Performance Evaluation of Traditional Caching Policies on A Large System with Petabytes of Data

Ribel Fares [1], Brian Romoser [2], Ziliang Zong [*3], Mais Nijim [4], Xiao Qin [5]

[1,2,3]*Computer Science Department, Texas State University (*: corresponding author)*
[4]*Department of Electrical Engineering and Computer Science, Texas A&M University-Kingsville*
[5]*Department of Computer Science and Software Engineering, Auburn University*
[1,2,3]{rf1190, br1170, zz11}@txstate.edu
[4]mais.nijim@tamuk.edu [5]xqin@auburn.edu

***Abstract*—Caching is widely known to be an effective method for improving I/O performance by storing frequently used data on higher speed storage components. However, most existing studies that focus on caching performance evaluate fairly small files populating a relatively small cache. Few reports are available that detail the performance of traditional cache replacement policies on extremely large caches. Do such traditional caching policies still work effectively when applied to systems with petabytes of data? In this paper, we comprehensively evaluate the performance of several cache policies, which include First-In-First-Out (FIFO), Least Recently Used (LRU) and Least Frequently Used (LFU), on the global satellite imagery distribution application maintained by the U.S. Geological Survey (USGS) Earth Resources Observation and Science Center (EROS). Evidence is presented suggesting traditional caching policies are capable of providing performance gains when applied to large data sets as with smaller data sets. Our evaluation is based on approximately three million real-world satellite images download requests representing global user download behavior since October 2008.**

I. Introduction

Today, we are in an era of data explosion. Google has recently reported that the massive amounts of data accessible via the Internet has rapidly increased from 5 exabytes in 2002 to 281 exabytes in 2009 [1]. On the one hand, an increasing demand of data-intensive applications requires consistent innovation on storage technologies for efficient sensing, storing, and sharing information. For instance, Amazon developed its specialized distributed database management system, ***Dynamo*** [2], to fit the needs of its expansive e-commerce services. Facebook designed a custom object store infrastructure, ***Haystack*** [3], to both service the half million photo requests per second and consistently maintain the more than 1.5 petabytes of image data.

On the other hand, many legacy systems and applications, employing older, more traditional storage technologies, are facing or will soon face the challenge of dealing with vast amounts of data at petabyte level. Whether or not traditional technologies can survive in such large systems deserves a much more comprehensive level of research. For example, in an environment where only a small subset of a large amount of data is required, making use of a small, high-speed intermediary storage device as the cache for performance gains [23], [29] is a well-researched area in computer science. However, most existing cache performance studies evaluate fairly small files populating a relatively small cache. Few reports discussed the performance of traditional cache replacement policies on extremely large systems. Do traditional caching policies still work effectively when applied to real world applications with petabytes of data?

In this paper, we comprehensively evaluate the performance of three traditional caching replacement policies, which include First-In-First-Out (FIFO), Least Recently Used (LRU) and Least Frequently Used (LFU), on the global satellite imagery distribution application maintained by the U.S. Geological Survey (USGS) Earth Resources Observation and Science Center (EROS). This application is a collaborative effort between USGS and the National Aeronautics and Space Administration (NASA). Starting from 1972, NASA's Landsat program has succeeded in archiving over 4 petabytes of satellite imagery of Earth over the past 39 years. These satellite images contain critical information about Earth's topographic, climactic, and geographic conditions past and present. Before 2008, the workload of USGS EROS storage center was relatively low because only a small group of researchers could afford to access these costly satellite images. However, in October 2008, EROS decided to make their vast amounts of satellite data freely available to global researchers through the EarthExplorer [4] and the Global Visualization Viewer [5] web portals. This new policy caused dramatic changes in workload due to rapidly increased global download requests.

In order to provide fast and reliable satellite imagery distribution service to global researchers, the USGS EROS proposed to house its massive satellite images in long term archived (LTA) format using a Tiered Storage System (TSS). Table I outlines the tiered storage components in the USGS EROS satellite imagery distribution system. All satellite images are first stored in the tape system in their raw format, which is also referred to as raw data. Once a satellite image (a.k.a. a scene) is requested, its raw data will be found at the tape system and then loaded to the SATA system for image processing and optimization. The processed images will then be fetched to the FTP server for user download. Since the proposed size of the FTP server (100 TB) is merely one percent of the size of the tape system (10 PB), the FTP server may be viewed as a cache for the tapes. In addition, the penalty of missing a scene in the FTP server is significantly high (usually about 20 - 30 minutes) because finding and processing a satellite image from its raw format to a download-ready format is very time consuming. Therefore, effectively caching popular scenes on the FTP server will play a critical role in improving the performance of the EROS satellite image distribution system.

TABLE I
USGS EROS Long Term Archival (LTA) Storage Architecture

Tier	Model	Capacity	Hardware	Bus Interface
1	Sun/Oracle F5100	100 TB	SSD	SAS/FC
2	IBM DS3400	1 PB	HDD	SATA
3	Sun/Oracle T10K	10 PB	Tape	Infiniband

The organization of the remainder of this paper is as follows. In Section II, we present the background related to caching strategies and the EROS system. Section III explains caching algorithms implementation and the experimental test bed. Section IV provides a comprehensive analysis of the experimental results. Section V discusses the related work and Section VI concludes our study and discusses future research directions.

II. Background

A. Traditional Caching Strategies

In this study, we implement three well-known and widely used cache replacement algorithms.

