# 一种时间敏感的 SSD 和 HDD 高效混合存储模型

杨濮源 金培权 岳丽华

(中国科学技术大学计算机科学与技术学院 合肥 230027)

**摘 要** 基于闪存的固态硬盘(Solid State Driver,SSD)已成为目前广泛使用的一种持久存储设备.但是由于闪存不对称的 I/O 特性以及价格因素,SSD 还不能完全取代传统硬盘(Hard Disk Driver,HDD).因此,由 SSD 和 HDD 组成的混合存储系统逐步成为目前研究的重点.文中针对 SSD 和 HDD 混合存储问题,提出了一个时间敏感的混合存储模型用来有效地利用 SSD.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.作者分别在基于高端 SSD 和中端 SSD 的混合存储系统上实现了提出的混合存储模型,并进行了性能评测.实验结果显示,作者提出的模型可以实现对数据页更准确的分类,可以有效地降低页面迁移代价,在较少的 SSD 存储条件下取得了显著的性能提升.

**关键词** 混合存储系统;闪存;固态硬盘;时间敏感;性价比

**中图法分类号** TP311 **DOI 号**: 10.3724/SP.J.1016.2012.02294

# A Time-Sensitive and Efficient Hybrid Storage Model Involving SSD and HDD

YANG Pu-Yuan JIN Pei-Quan YUE Li-Hua

(*School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027)

**Abstract** Solid state driver (SSD) based on flash memory has become a persistent storage device widely used. But it can not take place the magnetic disks absolutely due to the imbalance I/O character and the high price of flash memory. The hybrid storage system consisted of SSD and HDD is gradually becoming the research issue. For the hybrid storage with SSD and HDD, this paper proposes a time-sensitive hybrid storage model to efficiently take use of SSD. This model takes SSD and HDD at the same level of the memory hierarchy. According to the page access counter and access temperature, the model achieves accurate page classification and placement which places the hot and read-intensive pages SSD and places the cold or write-intensive pages to HDD. So the model takes advantage of the asymmetric I/O properties of HDD and SSD to reduce total I/O latency of system. We separately realizes the model on the hybrid storage based on high-end SSD and middle-end SSD and finish the performance evaluation. The experimental result shows that our model can achieve more accurate page classification, reduce the migration cost and get obvious performance improvement with fewer SSD.

**Keywords** hybrid storage system; flash; SSD; time-sensitive; performance price ratio

---

收稿日期:2012-06-05;最终修改稿收到日期:2012-08-21.本课题得到国家自然科学基金(60833005,61073039)资助.**杨濮源**,男,1985 年生,博士研究生,主要研究方向为闪存数据库、混合存储管理.E-mail: yangpuy@mail.ustc.edu.cn. **金培权**(通信作者),男,1975 年生,博士,副教授,主要研究方向为移动对象数据库、时空数据库、闪存数据库、互联网信息抽取与搜索等.E-mail: jpq@ustc.edu.cn. **岳丽华**,女,1952 年生,教授,博士生导师,主要研究领域为数据库系统及其应用、信息集成、实时数据库、闪存数据管理应用.

# 1 引　言

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,即闪存芯片不能原位更新,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[1-3],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[4],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的.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.

在混合存储系统中,页面分类算法和迁移策略是非常重要的部分.目前已有一些研究者对该问题进行了研究[5-6].特别地,文献[7]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[8-10].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[11-12],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.

已有的研究工作表明,混合存储系统一定会带来额外的迁移代价[13].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.

基于一种改进的 I/O 代价计算模型和 hot/cold 概念,本文提出了一种混合存储模型,主要贡献如下:

(1)页面热度状态转换.为了准确地区分页面热度,我们在传统的 hot/cold 概念中引入了“warm”状态.基本的方法是设置一个时间阈值 $t$,一个页面的两次连续的物理访问之间的时间间隔如果大于该阈值,则该页面会由 hot 变为 warm 或者由 warm 变为 cold;相反时间间隔如果小于该阈值,则页面会由 cold 变为 warm 或者有 warm 变为 hot.页面的热度状态转换如图 1 所示,该机制能够有效识别出偶尔变热的 cold 页面从而减少不必要的迁移操作.

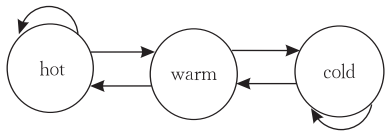



图 1　页面热度状态转换

(2)时间衰减因子.基于前文提到的迁移模型[7],我们在 I/O 统计和 I/O 代价计算方法中引入了衰减因子.如果一个页面的热度状态在 hot/cold 之间转换或者物理访问之间的时间间隔过长,那么我们将会根据该衰减因子降低 I/O 代价计算的值对页面分类的影响并且重置 I/O 统计值.因此,在本文的方法中,当一个页面访问稀疏时,它的 I/O 统计对页面当前的分类判断影响很微弱,这就避免了统计的累积效应对当前负载分析的不利影响.

(3)性价比分析.因为 SSD 高昂的价格,我们的目标是只将有限的 SSD 引入基于磁盘的存储体系来使存储系统获得明显的性能提升.为了充分利用有限的 SSD,对页面的分类就必须准确.在局部时间

内，我们选择读倾向的页面存储在 SSD 上，写倾向的页面存储在 HDD 上. 另一方面，由于 SSD 的存储空间有限，我们需要考虑 SSD 上的置换操作，因为闪存的擦除操作是以一个 block 为单位，所以我们设计一个 block 级的 LRU 队列来管理 SSD 上的页面，当 SSD 达到存储上限时，LRU 队列尾部的 block 将会被置换出迁移到 HDD 上. 为了衡量系统的性价比，我们列出如下公式

$$ratio=\frac{Improvement\text{-}of\text{-}Hybrid\text{-}Storage}{Price\text{-}of\text{-}SSD\text{-}used} \quad (1)$$

在第 3 节，我们会将 SSD 和 HDD 以各种容量比例进行组合并且分析各个组合的性价比，从而获得混合存储系统合适的组合方式.

本文第 2 节给出混合存储模型，并详细讨论其中的迁移模型和页面分类方法；第 3 节给出该混合模型的实现和相关的实验结果；最后第 4 节对本文的工作进行总结.

