**负载均衡功能说明**

1. **数据结构**

**（在本算法中未使用的部分数据成员不加以说明）**

1. **负载结构**

typedefstruct {

charmetric\_name[40]; // 维度名称

double used; // 占用容量

double total; // 总容量

double reserved; // 保留值

} st\_metric;

1. **物理机结构**

typedefstruct{

st\_host host;

/\*所在组信息\*/

pmgPtrat\_p\_pmgrps[10]; /\*物理机所在的组\*/

charat\_gr\_ids[100]; /\*denotes which group the PM belongs to，一个PM可以放置多个组\*/

intcountofpmgrp; /\*物理机所在的组个数\*/

intat\_pmlist\_id; /\*在全局pmlist中的下标\*/

/\*所在框信息\*/

framePtrat\_p\_fr; /\*物理机所在的框\*/

OBJECTIDat\_fr\_sid; /\*denotes the frame where PM located， PM对应1个框\*/

/\*所包含虚拟机信息\*/

vmPtrp\_vms[100];/\*每个PM最多能容纳100个VM\*/

intcountofvm;/\*每个PM中包含的VM数\*/

/\*PM COST预留字段\*/

double performance;

double cost;

} st\_pm;

1. **虚拟机结构**

typedefstruct{

st\_host host;

st\_net\_element net\_elm; //The net element which VM belongs to

intat\_vmgid;/\*the vmgroup that the VM belongs to\*/

int head; //指向afflist的链表头下标

int at; //<!—放置位置--> if s\_at is null then means for initial placement

//在迁移时，at表示源物理机，at\_pmid表示目标物理机

Int at\_pmid;//The PM where VM located. if null means initialplace

int ask;/\*临时增加，用于生成数据\*/

int n\_aff\_fr\_list[300];/\*用于存放完整的不同框反亲和性信息\*/

int n\_aff\_fr\_total;

int aff\_pm\_head,n\_aff\_pm\_head,aff\_fr\_head,n\_aff\_fr\_head,aff\_gr\_head,n\_aff\_gr\_head;

/\*用于存放各类亲和性\*/

} st\_vm;

1. **虚拟机组结构**

typedefstruct {

int groupid;

int vm\_id\_list[MAX\_VM\_COUNT\_PER\_GROUP];/\*该组所有VM点的集合\*/

int edge\_id\_list[MAX\_EDGE\_PER\_GROUP];/\*该组的边集合,从g\_v2vlist[N]数组中对应的边的序号\*/

int vm\_total;//该组VM 的个数

int edge\_count; //该组边的个数

int is\_kernel; //该vmg根据亲和性是至少两个冲突组的放置核心

double itraffic;//组内流量

double otraffic;//组外流量

int at\_pmid;/\*初试放置决策后获得的pmid\*/

st\_metric load[DIMENSION]; /\*loaday\*, \*denote vmCpu, vmMem, vmDiskio, vmFpga.\*/

int head;

} st\_vm\_group;

1. **主要功能函数说明**
2. **负载均衡函数**

函数声明：

void loadbalance()

函数功能：

该函数是入口函数。负载均衡的主要流程都集成在该函数内。

1. **物理机内分组函数**

函数声明：

Void grouping\_in\_pm(intpm\_id)

函数功能：

该函数的作用是将物理机内的虚拟机按照流量和亲和性分组。考虑到负载均衡处理的都是已完成放置的虚拟机，因此不同物理机上不会有同一组的虚拟机，只需按照单台物理机上的。

1. **虚拟机组内负载求和函数**

函数声明：

Void SumVMG(int)

函数功能：该函数将各个虚拟机组的负载之和更新到虚拟机组的数据结构中。

1. **计算当前系统均衡度**

函数声明：

double getSysBalance()

函数功能：该函数的作用是计算每个维度偏离该维度平均负载的方差，作为系统当前均衡度。

1. **计算单台物理机的负载均衡度**

函数声明：

double getPMSkw(pmPtr pm)

函数功能：该函数计算出单台物理机每个维度偏离系统平均负载的方差。

1. **选择负载均衡度最差的物理机**

函数声明：

int selectPMbyLoad(int flag)

函数功能：flag为0时选择负载均衡度最高的物理机，返回PMID；flag为1时选择负载均衡度最低的物理机。

1. **从指定物理机上选择最适合迁移的VM组**

函数声明：

int selectVMGfromPM(int pmname, int flag)

函数功能：逐个尝试计算删去某个虚拟机组后，该物理机的负载均衡度，以此决定需要移除的虚拟机组。该函数会计算最佳和次佳选择，当flag为0时返回最佳，为1时返回次佳。

1. **为选择好的vm选择一台pm迁入，不能放回原pm**

函数声明：

int selectPMforVMG(int pmname\_o, int vmgname)

函数功能：根据负载能力和亲和性筛选可供放置的物理机，并计算假设放置后该物理机的均衡度，以选择最佳放置目标PM。

1. **迁出虚拟机组函数**

函数声明：

void getVMGoutofPM(vmgPtr vmg, pmPtr pm)

函数功能：该函数的作用是将虚拟机组迁移出物理机，并更新负载，物理机上的虚拟机列表等信息。

1. **虚拟机组放置函数**

函数声明：

void moveVMGtoPM(vmgPtr vmg, pmPtr pm)

函数功能：该函数的作用是将虚拟机组放置到物理机，并更新负载，物理机上的虚拟机列表等信息。

1. **处理流程**
2. 虚拟机分组预处理，组负载信息更新。
3. 计算出单台物理机每个维度偏离系统平均负载的方差
4. 选择负载均衡度最差的物理机
5. 从指定物理机上选择最适合迁移的VM组
6. 为选择好的vm选择一台pm迁入
7. 将虚拟机组取出，放入待迁移虚拟机组列表，重复做2-5直到无热点
8. 将决策存入决策列表中
9. 输出结果
10. **使用说明**
11. 入口为cls\_st\_loadbalance类的loadbalance（）函数，初始化后即可调用，方式如下

cls\_st\_loadbalance lb;

lb.sd = \_scthis->sd;

init\_loadbalance(&lb);

输入需要：xml1, xml2, xml21, xml22

1. 输出：xml3
2. 现用输入输出目录：./ data/input/loadbalance
3. 除了正常迁移外有二种特殊情况：
   1. 负载已较为均衡，不需要迁移，输出结果为空
   2. 负载不均衡的物理机，由于其他物理机资源不足或亲和性要求，不能迁出虚拟机组，则停止负载均衡过程