源的一半。例如：c3.8xlarge.2 类型的虚拟机需要被部署在一台服务器的 A 和 B 两个节点上，A、B 节点分别提供 16C，32G 资源。请注意，一种类型的虚拟机是单节点部署还是双节点部署与其规格没有必然联系，但是双节点部署的虚拟机保证其 CPU 和内存需求量都是偶数。

## 3.3 资源规划和调度

- **容量约束：**服务器可以用来容纳用户的虚拟机，但是服务器上的任意一个节点(A 和 B)上的资源负载(CPU 和内存)均不能超过其容量上限。
- **请求类型：**用户的请求共分为两种类型：创建请求和删除请求。创建请求表示用户向公有云平台提交的一个创建虚拟机的请求；删除请求表示用户提交的删除一台之前创建的虚拟机的请求。
- **请求序列：**由一系列请求构成的序列。题目会给出接下来若干天中每一天用户的请求序列，根据每天的请求序列，你需要进行相应的资源规划和调度。
- **数据中心扩容：**在得知了一天的请求序列后，你可以在实际进行调度前进行一次数据中心扩容。即购买一些新的服务器来容纳后续用户请求的虚拟机，同时你需要付出所购买服务器相应的硬件成本。你需要指定购买哪些类型的服务器以及购买的数量。初始时你没有任何服务器。
- **虚拟机迁移：**在完成扩容后，在处理每一天的新请求之前，你还可以对当前存量虚拟机进行一次迁移，即把虚拟机从一台服务器迁移至另一台服务器。对于单节点部署的虚拟机，将其从一台服务器的 A 节点迁移至 B 节点(或反之)也是允许的。迁移的目的服务器和节点必须有足够的资源容纳所迁移的虚拟机。迁移的虚拟机总量不超过当前存量虚拟机数量的百分之三。即假设当前有  $n$  台存量虚拟机，每天你可以迁移的虚拟机总量不得超过  $3n/100$  向下取整。

- **部署虚拟机：**在完成扩容和迁移之后，你需要按顺序处理当天所有的新请求。对于每一个创建虚拟机的新请求，你要为虚拟机指定一台服务器进行部署。若虚拟机是单节点部署的，你还需要指明部署在服务器的 A 节点还是 B 节点。处理请求的过程中，任意一台服务器上每个节点容纳的虚拟机资源总和都不能超出节点本身的资源容量（指 CPU 和内存两个维度）。
  - **未知请求序列：**在初赛，我们面对的是提前知晓了未来所有用户请求序列的场景，这在实际中是很难作到的。现实场景中，我们往往只能预测后续较短的一段时间内用户可能的请求序列。所以在复赛中，你需要面对这种存在未知的场景，进行云上的资源规划和调度。
- 

# 4 交互协议

## 4.1 协议说明

## 4.2 交互示例

### 4.1 协议说明

本题使用交互式评测，你的程序应该从标准输入读取输入，并将输出打印至标准输出。交互协议如下：

- 输入的第一行包含一个整数  $N(1 \leq N \leq 100)$ ，表示可以采购的服务器类型数量。
- 接下来  $N$  行，每行描述一种类型的服务器，数据格式为：**(型号, CPU 核数, 内存大小, 硬件成本, 每日能耗成本)**。例如(NV603, 92, 324, 53800, 500)表示一种服务器类型，其型号为 NV603，包含 92 个 CPU 核心，324G 内存，硬件成本为 53800，每日能耗成本为 500。CPU 核数，内存大小，硬件成本，每日能耗成本均为正整数。每台服务器的 CPU 核数以及内存大小不超过 1024，硬件成本不超过  $5 \times 10^5$ ，每日能耗成本不超过 5000。服务器型号长度不超过 20，仅由数字和大小写英文字符构成。
- 接下来一行包含一个整数  $M(1 \leq M \leq 1000)$ ，表示售卖的虚拟机类型数量。
- 接下来  $M$  行，每行描述一种类型的虚拟机，数据格式为：**(型号, CPU 核数, 内存大小, 是否双节点部署)**。是否双节点部署用 0 和 1 表示，0 表示单节点部署，1 表示双节点部署。例如(s3.small.1, 1, 1, 0)表示一种虚拟机类型，其型号为 s3.small.1，所需 CPU 核数为 1，所需内存为 1G，单节点部署；(c3.8xlarge.2, 32, 64, 1)表示一种虚拟机类型，其型号为 c3.8xlarge.2，所需 CPU 核数为 32，所需内存为 64G，双节点部署。CPU 核数，内存大小均为正整数。对于每种类型的虚拟机，数据集保证至少存在一种服务器可以容纳。虚拟机型号长度不超过 20，仅由数字，大小写英文字符和 '.' 构成。
- 接下来一行包含一个整数  $T(1 \leq T \leq 1000)$  和一个整数  $K(1 \leq K \leq T)$ ，分别表示共有  $T$  天的用户请求序列数据，初始你只知晓前  $K$  天的数据。
- 接下来会按顺序给出  $K$  天的用户请求序列，每一天的数据格式如下：
- 对于每一天的数据，第一行包含一个整数  $R$  表示当天共有  $R$  条请求。
- 接下来  $R$  行，按顺序给出每一条请求数据。请求数据的格式为：**(add, 虚拟机型号, 虚拟机 ID)**或**(del, 虚拟机 ID)**分别表示创建一台虚拟机或者删除一台虚拟机。



例如(add, c3.large.4, 1)表示创建一台型号为 c3.large.4, ID 为 1 的虚拟机; (del, 1) 表示删除 ID 为 1 的虚拟机。虚拟机的 ID 均为整数, 每个创建请求的虚拟机 ID 唯一, 范围不超过带符号 32 位整数表示的范围。对于删除操作, 数据集保证对应 ID 的虚拟机一定存在。