1) First in First out (FIFO): FIFO is the simplest cache replacement policy, which evicts the earliest entry in cache when cache replacement happens. No action is taken on *cache hit*s, i.e., if an entry in cache gets requested again, its position does not change. FIFO is known to be a less effective caching policy in many applications. We consider FIFO as the baseline policy in this paper. Other policies can be evaluated by their improvement compared to FIFO.

2) Least Recently Used (LRU): LRU has a small but critical difference than FIFO. First entry in the cache still gets evicted first except when there is a *cache hit*. In case of a *cache hit*, the entry is removed from its current position and re-inserted back to the cache. Thus, such an entry becomes the last entry in the cache. LRU leads to the least recently used entry being the first removed. This policy has proven to be effective where the most recently requested entries are more likely to be requested again (temporal locality). While many variations of LRU have been introduced [6], [7], [8], [9], [10], we have implemented it in its pure form as described above.

3) Least Frequently Used (LFU): LFU is another commonly used cache replacement algorithm. LFU exploits the overall popularity of entries rather than their recency. LFU caches sort their entries by their overall popularity. The least popular item is always chosen for eviction. In EROS system, each satellite image is required to stay in cache for a minimum of 7 days because users who requested the image may not be able to download it immediately. This policy makes our implementation of LFU slightly deviate from its original form. If many requests for the least popular satellite image in cache occur within less than 7 days, it is not removed. Therefore, in our implementation of LFU, the least popular image that is requested within more than 7 days is removed first. The LFU policy has been shown to be effective in cases where the popular items tend to be requested frequently.

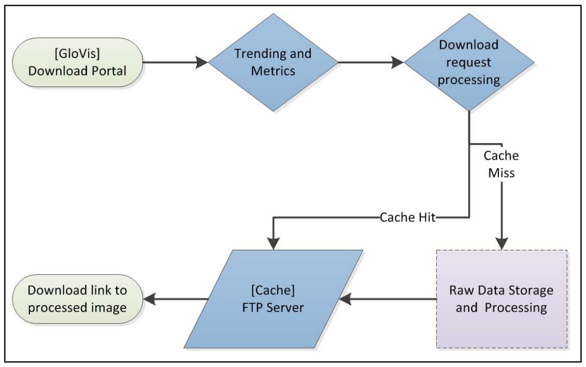

[GloVis] Download Portal
Trending and Metrics
Download request processing
Cache Miss
Cache Hit
Download link to processed image
[Cache] FTP Server
Raw Data Storage and Processing

Fig. 1. Flowchart of the download request process at EROS.

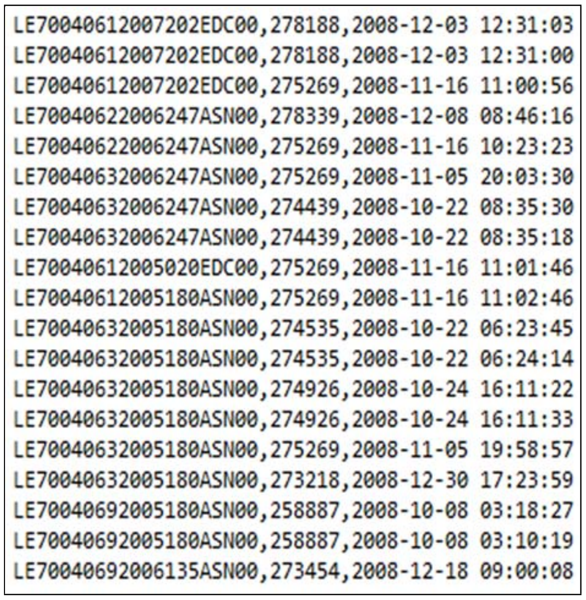
B. EROS System Overview

Fig. 1 illustrates how the EROS satellite image distribution system processes download requests. Global users of the system are able to send requests for a set of one or more satellite images from either the USGS EarthExplorer [4] web portal or the Global Visualization Viewer [5] web interface. Following submission , users are provided with a link to a publicly accessible FTP server from which they may download their requested images. However, the raw satellite imagery used to generate download-ready scenes must first be processed according to each individual user's specifications, which may impose significant delay between a user requesting a scene and being able to access it.

Once a user submits a set of requested images, each image is individually tested to determine if it is present in the FTP server, which acts as a cache for processed scenes. If a scene is present in the FTP server at the time it is requested, no further processing is required and a download link is generated for the satellite image files and sent to the user. If, however, a scene does not exist within the FTP disk, a request is sent to the image processing system, which must retrieve the relevant raw data from the magnetic tape and apply appropriate filtering and processing to create the download-ready satellite image that a user requests.

