An Empirical and Architectural Study of Using an SSD-Aware Hybrid Storage System to Improve the Performance of the Data Intensive Applications

A. Aldahlawi, E. El-Araby, S. Suboh, and T. El-Ghazawi

***Abstract*—Processors speed have increased in a steady pace year over year, storage system performance is still the major bottleneck for most computer systems including high performance computers. Many attentions have been given to optimize the storage system speed using various approaches such as caching, intelligent perfecting and scheduling techniques, nevertheless, the storage system remains the performance bottleneck for most computer systems. Solid State Devices (SSD) have lately been used as a cache layer located between the system main memory and the magnetic hard drives in order to create robust and cost effective hybrid storage systems. The reason comes from the growing density of the SSDs at lower prices with main advantage of high random read efficiency compared to magnetic hard drives. These new devices are capable of producing not only exceptional bandwidth, but also random I/O performance that is orders of magnitude better than that of standard rotating mechanical disks due to the absence of moving parts. In this paper, we have conducted an extensive empirical and comparative study of an I/O intensive workload running on hybrid storage system. We have configured an SSD-Aware real system with variable RAM, SSD, Working Sets configurations in order to evaluate the performance gain achieved by utilizing the SSD device as a middle layer between the RAM and the Hard Disk. This attractive middle layer has also motivated us to propose and simulate new SSD-Aware Hybrid Caching Architecture (HCA) that utilizes an SSD as an extended read cache to the main memory. We have developed a Hybrid Cache Simulator to explore the design space of the hybrid cache using both performance and cost metrics and test it for two I/O intensive real system workloads. Our simulated architecture along with the real system experiments have shown that SSD can effectively be used as a cache extension to the main memory to minimize the disk hits ratio that would otherwise cause substantial delay in workload performance.**

***Index Terms*—Hybrid storage system, multi-level caching, data intensive application, solid state devices, SSD-aware page replacement policy.**

I. Introduction

In recent years, Processor speeds have increased out pacing the speed of other computer system components. The speed gap between the CPU and the storage devices is negatively affecting the overall system performance, this gap is widen year after year as CPUs becoming much more sophisticated in their advanced internal design and high clock rate. Storage systems dependency on mechanical parts is a major bottleneck for the overall computer system performance, these devices have extremely low latency compared to the processor's speed, the processor has to wait for the data to be retrieved from the right location on the magnetic disk, this operation requires repositioning the reading head of the magnetic device to a new location and therefore wasting valuable processor's time. The absence of moving parts and substantial drop of the SSD's (Solid State Devices) cost has motivated researchers to integrate them within the current storage systems in order to minimize the latency generated by the magnetic devices. An interesting question to ask is how SSDs can be used within the existing memory hierarchy and how much performance gain can we achieve by integrating the SSD as a new layer within the current memory hierarchy to form 3-tiers Hybrid Storage System (HSS). In this paper we investigate this question by characterizing and evaluating the performance of 3-tier memory hierarchy based on Sun's ZFS storage model, we show that the SSD storage layer will improve the performance of an I/O intensive workloads such as pure random read workload and an OnLine Transaction Processing (OLTP) workload. We show that the hybrid storage model has increased the workloads' IOPS of the random read, the OLTP by 17.26x, 2.77x respectively. We also show that the file system IOPS have also increased by 17.08x, 2.76x respectively. The hybrid storage IOPS have also increased by 16.42x, 2.99x, respectively. The integration of SSD into the hybrid storage has also reduced the latency per I/O by 31.77%, 21.57%, respectively. A great gain from our experiment is a substantial reduction of the disk hits ratio; the two workloads have shown 73.01% and 34.67% drops in disk hits ratio respectively. We have also measured the cost in terms of dollar amount per single I/O and show that 33% cost reduction per single I/O for the two workloads. In order to further evaluate the performance of the new 3-tier memory hierarchy and have control over the data flow between storage layers, we extended our research by proposing a new SSD-Aware Hybrid Caching Architecture (HCA) that uses SSD as an extended read cache to main memory.

We have designed an SSD-Aware page replacement policy that takes into consideration the characteristics of the SSD. We have also developed a hybrid cache simulator to test our proposed architecture with two real trace workloads and explore the design space using both performance and cost metrics for variable architectural configuration. We have taken into consideration the continuous drop of the SSD cost

Manuscript received May 10, 2012; revised June 27, 2012. This work was supported in part by the High Performance Computing Lab (HPCL), The George Washington University. Washington DC, USA

A. Aldahlawi is a PhD candidate at Electrical and Computer Engineering Department, The George Washington University Washington DC, 20037, USA (e-mail: dahlawi@gwu.edu).

E. Al-Araby and T. El-Ghazawi, and S. Suboh are with the Department of Electrical Engineering at The Catholic University of America, Washington, DC 20034 USA (e-mail:aly@cua.edu, suboh@ gwu.edu, tarek@gwu.edu).

DOI: 10.7763/IJIEE.2012.V2.197

and studied its impact on the cost per I/O with variable RAM/SSD cost ratios. Our proposed HCA has shown significant performance improvement in both disk hits ratio reduction and SSD hits ratio along with low response time average for the proposed hybrid cache. The proposed architecture has also revealed some interesting correlation between the RAM/SSD size and cost ratios.

