IOWA STATE UNIVERSITY
Digital Repository

Graduate Theses and Dissertations

Iowa State University Capstones, Theses and Dissertations

2014

Hybrid drive design: an economics - workload based approach

Joy Shukla
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

 Part of the Computer Engineering Commons

Recommended Citation
Shukla, Joy, "Hybrid drive design: an economics - workload based approach" (2014). *Graduate Theses and Dissertations*. 14297.
https://lib.dr.iastate.edu/etd/14297

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

www.manaraa.com

Hybrid drive design: An economics - workload based approach

by

Joy Shukla

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:

Arun K. Somani, Major Professor

Phillip Jones

Govindarasu Manimaran

Iowa State University

Ames, Iowa

2014

www.manaraa.com

DEDICATION

I would like to thank my friends and family for their loving guidance and financial assistance during the writing of this work.

www.manaraa.com

TABLE OF CONTENTS

www.manaraa.com

www.manaraa.com

LIST OF TABLES

www.manaraa.com

LIST OF FIGURES

www.manaraa.com

LIST OF ALGORITHMS

www.manaraa.com

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis.

www.manaraa.com

ABSTRACT

Broader availability of solid state storage devices (SSD) enables an opportunity to improve the performance of the storage architectures. However, it not clear how exactly this development should proceed in terms of achieving greater cost-performance balance. Designers will benefit from a data-driven approach to understand how much SSD they should invest in to realize the most cost effective system.

This paper presents an analysis of actual workloads to deduce and derive guidance for an optimal investment strategy to balance the solid state and hard disk drive (HDD) to achieve the best cost and performance trade-offs. We show that while it is possible to determine this balance, it is heavily application dependent. For the workloads we studied, under certain assumptions, the preferred proportion of SSD varies from 8% to 60% for an 80% improvement in I/O performance (measured in terms of hits in SSD) compared to totally magnetic disks.

Further, we also propose three replacement strategies to keep the most accessed data in SSD. This replacement is determined using the past usage data. The goal is to make the best use of the available SSD, while minimizing the number of replacements. Our Simulation results shows that the best of our strategies provide 60% to 90% performance improvement compared to totally HDD across different workloads.

www.manaraa.com

CHAPTER 1. INTRODUCTION

Early civilizations recognized the importance of permanent storage of knowledge possessed by them to be passed on to future generations. Human communication has evolved from cave paintings to global information system that is in effect today. The digital universe is large and by 2020 will have as many bits as there are stars in our physical universe. With this rapid expansion comes a dilemma of storage of this abundant information.

Since the first demonstration of the Cathode Ray Tube (CRT) in 1948 that allowed volatile storage of data, the industry has evolved and has seen a range of devices Piramanayagam and Chong (2011). Tapes and the magnetic disks served need of the time. A desire to improve the reliability of storage at an affordable price continued. Hard Disk Drives (HDD) became the industry standard for storage device in early 1960s and have been there. Hard disk drives have in the past 50 years undergone crucial changes to accommodate the advancements in the computer processing power. In 1956 IBM launched RAMAC, spanning the size of 2 refrigerators and weighing almost a ton. It had an areal density of 2 kilobits/in^2 ram (1956).

Today's drives store data in densities as high as 0.25 terabits/in^2. This is a whopping 44% per annum compounded rate of increase in the last 50 years Wood (2009).

Solid State Drives (SSD)- the newer development offers benefits over the traditional HDDs. Flash storage devices deliver the performance that can match today's processing powers. The technology for now is still more expensive and impractical for consumer electronics. We will use SSD and flash interchangeably in the paper.

The two types of storage devices can be compared based on following four basic parameters. a) reliability or durability b) processing speed and one of the most important parameter for the industry, i.e. c) manufacturing cost and d) environmental impact.

www.manaraa.com

(a) Reliability or Durability.

Based on the inherent design on SSDs they are lighter in weight and also more durable than the traditional HDDs. Also because of them using semiconductors instead of a magnetic film they are stable in magnetic fields, which is a problem with the HDDs. In addition, owing to their semiconductor material based design, SSD's are stable in magnetic field whereas HDD's are not. Though SSDs may seem more reliable in the short term perspective but have a flaw that they can only have a set number of write cycles, beyond which they wear off. The HDDs can do many more write cycles compared to SSDs without significantly reducing the performance.

(b) Processing speed.

In the world of computational power the HDDs are the speed bumps. In the last 10 years the processing power has grown 30 times, whereas the HDDs have just done a meager rise of 30%. One of the biggest limitations of traditional magnetic drive is the high latency. Flash memory is the secondary storage that competes most closely with traditional magnetic hard disks. SSD's are typically twice as fast as HDD's Micheloni et al. (2013).

(c) Manufacturing Cost.

The cost of SSDs is a major restriction in using the technology where flash memory comes at 3$/GB, HDDs offer the economical alternative at 30 cents per gigabyte No (2012). Thus this limits the use of pure SSDs in consumer electronics and help HDDs maintain their position of the industry standard in storage drives. With technological advances,SSDs will become more affordable, but for now the alternative solution can only be sought in hybrid drives that give us the opportunity to use the virtues of both at once.

(d) Energy Consumption.

Due to the absence of any moving parts the overall energy consumption of SSDs is less than HDDs.

www.manaraa.com

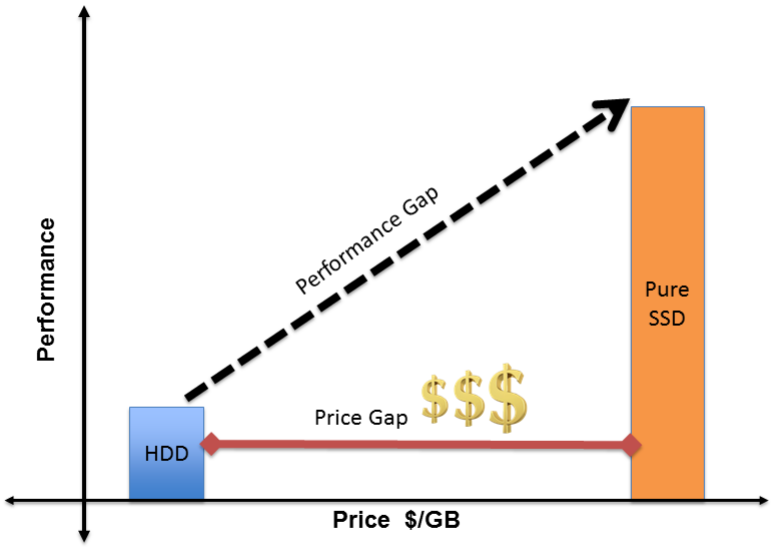


Figure 1.1: Price and Performance difference between traditional HDD and SSD enm (2014)

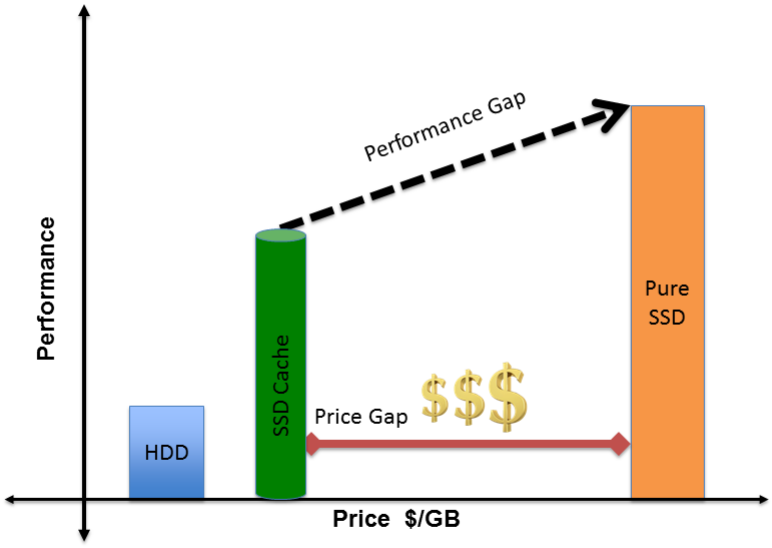


Figure 1.2: Using SSD as Cache enm (2014)

www.manaraa.com

Hybrid Solution and its advantages

The complementary features of the flash memory and hard disks have motivated several proposals on hybrid storage devices by combining the disks and flash memory. If the advantages of these two technologies could be bridged, high-end processing power can be made accessible to masses No (2012). To achieve this goal, hybrid drives that blend the speed of SSDs with the cost efficiency and storage capacity of HDDs are being developed. Hybrid Drives have two separate storage spaces, one is a small flash memory component, and the other is a traditional disk. Considering the wide range of possibilities that Hybrid Drives create, they can serve as the catalyst for the industry transition to a sustainable new generation of data storage.