## 2 SSD-HDD 混合存储模型

### 2.1 混合存储的体系结构

本文提出的混合存储模型架构如图 2 所示，图中箭头表示数据页面的流向，最小的正方形代表页面，较大的长方形表示数据块(block)，根据闪存的物理结构，我们将 64 个数据页面组成一个 block. 在缓冲区中，页面以一个 LRU 列表进行管理，刚进入缓冲区的页面置于 LRU 的队头，置换页面则从队尾选择. 当一个页面被读取到缓冲区中时，页面首先会被热度状态转换模块更新热度状态和访问时间信息，这部分将在 2.2 节论述. 当一个页面从缓冲区置换出来时，页面分类模块(placement trend calculation and decay)会计算根据该页面的 I/O 统计信息

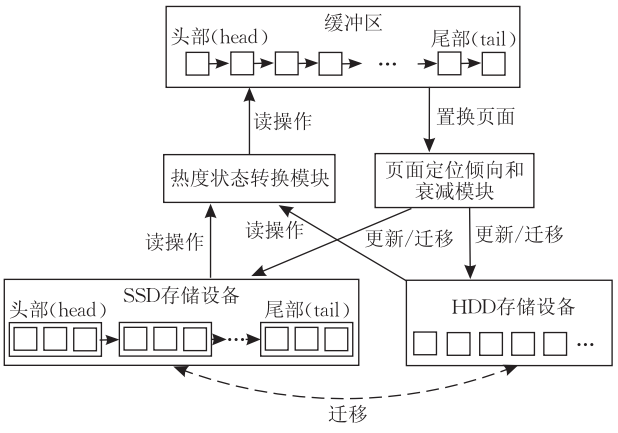



图 2 混合存储系统体系架构

和热度状态计算它的存储倾向，这部分将在 2.3 节论述. 页面分类模块是以热度状态转换模块为基础的.

另外，本文为 SSD 和 HDD 分别设计了存储管理模块，它们的地址空间也是分别独立的，任何时候，一个数据页面只能存在于其中一个设备上. 比如，如果一个页面分配在 HDD 上存储，那么它只有一个 HDD 上的地址，当它迁移到 SSD 上时，它将被分配一个 SSD 上的地址，同时 HDD 上的地址将被回收. 因为 SSD 上的存储空间有限，SSD 的地址空间可能将会用尽. 由于 SSD 上的页面是以 block 为单位以 LRU 列表管理，所以当 SSD“满”时，LRU 列表尾部的 block 将会被迁移到 HDD. 为了方便快速查询，我们需要记录页面的存储位置并分别为 SSD 和 HDD 上的页面建立索引，这将在 3.1 节论述. 当查询一个页面时，总是优先在 SSD 的索引上查找，因为该索引较小并且 SSD 的读速度较快. 如果该页面在 SSD 索引上未找到，再到 HDD 索引上查找.

基于上述的存储管理模块，迁移操作不仅将改变页面的存储位置，而且将更新 SSD 和 HDD 的索引信息，因此我们的迁移操作如算法 1 所示.

**算法 1.** 迁移操作.

```
migratePage(Page pg)
1. if pg.placement=SSD then
2.     pg.placement←HDD;
3.     Remove pg from SSD index and insert into HDD
           index;
4. else if pg.placement=HDD then
5.     pg.placement←SSD;
6.     remove pg from HDD index and insert into SSD
           index;
7. end if
8. set pg is dirty.
```

### 2.2 热度状态转换模块

已有的关于闪存存储的研究工作常常利用 hot/cold 的概念来对页面进行分类. 将传统的 hot/cold 概念引入混合存储系统中，因为 SSD 随机写的延迟较大，SSD 和 HDD 之间的迁移代价是必须考虑的问题. 比如，假设一个页面一直是 cold 状态，某次访问是一次热访问(即访问的时间间隔很小)，这次访问使系统误判它为 hot 页面，从而带来一次迁移操作. 但是紧接着该页面又回到 cold 状态，这可能又会带来一次迁移操作，所以不准确的页面分类会使迁移操作大大增加. 为避免这种情况，我们提出

了“warm”的概念来补充传统的 hot/cold 思想，这样页面就无法在 hot 和 cold 状态之间直接转换，引入“warm”状态对混合模型的作用将在 3.6 节实验讨论.

本文的热度转换模块是基于页面的物理访问时间的. 每个页面的物理访问时间都会被系统记录，参数 $t_2$ 表示页面的当前物理访问时间，参数 $t_1$ 表示页面的上一次物理访问时间. 如果两次连续的物理访问间隔大于阈值 $t$，则当前访问为 cold 访问，相反则为 hot 访问. 本文中的访问时间都是以页面的访问计数表示，例如从系统启动到当前为止有 $N$ 次页面访问请求，那么当前时间为 $N$.

对于页面的 hot/cold 状态，参数 $cold_1$ 和 $cold_2$ 分别记录上次 cold 访问时间和当前 cold 访问时间，参数 $hot_1$ 和 $hot_2$ 类似. 系统中 SSD 和 HDD 的容量以可以存储的页面数目衡量，参数 $ssdSize$ 表示 SSD 的存储容量，参数 $hddSize$ 表示 HDD 的存储容量. 每当页面发生物理访问时，它的访问时间都会被更新，如表 1 所示. 如果 $hot_2 > hot_1$，那么该页面处在 hot 状态，并且 $hot_1$ 是该当前 hot 状态的第 1 次访问时间. cold 状态的时间记录与之类似.

**表 1　物理访问时间更新**

| $t_1 \leftarrow t_2, t_2 \leftarrow current\ time$ 操作后的状态 | |
|---|---|
| hot 访问 | cold 访问 |
| $hot_2 = current\ time$ | $hot_1 = current\ time$ |
| $cold_1 = current\ time$ | $cold_2 = current\ time$ |

物理访问时间更新之后，页面的热度状态会根据它的访问时间进行更新. 图 1 所示的状态转换将根据表 2 所示的实现. 综合考虑表 1 和表 2，我们可以发现页面能够在 hot 和 cold 状态停留，但是不能保持在 warm 状态. 一个 cold(hot)页面只有经过两次连续的 hot 访问才能转换为 hot(cold)页面，因此偶尔的一次访问负载变化不会带来热度状态的变化，这样就能避免一些不必要的迁移操作.

**表 2　热度更新**

| | 停留状态 | | |
|---|---|---|---|
| | cold | warm | hot |
| $hot_2 > hot_1$ | warm | hot | hot |
| $cold_2 > cold_1$ | cold | cold | warm |

结合表 1 和表 2，我们提出了热度转换算法，如下所示.

**算法 2.**　热度转换.

```
updateState(Page pg)
1.  pg.t1←pg.t2;
2.  pg.t2←current time;
3.  interval←pg.t2−pg.t1;
4.  //update page access time
5.  if interval≤ssdSize then
6.      pg.hot2←current time;
7.      pg.cold1←current time;
8.  else if interval >ssdSize then
9.      pg.hot1←current time;
10.     pg.cold2←current time;
11. end if
12. //update page state;
13. pg.changeFlag←false
14. if pg.hot2>pg.hot1 then
15.     if pg.state=cold then
16.         pg.state←warm;
17.     else if pg.state=warm then
18.         pg.state←warm;
19.         pg.changeFlag←true;
20.     end if
21. else if pg.hot2<pg.hot1 then
22.     if pg.state=hot then
23.         pg.state←warm;
24.     else if pg.state=warm then
25.         pg.state←cold;
26.         pg.changeFlag←true;
27.     end if
28. end if
```

在算法 2 中，参数 $changeFlag$ 是状态转换的标志，它的默认值是 false，当页面状态由 warm 变为 cold 或者 hot 时，该参数的值变为 true，这个参数是为后文中 I/O 统计重置的标志. 1～11 行描述了页面访问时间的更新，14～28 行描述了热度状态的更新，实现了图 1 所示的状态转移.