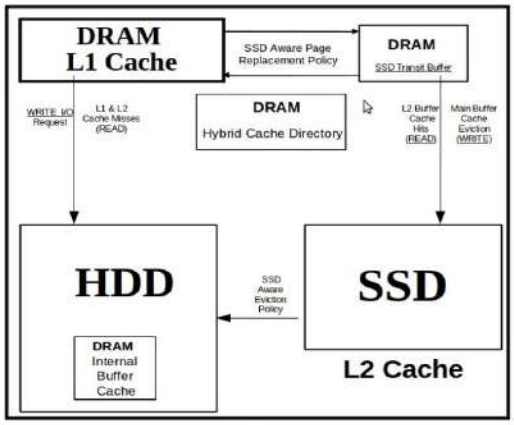

Fig. 1. Hybrid cache architecture.

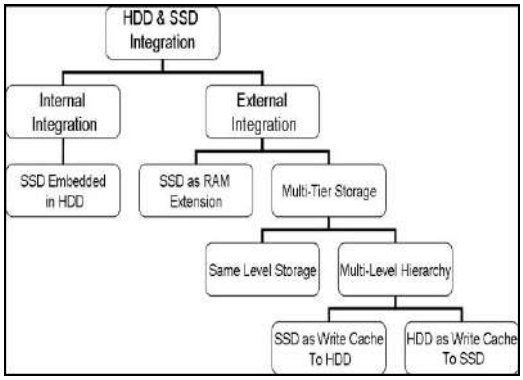

Fig. 2. HDD & SSD integration techniques.

II. Background

Even though storage systems have advanced in terms of their capacity and internal architecture, they still lag behind the speed of current advanced processors. One way to overcome this gap is to add more memory to the system in order to cache as much data as we can in order to minimize the disk accesses, however, even though adding more RAM lets systems store larger working sets in memory, the result is an expensive system that is highly unreliable due to the volatility of the RAM. As a result, many attentions has been given to optimize the storage system speed using various approaches such as caching, intelligent perfecting, and smart scheduling techniques [1]-[4], nevertheless, the storage system remains the performance nightmare for most computer systems performance analyst and designers. Current average seeks time for advanced server drive is usually 4 ms and around 8 ms for desktop computers [5], the rotation time on the other hand depends primarily on the total number of Rounds Per Minutes (RPM) that can be achieved by the storage device. The rotation delay also determines the maximum throughput that can be achieved by the storage device. For example, a high-end magnetic disk with 15K RPM can achieve up to 70MB/s [6]. The seek latency and maximum bandwidth are impairing the processor's speed and therefore reducing the total IOPS achieved by the processor. The problem gets even worse when a computer system is running I/O intensive applications [7] such as Data Base Management System (DBMS), Multimedia, scientific, or any other out-of-core application. As these applications process large amount of data; they require frequent rapid accesses to the storage device in order to run in an acceptable time. Application and system designers are always characterizing the access patterns of these I/O intensive applications in order to minimize the storage device latency. A new hope has emerged with the SSD devices as their internal design does not contain any moving parts, however, most I/O intensive applications developed over the last thirty years are heavily optimized around the model of a rotating disk. As a result, simply replacing a magnetic disk with SSD devices does not yield better performance. These applications have to be redesigned in order to leverage the potential of SSD [6], [8].

III. Related Work

Recent researches have looked into the potential of integrating SSDs with the current storage devices to form a hybrid storage model in order to exploit the potential of both types of storage devices. Integrating SSDs with the current storage systems that uses magnetic disks can be internal or external. See Fig. 2. For internal integration, SSD can be embedded within the hard disk device to provide extended cache for large and fast buffering [9], this approach has been implemented by some storage manufactures [10]-[12]. External integration however raises many fundamental questions to ask, these question are primarily focusing on how SSD can be integrated within the current memory hierarchy. For example, should the operating system consider the SSD as a memory extension [6]-[16] or disk extension?. Additionally, if the operating system considers SSD as disk extension, should it be used as same level storage extension [17], [18] or used as a multi-tier storage hierarchy. Furthermore, in case of multi-tier hierarchy, should SSD be used as a write-back cache to absorb and hide the synchronous write latency generated by some applications [19] or should a magnetic disk be used as a write cache to hide the SSD poor random write [6], [16] or should SSD be used as an external read cache to the storage device. Finally, how different designs are going to impact various applications with different access patterns. Some researchers believe that SSDs have revolutionized the memory hierarchy by presenting themselves as a new storage layer that perfectly fits between the memory and the storage devices and provide substantial improvement in many aspects such as total storage system cost, access latency, and power consumption [20]. Koltsidas and Viglas [13] has proposed a multi-level cache design using SSD, they have established a 3-tier (RAM, SSD and HDD) memory hierarchy and investigated the flow of pages across the memory hierarchy for different workloads. They have defined three invariant schemes namely inclusive, exclusive and lazy, their schemes are related to the coexistence of pages in the memory hierarchy; they argue that page replacement policy is orthogonal to their page coexistence schemes; therefore, their schemes may be used