C. EROS Data Format

Every time a user requests for a satellite image, a scene acquisition record will be appended to the log file. To date, EROS provided us a log file with a total of 3,958,831 records of global user download requests spanning a three year period of time from October 2008 to October 2011. Each line within the file represents an individual request for a processed satellite image file, as would be stored in the cache. Every line in the log file stores three comma-separated values, as shown in Fig. 2.



```
LE70040612007202EDC00,278188,2008-12-03 12:31:03
LE70040612007202EDC00,278188,2008-12-03 12:31:00
LE70040612007202EDC00,275269,2008-11-16 11:00:56
LE70040622006247ASN00,278339,2008-12-08 08:46:16
LE70040622006247ASN00,275269,2008-11-16 10:23:23
LE70040632006247ASN00,275269,2008-11-05 20:03:30
LE70040632006247ASN00,274439,2008-10-22 08:35:30
LE70040632006247ASN00,274439,2008-10-22 08:35:18
LE70040612005020EDC00,275269,2008-11-16 11:01:46
LE70040612005180ASN00,275269,2008-11-16 11:02:46
LE70040632005180ASN00,274535,2008-10-22 06:23:45
LE70040632005180ASN00,274535,2008-10-22 06:24:14
LE70040632005180ASN00,274926,2008-10-24 16:11:22
LE70040632005180ASN00,274926,2008-10-24 16:11:33
LE70040632005180ASN00,275269,2008-11-05 19:58:57
LE70040632005180ASN00,273218,2008-12-30 17:23:59
LE70040692005180ASN00,258887,2008-10-08 03:18:27
LE70040692005180ASN00,258887,2008-10-08 03:10:19
LE70040692006135ASN00,273454,2008-12-18 09:00:08
```

Fig. 2. Excerpt of several lines from the logs provided by EROS.

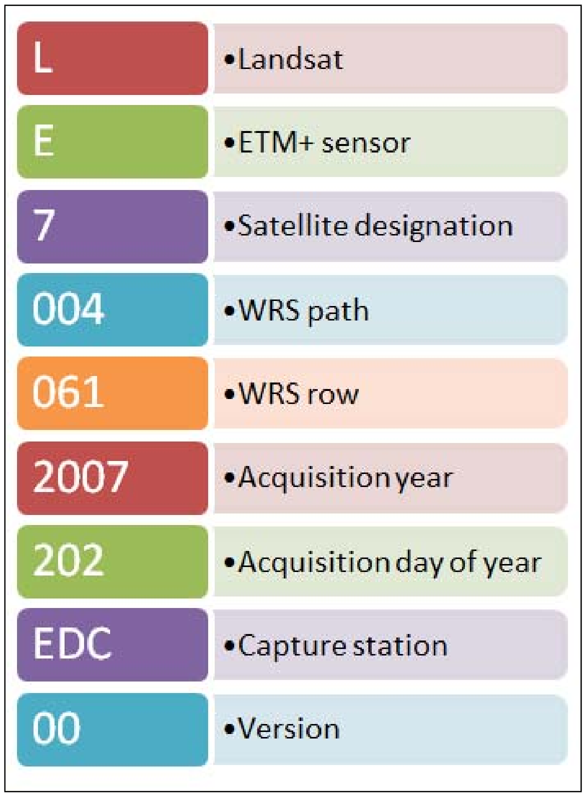


Fig. 3. Detail of an SAID within the logs provided by EROS.

The initial entry for a line in the log contains a scene-acquisition ID (SAID), which is itself a container for nine attributes defining the requested image, detailed in Fig. 3. Within a SAID, the first three characters tell which satellite and sensor captured the image being requested. Following the satellite designation, the next six digits establish the geographical location being imaged in the format defined by the *Worldwide Reference System* [11], a global grid system specified by NASA. Following the geographical coordinates, a further cluster of seven digits provides the date that a satellite image was acquired. A SAID's last five characters store the terrestrial station responsible for receiving the raw data, as well as a version notation.

The latter two values in each line within the log file represent the user responsible for the request submission as well as the date and time at which the request was submitted. Each user must login one of the aforementioned web portals using the unique identification number assigned to him/her by the EROS satellite imagery distribution system before they can send requests. We utilize the unique user id to monitor abnormal request patterns on a by-requestor basis and take such instances into special consideration. For example, we noticed that the number of satellite images requested by different users varies significantly. Therefore, we group users to two categories - *aggressive users* and *nonaggressive users*. We will describe the behavior of aggressive users and discuss the impact of aggressive users on the system performance in later sections.

III. Experiment Environment Description

In order to properly utilize the data provided by EROS, we build a simulator that will provide an environment as similar as possible to the real-world environment in which the data was generated. This ensures that the historical file requests are processed in a style consistent with their natural environment and the resulting measurements accurately reflect real system performance. Additionally, we also make some assumptions to simply our simulation without the loss of generality.

A. Simulation Assumptions

We attempt to recreate the effects of the procedures followed at EROS when processing the provided data. The major constraints and assumptions included in the simulator are as follows:

- In our simulator, *cache miss*es are assumed to be processed instantaneously and we assume the processing time of a satellite image from its raw format to download-ready format is 20 minutes in average. In other words, we assume that a scene missed in the FTP server will be available for download in 20 minutes. If a scene can be found in the FTP disk (i.e. a *cache hit*) after the initial request, the waiting time is negligible (zero). In the real-world system, after a satellite image is requested but missed on the FTP server, a work order will be generated and placed in a job queue to be scheduled. In addition, the processing time of satellite imagery may vary as well. For example, the processing time of a satellite image captured by Landsat 7 is longer than a satellite image captured by Landsat 4 or 5 because the

Landsat 7 satellite images can provide higher resolution. Therefore, the actual processing time of a satellite image maybe longer than 20 minutes in the actual EROS system.

