中南财经政法大学大学生创新训练项目申报书

二、项目论证（1.立项背景，国内外研究现状述评 2.项目研究意义。3．项目研究的主要内容、基本思路、研究方法、重点难点、基本观点。4．项目实施目标。5.前期相关研究成果和主要参考文献。限3000字左右。）

1.立项背景：

近年来固态硬盘（SSD）已经成为固态技术中的领先技术，最常见的SSD都是基于NAND FLASH芯片设计的。虽然SSD具有抗震无机械延迟等特点，但是由于闪存介质的物理特性，SSD的I/O性能具有不对称性。一般地，SSD具有较高的读性能和较差的随机写性能，所以随机写性能是SSD性能的瓶颈。此外，NAND型闪存具有写前擦除的特点，即闪存芯片更新，由于每块闪存芯片的擦除次数是有限的，所以擦除次数决定了闪存的寿命。也就是说，发生在SSD上的写操作不仅影响系统性能而且影响SSD的使用寿命。因此过多的更新必然会带来频繁的擦除操作，从而会明显地加大SSD的写延迟和降低SSD的使用寿命。相反地，传统磁盘HDD具有对称的I/O性能，写操作也没有如上所述的限制。虽然已有许多针对SSD磨损均衡的研究工作，但是引入HDD来弥补SSD的写性能缺陷仍是颇具吸引力的。另一方面，尽管HDD的读性能比SSD差不少，但是它在价格上具有明显的优势，所以目前主流的存储系统仍然是以基于HDD的为主。尽管如此，引入少量的SSD到现有的存储系统中是十分有价值的，这不仅能获得明显的系统性能提升，而且在价格成本上也是很有吸引力。综上，利用SSD和HDD各自优点来设计混合存储系统是存储系统未来的研究热点。

但是，在混合存储系统中，由于SSD价格的高昂以及擦除次数的有限性，现今混合存储系统仍旧以HDD作为主要的存储介质，因此页面分类算法和迁移策略是非常重要的部分。

表1. Intel系列型号参考价格

型号		参考价（美元）	每GB成本（美元）
Intel X25-E	32GB	460	14.38
Intel X25-E	64GB	900	14.06
Intel 710 SSD	100GB	679	6.79
Intel 710 SSD	200GB	1299	6.50
Intel 710 SSD	300GB	1999	6.63

目前已有一些研究者对该问题进行了研究。特别地，文献[13]基于页面的I/O统计信息提出了一种迁移模型，该模型根据页面的I/O统计数据分别计算该页面在SSD和

HDD 上的 I/O 代价，通过计算结果的分析来对页面进行分类，读倾向的页面会被分配到 SSD 上存储，写倾向的页面则保留在 HDD 上存储。很明显，该模型既能利用 SSD 优秀的读性能，又能借助 HDD 来减少 SSD 上的写擦除操作。但是，该模型中的 I/O 统计涵盖了该页面自生成以来的所有访问请求，这带来的累积效应将会使该模型在访问负载变化时不能快速反应。另一方面，因为闪存不对称的 I/O 特性，HOT/COLD 的概念很早就被研究者引入到关于闪存的研究中。一般地，被频繁访问的页面我们认为是 HOT 页面，非频繁访问的页面被认为是 COLD 页面。基本的研究思想是对存储在闪存上的频繁更新的 HOT 页面进行缓冲以实现批量更新，从而减少闪存上的物理写。已有大量的研究工作基于该思想来对闪存存储进行改进。近年来，已有一些研究者尝试将该思想应用到混合存储领域，如文献[6]、[7]他们的研究工作显示 HOT/COLD 的方法是值得考虑的。由于 HOT 页面有更高访问概率，那么当这些页面存储在 SSD 上时，因为 SSD 优秀的读性能，这就能带来明显的性能提升，混合存储系统一定会带来额外的迁移代价，例如文献[8]所阐述的。当 SSD 和 HDD 之间的迁移操作增加时，缓冲中的一些干净页面会发生迁移，这些迁移会带来额外的写操作，从而增大整个系统的 I/O 延迟，所以研究中迁移代价是必须考虑的因素，同时对页面准确分类也能有效减少迁移操作。另外，因为 SSD 高昂的价格，混合存储系统的性价比也是我们必须考察的因素，因此混合存储系统仍有许多值得研究的热点。