with any replacement policy. They also claim that their work is analytical study of the techniques engineered in Sun ZFS file system and therefore it is complementary to their work and can provide a suitable implementation basis. However, we observe some key differences between their work and ZFS implementation in terms eviction policy. In ZFS implementation which is based on Adaptive Replacement Cache [21], there is no eviction line between the memory and SSD, the ARC policy evicts page prematurely by monitoring the tail of the RAM cache in order to avoid SSD write latency. We argue here that their first and second schemes are not suitable for and I/O intensive application especially if these application have a random write access pattern unless it is implemented on a write-optimized SSD. In [17], Koltsidas and Viglas also proposed using hybrid storage for a database where they treat both SSD and magnetic disk at the same level of the memory hierarchy meaning that the SSD is not used as a cache for the magnetic disk. They have designed a set of online algorithm that can be applied at the storage controller level to filter out workloads based on their read or write intensity, pages with write intensive will be placed on magnetic to avoid poor write latency of SSD disk while pages with read intensive placed on SSD. Their algorithm is adaptive and will change placement based on the workload access patterns. In a recent work from Microsoft Research [16], Narayanan and Theresa, have found that replacing magnetic disks with SSD is not cost effective for any workload, interestingly; they have concluded that depending on different workloads, SSD capacity per dollar amount has to increase by a factor of 3-3000 in order to break even with magnetic disk. They have also looked at the cost-benefit trade-offs of various SSD and disk hybrid configurations, particularly, using SSD as a RAM cache extension and as write-back cache to hide the latency of the underlying storage device. They have found that only 10% of their workload can benefit from using SSD in a hybrid two-tier configurations due to the current high capacity per dollar SSD cost. Among all hybrid storage models and implementation, Sun has proposed a unique hybrid model [22] and implemented it in ZFS file system [23], their model is 3-tier memory hierarchy of RAM cache (ARC level-1 Cache), SSD(s) level-2 cache (L2ARC) and the underlying storage device(s), see Fig. 3. The uniqueness of their model is based on an Adaptive Replacement Cache [21] which stems from the fact that there is not eviction line between the ARC cache and L2ARC cache. Instead, ZFS file system will monitor the tail of ARC and evict (every 200 ms by default) data blocks prematurely and asynchronously from ARC to L2ARC in order to avoid poor write latency of SSD device. This innovative idea has a price to pay, the L2ARC will need some time to be filled up (warm up time), and this time depends on the workload access patterns and the ratio between the working set size and the ARC size.

IV. Proposed Hybrid Cache Architecture

In order to further evaluate the performance of a new 3-tier memory hierarchy and impalement our own page replacement policy, we have extended our research by proposing a new SSD-Aware Hybrid Caching Architecture that uses SSD as an extended read cache to main memory. A key issue in integrating the SSD within the current memory hierarchy is to optimize the I/O characteristics of the SSD by utilizing its exceptional random read performance while minimizing its write/update latency. When using SSD as a cache extension to RAM, a system designer will have to decide the size of the main memory and the size SSD cache taking into account the cost of each component. Another consideration is the price/performance trade-off for integrating the two caches.

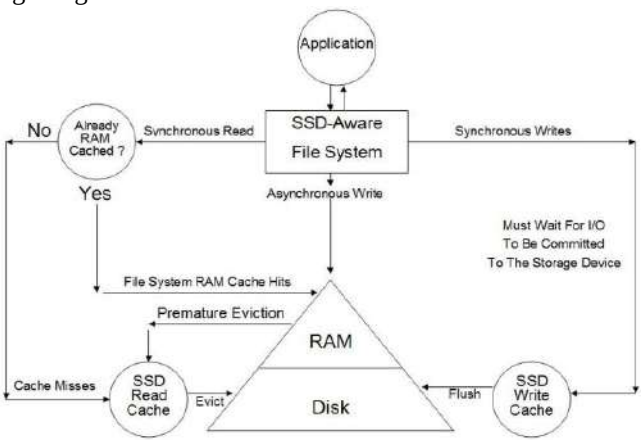

Fig. 3. ZFS implementation.

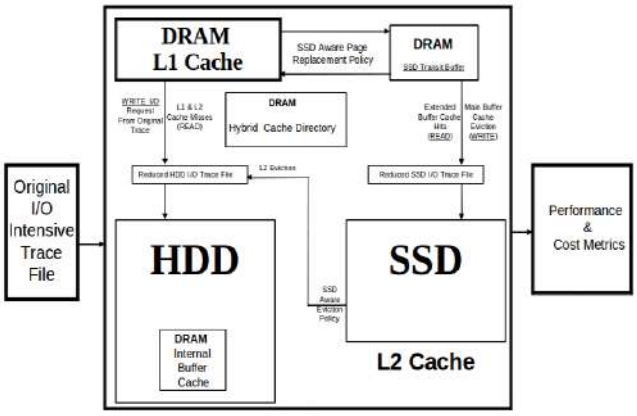

Fig. 4. HCA implementation.