- The EROS system implements a ”cache clean-up” policy in order to prevent the FTP server from reaching 100% capacity by removing a fixed percentage of entries in the cache once a threshold capacity is exceeded. Our simulator will initiate a clean-up procedure once the cache usage reaches 90% of its maximum capacity, at which point the cache is cleared until its usage is reduced to 45% of maximum capacity.
- Due to the fact that satellite images are not always immediately available, we cannot assume that a user will access a satellite image the moment it becomes available. Email notifications are sent to users immediately once the requested satellite images are populated to the FTP server, but the user may not be able to instantly begin download of the file. Therefore, we allow 7 days for users to download requested images through the e-mailed download link. As a result, each requested image is guaranteed to stay in the FTP server for a minimum of a 7-day period, regardless of clean-up policy.
- Real-world satellite image sizes vary on an image-by-image basis from 200MB to 300MB. In our simulator, we assume that the size of each satellite image is identical (250 MB).

B. Aggressive vs. Non-Aggressive Users

To utilize the unique user ID attribute contained within the log file provided by EROS, we find that satellite image requests originate from 63,447 unique users, resulting in an average of 59.6 satellite images requested per user with a standard deviation of 1192.9. We investigate the very large standard deviation and observe that there are disproportionately small amounts of users that generate a substantial portion of overall requests to the system. We term these high-volume users *aggressive users*, the top nine of which are surprisingly responsible for 17.5% of the all requests, as shown in Fig. 4.

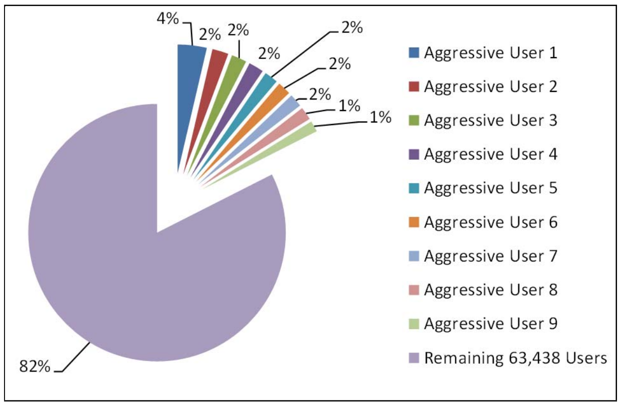


Fig. 4. Aggressive user proportions.

C. Data Pre-processing

We observe the high frequency of identical user ID and SAID combinations from the data provided by EROS. Such a high incidence of identical pairings implies that users often submit identical requests repeatedly. The proximity of these identical queries, in combination with the seven-day hold policy, ensures that these *duplicate requests* will never require additional processing time due to the fact that the ”original” image is guaranteed to remain accessible in the cache for seven days. As a result, we remove all duplicate requests from the provided log file when a user requested an identical SAID within seven days of another such request. After we clean the EROS log file, it contains 2,875,548 total requests. Duplicate requests represent 27.7% of all requests in the original log obtained from EROS. With a reduced overall amount of requests, the top nine *aggressive users* noted previously are now responsible for 12.3% of all requests. Although *aggressive users* can still be seen to generate a substantial number of image requests, cleaning the data did provide a 5.2% decrease in the ratio of *aggressive user* requests to other requests, suggesting that high-volume users could be more likely to repeat previous behavior.

D. Cache Replacement Policy

We implement three fundamental cache replacement policies - First In First Out (FIFO), Least Recently Used (LRU), and Least Frequently Used (LFU). As the cache will not reach 100% capacity, its contents will never require replacement at the block level (as is typical with traditional caches). Instead, replacement policies are used to order the entries in the cache such that when a clean-up occurs, the portion of the cache that is removed contains satellite images that least fit the implemented replacement policy. In our simulation, we simulate cache sizes of 30TB and 60TB to monitor the effect that cache size has on overall cache performance. The given size constraints result in a set of results modeling caches capable of containing 120,000 and 240,000 satellite images (the size of each satellite image is assumed to be 250MB).

IV. Experimental Results

In this study, we conduct a comprehensive performance evaluation of three traditional caching policies on the EROS satellite image distribution system using over 3 million real world download requests. Our experimental results are first analyzed to note how the differing caching policies would affect hit ratio over time. Due to the unique implementation of a ”clean-up” policy, there is no difference in the hit ratio between replacement policies until the first occurrence of a ”clean-up”, triggered once the cache reaches 90% capacity. The first ”clean-up” happens at the same point for all three cache policies, but subsequent ”clean-up”s no longer coincide across multiple policies. Hit ratios decrease for LFU, LRU, and FIFO after a ”clean-up”, naturally, because the cache is forcibly partially purged. Apart from studying the variance in hit ratios as a function of cache properties, we take interest in user behavior as well. *Aggressive users'* impact on cache

