作者：ddmm  
链接：https://zhuanlan.zhihu.com/p/356155386  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

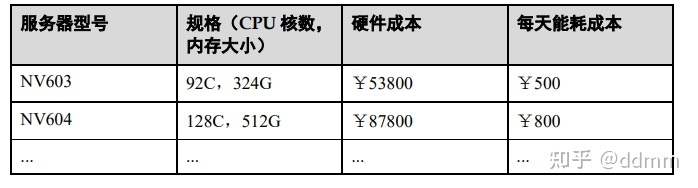
今天早上赛题出来后，我和群友一样马上研究起了题目，踩了无数的坑，终于在晚上9：30的时候提交出了一个天数。现将提交注意的事项，和楼主踩过的坑记录在这里。还有一件事，**武长赛区找队友。**

## **题目分析**

本题是一个优化问题，这类赛题在近期的许多比赛中都有出现，服务器和虚拟机的匹配类似于一个二分图匹配，需要在不超过服务器负载的情况下使得用户成功创建不同规格的虚拟机。

最重要的部分就是审题

## 1.NUMA 架构的服务器。



赛题组给的NUMA介绍如下：

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。

由于**服务器的资源是均匀分布**，所以我们每个服务器CPU核数 *c* 和内存大小 *m*，则它在A，B两个节点均分后，各有c/2 个cpu核数 和 m/2 大小的内存。

我采用的是unordered\_map来存储每种NUMA服务器的信息，虚拟机为了方便匹配服务器，设计的数据结构如下：

// 服务器信息  
unordered\_map<string,vector<int>> serverInfos;  
// 虚拟机信息  
unordered\_map<string,vector<int>> vmInfos;

在解析txt文件时，将可供购买的服务器类型和用户可以创建的虚拟机类型信息解析保存。

void generateServer(string &serverType,string &cpuCores,string &memorySize,  
 string &serverCost,string &powerCost){  
 string \_serverType="";  
 for(int i =1;i<serverType.size() -1;i++){  
 \_serverType += serverType[i];  
 }  
 int \_cpuCores =0,\_memorySize=0,\_serverCost=0,\_powerCost=0;  
  
 for(int i=0;i<cpuCores.size() -1;i++){  
 \_cpuCores = 10\*\_cpuCores + cpuCores[i] - '0';  
 }  
 for(int i=0;i<memorySize.size() -1;i++){  
 \_memorySize = 10\*\_memorySize + memorySize[i] - '0';  
 }  
 for(int i=0;i<serverCost.size() -1;i++){  
 \_serverCost = 10\*\_serverCost + serverCost[i] - '0';  
 }  
 for(int i=0;i<powerCost.size()-1;i++){  
 \_powerCost = 10\*\_powerCost + powerCost[i] - '0';  
 }  
 serverInfos[\_serverType] = vector<int>{ \_cpuCores/2 ,  
 \_cpuCores/2,\_memorySize/2,\_memorySize/2,\_serverCost,\_powerCost};  
}

在读取文件时，采用freopen进行重定向到txt文件，采用cin标准输入读取数据。代码如下。

std::freopen(filePath.c\_str(),"rb",stdin);  
 int serverNum;  
 string serverType,cpuCores,memorySize,serverCost,powerCost;  
 scanf("%d",&serverNum);  
  
 for(int i =0;i<serverNum;i++){  
 cin>>serverType>>cpuCores>>memorySize>>serverCost>>powerCost;  
 generateServer(serverType,cpuCores,memorySize,serverCost,powerCost);  
 }

## 2.数据扩容

题目描述：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

这里在输出答案会导致错误，在输出时，我们**每一天都要输出一个购买数量(purchase, T)**

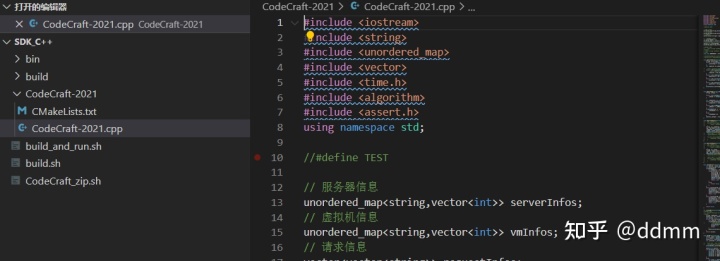
**和一个迁移数量(migration, N)** 。特别注意，这里购买数量(purchase, T) 的**T需要购买的服务器的种类数。**

## **3.文件读取路径问题**

这次在线上环境中，读取数据的形式和2020年的软挑形式不一样，我们不需要自己写读取文件的txt路径，而是类似于各大oj，系统自己会给你做输入，我们之间接收数据处理即可。

## **4. 关于本地编译的问题**

由于需要cmake编译，很多小伙伴之前没有接触过这些东西，导致上手起来比较慢，我在这里介绍一下我的编译方法。我是windows系统，使用git来运行sh build.sh进行C++ 项目的编译，编译之后，文件夹会出现build和bin两个文件夹，直接将sdk\_C++这个文件夹压缩成zip格式即可提交。我的提交文件夹结构如下。