2.研究意义：

目前，在信息存储领域，SSD 硬盘存储以其高速度，轻质量在存储领域掀起了技术革命。据调查，数据存储速度方面，在同样配置的电脑下，当按下电脑的电源开关时，搭载 SSD 固态硬盘的电脑从开机到出现桌面一共只用了 10 秒多，而搭载传统硬盘的电脑总共用时 20 秒多。进入系统后不管是运行程序或是打开文件也可以明显感觉到固态硬盘占有绝对优势。功耗方面，同样是因为少了很多机械部件，所以 SSD 的功耗上也要优于传统硬盘。重量方面，SSD 在重量方面更轻，与常规 2.5 英寸硬盘相比，重量轻 70-50 克。噪音上，由于 SSD 属于无机械部件及闪存芯片，所以具有了发热量小、散热快等特点，而且没有机械马达和风扇，工作噪音值为 0 分贝，这使得传统硬盘就要逊色很多。

而且由于储存介质和工作原理的不同，SSD 固态硬盘比传统旋转式硬盘消耗少 80%

电量，整体运行温度低，减轻系统的负担，延长系统生命周期，有助于环保节能，符合绿色环保的发展理念。

但是，不可否认的是，SSD 硬盘的价格高，读取速度快但是写入速度远远低于读取速度，且写入次数受限制。因此，目前的 SSD，HDD 混合存储系统兼具了两者的优点，具有很广阔的应用市场和前景。该存储模式将 SSD 和 HDD 相结合，取出读写较为频繁的，即热点区域的内容，放在读取速度较快，但容量有限的 SSD 中，而将一些读取不频繁，或者写入操作较多的内容放在 HDD 中，这样既保证了运行速度，又保证了 SSD 具有较长的使用寿命。研究表明，SSD 和 HDD 之间的相互迁移会产生迁移代价，因此，准确对页面进行“冷区”“热区”的判断能够有效的减少迁移操作，提升系统性能。而目前的研究对于混合存储模型中的热区判断，页面替换算法以及页面迁移等方法的针对性还明显不足，因此混合存储模型的算法研究仍然具有很高的研究价值和探索空间。

3．项目研究的主要内容、基本思路、研究方法、重点难点、基本观点

3.1 主要内容

我们项目的主要研究对象是一种基于热区跟踪替换算法的 SSD-HDD 高效储存模型，用于提升传统 SSD-HDD 混合存储系统的性能。我们设计并提出了热区跟踪算法，页面热度状态转换模型及页面定位倾向模型等。除了验证这些模型的有效性，还需要设计合适的调度算法来使得这些模型相互协调工作，研究的内容主要包括：

(1)根据热区的特性设计了热区跟踪(Hot Zone Tracing，HZT) 算法，用于跟踪标识热区的热度值。

(2)根据热区的热度值，结合改进的 I/O 代价计算模型和 HOT/COLD 概念，提出页面热度状态转换模型。

(3) 提出页面定位倾向模型，该模型根据页面的 I/O 统计，通过计算对比页面在 SSD 和 HDD 上 I/O 代价，得到页面定位倾向的计算值，从而确定页面的存储位置。