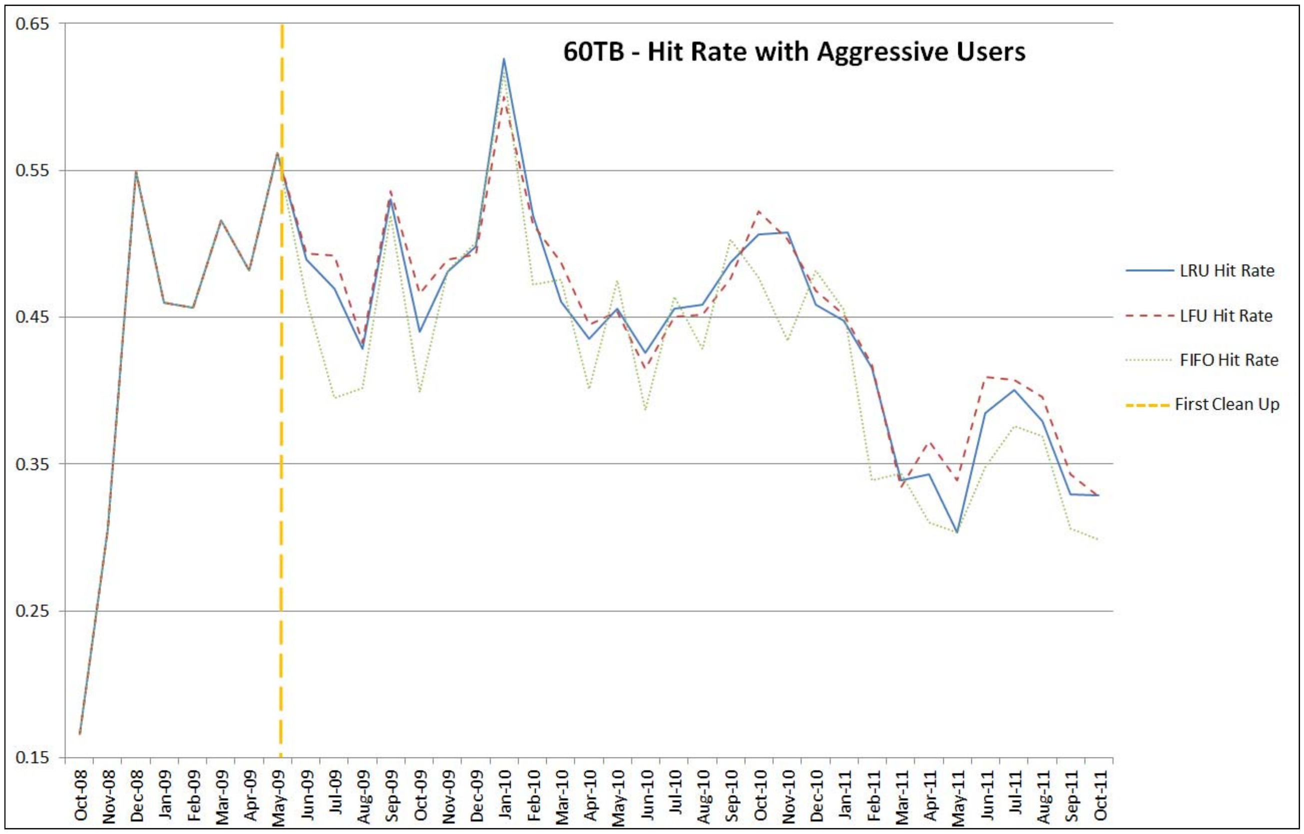


Fig. 5. Monthly hit ratios for all replacement policies using a 60TB cache, *aggressive users* included.

performance is investigated and we find that their inclusion has a noticeable impact on the hit ratio of the tested caching policies.

A. Monthly Hit Ratio Comparison

We conduct a detailed examination of the performance of the 60TB cache across the tested caching replacement policies using both the cleaned and unclaimed EROS log file. We graph the results in Fig. 5, which illustrates the hit ratios of all cache requests on a month-by-month basis. The dashed vertical line represents the month in which the first clean-up occurred, typically coinciding with a drop in hit rate due to the removal of many satellite images from the cache. We do, however note that there is no observable evidence that clean-up consistently leads to a drop in next month's hit rate. We suspect the absence of a drop in hit ratio coinciding with clean-ups may be because the cache policies are effective at preserving images tending to be requested again.

In Fig.6, the monthly hit rate of each of the three cache replacement policies is shown individually. Hollow bars represent months in which a "clean-up" occurred. Hilighting months in which "clean-ups" occur, we find that all replacement policies examined will execute a "clean up", on average, once every 1.8 months. Several brief periods were observed wherein FIFO and the pair of LRU and LFU will exhibit differences in their respective lengths of time between "clean-ups" being initiated, however differences in "rest times" were marginally small (two months at maximum). In addition, we confirm that LRU and LFU behave extremely similarly, with "clean-up" taking place in the same month for the two policies across all resultant data sets.

B. Impact of Aggressive Users

Upon aggregation of the results of our range of simulations, we first determine the hit ratio of each simulation for each combination of cache size and replacement policy, the results of which can be observed in Fig. 7. Across all cache sizes, FIFO preforms the worst, followed by LFU, with LRU preforming the best of the three. Furthermore, we observe that as cache size grows, performance improves as well for all three cache replacement policies tested. The results we observe appear consistent with previous research and do not exhibit any unexpected behaviors.

Using the smaller log file with duplicate requests removed, we calculate the same hit ratio evaluation on a second set of simulations, using the same cache policy and size combinations as above, the results of which are depicted in Fig. 8. We once again note that FIFO is the worst preforming of the three cache replacement policies examined, but with the reduced data set (requests from aggressive users have been

