# 分组切割及初始放置功能说明

## 数据结构说明

在实现分组切割以及初始放置的过程中使用到的数据结构体如下：

### 1.1 负载结构

定义及说明如下：

typedef struct {

char metric\_name[40]; // 负载名称

double used; // 已用量

double total; // 总量

double reserved; // 预留量

} st\_metric;

该结构用于记录物理机、虚拟机和虚拟机分组的负载信息。

### 1.2 虚拟机结构

定义及说明如下：

（在算法中未使用的数据成员不加以说明）

typedef struct{

int id; //虚拟机的ID

OBJECTID sid; //字符串形式的虚拟机ID/名称

st\_metric metricArr[DIMENSION]; //虚拟机的8项负载

char vnic\_deviceid[20];

char ip[30];

int link\_port;

OBJECTID at;

int at\_partid;//虚拟机所在的组号

int at\_pmid;//虚拟机所在的物理机

st\_net\_element net\_elm;

int TenaID;

int affinity\_type;

int affinity\_at;

} st\_vm;

该结构用于记录虚拟机的各项信息。

### 1.3 物理机结构

定义及说明如下：

（在算法中未使用的数据成员不加以说明）

typedef struct{

int id;//物理机ID

OBJECTID sid;//字符串形式的物理机ID/名称

st\_metric metricArr[DIMENSION]; //物理机的8项负载

char ovsmac[30];

char nicmac[PORT\_NUM][30];

char hostip[30];

char ip[PORT\_NUM][30];

int stat;

int rack\_id;

int group\_id;

int frame\_id;

} st\_pm;

该结构用于记录物理机的各项信息。

### 1.4 虚拟机到虚拟机的流量

定义及说明如下：

（在算法中未使用的数据成员不加以说明）

typedef struct{

int id;//该组流量关系的ID

st\_vm\* src\_vm;//指向该流量关系中起点虚拟机的指针

st\_vm\* dst\_vm;//指向该流量关系中终点虚拟机的指针

double bandwidth;

double trafficrate;//流量大小

double delay;

int affinity\_type;

int affinity\_at;

OBJECTID at;

} st\_vm2vm;

该结构用于记录虚拟机之间的流量、带宽、时延等信息。

### 1.5 虚拟机分组结构

定义及说明如下：

（在算法中未使用的数据成员不加以说明）

typedef struct {

int groupid;//该组的组号

int vm\_id\_list[MAX\_VM\_COUNT\_PER\_GROUP];//该组所有虚拟机的集合

int edge\_id\_list[MAX\_EDGE\_PER\_GROUP];//该组所有边的序号的集合

int vm\_total;//该组中虚拟机的个数

int edge\_count; //该组中边的个数

int at\_pmid;//该组所在的物理机编号

st\_metric metricArr[DIMENSION]; //该组的负载信息

} st\_part\_vm\_group;

该结构用于记录虚拟机进行分组与切割之后所得到的虚拟机组。

### 1.6 邻接表节点

定义及说明如下：

（在算法中未使用的数据成员不加以说明）

typedef struct{

int dst;//终点虚拟机ID

double traffic;//流量大小

}st\_adj\_list\_node;

该结构是之后将要提到的邻接表结构的一个节点。

### 1.7 邻接表结构

定义及说明如下：

（在算法中未使用的数据成员不加以说明）

typedef struct{

int vm\_id;//当前虚拟机ID

st\_adj\_list\_node adj\_vm\_id\_list[MAX\_VM\_COUNT\_PER\_GROUP];//邻接表节点数组，即与该虚拟机有边的虚拟机以及该边表示的流量大小

int adj\_vm\_id\_count;//邻接表节点个数

} st\_adj\_vmid\_list;

该结构在分组算法中用于读取虚拟机之间是否存在联系，在切割算法中还需要从这个结构中得到流量信息并进行相应的修改与合并操作。

## 主要函数说明

### 2.1 邻接表生成函数

函数声明如下：

void create\_adj\_vm\_list();

该函数的作用是从记录虚拟机到虚拟机流量的数组 g\_v2vlist 中获取所有流量信息，为实行分组算法做准备。

### 2.2 分组算法函数

函数声明如下：

void part\_component();

该函数的作用是遍历所有虚拟机节点，找出其中没有访问标记的节点，然后用深度优先搜索算法（DFS）找出与该虚拟机直接或间接有流量关系的虚拟机节点，并将这些节点合为一组，并设置访问标记。当所有虚拟机都被访问过之后，结束算法。

### 2.3 虚拟机组装箱算法函数

函数声明如下：

void pack\_group(int part\_no);

该函数的作用是找到能装下该组虚拟机并且VOL指标最大（即最适合装该组虚拟机）的物理机，并将该组内的全部虚拟机放入找到的物理机。如不存在任意一台物理机能够放置该组虚拟机，则调用2.4中将要提到的分组切割算法，将当前组分割为两个组，然后对分割结果中的第一组重复进行以上操作，直到被切割的足够小并能放入某一物理机为止。

注：VOL指标（part\_no为该虚拟机组的ID，i为物理机ID），只考虑前四个基本负载维度。



### 2.4 分组切割算法函数

函数声明如下：

double group\_cut(int part\_no);

该函数使用了计算无向图最小割的Stoer-Wagner算法。该算法的基本思想是：

1.最小割值记为min（其初始值为INF，无穷大），并固定一个顶点P。

2.将记录每个点的总流量的数组wage[i]初始值设置为0，然后找出和P点相连的所有顶点，将其对应的wage值增加P到该顶点的流量值。

3.找出当前wage值最大且未访问过的的顶点U，然后找出和U点相连的所有顶点，将其对应的wage值增加U到该顶点的流量值。重复该步骤直到该组中所有的顶点都已经被访问过。在这个过程中，需要记录最后访问的两个顶点S与T，并将T从该组中切除。

4.计算最后访问的顶点T的切割值（即与此顶点相连的所有边权和，即流量总和wage[T]），若该流量总和比min小，则更新min，同时将目前已从该组中切除的所有顶点加入新的分组。

5.合并最后访问的两个顶点S,T为一个顶点（这样的操作并不会影响最终的切割结果）。

6.转到2，S与T合并N-1次后结束

7.min即为所求的最小割，即分割出的两个组的最小东西流量。

基于这种算法，我们将一个虚拟机组切割成了具有最小东西流量的两个组。

### 2.5 输出函数

函数声明如下：

void output\_parts(VMGPtr ppart);

该函数将初始放置的结果进行输出。输出以分组算法和分组切割算法所得到的虚拟机组为单位来输出。首先输出总分组数，然后输出每一组所包含的虚拟机编号、流量关系、负载以及该组所在的物理机。

## 处理流程

1. 从文件“xml2.xml”中读取物理机和虚拟机的信息。
2. 从文件“xml22.xml”中读取虚拟机之间的流量关系。
3. 调用邻接表生成函数，根据是否有流量关系构建一个虚拟机邻接表。
4. 调用分组算法函数，根据有无流量关系将虚拟机分为若干个组。
5. 调用虚拟机组装箱算法函数放置虚拟机组，若虚拟机组无法放下则调用分组切割算法函数，并对新分组重复以上操作，直到该虚拟机组可以被放置到某一物理机。
6. 调用输出函数，按组输出虚拟机放置结果。