基于本节的热度转移模块，本文接下来将讨论页面的定位倾向(SSD 或 HDD). 这一节主要从时间角度讨论了页面的负载，下一节将从空间角度讨论该问题.

### 2.3　页面定位倾向和衰减

因为闪存不对称的 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. 在此我们引入一个概念：“页面定位倾向”，它表示页面存储在 SSD 或者 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 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.

类似于文献[7,13]所讨论的，页面定位倾向是根据页面负载的 I/O 代价来计算的. 文献[7]提出

的迁移模型基于页面的访问统计,有效地计算出了页面的 I/O 代价和读写倾向.但是因为该模型中的计算是基于页面分配以来的 I/O 统计的,所以它具有明显的累积效应.这意味着以前长时间的访问负载和当前的访问负载对当前的页面定位有着相同的影响.由于累积效应,页面以前的访问负载积累的 I/O 统计值将会很大,从而使该模型对当前的访问负载变化不够敏感,这将会导致不准确的页面定位,特别是当 SSD 容量有限时,会造成 SSD 存储空间的浪费.

因为 SSD 高昂的价格,本文假设设计环境是基于有限容量的 SSD 的.因此我们需要构建一个准确的、敏感的页面定位倾向计算模型.以文献[7]的模型为基础,本文引入时间衰减因子来改进方案.首先本文引入了页面 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被列入统计,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有更大的影响.如算法 3 所示,*ssdCost* 和 *hddCost* 分别表示页面在 SSD 和 HDD 上访问的 I/O 代价.

对于一个页面,假定一个逻辑访问操作有一定概率变为物理访问.为了得到更准确的结果,这个概率和文献[7]模型中所提的概率并不一样.文献[7]中的概率是一个常量,它的计算式为 $1-b/n$,$b$ 表示缓冲区的容量,$n$ 表示页面的总数目,文献[7]利用该常量作为逻辑操作转变为物理操作的概率.对应地,本文所提的概率并不是一个常量,由算法 3 的第 2 行所示,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计数,$N$ 表示页面所有访问操作的计数.所以不同的页面拥有不同的概率值,这也反映了不同页面各自的访问负载.如算法 3 所示,本文利用这个概率值来衡量逻辑访问操作对 I/O 代价计算的影响.

**算法 3.** 倾向计算.

```
trendCalculate(Page pg)
1.  flag←true //the flag of whether trend is reduced;
2.  q←1−(pg.lr+pg.lw)/pg.totalaccess;
3.            //calculate the I/O cost on SSD and HDD
4.  ssdcost←(pg.lr·q+pg.pr)·r_s+
5.            (pg.lw·q+pg.pw)·w_s;
6.  hddcost←(pg.lr·q+pg.pr)·r_h+
7.            (pg.lw·q+pg.pw)·w_h;
8.            //calculate the trend and its reduce
9.  if pg.state=hot then
10.   if pg.changeFlag=true then
11.     pg.trend←(ssdcost−hddcost)+
12.                 pg.pretrend;
13.     pg.pretrend←pg.trend·β;
14.     pg.changeFlag←false; flag←true;
15.   else
16.     pg.trend←(ssdcost−hddcost)+
17.                 pg.pretrend;
18.     flag←false;
19.   endif
20. else if pg.state=warm then
21.   pg.trend←(ssdcost−hddcost)+
22.              pg.pretrend;
23.   flag←false;
24. else if pg.state=cold then
25.   if pg.changeFlag=true then
26.     pg.trend←(ssdcost−hddcost)+
27.                 pg.pretrend;
28.     pg.pretrend←pg.trend·β;
29.     pg.changeFlag←false; flag←true;
30.   else
31.     interval←pg.cold_2−pg.cold_1;
32.     if interval/ssdSize<hddSize/ssdSize then
33.       pg.trend←(ssdcost−hddcost)+
34.                   pg.pretrend;
35.       flag←false;
36.     else
37.       pg.pretrend←pg.trend·β·
38.                   (ssdSize/interval);
39.       pg.trend←(ssdcost−hddcost);
40.       flag←true
41.     end if
42.   end if
43. end if
44. //reset the counter
45. if flag=true then
46.   pg.lr←0; pg.lw←0; pg.pr←0; pg.pw←0;
47. end if
48. if |pg.trend|>w_s+w_h and pg.trend<0 then
49.   return SSD-trend;
50. else if |pg.trend|>w_s+w_h and pg.trend>0
51.   return HDD-trend;
52. else
53.   return pg.placement;
54. end if
```

每当页面被访问,它的访问计数就会被更新.对于每个迁移的页面,我们假定它还会迁移回来.比如,当 HDD 上的一个页面被迁移到 SSD 上,这会有一次 SSD 上的写代价,然后该页面可能会被迁移回来,这又会带来一次 HDD 上的写代价.所以一个页面迁移代价是 SSD 和 HDD 上写代价的和.如算法 3 的 48 和 50 行所示,$trend<0$ 表示页面存储在

SSD 上 I/O 代价较小，$trend>0$ 表示页面存储在 HDD 上 I/O 代价较小. 参数 $trend$ 表示迁移操作带来的收益. 所以迁移操作只能在收益 $trend$ 超过迁移代价时发生.

如算法 3 所示，本文的模型为每个页面维护了 5 个计数器：$lr$ 和 $lw$ 统计页面的逻辑读写操作；$pr$ 和 $pw$ 统计页面的物理读写操作；$totalaccess$ 统计所有的访问操作；$r_s$，$r_h$ 分别表示 SSD 和 HDD 上的物理读代价；$w_s$，$w_h$ 分别表示 SSD 和 HDD 上的物理写代价. 两个基本的参数 $ssdcost$ 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，每当页面从缓冲区置换时该参数被更新. $trend<0$ 表示 $ssdcost$ 比 $hddcost$ 小，这意味着页面应该存储在 SSD 上，反之，页面就应该存储在 HDD 上. 当考虑到迁移代价时，实际的定位倾向判断会更复杂一些，正如算法 5 所示. 另一参数 $pretrend$ 存储了衰减后的定位倾向值，当 $pretrend$ 被更新时，访问计数会被重置. 参数 $\beta$ 表示衰减因子，它是一个小于 1 的常量（实验中设置为 0.1），页面定位倾向在 3 种状态下的计算由算法 3 中 8～43 行所示，在两种情况下，$trend$ 会按因子 $\beta$ 减小.