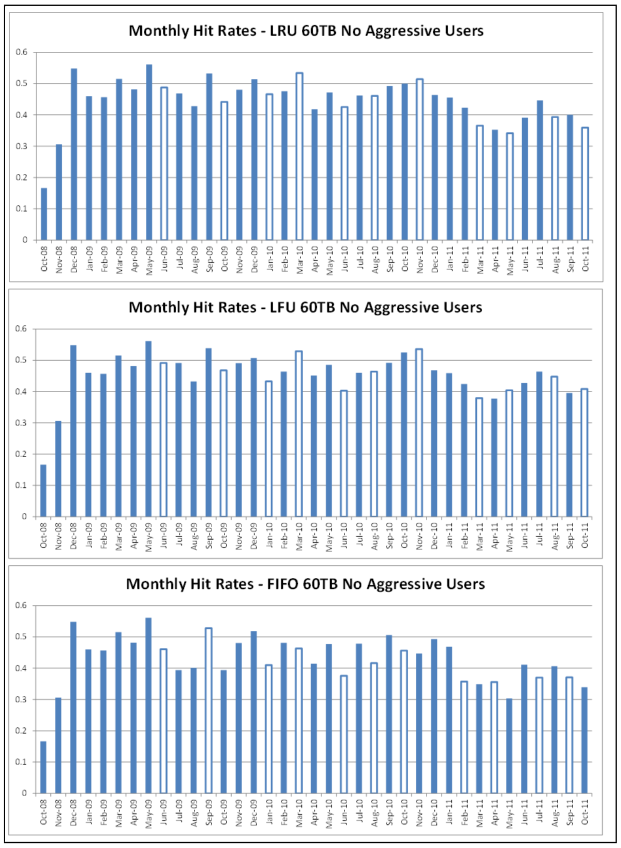


Fig. 6. Monthly hit ratios for all replacement policies using a 60TB cache, *aggressive users* excluded. Months in which ”clean-ups” are denoted by hollow bars.

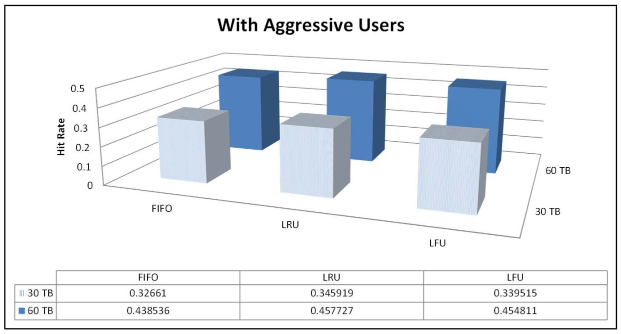
eliminated), LFU now outperforms LRU.

In Fig. 9, we observe the variation amongst monthly hit ratio for the different cache replacement policies. It should be noted that the exclusion of *aggressive users* results in a measurably lower variance for all replacement policies. This seems to suggest that aggressive behavior destabilizes monthly hit rates. Processing users deemed as aggressive in a manner separate from users considered unaggressive may then produce improved performance.

In addition, we find that the LFU cache replacement policy maintains the lowest standard deviation of monthly hit rates whether or not *aggressive users* are considered. Similarly, FIFO maintains the highest range between monthly hit ratios regardless of the inclusion or exclusion of *aggressive users*. Overall, we find that using the LFU cache replacement policy provides us with the most reliably consistent monthly hit ratio whether or not *aggressive users* are processed in the same manner as other users.

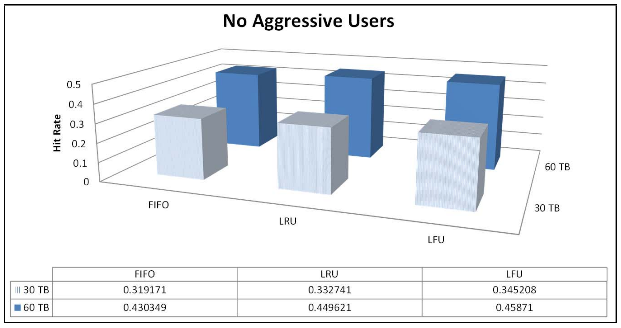
C. Impact of Cache Size

When cache size is shifted from 30 terabytes to 60 terabytes, we record an 11-12% increase in hit ratio, depending on the replacement policy utilized. This shift is demonstrated in Figs. 7 and 8. FIFO, LRU, and LFU all exhibited the same positive shift in hit ratio by an approximately equal amount, suggesting that none of the three policies is more strongly affected by changing cache sizes.



	FIFO	LRU	LFU
30 TB	0.32661	0.345919	0.339515
60 TB	0.438536	0.457727	0.454811

Fig. 7. Average monthly hit ratio with *aggressive users* included.



	FIFO	LRU	LFU
30 TB	0.319171	0.332741	0.345208
60 TB	0.430349	0.449621	0.45871

Fig. 8. Average monthly hit ratio with *aggressive users* excluded.

Standard Deviation of Monthly Hit Rates	LRU	LFU	FIFO
With Aggressive Users	0.08662	0.083058	0.09047
No Aggressive Users	0.075853	0.072411	0.079171

Fig. 9. Average variance in monthly hit ratios for a 60TB cache.

V. Related Work

Caching is one of the most popular technologies to improve I/O performance. The basic idea of caching is to access data on storage components with higher speed, thereby avoiding the penalty of accessing low speed storage components. Caching has been widely accepted and commonly used in many systems and applications, which include web proxy server and web applications [12], [35], [36], database applications [13], data compression applications [16], file systems [14], and operating systems [15]. In a large system that supports data-intensive applications, it is a wise decision to store the majority of data on cheaper low-speed storage components and develop effective caching policies to achieve the twin goals of maximizing performance and minimizing cost. For example, in the EROS satellite imagery distribution application, solid state disks (SSDs) are used to serve as the cache of tapes.