Furthermore, when given a specific budget, the question to ask is what is the right balance that will lead to the best performance while maintaining the lowest cost?. Another issue the system designer has to decide is the rules that govern the migration of data between the two storage layers, to elaborate more, a replacement policy must take into consideration the physical characteristics of the SSD cache in order to utilize its excellent read latency while avoiding excessive update that will lead to poor performance. We are proposing a hybrid cache architecture that utilizes the co-existence of SSD with the memory hierarchy. An abstract view of our proposed Hybrid Cache Architecture is shown in Fig. 1, it consists of a main memory used as L1 Cache, an SSD used as an extended L2 Cache, and magnetic disk (with small internal cache) used for persistent storage. An additional temporary storage used as a Transient Buffer (TB), The transient buffer is a DRAM side buffer that is used to (i) store evicted pages from RAM cache in order to avoid excessive SSD update, (ii) process and evict pages based on their locality to minimize the number of SSD blocks that will

hold the evicted pages from transient buffer. In order to further reduce the probability of excessive SSD update, our architecture adopts the exclusive principle of multi-level caching, i.e., a page cannot coexist in the both caches at the same time, it can either be in RAM Level-1 (L1) Cache or SSD Level-2 (L2) cache. The Architecture also includes a hybrid cache directory that keeps tracks of the location of each page in the hybrid cache. A detailed page replacement algorithm that will utilize the proposed hybrid cache architecture is shown below. In our replacement policy algorithm, when a page is referenced by an application, it is first looked for in L1 Cache, if the page is found, a RAM hits occurs and a page is returned to the application. If the page is not found, it is looked for in the L2 SSD cache, if found, a space needs to be freed (in case L1 Cache is full) in L1 cache before moving the page up in the hierarchy, a hybrid cache directories is also updated to reflect the most recent locations of migrated pages. If the page is not found in either L1 cache or L2 cache, the transient buffer is searched as well, the page is returned if found, otherwise the page is retrieved from the magnetic disk. L1 RAM cache is always checked for free space and a page is evicted (based on Least Frequently Used LFU Policy) to the transient buffer if space is needed to store new page in L1 cache. We varied the transient buffer size during our experiments and observed some improvement in the extended cache average response time and IOPS. Detailed results will follow in subsequent sections.

V. Experimental Settings

In this section, the experimental settings are explained for both real system experimentation as well as the simulation based proposed architecture.

A. SSD-Aware Real System

For a real system setup, we will characterize and evaluate the performance of the hybrid storage model implemented by the Sun's ZFS file system. Our main aim is to investigate the impact of using SSD as a RAM cache extension (L2ARC). We have used Filebench benchmark tool [24] to generate two different workloads namely Random Read and OLTP workloads with 10 GB working set for each workload. We have ran each workload twice, a baseline run with standard 7500 (Round Per Minutes) RPM disk and another run with a hybrid storage by using off-the-shelf Lexar (LJDTT32GB-000-1001D) 32 GB SSD flash. The whole experiments implemented on a Sun Fire X2270 Server with Opensolaris (2009.06). Each experiment ran for six hours to guarantee enough warm-up time for the SSD cache. It is worth indicating that using off-the-shelf SSD was deliberate since our aim is to use the SSD as a read level-2 cache extension to improve the workloads' read performance and therefore not to worry about poor write performance of this cheap device. Although our server has 6 GB of RAM, we have limited the ARC cache to 2 GB to maintain a 5x ratio between the working set size and the ARC cache available to the file system which will guarantee ARC cache misses and therefore L2ARC hits. The file system record size was set to 8K bytes for the OLTP workload benchmark to emulate a real OLTP applications, the other workload was also ran on 8K record size to maintain fair comparison between the two workloads. All the performance data was collected using commands that capture snapshots from the operating system's Kernel. Finally and as an extension to our work, We have conducted twenty seven additional experiments by configuring the RAM sizes to 2,4,6 GB, SSD size to 4,6,8 GB and varying the working sets size to equal 10,15,20 GB. The main objective of these additional experiments is to configure variable RAM, SSD, Working set combinations and monitoring the performance gain achieved by each configuration. We have also used Filebench benchmark to generate the workloads; again, we have ran each workload twice, a baseline and a hybrid storage system by using the same off-the-shelf Lexar SSD. The experiments were implemented on the same machine.

B. HCA Simulation Setup

In simulating our proposed Hybrid Cache Architecture, we have developed a simulator to evaluate the performance of our architecture and integrated it with the standard Disksim simulator [25]. The L2 SSD cache and magnetic disk response times were collected from Disksim along with IOPS for both devices. On the other hand, the hits ratios for each system components were monitored by our simulator. The interaction between our simulator and Diskism is in the form of reduced trace file. Whenever, an I/O miss occurs in either L1 RAM cache or L2 SSD Cache, a new I/O request is added to the corresponding trace file, these two reduced size trace files are passed over to Disksim and performance metrics is gathered and used by our simulator. A detailed view of the interaction with Disksim is shown in Fig. 4. Our simulator was implemented in standard C language and all performance evaluations ran on 131 nodes SUN X2200 M2 X64 cluster machine in order to test large number of experiments. A variable RAM/SSD size ratios were used to evaluate our architecture, we have also used variable RAM/SSD cost ratio in order to consider future anticipated reduction in the SSD cost. We have varied the Transient Buffer size in order to evaluate its impact of our performance metrics. Finally, we ran each experiment twice to compare the baseline with the hybrid cache configuration for two I/O intensive workloads; a financial trace and a Web Server trace obtained from the Storage Performance Council [26].