(1) 当页面热度状态从 warm 变为 hot 或者 cold 时，这暗示着页面的当前访问负载可能发生变化，读写倾向性发生改变，所以该页面先前的定位倾向和访问统计对当前的计算不应该有太大的影响. 定位倾向参数 $trend$ 按因子 $\beta$ 减小，并且访问统计重置.

(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 too cold，并且它的定位倾向发生衰减，如算法 3 中 36～41 行所示. 在这种情况下，衰减因子 $\beta$ 的影响会增大，它的值会被减小 $ssdSize/interval$ 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 cold 或者 hot 的状态，页面的定位倾向就会正常更新，如算法 3 中所示. 基于上述讨论，我们将在下一小节提出本文的整个模型.

### 2.4　混合存储模型

这一节我们将讨论整个的混合存储模型（算法 4、5）. 如算法 5 中所示，页面必须在热度状态为 warm 或者 hot，并且定位倾向为 $SSD\text{-}trend$ 时，该页面才能被迁移到 SSD 上存储. 而从 SSD 到 HDD 的迁移触发条件则不相同，如算法 5 的 8～14 行所示，一旦 SSD 页面的定位倾向为 $HDD\text{-}trend$，则该页面被迁移到 HDD，此外对某些写操作性能不输于读操作的高端 SSD 而言，热度状态的变化也是迁移操作的触发条件，当页面变为 cold 时迁移发生. 也就是说，本文的系统是兼容高端 SSD 和中端 SSD 的，在第 4 节的实验部分有更详细的讨论.

**算法 4.**　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 5.**　evictPage( ).

```
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.plcaement;
17. end if
```

如算法 4、5 所示，当页面被读到缓冲区中时，该页面的访问时间信息和热度状态都会被更新. 页面

从缓冲区置换时，算法 3 被调用更新页面的定位倾向值，从而确定页面的存储位置. 算法 5 中的第 6，10，12 行所示，迁移将会被触发，页面的相关信息也会被更新，同时页面被置为 dirty 以写入对应地硬盘.

## 3 实 验

本文分别在 TPC-C 和 OLTP 数据集上实现本文的模型，并与文献[7]提出的混合模型进行对比.

### 3.1 混合存储模型

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，并利用第 2.1 节中提到的 $B^+$-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 4KB. 缓冲区大小设定 4MB，并带有 LRU 管理器. 实验运行在硬盘仿真系统上，仿真实现包括常规 HDD(RU-HDD)、中端 SSD(SAMSUNG-32G SSD)和高端 SSD(Intel X25E SSD)在内的 3 种存储设备①，并使用工具软件 uFlip[14] 来测量硬盘仿真系统的 I/O 性能参数. 本文实现了第 2 节中提出的方法和文献[7]提出的混合模型，并分别运行在两种不同类型的混合存储设备上，包括 RU-HDD 与 SAMSUNG-32 G SSD 的混合存储以及 RU-HDD 与 Intel X25E SSD 的混合存储，最后对比分析本文的模型和以前的混合模型的实验结果.

存储设备性能：表 3 描述了 3 种存储设备 RU-HDD、SAMSUNG-32G SSD 和 Intel X25E SSD 的实际读写延迟时间和价格参数. 表 4 分别给出了 RU-HDD 与 SAMSUNG-32G SSD 以及 RU-HDD 与 Intel X25E SSD 的读写代价性能对比，并以最小代价的倍数值进行归一化描述. 其中，读写延迟时间越大，读写速度越慢，读写代价值越大. 例如，SAMSUNG-32G SSD 读页面的速度是 RU-HDD 的 107 倍，而 RU-HDD 写页面的速度大约是 SAMSUNG-32G SSD 的两倍；Intel X25E SSD 写页面速度比读页面速度大约快 3 倍，且其写页面速度是 RU-HDD 读页面速度的 108 倍，是 RU-HDD 写页面速度的 297 倍.

**表 3 SSD 和 HDD 的读写延迟和价格参数**

| 设备类型 | 读延迟/μs | 写延迟/μs | 价格/($·GB$^{-1}$) |
|---|---|---|---|
| RU-HDD | 19917 | 7257 | 0.125 |
| SAMSUNG-32G | 187 | 9619 | 16.000 |
| Intel X25E | 199 | 67 | 13.000 |

**表 4 SSD 和 HDD 的读写代价**

| 设备类型 | 读代价单元数 | 写代价单元数 |
|---|---|---|
| RU-HDD | 107 | 39 |
| SAMSUN-32G | 1 | 51 |
| RU-HDD | 297 | 108 |
| Intel X25E | 3 | 1 |

因此，第 2.3 节中提到的 I/O 代价参数可使用表 4 中的倍数值描述，例如对 RU-HDD 与 SAMSUNG SSD 的混合存储类型有：$r_s=1$，$r_h=107$，$w_s=51$，$w_h=39$. 实验记录了硬盘仿真系统中的读写操作数目，并结合表 3 中的读写延迟时间来仿真计算总的运行时间.

数据集的类型：本文使用两种类型的数据集：TPC-C 和 OLTP. TPC-C 数据集通过修改开源数据库 PostgreSQL 7.4.29 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. 表 5 分别描述了 TPC-C 和 OLTP 数据集中的不同操作类型的统计数值. 可以看到，两个数据集的读写操作数都分别约占总操作数的 77%和 23%. 下面分别介绍本文的模型在这两种数据集上运行的性能情况分析.

**表 5 TPC-C和OLTP数据集的读写操作数及页面数目统计值**

| 数据集类型 | 读操作次数 | 写操作次数 | 总操作次数 | 页面数目 |
|---|---|---|---|---|
| TPC-C | 1351144 (77.1%) | 400738 (22.9%) | 1751882 | 33892 |
| OLTP | 470678 (77.5%) | 136713 (22.5%) | 607391 | 51878 |

### 3.2 实验性能分析

实验中，首先在单独的 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. 为了方便描述，后文使用 Previous Hybrid 代指文献[7]提出的混合模型.

图 3 描述了 TPC-C 和 OLTP 数据集在单个 HDD 设备或 SSD 设备上的总运行时间. 可以看到，RU-HDD 上运行性能最差，IntelX25E 性能最好，SAMSUNG-32G 其次，它们之间的性能差距是由

① http://uflip.inria.fr/uFLIP/results/

SSD 相对于 HDD 的优点带来的. 正如前文所提到的，中端 SSD(SAMSUNG-32G SSD)与高端 SSD(Intel X25E SSD)的性能差距较大，因此需要分别在 RU-HDD 与 SAMSUNG-32 G SSD 混合存储设备、RU-HDD 与 Intel X25E SSD 混合存储设备上测试本文模型.