www.manaraa.com

CHAPTER 2. RELATED WORK

Hybrid storage is becoming more and more attractive because it can leverage the advantages from both technologies. There are several existing approaches attempting to better utilize the memory hierarchy in flash-based hybrid storage systems.

FaCE (Flash as Cache Extension), a low overhead caching method, uses SSD as an extension of DRAM buffer or a cache layer between DRAM and disk Kang et al. (2012). FaCE utilizes SSD in a FIFO manner to take advantage of the high sequential write performance of SSD. Additionally FaCE proposes GSC (Group Second Chance) to increase hits on SSD. GSC gives a valid page second chance before being dequeued from the cache, if the page has been referenced while staying in the SSD. From this study the crucial observations were, that adding flash memory as cache extension is more cost effective technique over increasing the size of DRAM buffer. The main drawback of these designs is they do not make full use of the storage hierarchy. All the pages replaced out of main memory will be kept on the flash no matter whether they will be reused again

SSDAlloc, another recent study, uses a similar approach to treat SSD as an extension of the RAM in the system SSDAlloc exposes flash memory using page-based virtual memory manager interface Badam and Pai (2011).

Another empirical approach to manage the buffer in flash-based hybrid storage systems, named Hotness Aware Hit (HAT) Lv et al. (2013). HAT utilizes a page reference queue to maintain the historical access information i.e. hot, warm and cold, and the queue itself is divided into hot region and warm region. The HAT approach updates the page status and deals with the page migration in the memory hierarchy according to the current page status and hit position in the page reference queue.

www.manaraa.com

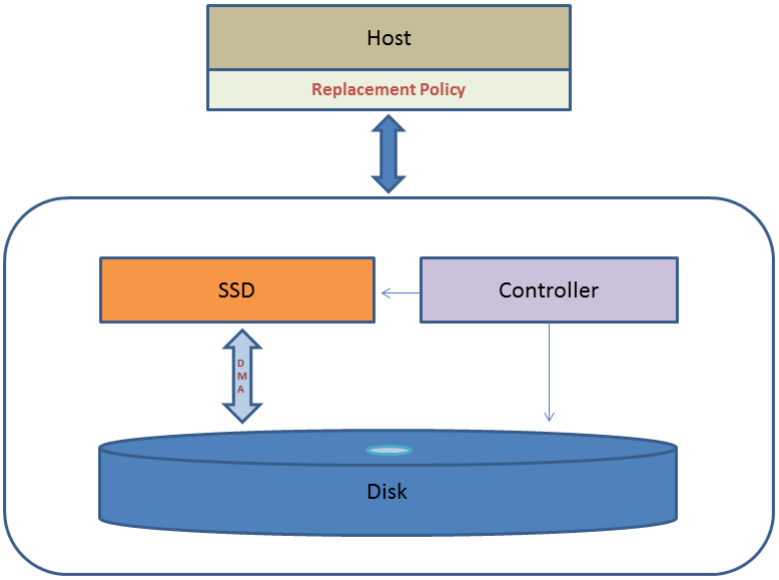


Figure 2.1: Replacement in Host

2.1 Our Approach

Above approaches make replacement decisions in operating system page replacement algorithm.. They propose to make changes the way storage gets handled by the host in the OS. Also, as these decisions are made at the operating systems level, it increases computation overhead for hosts. Figure 2.1 shows a typical structure of the hybrid drive where replacement decision is made at host level. In this arrangement, the host needs to keep track of the meta-data for replacement. This results into both memory and computational overhead. The DMA (Direct Memory Access) between SSD and HDD also needs to be controlled by the host for replacements.

Figure 2.2 shows our approach in which replacement decisions are made in the disk drive controller. Disk drive controller already maintains meta-data about all the sectors in the drive. This requires comparatively less overhead to keep meta-data related to replacement policy. Also, the host side computational overhead is reduced by making replacement decision in the disk drive controller.

www.manaraa.com

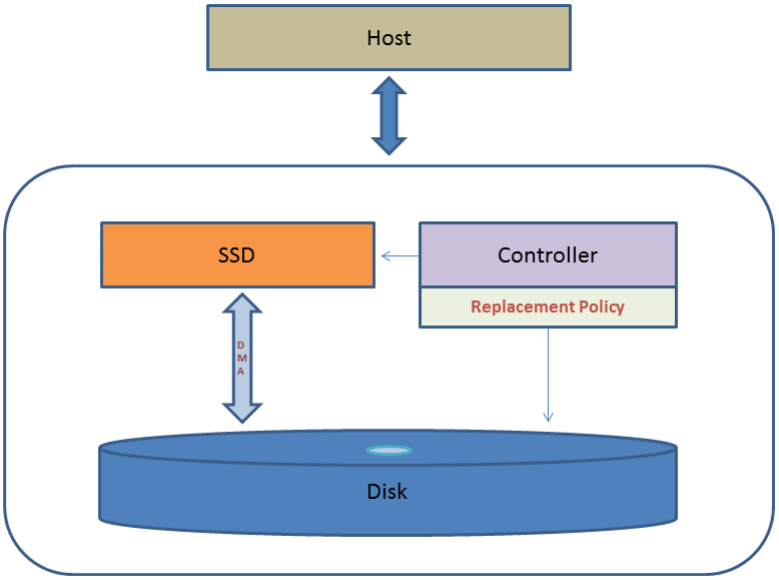


Figure 2.2: Replacement in Disk

An advantage of our approach is that the host does not need to know which kind of disk it is dealing with, whether it is a hybrid disk or a traditional HDD or an SSD.

Our approach proposes to make such decisions related to what data should be kept in SSD and what data should be kept in HDD in disk drive itself. We propose to make smarter hybrid disk which monitors access patterns by the host and based on that makes replacement in SSD. This way host does not need to know what kind of disk it is dealing with, a hybrid disk system, an HDD or a SSD.

The rest of the paper is organized as follows. Chapter 3 describes our modeling and analysis for determining the SSD size. Chapter 4 describes proposed replacement policies and analysis of simulation results. Lastly, we conclude in Chapter 5.

www.manaraa.com

CHAPTER 3. DETERMINING SSD SIZE

Hybrid disk drive give a good balance of performance and affordability compared to traditional magnetic drive solution and all SSD solution. But how much SSD and how much HDD should be there in our hybrid drives will depend on performance requirement and the workload. In this section we analyze spatial locality of the workload to help us make decision about SSD-HDD proportion in hybrid disk drive.

3.1 Spatial Locality Analysis

Spatial locality refers to location of the data accessed. If a particular memory location is referenced at a particular time, then it is likely that nearby memory locations will also be referenced in the near future. In this case it is common to attempt to guess the size and shape of the area around the current reference for which it is worthwhile to prepare faster access. Such references to nearby memory locations can be grouped as access to a certain block or page or other granularity of transfer between SSD and HDD. We call that granularity a bank. All references to any location in a bank can be considered as an access to the corresponding bank.