(4) 根据页面的热度状态转换模型和页面定位倾向模型，提出混合储存模型。

3.2 基本思路

我们的模型主要包括：热区跟踪算法，页面热度转换模型，页面定位倾向模型和页面迁移算法等，项目的总体框架如图 1 所示：

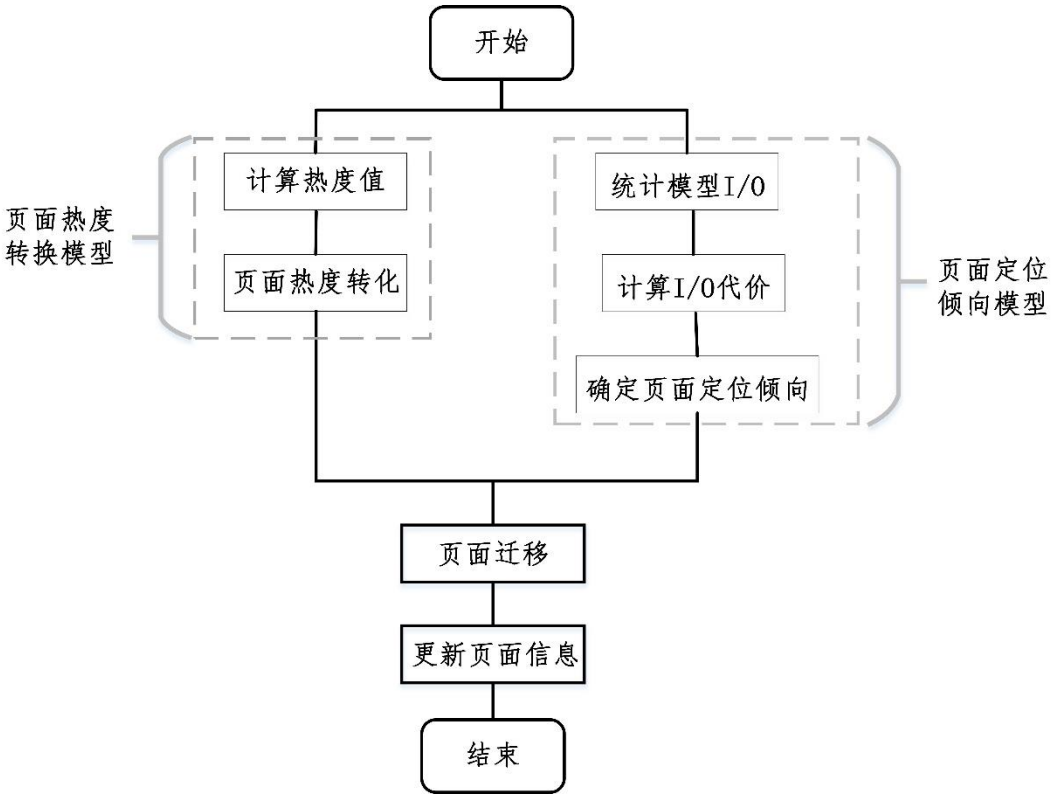

图 1. 模型总体框架图

本研究中主要包括三大部分：页面热度状态转换，确定页面定位倾向，结合热度值和页面定位倾向进行页面迁移及页面信息更新。整个模型概述如图 1 所示。

（1）首先，我们利用类似于 Linux 内核的页高速缓存机制的基树(Radix Tree)的数据结构来管理区访问历史，基于此设计了热区跟踪算法，得到页面的热度值。

（2）利用页面热度转换模型和页面当前热度值，进行页面状态转化。

（3）根据提出的一种改进性迁移模型，该模型根据页面的 I/O 统计数据以及转换后的页面热度值分别计算该页面在 SSD 和 HDD 上的 I/O 代价，通过计算结果的分析来对页面进行分类，读倾向的页面会被分配到 SSD 上存储，写倾向的页面则保留在 HDD 上存储。

最后在单独的 HDD 设备或 SSD 设备上分别运行两个数据集，获得单个设备的运行

性能。然后在混合存储设备上实现本文的模型和文献[7]提出的混合模型，并分别运行两个数据集，以此来验证该模型的有效性。

3.3 研究方法

3.3.1 热区跟踪算法的设计

根据热区的定义，在一段时间内热区的访问次数要高于冷区的访问次数。我们可以简单地把热度值当成一个区内的访问次数。但这样做有两个问题: 一是不能有效识别当前时间段的热区; 二是查找热区效率低。

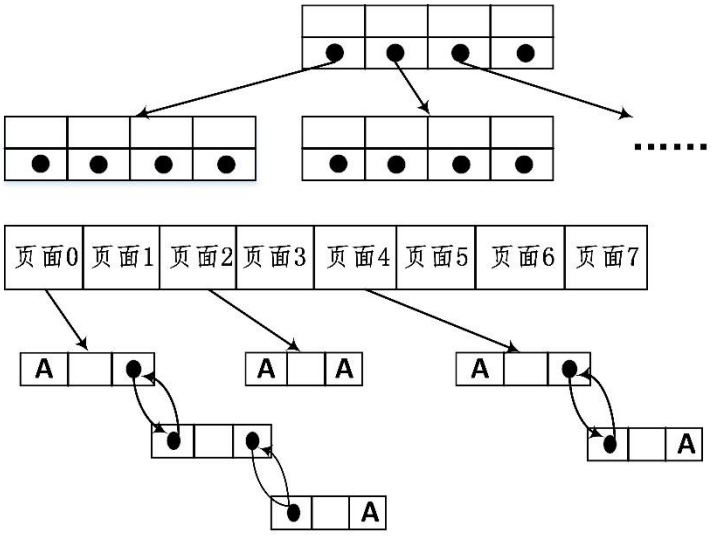

图 2. 页面的管理结构图

为提高内存利用效率，我们使用了类似于 Linux 内核的页高速缓存机制的基树(Radix Tree)的数据结构来管理区访问历史。基树的叶子节点指向每个页面，每个叶子节点记录页面的热度值(访问计数)、页面数据块计数、节点访问计数。 非叶子节点记录其所有子节点的汇总信息。每个节点包含的子节点的个数称为基，基树的高度由基树的基和页面的个数决定。基数的叶子节点包含一个指向该页面数据块 LRU 链表的指针。如图 2 所示，给出了一个基为 4 的 2 层基树。当一个页面内的数据块被访问时，该页面的热度值增加，其所有父节点的热度值也将增加。使用基树数据结构的优点是查找速度快、占用空间小。设基树的基为 R，层数为 L，区的个数为 A，则$L=\log_R A$。热度值需要反映区内数据块的近期访问和远期访问情况。设 $h_{i,j}$ 表示第 i 层第 j 个节点的热度值。热度值的计算方法：