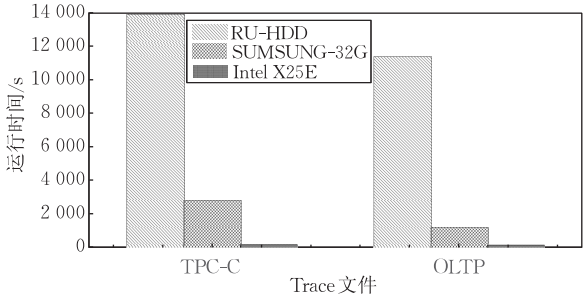
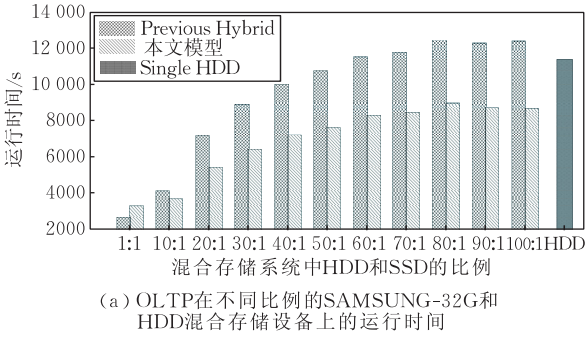



图 3　数据集在单个设备上的运行时间

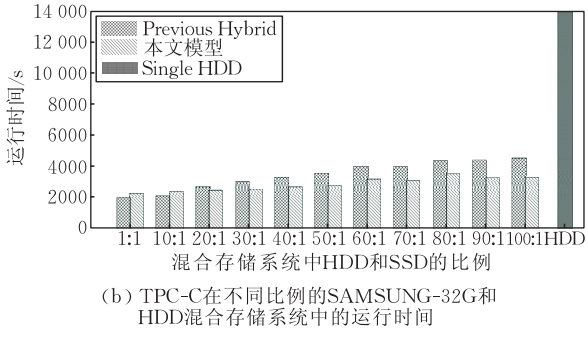
为了分析混合存储设备中 HDD 与 SSD 所占的空间比值(*HDDsize*/*SSDsize*)不同对实验性能的影响，我们为混合存储设备设定了 11 种不同的 *HDDsize*/*SSDsize* 比值. 其中，HDD 大小取值与数据集大小相同，即为 1 GB，SSD 大小由不同的 *HDDsize*/*SSDsize* 比值决定.

图 4(a)描述了在 RU-HDD 与 SAMSUNG-32 G SSD 混合存储设备上分别实现本文模型和 Previous Hybrid 模型，运行 OLTP 数据集的总运行时间；图 4(b)描述了运行 TPC-C 数据集的总运行时间. 图 4(a)和图 4(b)中前 11 组实验结果分别对应 11 种不同 *HDDsize*/*SSDsize* 比值大小的混合存储设备，而最后一组深色条形则表示数据集在单个 HDD 设备上的总运行时间. 对比发现，除 OLTP 数据集在基于 Previous Hybrid 模型的部分混合存储设备上运行时间较长以外，SSD 和 HDD 混合存储设备上的运行性能总是优于单个 HDD 设备的. 其次，不论哪种数据集，随着 SSD 大小的降低，实验性能逐渐变差，这进一步说明 SSD 的引入可以带来更高的性能. 此外，只有当 SSD 大小接近 HDD 大小时，Previous Hybrid 模型的性能略优于我们的模型，随着 SSD 大小的降低，本文模型的性能逐渐优于 Previous Hybrid，且性能差距逐渐增大. 这充分说明本文模型能够更有效地利用有效的 SSD 存储空间，其利用更小的 SSD 存储空间获得更多的性能提升.

图 5(a)和图 5(b)分别描述了 OLTP 和 TPC-C 数据集在 RU-HDD 与高端 SSD(Intel X25E SSD)混合存储设备上的运行结果. 第 2.4 节中描述的模型的特殊设计部分在该处实验中完成了从 SSD 到 HDD 的迁移. 图 5 中的实验结果与图 4 类似. OLTP 数据集在本文模型上相对于 Previous Hybrid 上有明显的改进；对于 TPC-C 数据集，当 *HDDsize*/*SSDsize* 比值小于 40∶1 时，本文模型仍优于 Previous Hybrid，但是随着 SSD 大小的增加，本文模型性能提升微弱，且不如 Previous Hybrid 模型的性能提升来得快. 这是因为 Intel X25E SSD 的写性能大大优于读性能，而且 TPC-C 数据集相对于 OLTP 数据

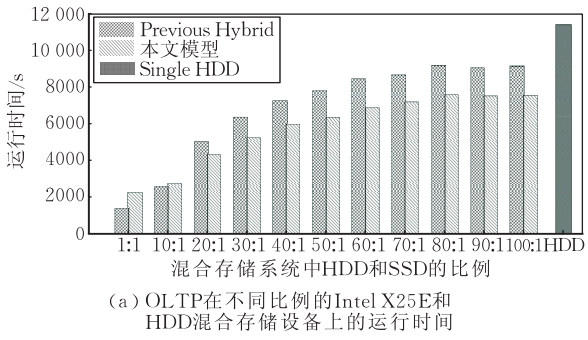



(a) OLTP在不同比例的SAMSUNG-32G和HDD混合存储设备上的运行时间

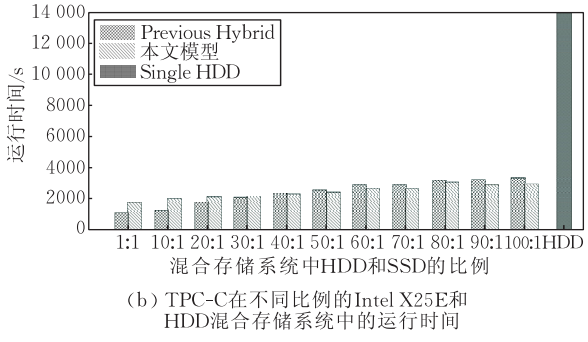



(b) TPC-C在不同比例的SAMSUNG-32G和HDD混合存储系统中的运行时间

图 4　数据集在 SAMSUNG-32 G 与 HDD 混合存储系统上的运行时间




(a) OLTP在不同比例的Intel X25E和HDD混合存储设备上的运行时间




(b) TPC-C在不同比例的Intel X25E和HDD混合存储系统中的运行时间

图 5　数据集在 Intel X25E 与 HDD 混合存储系统上的运行时间

集来说有大量的随机访问操作，从而导致由访问热度引起的迁移操作较少，而大部分迁移操作都是由本文模型和 Previous Hybrid 模型共同具备的页面定位倾向模块所触发引起，因此本文模型的优势在 TPC-C 数据集中体现不明显. 综合分析图 4 和图 5 可知，混合存储性能优于单个 HDD 设备，且本文模型相对于 Previous Hybrid 模型在 SSD 存储空间更小的情况下能够带来更大的性能提升.