- 当读取完前 K 天的请求序列数据后, 你的程序需要输出第一天的决策信息, 每天的决策信息格式如下:
- 首先第一行输出(purchase, Q), 其中 Q 是一个整数表示你需要扩容购买多少种类型的服务器。Q 不能为负数且不能大于 N。
- 接下来 Q 行, 每行的格式为: (服务器型号, 购买数量)。例如(NV603, 2)表示扩容购买两台 NV603 服务器。购买数量必须为正整数, 在整个用例中, 你所购买的服务器总量不能超过  $10^5$ 。
- 你所购买的每一台服务器都会被分配一个编号, 从零开始。例如你在第一天输出了

```
(NV603, 2)
(NV604, 1)
```

则你会拥有两台 NV603 服务器, 编号分别为 0 和 1; 一台 NV604 服务器, 编号为 2。在第二天输出了

```
(NV603, 3)
```

则会新增三台 NV603 服务器, 编号分别为 3, 4, 5。

- 接下来一行输出(migration, W), 其中 W 是一个整数表示你要迁移的虚拟机的数量。
- 接下来 W 行, 每行表示一个虚拟机的迁移, 格式为(虚拟机 ID, 目的服务器 ID)或(虚拟机 ID, 目的服务器 ID, 目的服务器节点)。例如(3, 1)表示将 ID 为 3 的虚拟机从当前所在服务器迁移至 ID 为 1 的服务器, 该虚拟机必须是双节点部署的; (4, 1, A)表示将 ID 为 4 的虚拟机从当前所在服务器迁移至 ID 为 1 的服务器的 A 节点, 该虚拟机必须是单节点部署的。
- 接下来按输入中请求的顺序, 输出对于当前这一天的每一个创建请求, 该虚拟机部署的服务器 ID, 格式为(服务器 ID)或者(服务器 ID, 部署节点)。对于双节点部署的虚拟机, 你只需要输出其部署的服务器 ID, 对于单节点部署的虚拟机, 你还需要输出部署的节点(A 或 B)。
- 当你的程序完整输出了第一天的决策信息后, 你将会读取到第 K+1 天的请求序列数据。然后你需要输出第二天的决策信息, 并且读取第 K+2 天的请求序列, 以此类推。

#### ⚠ 注意

当你的程序完整输出了一天的决策信息后, 请务必清空输出缓存区, 对于 C/C++ 语言, 你可以调用 `fflush(stdout)`; 对于 Java 语言, 你可以调用 `System.out.flush()`; 对于 Python 语言你可以调用 `sys.stdout.flush()`。

- 当你已经完整地读取了所有 T 天的请求序列数据后, 你只需要按顺序输出剩余的决策信息, 不再需要作读入。
- 用户创建请求数量总数不超过  $10^5$ 。

### 说明

裁判程序在进行合法性判断时，严格按照输入和输出的请求序列顺序进行模拟，所有的操作(包括创建，删除和迁移)都是按顺序串行生效的。

## 4.2 交互示例

<---- input ----> (用于说明，该行非实际交互数据)

```
2
(NV603, 92, 324, 53800, 500)
(NV604, 128, 512, 87800, 800)
2
(c3.large.4, 2, 8, 0)
(c3.8xlarge.2, 32, 64, 1)
3 2
2
(add, c3.large.4, 5)
(add, c3.large.4, 0)
```

```
2
(del, 0)
(add, c3.8xlarge.2, 1)
```

<---- output ----> (用于说明，该行非实际交互数据)

```
(purchase, 2)
(NV603, 1)
(NV604, 1)
(migration, 0)
(0, A)
(0, B)
```

<---- input ----> (用于说明，该行非实际交互数据)


```
3
(add, c3.large.4, 2)
(del, 1)
(del, 2)
```

<---- output ----> (用于说明，该行非实际交互数据)

```
(purchase, 0)
(migration, 0)
(1)
```

```
(purchase, 0)
(migration, 0)
(1, B)
```

# 5 评分规则

- 
- 总成本低的方案胜出。总成本包含两部分：购买服务器的整体硬件成本以及服务器消耗的整体能耗成本。整体硬件成本即将选手输出的方案中所有购买的服务器的硬件成本相加。整体能耗成本的计算方式为：在处理完每一天的所有操作后(包括迁移，创建和删除)，裁判程序会将当前有负载(至少部署了一台虚拟机)的服务器视为开机状态，没有任何负载的服务器视为关机状态，以此计算当天的能耗成本。整体能耗成本即每一天的能耗成本的总和。
  - 若参赛选手的方案对应的总成本相同，则迁移虚拟机数量较少者胜出。
  - 若参赛选手的方案中迁移虚拟机的数量依然相同，则参赛选手程序计算出方案的所用时间少者胜出(单位为 ms)。
  - 若参赛选手程序计算运行时间也相同，则先提交代码的队伍胜出。
  - 对于多组测试数据，采用结果相加排名。
  - 对于每组测试数据，选手的程序所有计算步骤(每一步用时的定义为从裁判程序完成当前的输出到读取到选手的对应输入之间的用时)所用时间相加总和不能超过 90 秒，若程序运行超时，运行出错或者输出了不合法的解，则记无成绩。对于多组测试数据，选手的程序在任意一组数据上无成绩则记整体无成绩。

## ⚠ 注意

因为通信交互存在一定的时间开销，用时统计从裁判程序侧和选手程序侧可能存在差异。建议选手控制算法用时留有一定的冗余。