Caching replacement policy is one of the key factors to determine the effectiveness of a caching policy. There are a number of cache replacement policies have been proposed in past research, which include First-In-First-Out(FIFO), Least Recently Used(LRU) [6], [7], [8], [9], [10], Least Frequently Used(LFU) [17], Most Recently Used (MRU) [18], Frequency Based Replacement(FBR) [19], Least-kth-to-last Reference (LRU-k) [20], Least Frequently Recently Used (LFRU) [21] and the Two Queue (2Q) algorithm [22]. Since FIFO, LRU and LFU are the three fundamental cache replacement policy (others are derived from them), we only evaluate the performance of these three policies in our study. Moreover, the implementation of these three cache replacement policies in our study is different from typical implementation because the cache replacement occurs at the file level instead of the block level. This is primarily because the FTP server, which stores satellite image files, is considered as a huge cache of tapes.

VI. Conclusions and Future Work

After observing and analyzing the results generated from the real-world data provided by EROS, we can make the conclusion that traditional approaches to caching can be used to successfully improve performance in environments with large-scale data storage systems. We find that a "clean-up" policy that removes approximately half of the entries in a cache can be successfully implemented without significant decrease in hit ratio. By properly selecting a replacement policy to select items for deletion, we believe that it is possible to virtually eliminate the penalty induced for partially emptying a cache. Of the cache replacement policies tested, we find LFU to be the most tolerable of the "clean-up" policy in terms of resulting reduction in hit ratio.

Throughout the course of evaluation, the FIFO cache replacement policy frequently resulted in a markedly lower hit ratio than either LRU or LFU, although there are a small subset of data points wherein FIFO had a superior hit ratio. The LRU cache replacement policy garnered the highest ratio of all tested replacement policies, but we find that it results in a higher standard deviation in monthly hit ratios than an LFU policy. The Least Frequently Used cache replacement policy results in the best average monthly hit ratio when excluding *aggressive users*, and is outpaced by only 0.29% when *aggressive users* are included with LRU. Furthermore, we find that LFU results in the lowest standard deviation for hit ratio, implying that it would be most suitable for applications requiring a consistent hit ratio.

Doubling the size of cache available resulted in, at most, a 12 percent gain in hit ratio. When employing caches with sizes as large as 60TB, many hard disks must be linked together to provide the required amount of space. Such constraints of scale require us to consider the environment in which the system being studied will operate and the costs involved. We suggest that these additional requirements may reduce the value of the performance gain by simply expanding cache size.

Having established that traditional caching approaches may be successfully applied to large-scale caches and data sets, we believe that a more thorough examination of specialized caching policies would allow us to focus on the optimization of large-scale cache sizes. We observed a relatively small change in hit ratio by doubling the size of cache used, suggesting that shrinking overall cache size in tandem with employing a more effective caching policy would save substantial space and reduce the number of active disks needed to serve as a cache.

Prefetching is another technology that has the potential to significantly increase cache performance [24], [25], [26], [27], [28]. In fact, prefetching can be more effective than simply loading the cache with the popular documents [33]. An effective prefetching policy can assist a cache to improve its hit ratio by as much as 50 percent [32]. Intelligently preloading data enables the realization of performance gains without a higher cost, as a cache that uses prefetching can be equally effective as a cache twice its size that does not [30]. The use of an effective prefetching scheme has been shown to significantly reduce the difference amongst hit ratios of different cache replacement policies, reinforcing the significance of a well-formed prefetching algorithm [33].

To further reduce the size of cache needed and the waiting time required for requested satellite images, historical data on image requests may be mined to generate an intelligent prefetching scheme. By preforming pattern analysis on the log of download requests, we could establish a correlation amongst request behavior and adaptively queue images for processing, eliminating the EROS system's wait time and reducing the miss ratio in cache. Previous research [30], [31], [32], [33], [34] suggests that data mining techniques has the potential to successfully improve performance of storage systems. As future work, we plan to explore the impact of data mining based prefetching techniques on the hit ratio of FTP servers in EROS system.

Acknowledgment

The authors sincerely appreciate the comments and feedback from the anonymous reviewers. Their valuable discussions and thoughts have tremendously helped in improving the quality of this paper. The work reported in this paper is supported by the U.S. National Science Foundation under Grants No. CNS-1212535, CNS-0915762, CNS-1048432, CCF-0845257 (CAREER), CNS-0917137 (CSR), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341(CCLI), and DUE-0830831 (SFS), and the Texas State University Library Research Grant. We also gratefully acknowledge the support from the U.S. Geological Survey (USGS) Earth Resources Observation and Science (EROS) Center.

References

[1] Marissa Mayer, "Innovation at Google: the physics of data", *PARC Forum*,2009, http://www.parc.com/event/936/innovation-at-google.html.

[2] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swami Sivasubramanian, Peter Vosshall and Werner Vogels, “Dynamo: Amazon’s Highly Available Key-Value Store”, *in Proceedings of the 21st ACM Symposium on Operating Systems Principles*, Stevenson, WA, October 2007.

[3] Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel, “Finding a needle in Haystack: Facebooks photo store”, *in Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, Vancouver, Canada, October 2010.

[4] http://earthexplorer.usgs.gov/

[5] http://glovis.usgs.gov/

[6] Alan Jay Smith, “Design of CPU Cache Memories”, *In Proceedings of IEEE TENCON*,1987.

[7] Hong-Tai Chou and David J. Dewitt, “An Evaluation of Buffer Management Strategies for Relational Database Systems”, *in Proceedings of International Conference on Very Large Databases (VLDB)*,1985.

[8] Shaul Dar, Michael J. Franklin, B. Jonsson, Divesh Srivastava, and Michael Tan, “Semantic Data Caching and Replacement”, *in Proceedings of International Conference on Very Large Databases (VLDB)*,1996.