VI. Results For Real System Experiments

As we have indicated above, each workload ran for six hours in order to allow enough time for the SSD cache to be populated. Based on observations from our experiments, we have seen that L2ARC cache is populated at different rate for each workload, for example, the random read required 4.5 hours to fully migrate the working set from level-1 ARC cache to L2ARC SSD cache, the OLTP workload however, required almost the same time to migrate 80% of the working

set to the L2ARC SSD cache, the remaining 20% working set could not be migrated to the SSD during the remaining 1.5 hours, we characterize this to the fact that the OLTP workload includes updating some data blocks which means that the workload is invalidating these dirty blocks and constantly dropping them from L2ARC SSD due to the workloads behavior.

A. Performance Metrics

In our experiments, we have collected snapshot from the Kernel's performance data using some system commands that reflects real time performance data every 60 seconds. The main aim of our empirical study was to evaluate the hybrid storage model performance using some critical metrics such as, the ARC hits ratio, the L2ARC hits ratio, the disk hits ratio, the workload's IOPS, the file system's IOPS, the storage system IOPS, the latency per IO, and the dollar amount cost per IO. This last metric was calculated by dividing the total combined cost of the hard disk and SSD ($250) by the total achievable IOPS for all the 360 snapshot intervals. We have measured all these metrics in the baseline run and the hybrid run and compared the performance gain and the reduction in Random Read Workload.

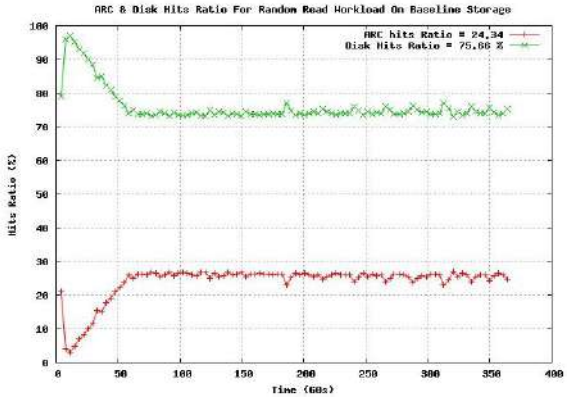

Fig. 5. Baseline hits ratio for random read workload.

This workload generate a pure random read on a 10 GB working set, it involves heavy random seeks within a single large file. Table I shows the improvement of our metrics, we clearly see a substantial improvement of IOPS rate by 17.26x for the workload , the IOPS for file system and the storage system have also improved by 17.08x and 16.42x respectively.

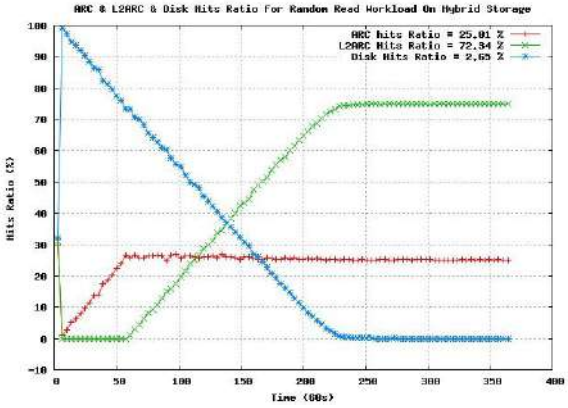

Fig. 6. Hybrid storage hits ratio for random read workload.

Fig. 5 shows that the ARC and Disk Hits ratio for the baseline storage is stable overtime; we also see that the disk hits ratio is more than 75% which is due to the large working set and the limited available ARC cache.

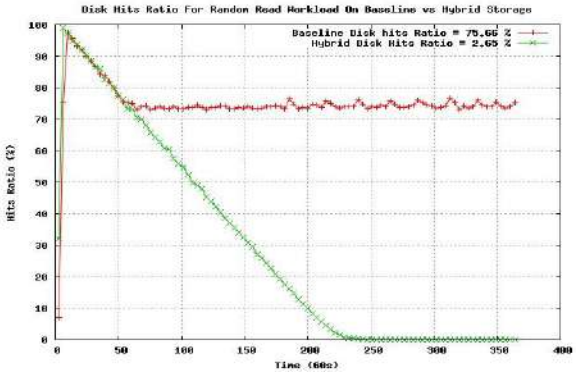

Fig. 7. Disk hits ratio reduction for random read workload.