Our approach is to monitor traffic on all such logical banks in the disk. We want to find the portion of most busy banks amongst them. Naturally if we move data from those busier banks to SSD and keep other data in HDD, we will get most performance improvement. As an example, if for some type of application top 20% of banks handle 90% of total traffic, we can conclude that for the application 20% SSD and 80% HDD combination will give 90% performance improvement over traditional HDD by employing just 20% extra SSD. We do cost vs performance analysis considering latency as a performance metric.

Cost and performance metric is defined as

www.manaraa.com

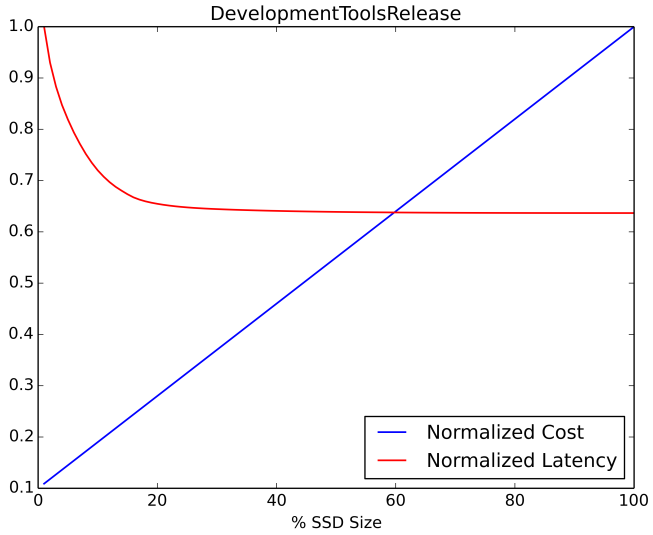


Figure 3.1: Cost-Latency vs SSD Size

$$Cost = \alpha * SSDsize\% + \beta * HDDsize\%$$

$$Latency = \gamma * \%AccessInSSD + \delta * \%AccessInHDD$$

Here, α and β are unit cost of SSD and HDD respectively. γ and δ are the latency of SSD and HDD respectively. Our model calculates ideal *AccessInSSD* for different proportion of SSD in the hybrid drive. *AccessInHDD* will be calculated by below formula

$$AccessInHDD = TotalAccess - AccessInSSD$$

We assume that SSD is 10 times costlier than HDD i.e. $\alpha = 10 * \beta$. As SSD is typically twice fast as HDD, we assume $\gamma = \delta/2$. As we increase SSD proportion in hybrid drive, cost increases. At the same time, more access will happen in SSD with lower latency ultimately reducing hybrid drive latency. Plot in Figure 3.1 shows this property. Interestingly, if we keep increasing SSD proportion, after certain point latency will not reduce although cost will keep increasing linearly.

www.manaraa.com

3.2 Data Driven Design

We conduct analyses on workload traces provided by SNIA(Storage Networking Industry Association). These traces consist of workload of different applications. Each trace is collected over a period of a day or two. They capture primarily disk IO events. It gives information about each IO access like TimeStamp, LBA(Logical Block Access), IOSize, etc. We access these traces and generate our meta-data for our analysis.

1. We record the accesses to each bank;
2. We sort the banks in descending order of access counts;
3. We plot the percentage of total accesses handled by the top x% banks

For example, consider 4 banks B_0, B_1, B_2 and B_3. Let the accesses to each bank after monitoring complete trace be 100, 250, 50 and 70. We sort these banks as B_1, B_0, B_3 and B_2. Now if we had 25% SSD, we will have only B_1 in SSD and other banks in HDD. In this scenario, our percentage of total access will be $250/(100+250+50+70) = 53.19\%$. We do similar calculation for 50% SSD - 74.47%, 75% SSD - 89.36%. We do such analysis in two phases. First we gather the data and populate it in map data structure. This data is used in second phase for the analysis. Next, we perform sorting of banks based on above-mentioned traffic and analyze how much traffic is handled by how much percentage of the banks.

3.2.1 Gathering Data For The Analysis

We first gather data to implement our approach. Our model counts Reads and Writes in each banks and put them in a map data structure for easy access for analysis. Algorithm 1 shows how the map data structure was created. After the execution of Algorithm 1 we have a map object which have *BankNumber* as the key and *AccessCount* and *AccessBytes* as values for each bank that was accessed during the trace.

www.manaraa.com

Algorithm 1 Spatial Locality Analysis: gathering data

Input - Operation Type(Read/Write), Logical Block Address(CurrentLBA), Accessed Bytes(CurrentAccessBytes)
Output - Bank[] [AccessCount] [AccessBytes]
while end of trace **do**
  Read currentAccess
  CurrentBank = FindBank(CurrentLBA)
  incr Bank[CurrentBank][AccessCount]
  Bank[CurrentBank][AccessBytes]+=CurrentAccessBytes
  Next Access
end while

Algorithm 2 Spatial Locality Analysis: Analyze

Input - Bank[] [ReadCount] [WriteCount] [ReadBytes] [WriteBytes]
TopPerc = 1
while TopPerc < 100 **do**
  RunningSum[TopPerc] = Cumulative count of all accesses to the TopPerc% of banks

  plot fraction of RunningSum[TopPerc] and total accesses Vs TopPerc
  incr TopPerc
end while