[9] Nimrod Megiddo and Dharmendra S. Modha, “ARC: A Self-Tuning, Low Overhead Replacement Cache”, *in Proceedings of USENIX Conference on File and Storage Technologies (FAST)*,pp. 115-130, 2003.

[10] Yuanyuan Zhou, James Philbin, and Kai Li, “The Multi-Queue Replacement Algorithm for Second Level Buffer Caches”, *in Proceedings of USENIX Technical Conference*, 2001.

[11] The NASA Worldwide Reference System, http://landsat.gsfc.nasa.gov/about/wrs.html.

[12] P. Cao and S. Irani, “Cost-aware WWW proxy caching algorithms”, *in Proceedings of USENIX Conference*, 1997.

[13] J. Z. Teng and R. A. Gumaer, “Managing IBM database 2 buffers to maximize performance”, *IBM System Journal*,vol. 23, no. 2, pp. 211-218, 1984.

[14] M. N. Nelson, B. B. Welch, and J. K. Ousterhout, “Caching in the Sprite network file system”, *ACM Transactions on Computer Systems*,vol. 6, no. 1, pp. 134-154, 1988.

[15] M. J. Bach, “The Design of the UNIX Operating System”, *Englewood Cliffs*, 1986.

[16] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, “A locally adaptive data compression scheme”, *Comm. ACM*,vol. 29, no. 4, pp. 320-330, 1986.

[17] D. L. Willick, D. L. Eager, and R. B. Bunt, “Disk Cache Replacement Policies for Network File Servers”, *in Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 1993.

[18] P. J. Denning, “The Working Set Model for Program Behavior”, *Comm. ACM*, 1968.

[19] J. Robinson and M. Devarakonda, “Data Cache Management Using Frequency-Based Replacement”, *in Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, 1990.

[20] E. J. O’Neil et. al., “The LRU-K Page Replacement Algorithm for Database Disk Buffering”, *in Proceedings of SIGMOD*, 1993.

[21] D. Lee et. al., “On the Existence of a Spectrum of Policies that Subsumes the Least Recently Used (LRU) and Least Frequently Used(LFU) Policies”, *in Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, 1999.

[22] T. Johnson and D. Shasha., “2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm”, *in Proceedings of International Conference on Very Large Databases (VLDB)*, 1994.

[23] M. Baker, S. Asami, E. Deprit, J.K. Ousterhout, and M.I. Seltzer, ”Non-Volatile Memory for Fast, Reliable File Systems”, *In Proceedings of ASPLOS*, pp.10-22, 1992.

[24] Butt, A.R., Gniady, C., Hu, Y.C., “The Performance Impact of Kernel Prefetching on Buffer Cache Replacement Algorithms”, *IEEE Transactions on Computers*, vol. 56, no. 7., pp. 889-908, Jul. 2007.

[25] Grimsrud, K.S., Archibald, J.K., Nelson, B.E., “Multiple Prefetch Adaptive Disk Caching,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 1, pp. 88-103, Feb.1993.

[26] Jeon, H.S., “Practical Buffer Cache Management Scheme Based on Simple Prefetching”, *IEEE Transactions on Consumer Electronics*, vol. 52, no. 3, Aug. 2006.

[27] Jeon, J., Lee, G., Cho, H., Ahn, B., “A Prefetching Web Caching Method Using Adaptive Search Patterns”, *in Proceedings of the IEEE Pacific Rim Conference On Communications, Computers, And Signal Processing*, vol. 1, pp. 37-40, Aug. 2003.

[28] Cao, P., Felten, E.W., Karlin, A.R., Li, K., “A Study of Integrated Prefetching and Caching Strategies”, *in Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, vol. 23, no. 1, pp. 188-197, May 1995,.

[29] Gonzalez, A., Aliagas, C., Valero, M., ”A Data Cache with Multiple Caching Strategies Tuned to Different Types of Locality”, *in Proceedings of the Ninth International Conference on Supercomputing*, pp. 338-347, 1995.

[30] Griffoen, J., Appleton, R., ”Reducing File System Latency Using a Predictive Approach”, *in Proceedings of the Summer USENIX Technical Conference*, vol. 1, pp. 13-13, 1995,.

[31] Azevedo, D., Oliveira, J., ”Application of Data Mining Techniques to the Storage Management and Online Distribution of Satellite Images”, *in Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*, pp. 955-930, 2007.

[32] Nanopoulos, A., Katsaros, D., Manolopoulos, Y. ”A Data Mining Algorithm for Generalized Web Prefetching”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 5, pp. 1155-1169, September 2003.

[33] Lan, B., Bressan, S., Ooi, B., Tan, K., ”Rule-Assisted Prefetching in Web-Server Caching”, *in Proceedings of the Ninth International Conference on Information and Knowledge Management*, pp. 504-511, 2000.

[34] Megiddo, N., Modha, D., ”Outperforming LRU with an Adaptive Replacement Cache Algorithm”, *Computer*, vol. 37 no. 4, pp. 58-65, April 2004.

[35] Arlitt, M., Friedrich, R., Jin, T., ”Performance Evaluation of Web Proxy Cache Replacement Policies”, *Performance Evaluation*, vol. 39 no. 1-4, pp. 149-164, February 2000.

[36] Rizzo, L., Vicisano, L., ”Replacement Policies for a Proxy Cache”, *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 158-170, April 2000.