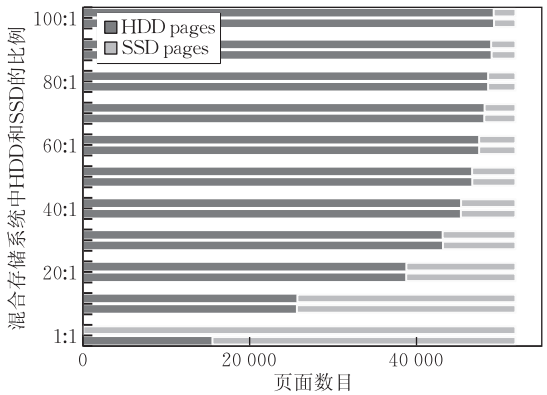
### 3.3 页面分类和物理读写分布

在试验中，存储设备的存储容量以页面数目来衡量. 如 3.2 节所述，HDD 的存储容量设置为 1 GB 即 262 144 个页面，SSD 的存储容量随着我们设定的比例变化. 我们对本文提出的混合存储模型和 Previous Hybrid 模型的页面分类进行了对比.

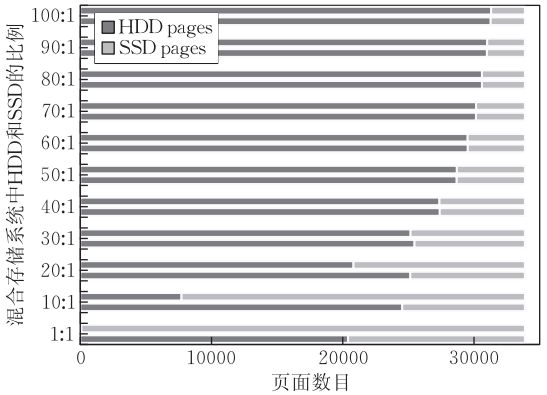
图 6 显示了分别在 OLTP 和 TPC-C 的 trace 中混合存储系统的页面分类结果. 在(a)～(d) 4 个图中，每一个纵坐标值对应两个柱状，其中上面的柱状表示在该比例情况下 Previous Hybrid 的页面分类结果，下面的一个柱状表示本文混合系统的页面分类结果.

从 4 个图中可以明显看出，尽管 trace 不同，SSD 设备也不同，但是随着 HDD 和 SSD 的比例变化，两个混合存储模型的页面分类结果有着相似的变化趋势. 二者明显的区别体现在当 HDD 和 SSD 的比例接近时，特别是在 1∶1 的情况下，Previous hybrid 将几乎所有页面都分配到了 SSD，这种情况下 trace 中的所有写操作也将发生在 SSD 上，这显然是不合适的，这样的结果和前文我们关于统计的积累效应的分析是一致的；本文的模型则只是将大部分页面分配到 SSD，这是因为衰减因子和热度计算的限制，从而避免了所有页面分配到 SSD. 当 HDD 和 SSD 的比例变大时，因为 SSD 的存储容量有限，所以 SSD 很容易被“填满”，一旦“填满”SSD 就会发生“溢出”必须进行迁移. 在实验结果中两个混合存储模型分配到 SSD 的页面数目逼近 SSD 的存储容量，表明受 SSD 容量限制，两个模型的页面分配比较接近. 结合 HDD 和 SSD 的 1∶1 比例时的实验结果，我们可以预测当比例较大时，在实验运行中 Previous Hybrid 的 SSD 上因“填满”发生的迁移将大大多于本文的模型. 关于迁移代价将在下一节讨论.

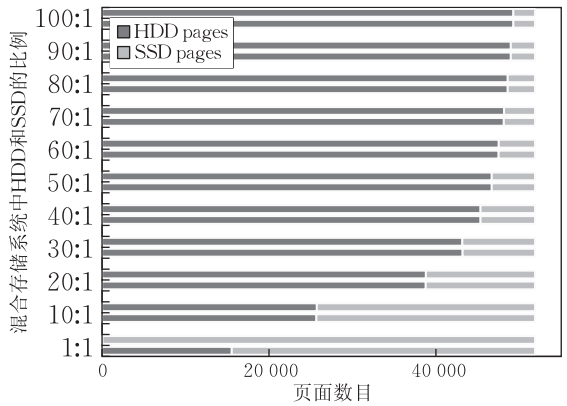
混合存储系统中页面的分类和读写访问的分布



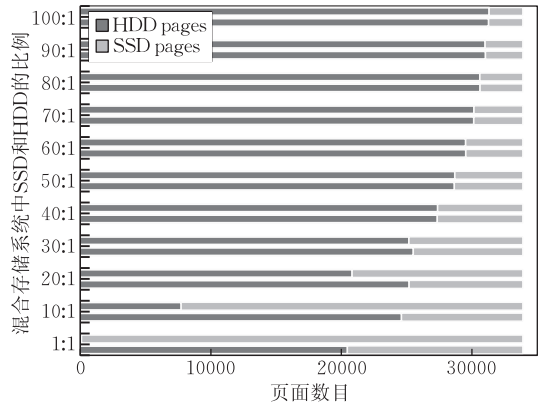

(a) OLTP在不同比例的Intel X25E和HDD上的页面分类




(b) TPC-C在不同比例的Intel X25E和HDD上的页面分类



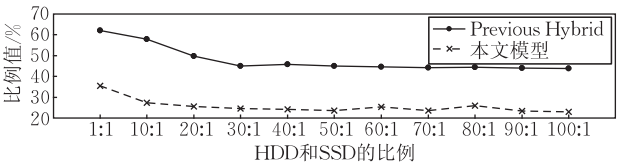

(c) OLTP在不同比例的SAMSUNG-32 G和HDD上的页面分类



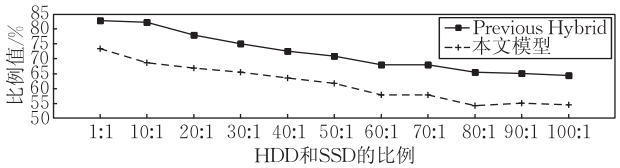

(d) TPC-C在不同比例的SAMSUNG-32 G和HDD的 上的页面分类

图 6 混合系统上的页面分配

是相关的，设计有效页面分配方法是为了实现合理的物理读写访问分布. 因为在混合存储系统中 SSD 的物理读写分布情况是决定系统性能的关键，所以我们在 TPC-C 的 trace 上分别统计了两个混合存储模型中 SSD 的物理读写访问次数，计算 SSD 的物理读写次数在系统总物理读写次数中的比例. 我们以 Samsung32 G 的 SSD 在 TPC-C 的 trace 上的实验来分析，结果如图 7 所示.




