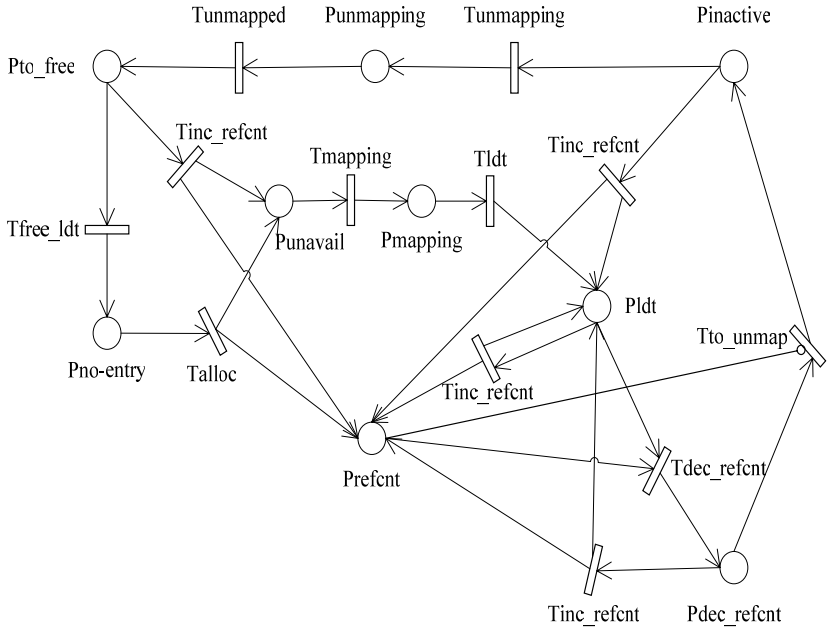


图 4.3 宿主权限项的状态转换图

图 4.3 中与图 4.2 中相同位置名的位置，具有相同的含义。除此之外的位置的含义为：

表 4.4 宿主权限项状态转换图的位置含义

| 位置名 | 含义 |
|---|---|
| Prefcnt | 该项的引用计数值 |
| Pdec_refcnt | 引用计数被减少 |
| Pinactive | 引用计数为 0，对应元数据不活跃 |
| Punmapping | 正在与 BS 通信，要求 BS 解除该元数据的分布管理 |
| Pto_free | 等待释放内存。此时已经从哈希链表摘下，不会与新分配内存结构冲突 |

同样，图 4.2 中没有包括的转换变迁的含义为：

表 4.5 宿主权限项状态转换图的变迁含义

| 变迁名 | 含义 |
|---|---|
| Tdec_refcnt | 请求使用完毕该项，减少引用计数 |
| Tinc_refcnt | 请求需要使用该项，增加引用计数 |
| Tto_unmap | 引用计数为 0，可以请求 BS 释放分布管理 |
| Tunmapping | 与 BS 通信，释放该项的分布管理 |
| Tunmapped | 释放请求结束，可以释放其内存结构 |

系统同时存在多个请求使用分布信息项，这表现在表 4.1 的 me_refcnt。Me_refcnt>0 表示有请求使用该项，元数据处于活跃状态，此时它链接在相应的哈希链表和 entry_in_use 链表。位置 Prefcnt 记录引用计数，其标记数目等同于 me_refcnt 的值。Talloc 将它初始化为 1，因为此时必然有请求正在使用该项。当分布信息项处在 Pldt 时，其他请求可以使用该项。本请求也可能使用完毕，所以增加或者减少引用计数都是可以进行的。只有在引用计数为 0（位置 Prefcnt 没有标记）时，该信息变成不活跃的，这表现在位置 Prefcnt 和变迁 Tto_unmap 之间的禁止弧。当 me_refcnt 为 0 时，它将通过其 me_list

连接到 entry_unused_hosted 链表，但此时仍通过 me_hash 链接到相应的哈希链表。在与 BS 真正开始解除不活跃元数据分布之前，可能有请求访问该元数据，其再度活跃。表现在位置 Pinactive 和位置 Pldt、Prefcnt 间的变迁 Tinc_refcnt。

当宿主权限项不活跃时，MS 主动要求 BS 释放对该项的分布管理。它首先需要通知 BS 解除对它的管理。完成后，首先将不活跃项从链表 entry_unused_hosted 摘下，然后再释放内存结构。最后，Tfree_ldt 进行时，才将分布信息项从哈希链表摘下，释放其内存结构。