文件夹结构

## **5.优化方向**

如何初始化购买服务器，如何进行虚拟机的迁移，还有扩容策略我在这版代码中没有考虑，方便各位有缘人发挥自己的创造力，各种解决NP-Hard问题的智能算法是一个值得尝试的方向，同时，强化学习的方法来解决这个问题，在解决模型泛化的情况下好像也是一个不错的策略。

## **6.Baseline**

最后附上我的baseline



代码地址：

[Ubuntu Pastebinpaste.ubuntu.com](https://link.zhihu.com/?target=https%3A//paste.ubuntu.com/p/jx4PjcK7fN/)

## **结语：**

不是c++大佬，代码写的不是很好，时间仓促。如果能对大家有帮助，我就很开心了。另外打个广告，我主要使用C\C++,python，有类似比赛经验，武长赛区有同学想带带我的话，可以私信我哦。还有哦，代码有什么问题欢迎大家在评论区指出来，同时，评论区也欢迎各位在这里分享思路。我也将不定期更新自己的进度，欢迎大家持续关注。

## 3.11日更新内容

## 根据官方的说明，我们可以一次性读取所有天数的用户请求，然后对每天当中的用户请求按顺序处理。所以，当一个虚拟机被释放后，可以立即使用释放的服务器资源进行再分配。

为了方便大家值观理解，画了个图，serverInfos 和 vnInfos的数据结构如下图所示。

![](data:application/octet-stream;base64,)

**修复释放虚拟机**的bug后Baseline如下，同时将输出信息放在了最后统一进行输出。代码的效率得到了提升（那代码的输入是不是也可以一次性读完再按天处理呢）。



同时发现vector存储服务器和虚拟机的信息可读性太差，不方便直观的理解各个变量的意义，计划未来将服务器和虚拟机的信息用**结构体进行保存**。

#### 直播提示

* bin-packing 算法
* 数据集全部读入，可以后面一次性输出
* 输入的逗号括号需要自己处理
* 开放性求解局部最优。例如：贪心，动态规划。
* 今年无最优解，所以每部分都要优化。 每个人都要掌握全队的思路，要跟队友多交流
* 组合优化类算法
* 单双节点最好分开来做
* 选一两种服务器就可以达到最优解

2021.3.16

（1）数据结构设计 vector:{ID, core, mem, cost, power} lzy

typedef long long ll;
  
   
//供选的服务器
  
vector<serverInfo> serverInfos; //这里string是唯一的
  
//1、按照自定义的标准（性价比： 单位核的成本（硬件+100天能耗）和单位内存的成本）排序 这部分gxf
  
//---2、直接按照规格大小排序，然后每隔多少步取一种。假如取四种---
  
//排序之后映射string->index;
  
unordered\_map<string, int> serverType2Info; //string to index
  
struct serverInfo {
  
 string typeId;
  
 struct Node {
  
 int core, memory;
  
 } A, B;
  
 int serverCost, powerCost;
  
};
  
   
//虚拟机类型
  
unordered\_map<string, vmInfo> vmInfos;
  
struct vmInfo {
  
 string typeId;
  
 int core, memory, isTwoNode;
  
 int
  
};
  
   
enum reqType {ADD, DEL};
  
struct requestInfo {
  
 string vmType; //ADD时需要用
  
 reqType type;
  
 int id; //虚拟机id
  
};
  
   
//当前存在的服务器（已经购买的）
  
//vector<serverInfo> serverResource;
  
vector<serverInfo> server4OneNode;
  
vector<serverInfo> server4TwoNode;
  
   
//当前存在的虚拟机 虚拟机有ID
  
   
unordered\_map<int, int> vmInfos; //虚拟机ID->服务器ID
  
   
serverInfo readServer() {
  
   
}
  
   
vmInfo readVm() {
  
   
}
  
   
requestInfo readRequest() {
  
   
}

（2）输入解析，写成函数。readServer, readVm, readRequest gxf 先写这个

serverInfo readServer() {
  
 freopen();
  
}

（3）整理逻辑（流程）

//每一天的工作：购买服务器、分配虚拟机、（迁移）、计算成本 tw

（4）优化方向：（先考虑减小成本、运行时间目前只需要在90s内跑完就行）

* 购买（扩容）服务器（每天都可以购买，也可一次性购买很多）:购买服务器规格大小（可以选大的小的搭配） gxf
* 分配服务器（把服务器分成两组，单节点部署的全部在一组服务器上，双节点部署在另一组）
* 迁移虚拟机（今天先不管）lzy 今天先看看

（5）优化思想：贪心、动态规划？（先去百度一下）

* 是否可以模拟操作系统分配资源的算法：比如分配内存的最优适配（每次分配使用最小的可用资源）
* 直播有说这是个bin-packing（NPC） 问题，需要考虑的是在服务器容量的约束下，最小化购买成本和能耗成本的

（6）每天最新的发到群里，命名日期加名字+版本