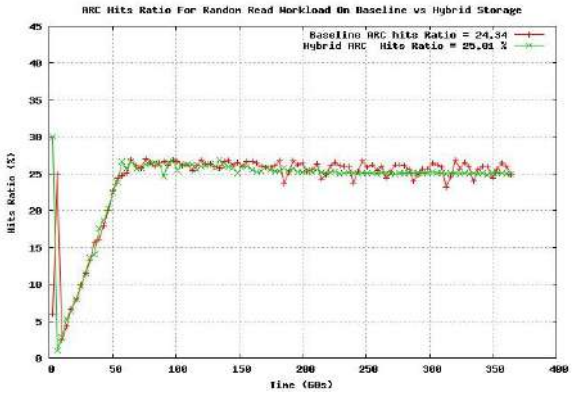

Fig. 8. ARC hits ratio for random read workload.

TABLE I: RANDOM READ WORKLOAD

Experimental Metrics	Baseline	Hybrid	Improvement
Workload AVG IOPS	138.5	2390.4	17.26X
File System AVG IOPS	135.22	2309.6	17.08X
Storage AVG IOPS	105.22	1728.09	16.42
ARC Hits Ratio (%)	24.34	25.01	1.03X
L2ARC Hits Ratio (%)	N/A	72.34	-
Disk Hits Ratio (%)	75.66	2.65	< 73.01 %
AVG Latency Per IO (ms)	0.09	0.04	< 30.77 %
AVG Cost Per IO ($$)	1.89	0.94	< 33.22 %

TABLE II: OTLP WORKLOAD

Experimental Metrics	Baseline	Hybrid	Improvement
Workload AVG IOPS	378.6	1050	2.77X
File System AVG IOPS	184.3	509.31	2.76X
Storage AVG IOPS	140.09	418.51	2.99X
ARC Hits Ratio (%)	53.26	50.29	> 0.94X
L2ARC Hits Ratio (%)	N/A	37.64	-
Disk Hits Ratio (%)	46.74	12.07	< 34.67 %
AVG Latency Per IO (ms)	0.80	0.22	< 21.57 %
AVG Cost Per IO ($$)	1.44	0.71	< 33.02 %

Fig. 6 and Fig. 7 show the different hits ratios when we ran this workload on a hybrid model, the disk hits ratio has been dropped to only 2.65% which is a significant reduction in disk access and substantial performance gain for both the workload and the file system. Fig. 6 also shows how SSD hits ratio is improving over time as data migrates and therefore retrieved from SSD. Fig. 5 and Fig. 8 show that the ARC hits ratio is almost the same for both the baseline and the hybrid storage, we believe that the ratio between the working set size and the ARC size the crucial factor that determine the ARC

hits ratio regardless of the absence or the presence of SSD L2ARC cache. Fig. 9 shows clear speed up of the file system IOPS jumping from 135 to 2309 IOPS as data block being migrated to L2ARC cache. Similar gain is shown in Fig. 10 for the hybrid storage reaching 1728 average IOPS. Fig. 11 also proves that latency per single IO is being reduced over time which is also applicable to the dollar mount cost per single IO shown in Fig. 12. Fig. 13 is interestingly showing the source of retrieving data overtime; it shows that as data blocks being migrated to L2ARC cache, the storage system will retrieves data blocks from SSD rather than hard disk storage which means higher IOPS rate.

Fig. 9. File system IOPS for random read workload.

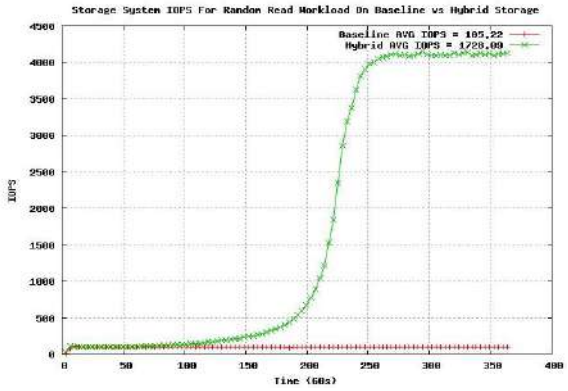

Fig. 10. Storage system IOPS for random read workload.

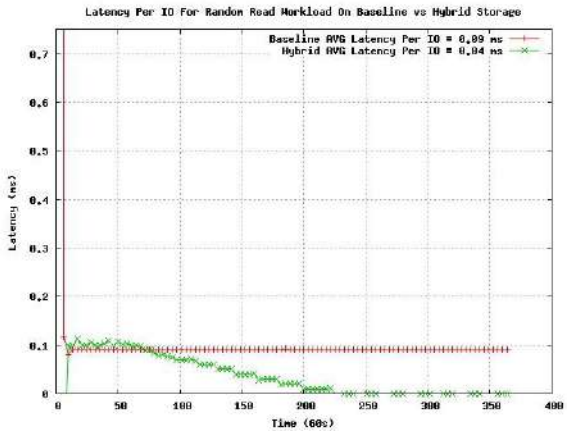

Fig. 11. Latency per I/O for random read workload.