1) 每次对区内数据块的访问，该区的热度值加 1，该节点所有父节点的热度值也相应加 1:

$$h_{R-i,\frac{n}{R^i}} = h_{R-i,\frac{n}{R^i}} + 1(1 \le i \le L) \qquad (1)$$

2) 当热度值溢出时，本节点所有热度值右移 1 位(除以 2):

$$h_{i,j} = \frac{1}{2} h_{i,j} (\frac{i}{R} \le j \le (\frac{i}{R} + R - 1)) \qquad (2)$$

3) 当节点访问计数大于一定阈值时，节点所有热度值右移 1 位，同时访问计数清“0”，热度值得计算与公式(2)相同。随着访问量增加，热度值可能溢出(尤其是上层节点)，节点内所有热度值右移 1 位不会改变本节点内的热度排序，所以不需要修改同层不同节点和不同层节点。当节点访问计数大于一定阈值时节点内所有区的热值右移 1 位可以隔离历史访问信息。热值右移 1 位相当于使这一个时刻之前的热度值的权重降为 1/2，使近期的访问量在热度值中占有更高的权重。当一个区内的数据块长时间没有访问时，其热度值将不断变小直至变为 0，成为冷区。

3.3.2 页面热度转换模型

基于热区跟踪算法和 COLD/HOT 概念,本文提出页面热度转换模型。为了准确 地区分页面热度，我们在传统的 COLD/HOT 的概念之上，引入“warm”状态，基本方法是设计设置一个热度阈值 heat。在一次单位时间的物理访问之后，当热度值的增加量大于 0，页面从 cold 页面转化成 warm 页面或者从 warm 页面转化成 hot 页面。相反热度值的增加量小于 0，则 warm 页面转化成 cold 页面或者从 hot 页面转化成 warm 页面。页面热度转换如图所示，该机制能够有效识别出偶尔发热的 cold 页面，以减少不必要的迁移操作。

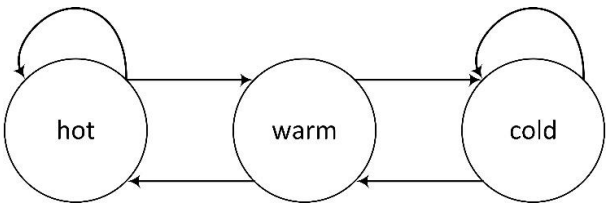

图 3. 页面热度状态转换图

本文中的页面热度转换模块是基于单位时间内页面热度的增加量来实现的。每个页面的当前的热度值将会被系统记录下来。在经过单位时间 T 之后，页面的热度值将会

发生改变，增量值即为 addedHeat。页面的热度状态将会随着 addedHeat 的取值发生改变，具体的热度状态变化如表 1 所示。图 3 的状态改变将会根据表 1 所示的实现。

表 2. 热度更新表

	停留状态		
	Cold	Warm	Hot
addedheat	Warm	Hot	Hot
addedheat	Cold	Cold	Warm

结合图 3 和表 2，我们提出热度转换算法，如下所示。

算法 1：热度转换算法

```
1. pg.changeFlag←false
2. if  addedHeat＞0  then
3.     if  pg.state＝cold  then
4.         pg.state←warm;
5.     else if  pg.state＝warm  then
6.             pg.state←hot;
7.             pg.changeFlag←true;
8.     end if
9. else if  addedHeat＜0  then
10.    if  pg.state＝hot  then
11.        pg.state←warm;
12.    else if  pg.state＝warm then
13.            pg.state←cold;
14.            pg.changeFlag←true;
15.    end if
16. end if
```

在算法 1 中，参数 changeFlag 是状态转换的标志，它的默认值为 false，页面的状态从 warm 状态转换为 cold 或者 hot 状态时，该参数的值会变为 true。这个参数是为后面 I/O 统计重置的标志。

3.3.3 页面定位倾向模型

因为闪存具有不对称的 I/O 特性，不仅页面的访问热度而且页面的 I/O 负载也会影响页面的定位。因此我们引入一种概念：“页面定位倾向”，它表示页面存储在 SSD 或 HDD 上的倾向性。根据页面的 I/O 统计，通过计算对比页面在 SSD 和 HDD 上的 I/O 代价，我们可以得到页面定位倾向的计算值从而确定页面的存储位置。我们的模型是基于文献[13]做出改进。