### 4.4.2 非宿主权限项的状态转换

非宿主权限项的状态转换 Petri Net 描述如图 4.4 所示，起始状态是位置 Pno-entry 具有 1 个标记。相对于宿主权限项的状态转换图而言，变迁 Tfree_rdt 的作用和 Tfree_ldt 等同，非宿主权限项通过变迁 Trevalidate 驱动分布信息的刷新。

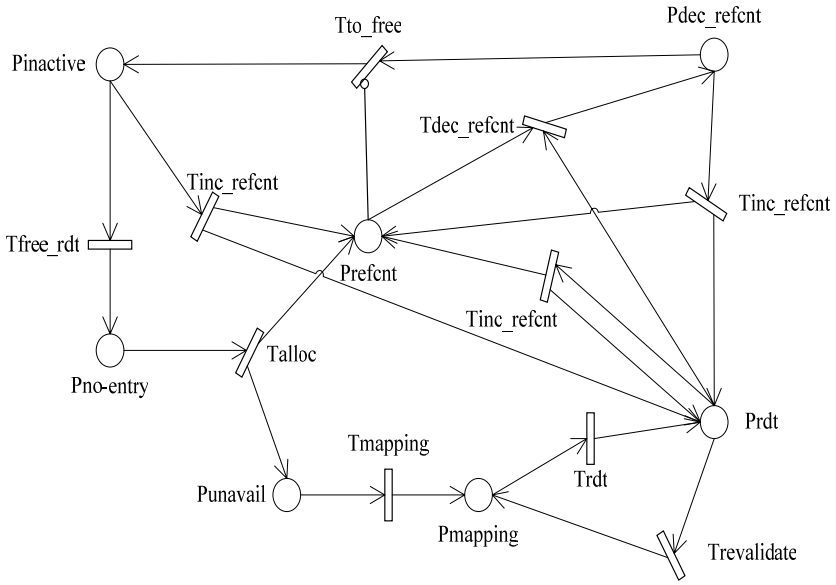


图 4.4 非宿主权限项的状态转换图

表 4.6 是图 4.2 和图 4.3 都没有的变迁的含义。

表 4.6 非宿主权限项状态转换图的变迁含义

| 变迁名 | 含义 |
|---|---|
| Trevalidate | 需要刷新非宿主权限项的信息 |

相对于宿主权限项而言，非宿主权限项的释放不需要请求 BS 解除请求分布。但是，由于元数据分布信息的动态变化，RDT 中的信息可能过时，需要向 BS 请求刷新信息，表现为位置 Prdt 和位置 Pmapping 间的变迁 Trevalidate。但是，刷新结果不会将该项转成宿主权限项，它获得的仅是新的宿主 MS 的信息。

## 4.5 状态转换图的活跃性分析

Petri Net 的活跃性分析可以通过求解其可达树，再合并可达树中的相同节点获得可达图完成。如果可达图中所有节点都有进和出的弧，从任意节点开始，经过一定变迁序列达到其他节点，Petri Net 是活跃的。Petri Net 可达树求解算法[Murata1989][LinChuang2001]如表 4.7 所示。

表 4.7 Petri Net 的可达树求解算法

1）根节点 r 由 M0 标注。

2）一个标注 M 的结点 x 是一个叶结点，当且仅当不存在 t∈T，t 在 M 是可实施的或者在从 r 到 x 的路径上存在一个结点 y≠x，但节点 y 也是由 M 标注的。

3）如果一个标注 M 的结点 x 不是一个叶结点，那么对于所有 t∈T 使得在 M 下可实施的 t 实施而产生一个新的结点 y，且在从 x 到 y 新产生的弧上标注 t。y 结点标注的标识 $M'$ 可由 $M''$ 来计算，$M''$ 满足于 $M[t>M''$，即 $\forall s \in S : M''(s) = M(s) - W(s,t) + W(t,s)$。$M'$ 的计算可区别为两种情况：

①在从 r 到 y 的路径上，如果存在标注 $M'''$ 的结点 z≠y 且 $\forall s \in S$：$M''(s) \geq M'''(s)$，那么：

$$M'(s) = \begin{cases} M'(s), \text{如果} M'(s) = M'''(s) \\ \omega, \text{其他} \end{cases}$$