B. OLTP Workload

This workload emulate the I/O access patterns of Oracle 11g database, it read from 9 files with 1 GB each plus an additional 1 GB used to mimic a database log file. Table II summarizes the result of running this workload on the baseline and the hybrid storage. We notice that the IOPS performance gain for the workload, file system and storage system is almost 3X. We attribute this relatively low improvement to the fact that the workload is not only performing IOs, it generates fake CPU cycles, deliberate I/O wait and writes logging information to emulate real Database Management System. These additional overheads will not fully utilize the hybrid storage model like the pure random read. Nevertheless, a 3X improvement for a ($55) amount is an interesting gain.

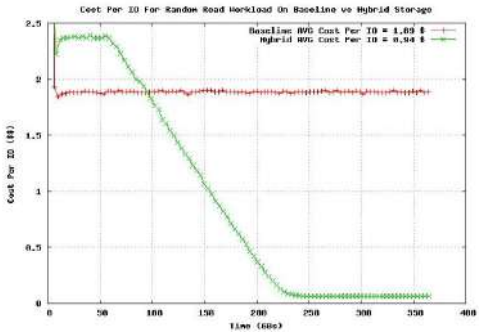

Fig. 12. Cost per I/O for random read workload.

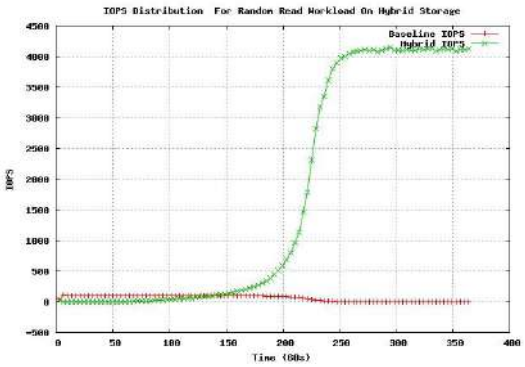

Fig. 13. I/O Source for random read workload.

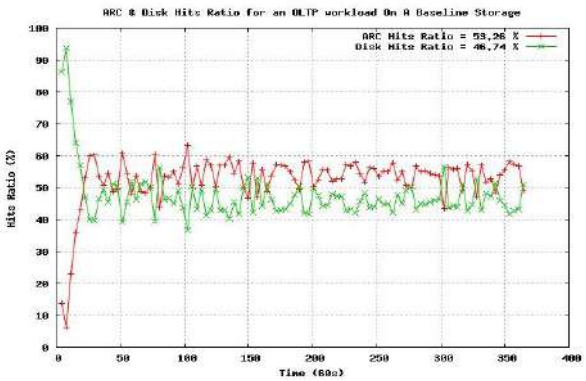

Fig. 14. Baseline hits ratio OLTP workload.

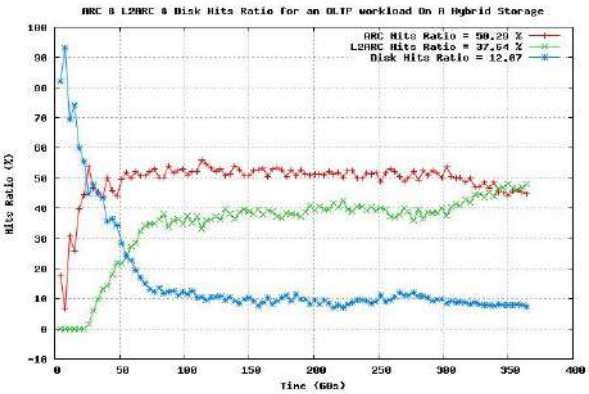

Fig. 15. Hybrid storage hits ratio for OLTP workload.

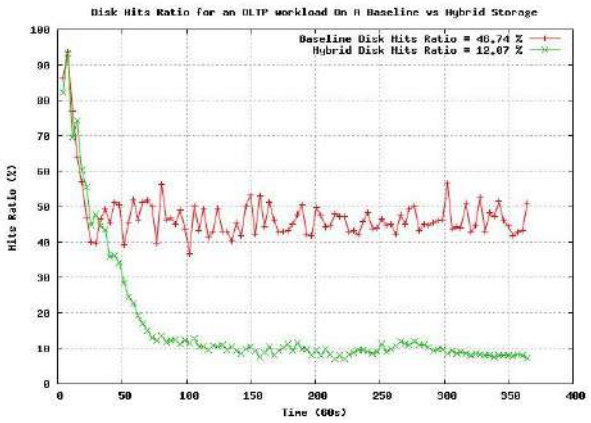

Fig. 16. Disk hits ratio reduction for OLTP workload.