本文基于有限的 SSD 容量，提出一种准确，敏感的页面定位倾向计算模型。本文还引入时间衰减因子来改进方案。首先引入了页面 I/O 代价计算方法。对于一个数据页面，它的物理访问和逻辑访问都列入统计，并且一个物理访问操作对 I/O 代价的计算比逻辑访问操作的影响要大。在算法 2 中，ssdCost 和 hddCost 分别代表页面在 SSD 和 HDD 上访问的代价。对于一个页面，假定逻辑访问有一定的概率变成物理访问。为了得到更准确的结果，本文提出的概率并不是一个常量，每个页面的概率都以公式 $1-l/N$ 来计算，l 表示单个页面的逻辑访问操作次数，N 表示页面所有的访问操作次数，故不同页面有不同的概率值，这反映了不同页面各自的访问负载。在算法 2 中，本文利用这个概率值来衡量逻辑访问操作对于 I/O 计算代价的影响。

算法 2：页面倾向计算算法

```
trendCalculate(Page pg)
1. flag←true //the flag of whether trend is reduced;
2. q← 1 −(pg.lr＋lg.lw) / pg.totalaccess;
3.                 //calculate the I/O cost on SSD and HDD
4. ssdcost←(pg. lr · q＋pg. pr) · rs+ (pg.lw · q＋pg.pw) · ws;
5. hddcost←(pg.lr · q＋pg.pr) · rh+ (pg.lw · q＋pg.pw) · wh;
6.                 //calculate the trend and its reduce
7. if   pg.state＝hot   then
8.          if   pg.changeFlag＝true   then
9.                pg.trend←(ssdcost－hddcost)＋pg.pretrend;
10.         pg.pretrend←pg.trend · β;
11.              pg.changeFlag←false；flag←true;
12.   else
13.              pg.trend←(ssdcost－hddcost)＋pg.pretrend;
14.             flag←false;
15.     end if
16. else if   pg.state＝warm   then
17.   pg.trend←(ssdcost－hddcost)＋pg.pretrend;
18.   flag←false;
19. else if   pg.state＝cold   then
20.   if   pg.changeFlag＝true   then
21.               pg.trend←(ssdcost －hddcost)＋pg.pretrend;
22.              pg.pretrend←pg.trend · β;
23.              pg.changeFlag←false；  flag←true;
24.   else
```

```
25.            interval←pg.cold2－pg.cold1;
26.            if  interval／ssdSize＜hddSize／ssdSize  then
27.                pg.trend←(ssdcost－hddcost)＋pg.pretrend;
28.                flag←false;
29.            else
30.            pg.pretrend←pg.trend · β · (ssdSize／interval) ;
31.                pg.trend←(ssdcost －hddcost) ;
32.                flag←true;
33.            end if
34.        end if
35. end if
36. //reset the counter
37. if  flag＝true  then
38.    pg.lr←0；pg.lw←0；pg.pr←0；pg.pw←0;
39. end if
40. if  ｜pg.trend｜＞ws＋wh and pg.trend＜0  then
41.    return SSD-trend;
42. else if  ｜pg.trend｜＞ws＋wk and pg.trend＞0
43.    return HDD-trend;
44. else
45.        return pg.placement;
46. end if
```

分析以上算法可以看出，如 40 和 42 行所述，*trend*<0 表示页面存储在 SSD 上的 I/O 代价比较小，*trend*>0 表示页面存储在 HDD 上的 I/O 代价比较小。*trend* 表示页面迁移操作带来的收益。所以迁移操作只有在收益 *trend* 大于迁移代价时才会发生。

如算法 2 所示，页面定位倾向模型需要为每个页面维护 5 个计数器：*lr* 和 *lw* 统计页面的逻辑读写操作；*pr* 和 *pw* 统计也买呢的物理读写操作；*totalaccess* 统计所有的访问操作；r_s，r_h 分别代表 SSD 和 HDD 上的物理读代价；w_s，w_h 分别代表 SSD 和 HDD 上的物理写代价。两个基本参数 *ssdcost* 和 *hddcost* 分别表示页面在 SSD 和 HDD 上访问的 I/O 代价，参数 *trend* 表示页面的定位倾向，每当页面从缓冲区被置换时，该参数将会被更新。*trend*<0 表示 *ssdcost* 比 *hddcost* 小，意味着页面应该被存储在 SSD 上，反之应该被存储在 HDD 上。参数 β 表示衰减因子，它是一个小于 1 的常量（实验中被设置为 0.1），页面定位倾向在三种状态下的计算由算法 2 的 6~35 行所示，在两种情况下，*trend* 会按照因子 β 减少。

