**热点消除功能说明**

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**

typedef struct {

char metric\_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**

typedef struct{

st\_host host;

/\*所在组信息\*/

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

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

int countofpmgrp; /\*物理机所在的组个数\*/

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

/\*所在框信息\*/

framePtr at\_p\_fr; /\*物理机所在的框\*/

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

/\*所包含虚拟机信息\*/

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

int countofvm;/\*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**

typedef struct{

st\_host host;

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

int at\_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**

typedef struct {

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. **热点消除函数**

函数声明：

void hotsolve(void)

函数功能：

该函数是驱动函数、入口函数。热点消除的各种函数集成在该函数内。

1. **物理机内分组函数**

函数声明：

void grouping\_in\_pm(int pm\_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。考虑到热点消除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和更新到虚拟机组的数据结构中。

1. **寻找热点物理机**

函数声明：

BOOL FindHotPm(int \*pmID, int \*load)

函数功能：该函数的作用是寻找出热点物理机及其最热维度，若无热点则返回-1.

1. **选择迁出虚拟机组函数**

函数声明：

st\_vm\_group \* selectVMG(pmPtr pm, int load)

函数功能：该函数的作用是在一直物理机及其最热维度的情况下，选取一组恰好消除热点或者接近恰好消除热点的虚拟机组。

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 placeVMGtoPM(vmgPtr vmgs, int n\_vmg, pmPtr pms, int n\_pms, vm2pmPtr ret,int\* n\_ret)

函数功能：该函数的作用是给带迁移虚拟机组选择符合亲和性关系且大小合适的物理机放置。

1. **处理流程**
2. 虚拟机分组预处理，组负载信息更新。
3. 寻找热点物理机，选取合适虚拟机组以消除该物理机上的热点
4. 将虚拟机组取出，放入待迁移虚拟机组列表，重复做2、3直到无热点
5. 依次放置待迁移虚拟机组列表中的虚拟机组
6. 将决策存入决策列表中
7. 输出结果
8. **使用说明**
9. 入口为cls\_st\_hotsolve类的hotsolve（）函数，初始化后即可调用，方式如下

cls\_st\_hotsolve hsl;

init\_hotsolve(&hsl);

hsl.sd=\_scthis->sd;

hsl.hotsolve();

1. 输入需要：xml1, xml2, xml21, xml22
2. 输出：xml3
3. 现用输入输出目录：./Data/input/hotsolve/
4. 处理前后的热点物理机数量会打印在屏幕上，不输入到xml中
5. 除了正常迁移外有三种特殊情况：
   1. 没有热点，不需要迁移，输出结果为空
   2. 有消除不掉的热点，原因是物理机资源不足，错误信息会输出到xml3中
   3. 有消除不掉的热点物理机，原因是物理机上的虚拟机均不可迁移（即stat=2或有AT要求），错误信息会打印在屏幕上（debug=1时会打印），不输入xml中