(a) SAMSUNG-32 G在TPC-C上物理写比例




(b) SAMSUNG-32 G在TPC-C上物理读比例

图 7　SSD 的物理读写比例

图 7 显示，随着 SSD 的容量减少，SSD 上分配的页面数目降低，两个模型中的 SSD 物理读比例都在下降. 两个模型相比较，本文的模型虽然物理读的比例较低些，但是物理写的比例一直维持在一个较低的水平；而 Previous Hybrid 则在获得较高的物理读比例的同时也带来了高达 50%左右的物理写比例，这样的结果对读写不对称，写代价较大的 SSD 来说是不利的，这也不利于系统的整体性能，同时这样的结果说明了本文模型的页面分配所产生的读写分布更为合理.

综合本节的实验结果，与 Previous Hybrid 相比本文的混合存储模型在页面分配方面准确性更强，带来的访问分布更加合理.

### 3.4　迁移代价

在整个混合存储系统中，所有的物理写操作由 2 类操作组成：(1) 缓冲区脏页的置换；(2) 页面在 SSD 和 HDD 之间的迁移. 对于特定的缓冲区大小和特定的 trace 文件，第 1 类写操作的数目是固定的，随着 SSD 容量的变化，第 2 类操作的数目将会发生改变，即混合存储系统的总的物理写数目将会变化.

在本文的实验环境下，我们先在单个硬盘上运行 trace 文件，记录下物理写的数目 $N$，该值就是这个 trace 文件运行中的缓冲区脏页置换次数，再在混合存储模型中以相同的缓冲区配置运行该 trace 文件，此时运行中的物理写数目 $M$ 与 $N$ 的差值即是迁移带来的物理写操作，也就是迁移代价. 以 SAMSUNG-32 G在 OLTP 的 trace 上的运行来进行实验，对不同的 HDD 和 SSD 比例下混合存储模型的物理写数目和单个硬盘的物理写数目的差值进行记录，结果如图 8 所示.

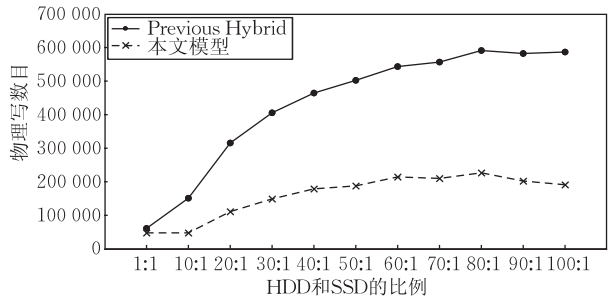



图 8　混合存储系统的迁移代价

在图 8 中，Previous Hybrid 的迁移代价明显高于本文的存储模型，这主要是由于 3.3 节所分析的 Previous Hybrid 较不准确的页面分类引起的. 在 HDD 和 SSD 比例为 1∶1 时，二者的迁移代价非常接近，这是因为在此比例下，SSD 的存储容量超过了 OLTP 的 trace 中的页面数目，不会发生 SSD“填满”的情况，从而迁移不会发生. 随着 SSD 容量降低，“填满”引起的迁移显著增加，本文的存储模型的准确性就体现了出来. 注意在比例超过 80∶1 后，本文模型的迁移代价又在缓慢下降，这是因为本文模型中的热度计算是和 SSD 容量大小联系起来的，当 SSD 容量过小时，只有很少的页面能进入 hot 状态从而迁移到 SSD，这就使页面从 HDD 到 SSD 的迁移减少，这会导致 SSD“填满”的情况也减少，从而 SSD 到 HDD 的迁移也因此减少了，所以整个系统的迁移都减少了.

结合 3.3 节和 3.4 节，本文的混合存储模型对页面的分类更为精确，对 SSD 的利用更加有效合理.

### 3.5　系统性价比分析

除了实验性能优劣的比较外，由于本文的目标致力于在更小的 SSD 存储空间中获得更高的性能提升，我们更加关心混合存储设备的性能价格比 *ratio*. 利用 *OLTP* 测试数据集在基于本文模型的混合存储设备上的运行性能以及表 3 中给出的设备价格，定义混合存储设备的性能价格比如下式所示

$$ratio=\frac{(hddTime-hsTime)/hddTime}{Price\text{-}of\text{-}SSD\text{-}in\text{-}Hybrid\text{-}Storage} \quad (2)$$

其中，*hddTime* 参数表示 OLTP 数据集在单个 HDD 设备上的运行时间，*hsTime* 参数表示 OLTP

数据集在混合存储设备上的运行时间.故公式中分子表示混合存储相对于单个设备的性能提升比例,分母表示 SSD 的价格且由混合存储设备的空间比值 *HDDsize*/*SSDsize* 大小决定.

图 9 分别描述了基于本文模型的两种混合存储设备在不同空间比值 *HDDsize*/*SSDsize* 情况下的性能价格比 *ratio* 值分布情况.其中中虚线代表 RU-HDD 与 Intel X25E SSD 混合存储设备;实线代表 RU-HDD 与 SamSung-32 G SSD 混合存储设备.RU-HDD 与 Intel X25E SSD 混合存储设备的 *ratio* 值明显高于 RU-HDD 与 SAMSUNG-32 G SSD 混合存储设备.当混合存储设备空间比值 *HDDsize*/*SSDsize* 接近 1 时,即 SSD 大小与 HDD 大小相等时,两种混合存储设备的性能价格比非常接近.这说明当 SSD 大小与 HDD 大小相等,此时若基于本文模型实现混合存储系统,选择中端 SSD 或高端 SSD 性能价格比是接近的.考虑相同大小的中、高端 SSD 价格相差较大,应当选择中端 SSD 完成设计.此外,随着空间比值 *HDDsize*/*SSDsize* 从 1∶1 增加到 50∶1,两种类型的混合存储设备的性能价格比均有快速提升,表明该空间比值范围适合基于本文模型的混合存储系统实现;随着空间比值 *HDDsize*/*SSDsize* 从 50∶1 增加到 80∶1,两种类型的混合存储设备的性能价格比均增长缓慢,表明该空间比值范围不适合基于本文模型的混合存储系统实现.故基于本文模型的混合存储系统实现时最佳空间比值为 1∶1 到 50∶1.

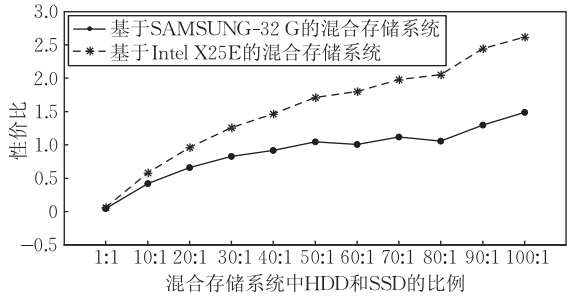



图 9 混合存储系统的性价比

### 3.6 "warm"对混合模型的作用

本文在页面状态中引入了"warm"状态,以避免 cold 页面骤然地转变为 hot 状态,从而减少不必要的迁移操作.所以带有"warm"状态的混合模型和没有"warm"状态的相比,迁移代价应该较小,运行时间较短.根据 3.4 节的讨论,迁移代价以模型运行中的物理写数目衡量,所以通过比较有"warm"的模型和无"warm"模型的运行时间、物理写数目就可以衡量"warm"对混合模型的作用.