②其他情况，$M' = M''$ 。

其中的 M 是所有位置的标记序列。W（x,y）是位置 x 到变迁 y，或者变迁 x 到位置 y 的弧权。M（s）表示在标记 M 中位置 s 的标记数量。

### 4.5.1 宿主权限项

根据图 4.3 的宿主权限项的状态转换图、表 4.7 的 Petri Net 可达树求解算法，宿主权限项的状态转化可达树如图 4.5 所示。

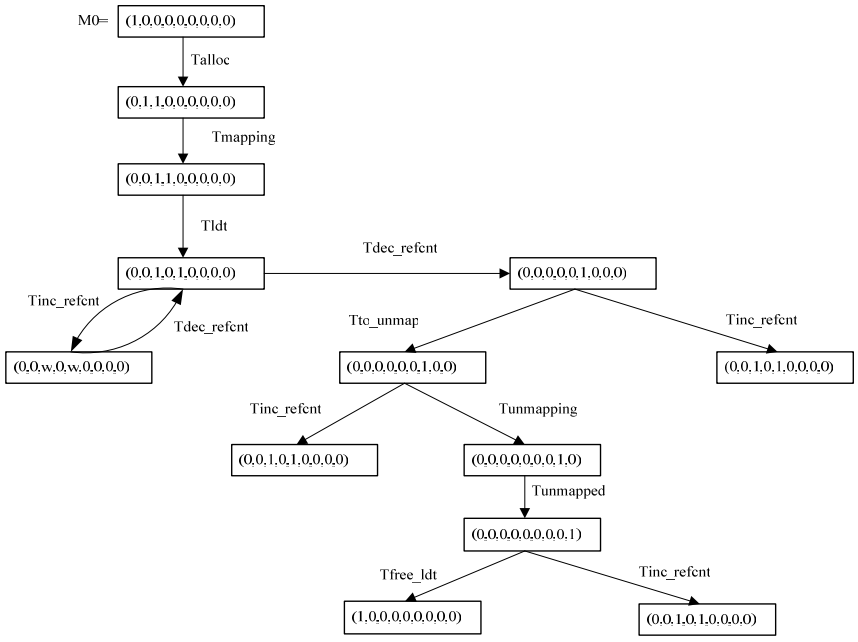


图 4.5 宿主权限项的状态转换可达树

图4.5中的节点是各个位置的标记数量的序列，其顺序为（Pno-entry, Punavail, Prefcnt, Pmapping, Pldt, Pdec_refcnt, Pinactive, Punmapping, Pto_free）。

将图4.5中相同节点合并，得到图4.6的宿主权限项的状态转换可达图。从图4.6可以看出，从宿主权限项状态转换可达图的任何节点出发，都可以通过一定变迁序列到达其它节点，状态转换图是活跃的。