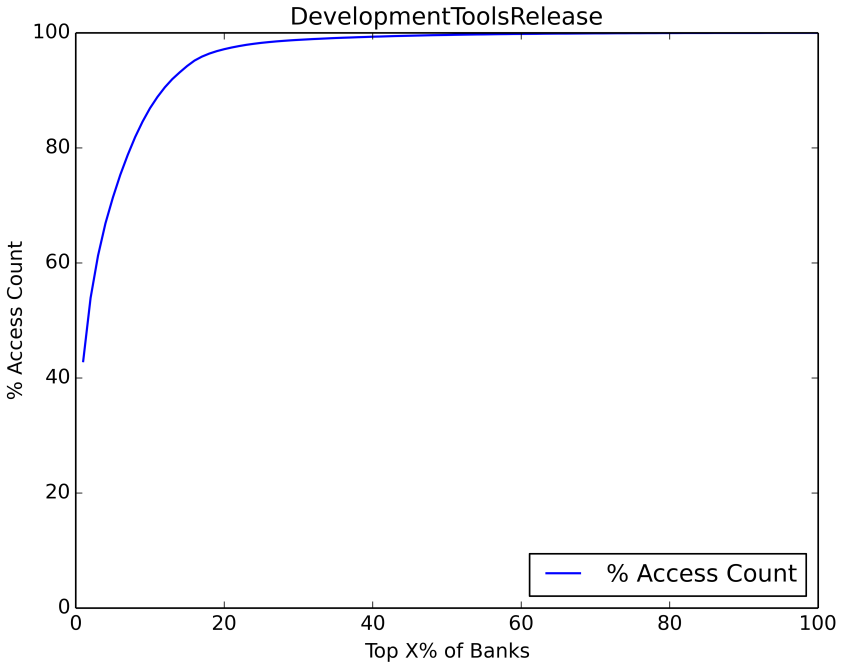


Figure 3.2: Sample Spatial Analysis

www.manaraa.com

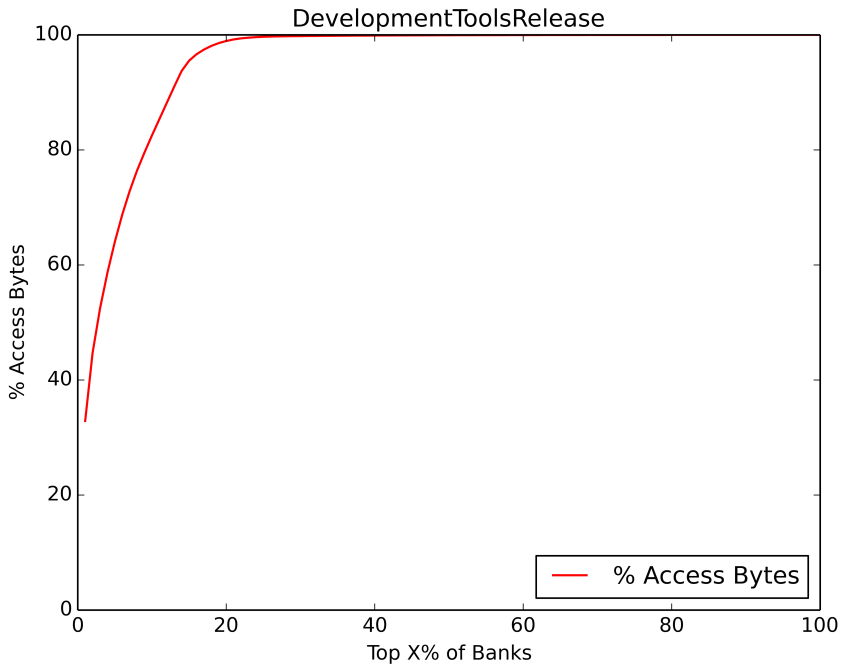


Figure 3.3: Sample Spatial Analysis of Bytes Accessed

3.2.2 Analysis

Next we analyze the generated data. The goal is to analyze how much traffic is handled by how much percentage of banks. So we sort the banks based on number of accesses. Banks with maximum accesses i.e. maximum traffic stays on the top of the list and least traffic will be on the bottom on the list. The idea is to have a complete HDD and keep adding some amount of SSD in that (removing same amount of HDD from it, i.e. replacing portion of HDD with SSD) and see how much traffic can be handled by that proportion of SSD. The traffic handled by SSD is the improvement as we can access that data with lower latency compared to an HDD. As we keep increasing amount of SSD, more portion of traffic gets handled by SSD resulting in more improvement.

Figure 3.2 shows a sample spatial analysis done using Algorithm 2 for Development Tools Release. The plot depicts Y% of traffic(i.e. access count) is handled by X% of sorted banks. This analysis is useful in deciding how much SSD portion should a Hybrid disk drive have for the performance requirement. For example, if certain application have locality characteristic

www.manaraa.com

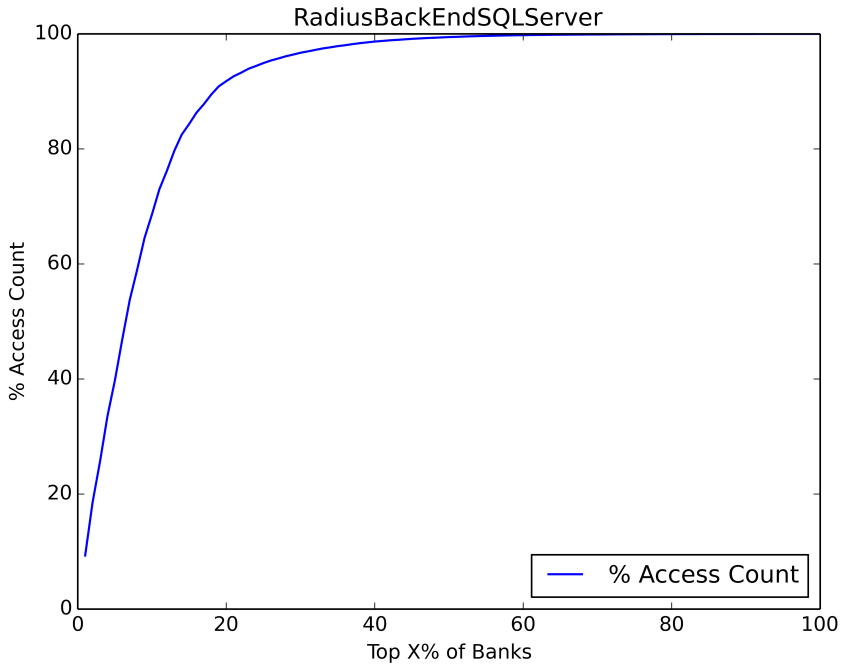
as in the plot on Figure 3.2, then we read from the plot that top 10% of banks handle 86% of traffic. If we move most used banks in SSD, we have 10% SSD to handle 86% access requests with low latency. This analysis provides understanding of the I/O workload to make the right investment in storage. The sample shows analysis for Read-Write count. Similar analysis can be done considering no of accessed bytes as well. Figure 3.3 shows such analysis. Instead of Read-Write counts we do our analysis on Read-Write IOSize. Such analysis can tell how much data traffic was handled by how much SSD. We can read from the plot in figure 3.3 that top 10% of banks handle 82% of data traffic.

The spatial locality analysis done above is for certain application. It provides us ideas on how much SSD our hybrid drive should have to get required speed. Practically we will have different kind of workloads executed on same infrastructure. For that we can do such individual application analysis and find the application which needs most amount of SSD for its requirement. When application changes or even in same application we need to shuffle data in SSD to get maximum performance. In the next section we discuss replacement strategies to be used for such shuffling.

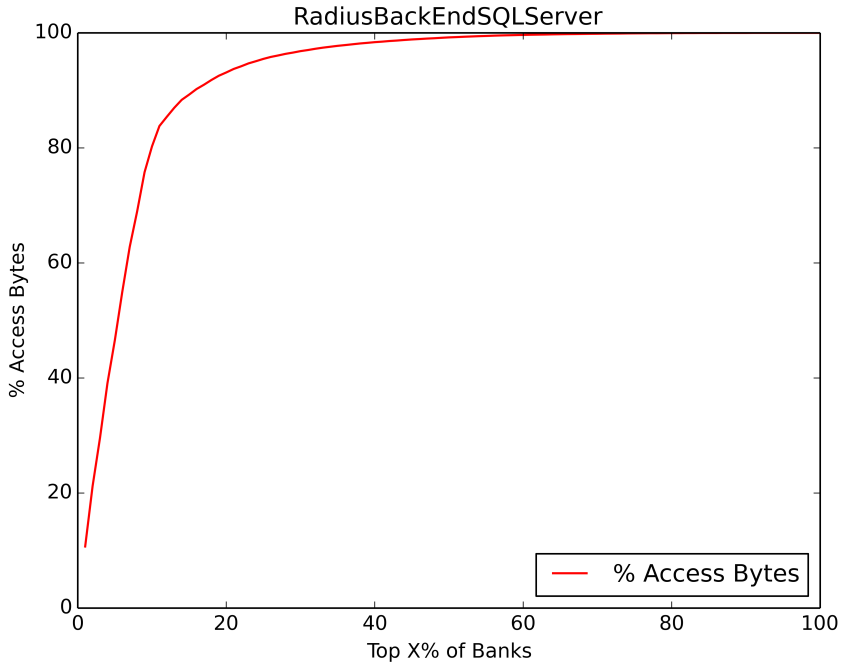
3.2.3 Spacial Locality Analysis Results

The plots in Figure 3.4 to 3.6 shows the spatial locality results for four different applications. X-axes shows top %banks and Y-axes shows fraction of access count and fraction of data. A point (x,y) on the curve in these plots denotes that the top x% banks handle y% of the total accesses. Therefore, as x tends to 100, y tends to 100. The application RadiusBackEndSQLServer shows very good spatially local workload. From the plot, *80%* improvement in %access count will require only *13%* SSD. Another application DisplayAdsPayload exhibits very low spatially local workload. Here, *80%* improvement in %access count will require *60%* SSD. The same way MSNStorageCFS requires *38%* SSD for *80%* improvement in access count compared to totally magnetic disk.