（1）当页面热度状态由 warm 状态转换为 hot 或者 cold 状态，这表明页面的当前访问负载可能发生变化，读写倾向性发生改变。故先前的定位倾向和统计计算影响变小，所以定位倾向参数 *trend* 按因子 β 减小，且访问统计被重置。

（2）HDD 的容量大小设为 *hddSize*，假定每个页面在 *hddSize* 次的访问中至少会被访问一次，那 么如果页面的两次相邻访问的间隔超过 *hddSize*，该页面被认为 *toocold*，并且它的定位倾向发生衰减，如算法 2 中 29～33 行所示。在这种情况下，衰减因子 β 的影响会增大，它的值会被减小 *hddSize/interval* 倍，这就使之前的负载对定位倾向的影响减小至几乎 0 。

如果页面一直保持在 hot 或者 cold 状态，页面的定位倾向就会正常更新。

3.3.4 混合存储模型

结合上述的算法和模型，这里将给出整个混合存储模型（算法 3，4）。如算法 4 所示，页面必须在热度状态为 warm 或 hot，且定位倾向为 *SSD-trend* 时，该页面才能被迁移到 SSD 上存储。而从 SSD 迁移到 HDD 上储存的迁移触发条件则不相同。如算法 4 的 8~14 行所示，一旦 SSD 的定位倾向为 *HDD-trend,*则该页面被迁移到 HDD。

算法 3：混合存储算法

```
accessPage(Page pg)
1. pg.totalaccess＋＋;
2. if   pg is found in buffer   then
3.          move pg to the head of buffer LRU;
4.          if   access is read quest   then
5.          pg.lr＋＋;
6.     else if   access is write quest   then
7.               pg.lw＋＋;
8.          end if
9. else
10.         if   buffer is full   then
11.              evictPage();
12.         end if
13.         fetch pg from disk;
14.         updateState(pg);
15.         add pg to the head of buffer LRU;
16.         if   access is read quest   then
17.              pg.pr＋＋;
18.         else   access is write quest   then
```

```
19.             pg.pw＋＋;
20.         end if
21. end if
```

算法 4：页面迁移算法

```
1. Page pg;
2. pg←tail of LRU;
3. place←trendCalculate(pg);
4. if  pg is on HDD  then
5.         if  pg.state is hot or warm and place＝SSD-trend
6.             migratePage(pg);
7.         end if
8. else if  pg is on SSD  then
9.         if  place = SSD-trend  then
10.            migratePage(pg);
11.        else if  pg is cold and SSD write performance is beyond HDD  then
12.            migratePage(pg);
13.        end if
14. end if
15. if  pg is dirty  then
16.        write pg to disk of pg.placement;
17. end if
```

如算法 3，4 所示，当页面被读入缓存区时，页面的访问时间信息和热度状态信息将会被更新。页面从缓存区置换时，算法 2 被调用计算页面的定位倾向，从而确定页面的存储位。算法 4 的第 6，10，12 行表示，迁移被触发时，页面的相关信息将会被更新，同时页面被置为 dirty 被写入硬盘。至此整个混合存储系统中各个算法及模型的功能原理及它们的协同运作被介绍完毕。

3.4 重难点

从主要研究方法的介绍中可知，本文提出了热区跟踪算法，页面热度状态转换模型，页面定位倾向模型等多个方法模型，旨在对传统的 SSD-HDD 混合存储系统做出改进。我们需要给出相应的模型对存储系统的子系统进行性能的提升，并且给出合适的算法使得各种模型和算法能够协同工作。在这个过程中有以下难点：

（1）如何合理地量化每个页面的被访问的热度。

（2）一次热访问可能导致 cold 页面被迁移到 SSD 上之后，页面迅速“变冷“又被

迁移到 HDD 上，如何解决这种情况。

（3）如何根据页面的 I/O 统计及历史定位倾向判断页面当前的存储倾向。

（4）结合页面热度状态和页面定位倾向，如何实现页面的迁移操作。

3.5 基本观点

我们的研究的目的是针对 SSD 和 HDD 混合存储问题，提出一种基于热区跟踪算法的 SSD 和 HDD 混合存储模型。该模型把 SSD 和 HDD 作为同级的存储设备，结合数据页的访问次数以及访问热度实现对页面的准确分类和分配，即将读倾向负载的 hot 页面分配到 SSD 存储，写倾向负载的页面或者 cold 页面分配到 HDD 存储，从而利用 SSD 和 HDD 不对称的 I/O 特性来降低系统总的 I/O 延迟。本项目将通过更多的实际数据对所提出的模型及相关算法进行检验。