本文以 HDD 和 SAMSUNG-32 G SSD 组成的混合存储系统作为实验对象,以 TPC-C trace 作为实验数据,在本文的混合模型有"warm"和无"warm"的情况下,分别在混合系统中 HDD 和 SSD 的比例为 10∶1,30∶1,50∶1 时进行实验,实验的结果如图 10 所示.

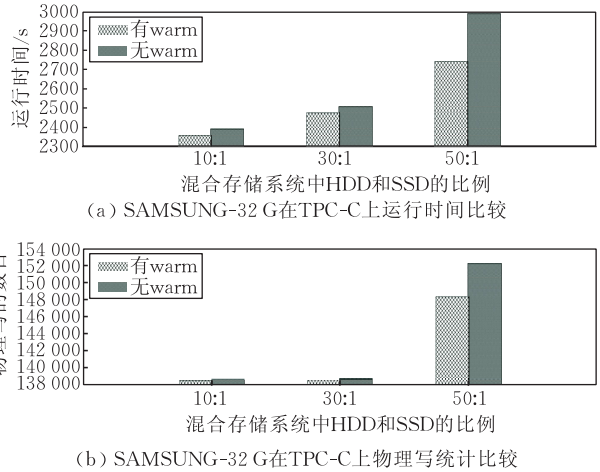



图 10 "warm"对混合模型的作用

图 10 显示,在无"warm"的混合模型中,物理写的数目更多,这表示迁移代价更大.这些额外的物理写是由 2.2 节第 1 段所分析的情况带来的,实验结果证实了直接的 cold/hot 状态转换会引起不准确的迁移.较大的迁移代价会降低模型的性能,这个结果在图 10(a)中也表现了出来.

通过图 10 的对比,"warm"状态的引入对混合模型是有着积极意义的.

## 4 总 结

本文提出了一个高性价比的 SSD-HDD 混合存储模型,该模型将有限容量的 SSD 引入磁盘存储体系从而获得明显的性能提升.我们通过引入 warm 的概念改进了传统的 hot/cold 思想并用于混合存储系统的设计,并且利用衰减因子修改 I/O 统计值来实现较为准确的迁移判断.最后本文对混合存储系统的性价比问题展开了讨论,这对于探讨 SSD-HDD 的最佳组合方式是有益的尝试.

未来,我们将对其它的混合存储架构展开研究.热度状态迁移的思想将会得到更多的关注.混合存储的性能价格比在以前的相关研究中没有提及,该方向具有一定的现实意义,我们将对其展开进一步的研究.

## 参 考 文 献


[1] 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-associative sector translation. ACM Transactions on Embedded Computing Systems (TECS), 2007, 6(3): 18

[2] 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

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

[4] Graefe G. The five-minute rule twenty years later, and how flash memory changes the rules//Proceedings of the 3rd International Workshop on Data Management on New Hardware (DaMoN'07). New York, USA, 2007: 6

[5] Kgil T, Roberts D, Mudge T. Improving NAND flash based disk caches//Proceedings of the 35th International Symposium on Computer Architecture (ISCA'08). Beijing, China, 2008: 327-338

[6] Kgil T, Mudge T. Flashcache: A NAND flash memory file cache for low power web servers//Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES'06). New York, USA, 2006: 103-112

[7] Koltsidas I, Viglas S D. Flashing up the storage layer. Proceedings of the VLDB Endowment, 2008, 1(1): 514-525

[8] Jung H, Shim H, Park S, Kang S, Cha J. Lruwsr: Integration of LRU and writes sequence reordering for flash memory. IEEE Transactions on Consumer Electronics, 2008, 54(3): 1215-1223

[9] Ou Y, Hörder T, Jin P. CFDC: A flash-aware replacement policy for database buffer management//Proceedings of the 5th International Workshop on Data Management on New Hardware (DaMoN'09). New York, USA, 2009: 15-20

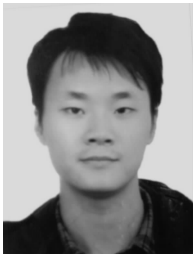
[10] Li Z, Jin P, Su X, Cui K, Yue L. CCF-LRU: A new buffer replacement algorithm for flash memory. IEEE Transactions on Consumer Electronics, 2009, 55(3): 1351-1359

[11] 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

[12] 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

[13] Soundararajan 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

[14] Bouganim L, Jónsson B, Bonnet P. uFLIP: Understanding flash IO patterns//Proceedings of the 4th Biennial Conference on Innovative Data Systems Research. Asilomar, CA, USA, 2009


**YANG Pu-Yuan**, born in 1985, Ph. D. candidate. His main research interests include flash memory database and hybrid storage management.

**JIN Pei-Quan**, born in 1975, Ph. D., associate professor. His main research interests include moving object database, spatio-temporal database, flash memory database and internet information extraction and search.

**YUE Li-Hua**, born in 1952, professor, Ph. D. supervisor. Her main research interests include information integration, real-time database and flash memory database.


**Background**

In recent years, the technology of flash and SSD has gotten much attention from academic circles and industrial community. The technology of flash before was usually based on the concept "flash replace magnetic disk absolutely". However, some recent experimental data and research result shows that the hybrid of flash and magnetic disk is the future of the storage system. Current correlative research does not clear the policy of the hybrid storage based on flash and magnetic disk and there is no research work having proposed an effective approach of data classification, which blocks the development and application of the flash device.

This paper, aiming at the problem, proposes some viewpoint for the hybrid storage taking SSD and HDD at the same level. This paper mainly investigate the problems such as approach of data classification, the best hybrid ratio of HDD to SSD, the performance price ratio of whole storage system and so on. Finally, this paper proposes a hybrid model to resolve these problems above. This paper proposes a time-sensitive hybrid storage model to efficiently take use of SSD. This model takes SSD and HDD at the same level of the memory hierarchy. According to the page access counter and access temperature, the model achieves accurate page classification and placement which places the hot and read-intensive pages SSD and places the cold or write-intensive pages to HDD. So the model takes advantage of the asymmetric I/O properties of HDD and SSD to reduce total I/O latency of system.

This work is supported by National Natural Science Foundation of China (60833005, 61073039). The both projects investigate the storage management based on the flash. They focus on the database technology such as buffer management, index, storage management and so on. Except that, the both projects also take much work on the problem of hybrid storage.