www.manaraa.com



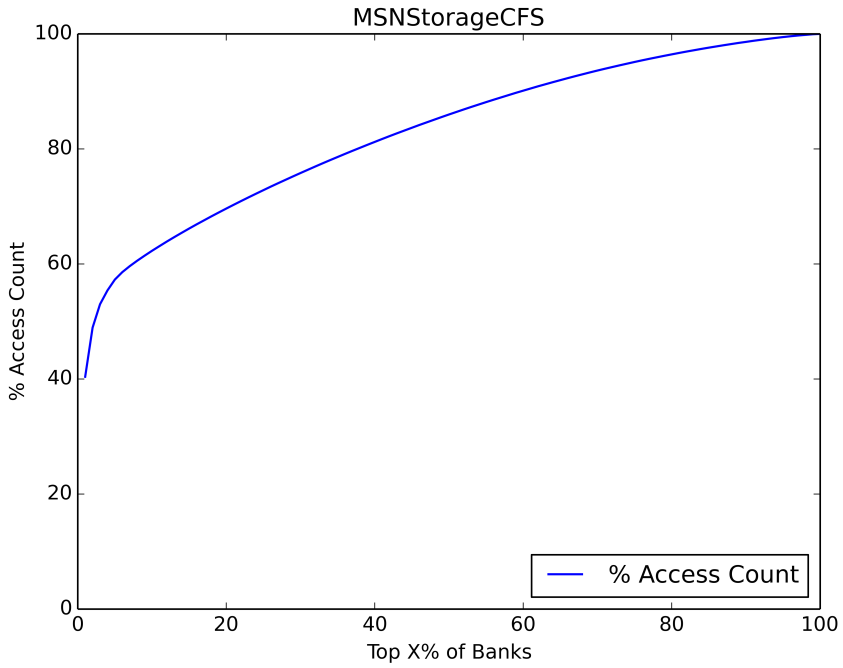
(a) % Access Count



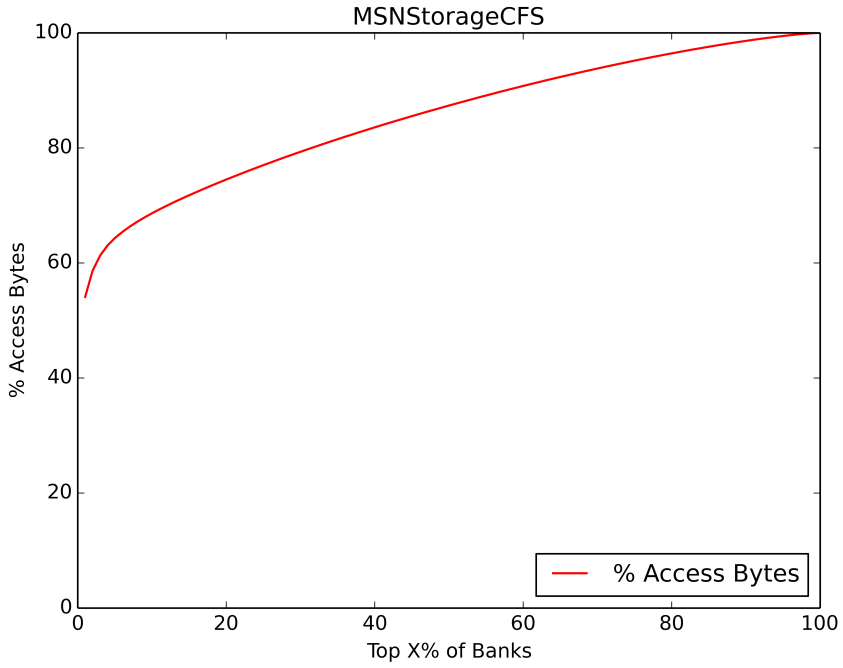
(b) % Access Data

Figure 3.4: Spacial Locality Analysis Results - RadiusBackEndSQLServer

www.manaraa.com



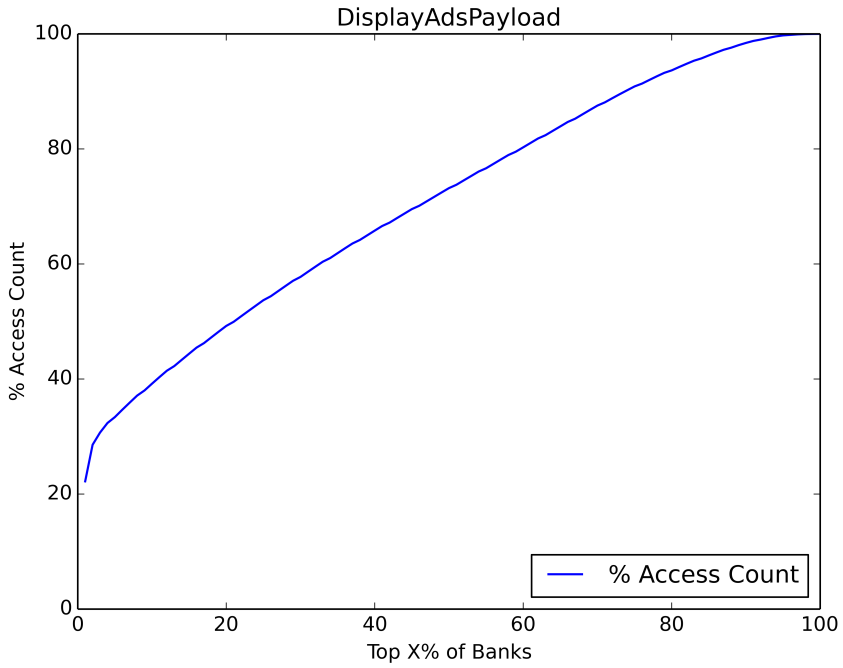
(a) % Access Count



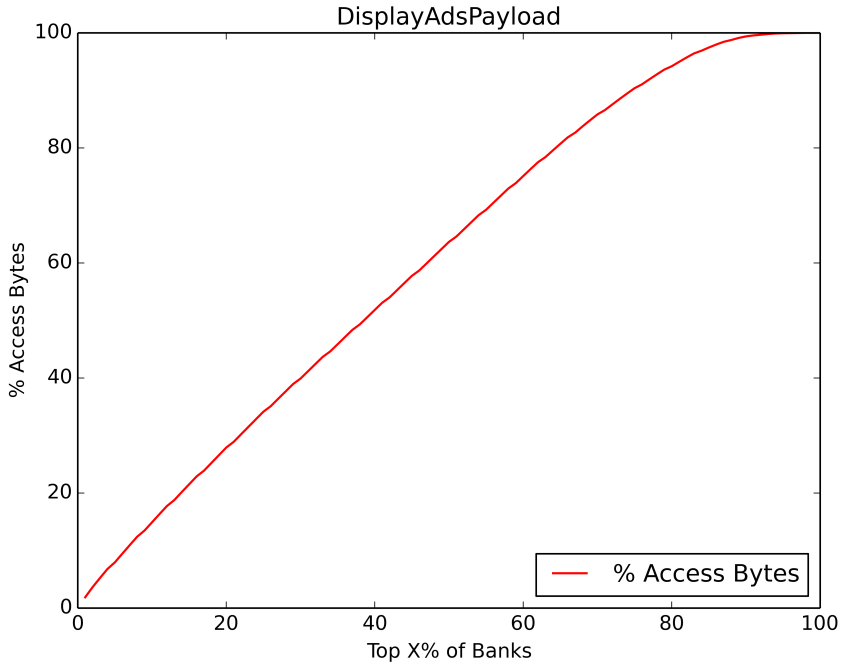
(b) % Access Data

Figure 3.5: Spacial Locality Analysis Results - MSNStorageCFS

www.manaraa.com



(a) % Access Count



(b) % Access Data

Figure 3.6: Spacial Locality Analysis Results - DisplayAdsPayload

www.manaraa.com

CHAPTER 4. REPLACEMENT STRATEGIES

The analysis done in Section 3.1 gives understanding of the I/O workload. It can give idea about how much SSD and how much HDD a hybrid drive storage system should have.The underlying assumption is the relevant banks need to be present in SSD when they are being accessed. It is an idealistic situation. If we had some mechanism that can tell which bank will be accessed in future, we might as well get 100% "improvement". So the above mentioned analysis is an indicator of a suitable size of SSD. In order to realize the above-stated improvement, we need to have smart replacement policies. These replacement strategies probabilistically determine when and what data will reside in SSD and HDD. Although the problem sound similar to replacement in cache, there are some fundamental differences. SSD is non-volatile, SSD have limited number of writes. Here, we will introduce some replacement strategies and discuss advantages and disadvantages of each.