4.实现目标

（1）设计热区跟踪算法，实现对页面热度量化的目标。

（2）构建页面热度状态转换模型及页面定位倾向模型，确定页面的存储倾向。

（3）提出一种基于热区跟踪算法的 SSD-HDD 高效混合存储模型。

（4）根据实验数据，通过与传统模型的对比实验，验证提出的模型的性能优点及其有效性。

5.前期相关研究成果及主要参考文献

本文分别在 TPC-C 和 OLTP 数据集上实现本文的模型，并与文献［13］提出的混合模型进行对比. TPC-C 数据集通过修改开源数据库 PostgreSQL7.4.29 获得，在其缓冲区管理器中增加操作记录模块，每当缓冲区管理器接到操作请求时，数据库便记录该操作请求和页面号，并提供输出操作记录至文件的函数．在 PostgreSQL 数据库中使用 BenchmarkSQL 软件模拟运行 TPC-C 测试，数据量设定为 1GB，并调用函数输出 TPC-C 操作记录文件. OLTP 数据集则是由 Gerhard Weikum 提供的真实的银行系统交易处理数据集。实验系统可分为存储管理器、缓冲管理器和混合迁移模型 3 部分，并利用上面提到的 B+-树完成对存储数据的索引. 存储管理器包括空闲空间管理器和非空闲空间管理器．实验代码使用 C++实现，并运行在 Debian GNU/Linux 2.6.21 操作系统上，页面大小为 4K B．缓冲区大小设定 4M B，并带有 LRU 管理器. 实验运行在硬盘仿真系统上。