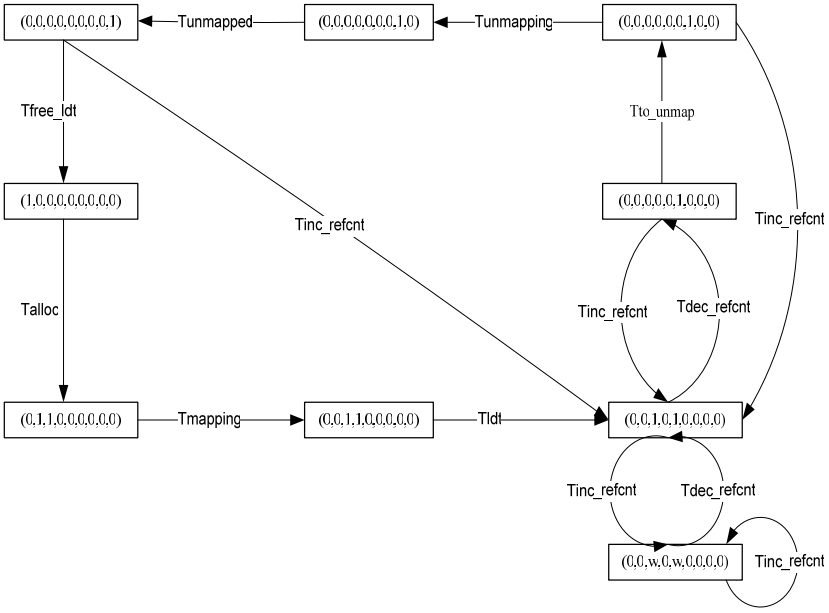


图4.6 宿主权限项的状态转换可达图

### 4.5.2 非宿主权限项

同样，根据图4.4和表4.7，非宿主权限项的状态转换可达树如图4.7所示。

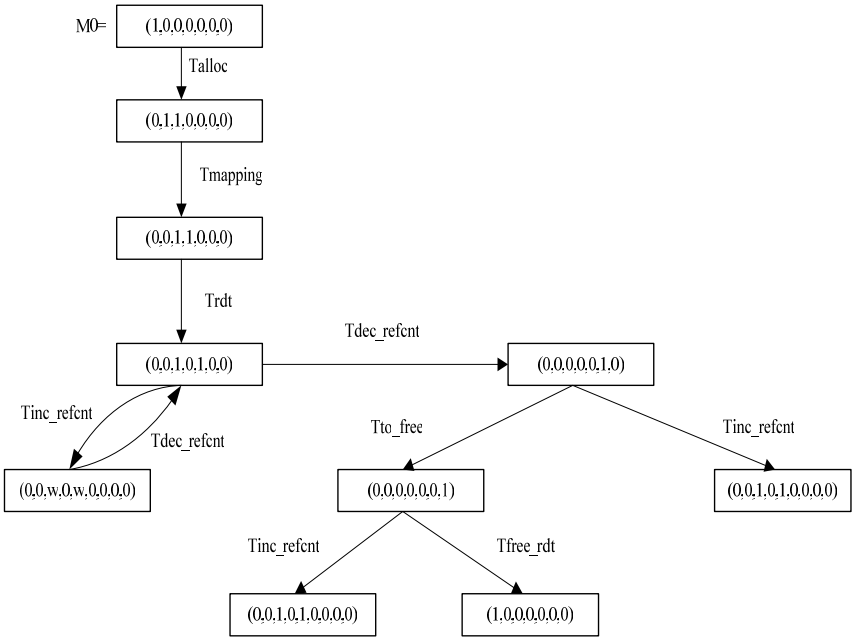


图4.7 非宿主权限项的状态转换可达树

节点包含各个位置的标记数量，其顺序为(Pno-entry, Punavail, Prefcnt, Pmapping, Prdt, Pdec_refcnt, Pinactive)。

合并图 4.7 中的相同节点，得到图 4.8 的非宿主权限项的状态转换可达图。同样，非宿主权限项的状态转换是活跃的，不会出现死锁情况。

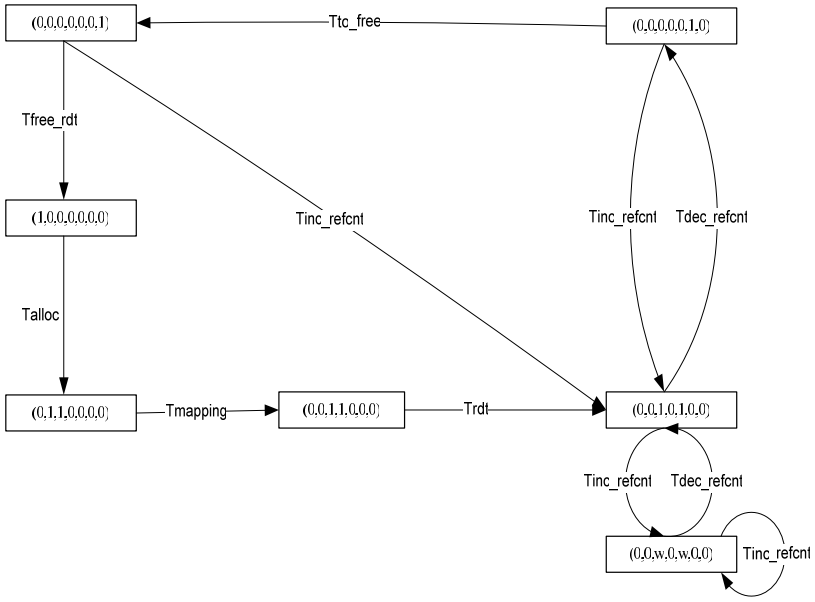


图 4.8 非宿主权限项的状态转换可达图

## 4.6 不活跃元数据信息的替换策略

不活跃元数据项的替换策略，将影响元数据分布信息缓存的命中率，从而影响 BS 的负载和元数据请求处理的时间延迟。

综合考虑内存开销、BS 负载等多种因素，元数据分布信息缓存中的不活跃元数据分布信息项的释放策略，由释放时间间隔、缓存的内存开销和不活跃分布信息项的数量等三个因素决定。每隔固定时间间隔，缓存总的数目、或者不活跃分布信息项的数目超过预定值时，缓存管理将根据 LRU 算法释放不活跃的分布信息项。

由于用户的元数据访问表现出很高的局部性，具有宿主权限的元数据被再次访问的概率，要远大于非宿主权限的元数据。并且，由于宿主权限的释放需要与 BS 的通信，时延较大。所以，在进行不活跃元数据项释放时，首先针对非宿主权限项进行。只有在系统内存压力非常大时，才选择宿主权限项进行释放。

结合内存压力和固定时间间隔策略，不活跃元数据的替换策略能够较好地保证系统对内存资源的消耗。非宿主权限项先于宿主权限项的释放策略，能够在保证缓存命中率的同时，降低缓存的内存开销。