4.1 Interval Least Frequently Used (LFU) Replacement

Under a perfect model, SSD will always contain the data that needs to be accessed in the memory. So we should put data which will be accessed most number of times in SSD to reduce latency for most accesses. Similar to Cache-DRAM model, in the context of SSD-HDD model, a 'hit' is said to occur when the data that needs to be accessed lies in SSD. A 'miss' is said to occur when the data that needs to be accessed does not lie in SSD, so it needs to be accessed from HDD with higher latency. To improve the performance, we need to have more number of hits and less number of misses. At the same time, we can not replace data in SSD very frequently like cache-DRAM because 1) block size is much higher(several pages) so transfer penalty is high, and 2) SSD has limited number of erase-write cycles. Hence, our objective is

www.manaraa.com

to maximize the hits while performing minimum number of replacements.

In Interval LFU, banks are replaced at regular periods or intervals. This is based on the assumption that banks accessed most in previous interval, will be again accessed most in current interval. So based on previous interval's spatial analysis, as discussed in Section 3.1, we determine the banks that should reside in SSD. Assume that SSD can accomodate x number of banks, top x banks accessed in i^{th} interval will reside in SSD for $(i+1)^{th}$ interval. Out of these banks some of them will already be in SSD. So we need to do replacement for those banks which should be in SSD but are not there in SSD. These replacements happens in background or in idle time based on on-line data collections. LFU replacement increases number of hits compared to no replacement which helps increase overall speed. Performance metric for the replacement policy would be Improvement percentage and total number of replacements. The main operations of interval LFU replacement is summarized in Algorithm 3.

$$Improvement = \frac{Hits}{TotalAccess}\%$$

Algorithm 3 Interval LFU Replacement

```
while 1 do
  Read currentAccess
  B = FindBank(CurrentLBA)
  incr AccessCount for Bank[B]
  if B is not in SSD then
    Load bank B from HDD //miss
  else
    Load bank B from SSD //hit
  end if
  if new interval then
    Sort Bank[][] in descending order of AccessCount
    Make replacements
  end if
  Next access
end while
```

www.manaraa.com

4.2 History based Interval LFU Replacement

Interval LFU replacement replacement policy base their replacement decisions on a single interval. History based interval LFU policy attempts to make replacement decision based on last k intervals compared to a single interval to make more reliable prediction. Intuitively, by considering last several intervals we can capture the usage pattern. For example, a bank that has been accessed for last few intervals versus another bank that was accessed in the last interval, former is more likely to be accessed in the current interval as well. Therefore, we propose a weighted score as follows. Weighted score for each back gets calculated by below formula.

$$\begin{aligned} WeightedScore &= W_i * AccessCount_i \\ &+W_{i-1} * AccessCount_{i-1} \\ &+W_{i-2} * AccessCount_{i-2} \end{aligned}$$

Here i denotes an interval and W_i denotes weight for the interval. Also, $W_i > W_{i-1} > W_{i-2}$ so that recent interval has the most weight. So for every interval, banks will be sorted in the descending order of *WeightedScore*. Similar to interval LFU replacement policy, replacements will be done for the banks which do not already reside in SSD. The main operations of history based interval LFU replacement is summarized in Algorithm 4.

4.3 Conservative Interval LFU Replacement

Interval LFU replacement relies on a critical assumption that the banks accessed most in previous interval, will continue to be accessed in the current interval as well. Sometimes this assumption is not true, which incurs wasteful replacements. Because SSD has limited erase-write cycles, wasteful writes would wear-off SSD's faster, while not improving performance. Conservative LFU replacement policy attempts to minimize the wasteful replacements while ensuring minimal performance degradation. Similar to interval LFU policy, replacement contender list is generated. This replacement contender list consists of banks that are accessed most in previous interval but was not residing in SSD. We will then keep analyzing subsequent accesses and

www.manaraa.com

Algorithm 4 History based Interval LFU Replacement

```
while 1 do
  Read currentAccess
  B = FindBank(CurrentLBA)
  incr AccessCount for Bank[B]
  if B is not in SSD then
    Load bank B from HDD //miss
  else
    Load bank B from SSD //hit
  end if
  if new interval then
    WeightedScore = CalculateWeightedScore(B)
    Sort Bank[][] in descending order of WeightedScore
    Make replacements
  end if
  Next access
end while
```

monitor if we access these replacement contender banks or not. If a bank is in replacement contender list and gets accessed in current interval, we make a replacement to get that bank in SSD. We check for certain "threshold" number of accesses to a bank, before making the replacement. After a bank gets accessed more than certain threshold, we are more confident that the replacement contender is actually being accessed in this interval so we replace least frequently used SSD entry (bottom entry in the sorted list) with the contender. We also make sure that we do not replace any contender with another contender to avoid other wasteful replacements. The main operations of conservative interval LFU replacement is summarized in Algorithm 5.

4.4 Conservative History based Interval LFU Replacemen

We discussed two variations of LFU replacement policy to improve our performance and reducing the number of replacements. History based approach tries to decide what should be in SSD by analyzing past k intervals compared to a single interval to make more reliable prediction. Conservative approach tries to limit the number of replacement by avoiding wasteful replacement. History based conservative approach tries to use both of these variations to make more reliable prediction as well as prevent wasteful replacements. The main operations of conservative history based interval LFU replacement is summarized in Algorithm 6.

www.manaraa.com

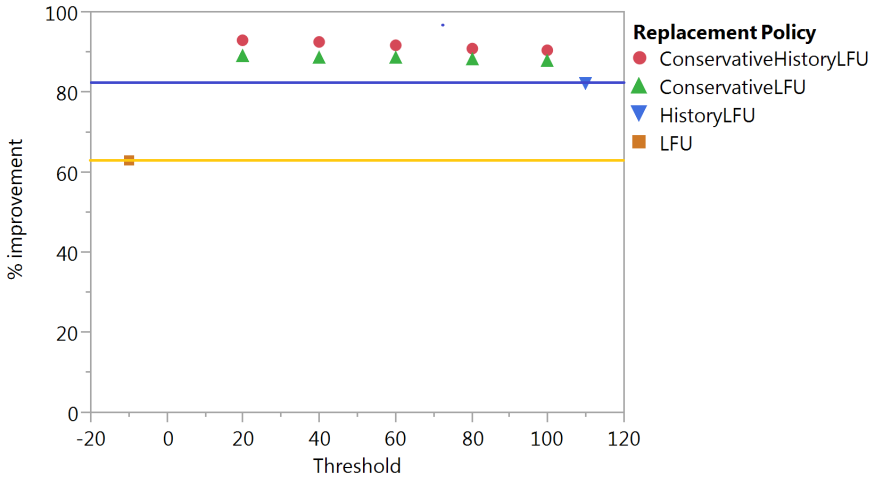
Algorithm 5 Write Conservative Interval LFU Replacement

```
while 1 do
  Read currentAccess
  B = FindBank(CurrentLBA)
  incr AccessCount for Bank[B]
  if B is not in SSD then
    Load bank B from HDD //miss
    if B is a Contender then
      if Bank[B][AccessCount] >= Threshold then
        Replace bottom entry with B
      end if
    end if
  else
    Load bank B from SSD //hit
  end if
  if new interval then
    Sort Bank[][] in descending order of AccessCount
    Replacement Contender = Top (Bank)
  end if
  Next access
end while
```