实验结果如下图所示：

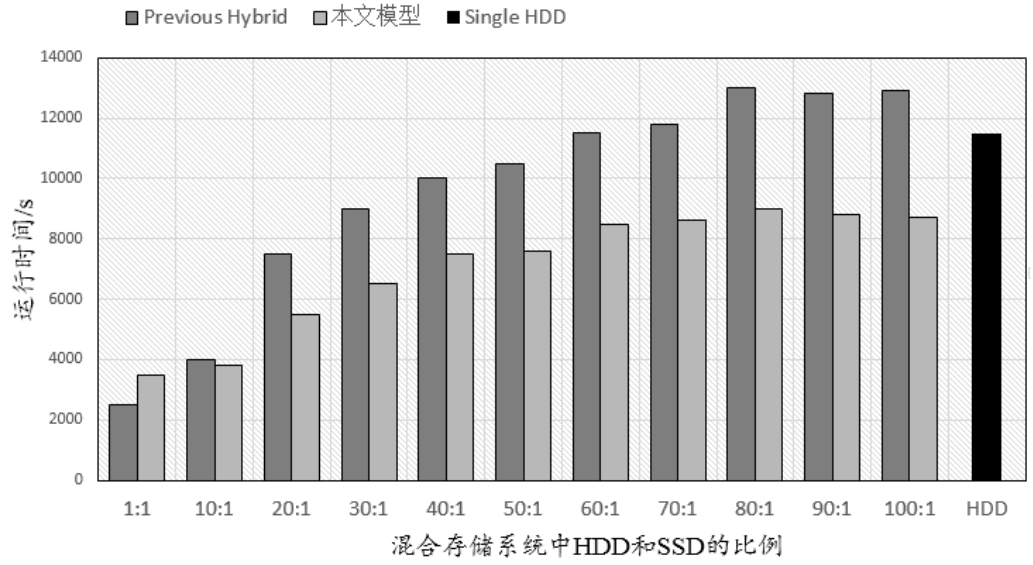

图 4. OLTP 在不同比例的 SAMSUNG-32G 和 HHD 混合存储设备上的运行时间

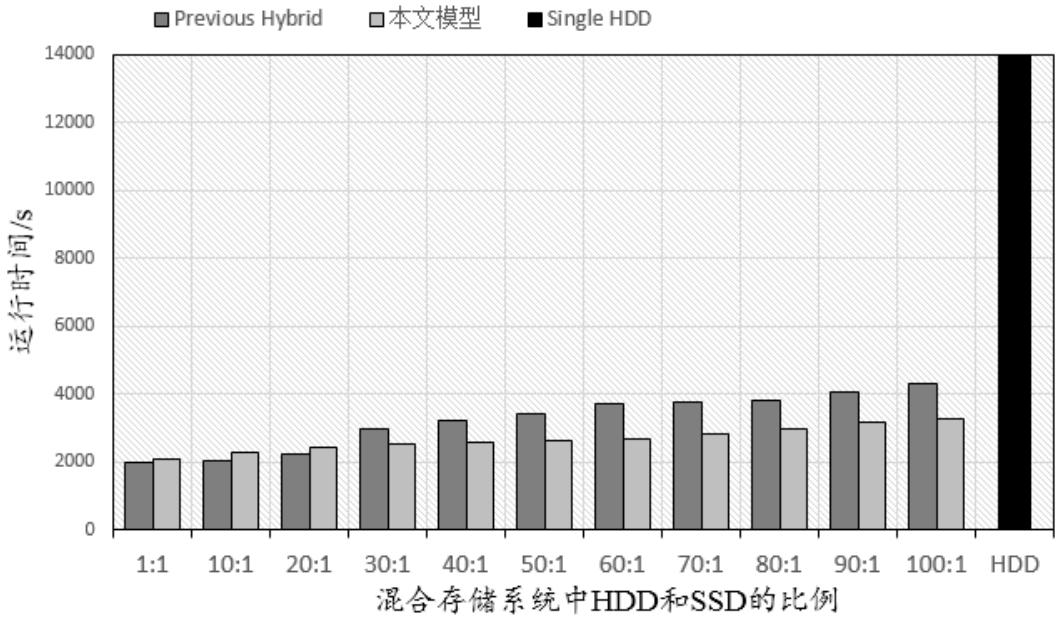

图 5. TPC-C 在不同比例的 SAMSUNG-32G 和 HHD 混合存储设备上的运行时间

主要参考文献：

[1] Kang S H,Koo D H,Kang W H, et al. A case for flash memory ssd in hadoop

applications. International Journal of Control and Automation. 2013

[2] He Yongqiang,Lee Rubao,Huai Yin,Shao Zheng,Jain Namit,Zhang Xiaodong,Xu Zhiwei.RCFile:A fast and space-efficient data placement structure in MapReduce-based warehouse systems.Proceedings of the 24th International Conference on Data Engineering (ICDE.11). 2011

[3] Lee S W, Park D J, Chung T S, Lee D H, Park S, Song H J. A log buffer-based flash translation layer using fully-asso-ciative sector translation. ACM Transactions on Embedded Computing Systems (TECS), 2007,6(3):18

[4] Park D, Debnath B,Du D. CFTL: A convertible flash translation layer adaptive to data access patterns//Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. New York, USA, 2010:365-366

[5] Ma D,Feng J, Li G. Lazyftl: A page-level flash translation layer optimized for nand flash memory//Proceedings of the 2011 International Confernence on Management of Data(SIGMOD11). Athens,Greece, 2011:1-121

[6] Cheong S K, Jeong J J,Jeong Y W, Ko D S,Lee Y H. Research on the I/O performance advancement of a low speed HDD using DDR-SSD. Communications in Computer and Information Science, 2011,184(2):508-513

[7] Canim M, Mihaila G A, Bhattacharjee B, Ross K A, Lang C A. SSD bufferpool extensions for database systems. Proceedings of the VLDB Endowment, 2010,3(1-2):1435-1446

[8] Soundararaian G, Prabhakaran V, Balakrishnan M, Wobber T. Extending SSD lifetimes with disk-based write caches//Proceedings of the 8th USENIX Conference on File and Storage Technologies(FAST'10). Berkeley, USA,2010:8-8

[9] Corporation I. Understanding the flash translation layer (FTL) specification [EB/OL]. http://www.embeddedfreebsd.org/Documents/Intel-FTL.pdf, 2011.

[10] chen F, Koufaty D A, Zhang X. Hystor: making the best use of solid state drives in high performance storage systems [A]. Proceedings of the International Conference on Supercomputing [C], New York, USA: ACM, 2011:22-32

[11] Blktrance [EB/OL]. http://linux.die.net/man/8/blktrance, 2011.

[12] Apache Hadoop. http://hadoop.apache.org . 2015

[13]Koltsidas,Viglsa S D.Flashing up the storage layer.Proceedings of the VLDB Endowment,2008,1(1):514-512

三、项目特色与创新点（200 字以内）

本文提出一种基于热区跟踪算法的 SSD-HDD 高效混合存储模型，该项目的特色和创新点在于：

（1）设计出热区跟踪算法，成功实现了对页面热度的量化。

（2）在传统的 cold/hot 概念中，引入 warm 概念，以避免 cold 页面骤然地变为 hot 页面，从而减少不必要的迁移操作，提升系统的性能 。

（3）将对页面的逻辑访问操作转换成物理访问操作的概念设置为变量，即每个页面的概率都以公式 1-*l*/*N* 来计算，*l* 表示单个页面的逻辑访问操作次数，*N* 表示页面所有的访问操作次数，提升了页面倾向计算的准确度。

（4）混合存储模型结合了页面热度，页面定位倾向，迁移收益及迁移代价等多个因素，实现了对页面的精准分类，可有效降低页面迁移代价，是系统性能获得显著提升。