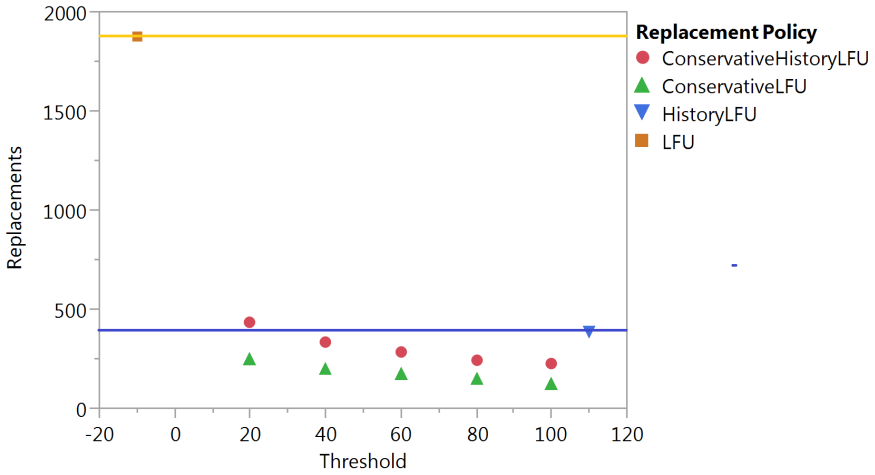
Algorithm 6 Conservative History based Interval LFU Replacement

```
while 1 do
  Read currentAccess
  B = FindBank(CurrentLBA)
  incr AccessCount for Bank[B]
  if B is not in SSD then
    Load bank B from HDD //miss
    if B is a Contender then
      if WeightedScore >= Threshold then
        Replace bottom entry with B
      end if
    end if
  else
    Load bank B from SSD //hit
  end if
  if new interval then
    = CalculateWeightedScore(B)
    Sort Bank[][] in descending order of WeightedScore
    Replacement Contender = Top (Bank)
  end if
  Next access
end while
```

www.manaraa.com



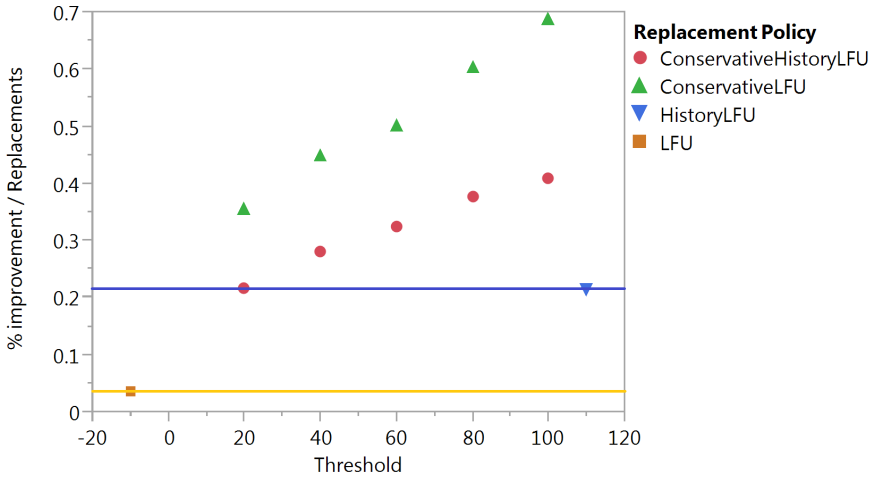
(a) % Improvement vs Threshold



(b) Replacements vs Threshold

Figure 4.1: Replacement Strategies Simulation Results-DisplayAdsPayLoad

www.manaraa.com



(c) PolicyEffectiveness vs Threshold

Figure 4.1: (Continued)

Replacement Strategies Simulation Results-DisplayAdsPayLoad

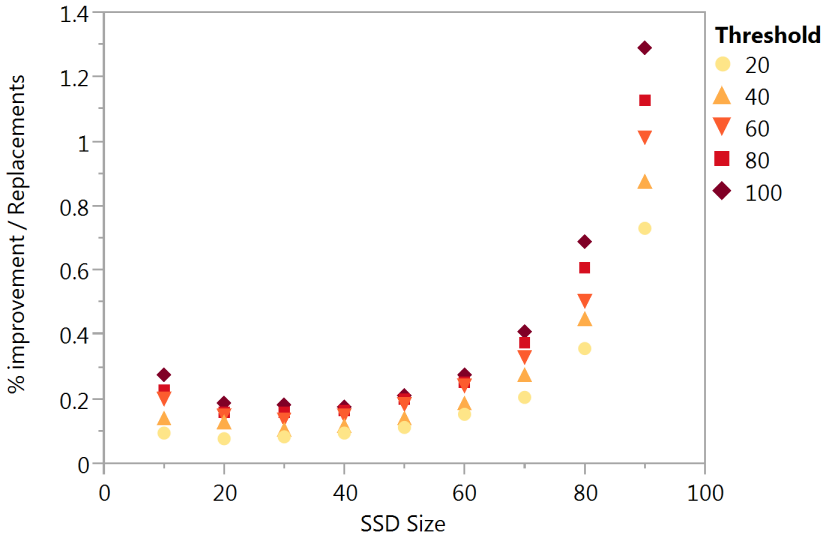


Figure 4.2: Conservative LFU Policy-DisplayPayload

www.manaraa.com

Table 4.1: Replacement Strategies Simulation Results Table

Replacement Strategy	***DisplayAdsPayload***			
	Threshold	***% Improvement***	***Replacements***	***%improvement/ Replacements***
LFU	N/A	63.11	1871	0.0337
Conservative LFU	20	89.21	250	0.3568
	40	88.80	198	0.4485
	60	88.59	177	0.5005
	80	88.24	146	0.6044
	100	88.06	128	0.6880
History LFU	N/A	81.97	386	0.2124
Conservative History LFU	20	92.76	431	0.2152
	40	92.32	331	0.2789
	60	91.51	283	0.3234
	80	90.97	241	0.3775
	100	90.60	222	0.4081

4.5 Replacement Strategies Simulation Results

All three replacement strategies are simulated on the same workloads used for spatial locality analysis. We use a quality metric defined as

$$PolicyEffectiveness = \frac{\%imrovement}{Replacements}$$

For a given SSD size, we compare effectiveness of each policy. The table 4.1 shows results for DisplayAdsPayload application for a one SSD size. We plot % improvement, number of replacements and *PolicyEffectiveness* in Figure 4.1. The plot shows that conservative LFU scheme gives more improvement with lower replacements. For example, for a threshold of 80 conservative LFU has 12.8x lesser replacements and 15% larger improvement compared to LFU replacement. In case of history based conservative LFU, the % improvement is 27% higher, where as the replacements are 7.8x lesser. So *PolicyEffectiveness* is higher for Conservative LFU and History based conservative LFU.

www.manaraa.com

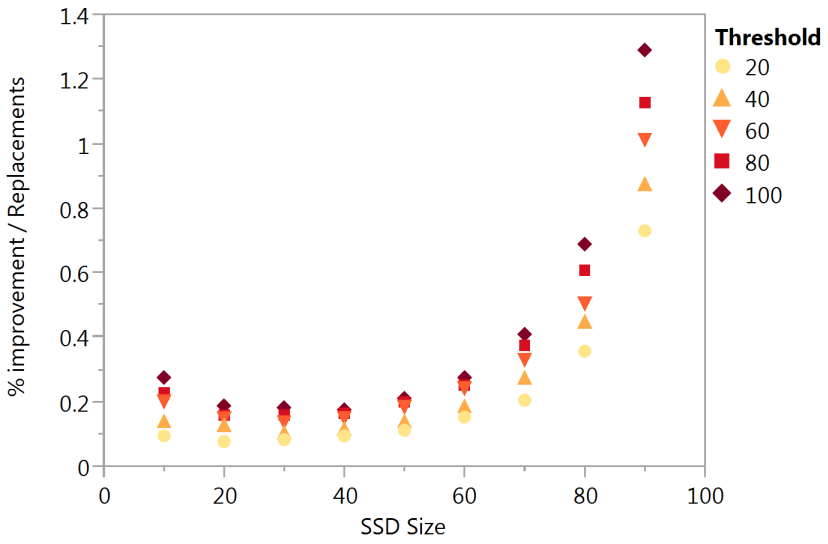


Figure 4.3: Conservative History based LFU Policy-DisplayPayload

Threshold Analysis

We analyze effect of threshold on *PolicyEffectiveness* for DisplayAdsPayload application. Figure 4.2 and 4.3 shows plot of *PolicyEffectiveness* vs SSD Size. Plot shows that *PolicyEffectiveness* is highest when threshold is highest for each SSD size. The reason behind, as the threshold increases, it becomes tougher for a contender bank to replace a bank in SSD. This reduces the number of replacements. As number of replacements decreases, *PolicyEffectiveness* increases.

The plot also shows that effect of different threshold is much more for higher SSD size compared to lower SSD size. For the higher SSD size, decrement in number of replacements as we increase threshold is much lower. As replacements does not decrease much, *PolicyEffectiveness* increases more as % improvement increases. For example, for conservative LFU replacement, replacements reduces by 369 as threshold increases from 20 to 100 for SSD size 40. But for SSD size 80 replacements reduces by 122. Therefore effect of increasing threshold is more for higher SSD size.

www.manaraa.com

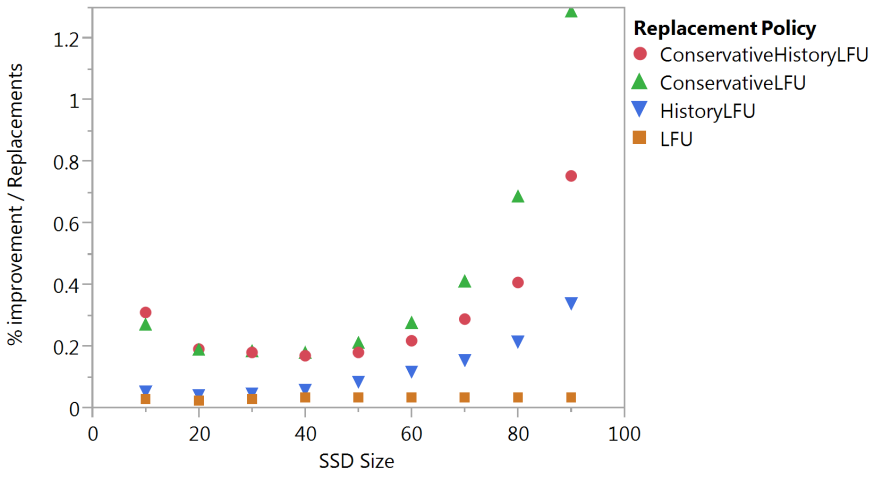


Figure 4.4: Comparison of all four policies - DisplayAdsPayload

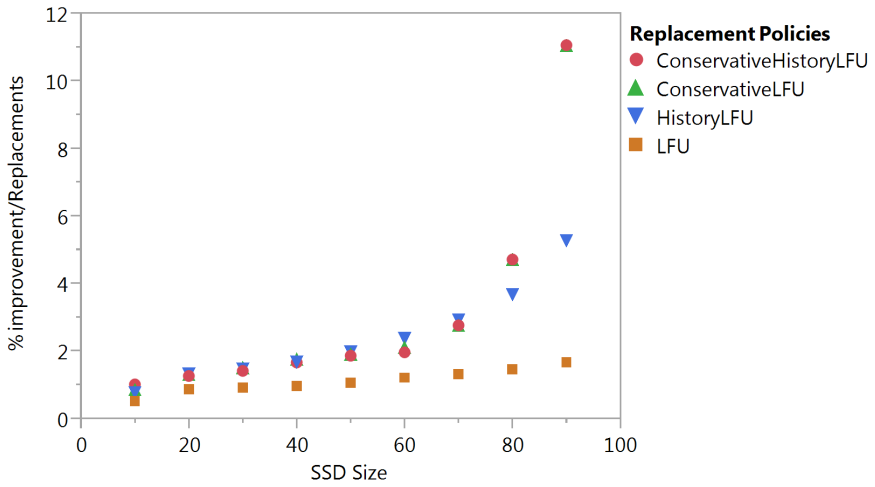


Figure 4.5: Comparison of all four policies - RadiusBackEndSQLServer

www.manaraa.com

Comparison

We compare all four policies for different SSD sizes to see how does each policy improve the performance as SSD size increases. Figure 4.4 and 4.5 shows plot of *PolicyEffectiveness* vs SSD size. DisplayAdsPayload application has lower locality and RadiusBackEndSQLServer has good locality according to our analysis in Section 3.1. The plots shows that *PolicyEffectiveness* is much higher for RadiusBackEndSQLServer compared to DisplayAdsPayload. The plots also shows that History LFU performs better than LFU replacement policy in all cases. Also, conservative LFU and conservative history based LFU policies perform better than LFU and history LFU policies.

www.manaraa.com

CHAPTER 5. SUMMARY AND CONCLUSION

With the emergence of SSD, hybrid drives that trade-off disk latency with the cost are gaining more importance. However, its not clear how much should be the proportion of SSD and HDD in the disk. We carried out a spatial locality based analysis to show that while some applications would be benefited in terms of improved disk access latency by $<10\%$ SSD, other applications, for the same benefit, would require $>60\%$ SSD.

On one hand, data in the SSD needs to be replaced at regular intervals to maintain its utility, on the other hand, SSD has very limited life write cycles before it wears off. To this effect, we proposed four different replacement policies to maximize the performance with minimal replacements. Our replacement policies demonstrate the trade-off between performance improvement and the number of replacements.

www.manaraa.com

BIBLIOGRAPHY

(1956). IBM 305 RAMAC. *IBM Hardware Archives*. 1

(2014). The Economics of Intelligent Hybrid Storage. *An Enmotus White Paper*. vi, 3

Badam, A. and Pai, V. S. (2011). Ssdalloc: Hybrid ssd/ram memory management made easy. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 211–224, Berkeley, CA, USA. USENIX Association. 5

Kang, W.-H., Lee, S.-W., and Moon, B. (2012). Flash-based extended cache for higher throughput and faster recovery. *Proceedings of the VLDB Endowment*, 5(11):1615–1626. 5

Lv, Y., Cui, B., Chen, X., and Li, J. (2013). Hotness-aware buffer management for flash-based hybrid storage systems. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1631–1636. ACM. 5

Micheloni, R., Marelli, A., and Eshghi, K. (2013). *Inside Solid State Drives (SSDs)*. Springer Series in Advanced Microelectronics. Springer Netherlands. 2

No, J. (2012). Nand flash memory-based hybrid file system for high i/o performance. *Journal of Parallel and Distributed Computing*, 72(12):1680–1695. 2, 4

Piramanayagam, S. and Chong, T. C. (2011). *Developments in data storage: materials perspective*. John Wiley & Sons. 1

Wood, R. (2009). Future hard disk drive systems. *Journal of Magnetism and Magnetic Materials*, 321(6):555 – 561. Current Perspectives: Perpendicular Recording. 1

www.manaraa.com