Fig. 14 shows that the baseline ARC and Disk hits ratio are consistent over time. The ARC Hits ratio of this workload (53.26%) is much more than the previous workload (24.34%), we believe that the high ARC hits ratio for the OLTP workload compared to the random read workload is caused by less data being placed on ARC by the OLTP workload due to its I/O access pattern. Additionally, when we move to the hybrid storage, the L2ARC picks more that 37% of the hits leaving only 12% as disk hits compared to 46% disk hits for the baseline storage. Fig. 15 and Fig. 16 clearly shows significant reduction of disk hits ratio over time dropping from more than 90% to less that 10% at the end of the benchmark. Fig. 15 also shows how data migrates to SSD over time and therefore improving the SSD hits ratio. Again, Fig. 17 shows that the ARC hits ratio is almost the same for both the baseline and the hybrid storage , we also believe that the ratio between the working set size and the ARC size the factor that determine the ARC hits ratio regardless of the absence or the presence of SSD L2ARC cache. Fig. 18 and Fig. 19 show the IOPS improvement for both file system and hybrid storage as data block migrated to SSD device over time. The latency and dollar amount per IO have also been improved as shown in Fig. 20 and Fig. 21. Finally, Fig. 22 is consistent with the fact that data block are retrieved from L2ARC as time progresses which leads to substantial improvement for the file system and the workload IOPS.

C. Results from Extended Experimental Work

Variable RAM, SSD and Working sets configurations revealed much more interesting results. We have conducted an additional twenty seven experiments with different configurations and gathered the results shown in Tables 3 to 8. These tables show the result of three metrics for the same workloads, however, different working set sizes are used in these workloads. It can be seen by examining the columns of Table III and Table IV that the disk hit ratio is decreasing with the increase of the SSD cache size for both workloads. These results are expected since the SSD is being used to retrieve data which means less access to the hard disk and therefore more achieved IOPS because of the faster SSD devices. Table III and Table IV show the SSD and disk hit ratio for the random read workload with different variation of SSDs and working sets sizes while restricting the RAM cache. These two tables show that the disk hit ratio is proportional to the working set size in a fixed SSD cache size. Additionally, the SSD hit ratio is inversely proportional to the working set size in a fixed SSD cache size since the SSD cache hits are less likely with an increasing working set size. However, the SSD hit ratio is increasing with an increasing size of the SSD cache as more data are migrated to the larger SSD cache. Table 6 also shows that the random read IOPS is inversely proportional to the working set size in a variable RAM and a fixed SSD cache sizes. This observation is expected since the achieved IOPS are determined by the sizes of the RAM and SSD caches. Additionally, the IOPS are proportional to SSD cache size because more data is successfully retrieved from these fast devices. Furthermore, the IOPS is somewhat proportional to the RAM cache size in a fixed SSD and working set sizes. A spike in the 8 GB SSD Cache size is attributed to more data being accessed from a larger SSD cache over longer time compared to smaller SSD caches. Table 6 and Table 7 respectively show the disk and SSD hit ratios for the OLTP workload with different variation of SSD and working sets sizes while restricting the RAM cache.

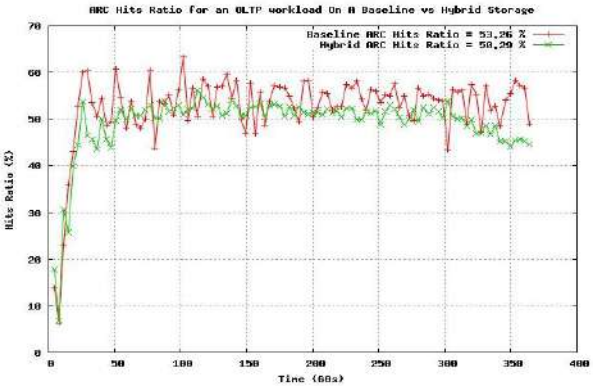

Fig. 17. ARC hits ratio for OLTP workload.

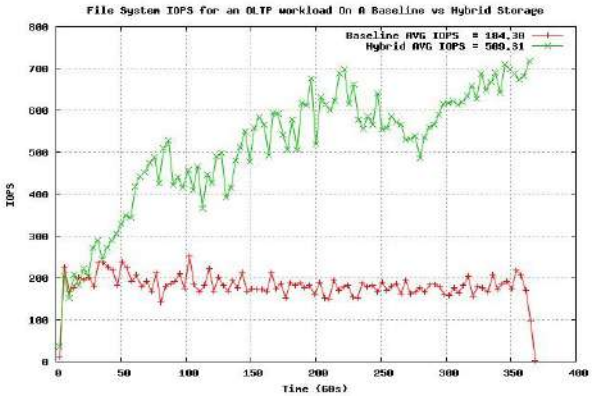

Fig. 18. File system IOPS for OLTP workload.

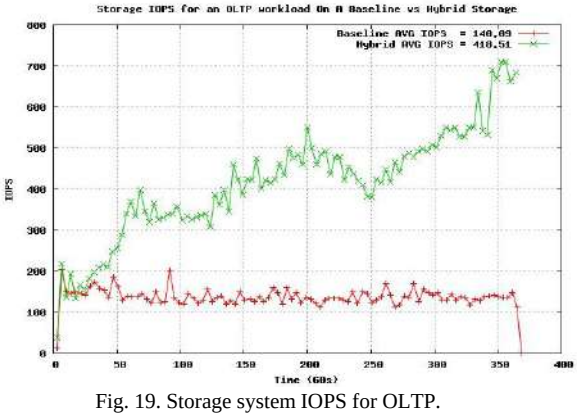

Fig. 19. Storage system IOPS for OLTP.

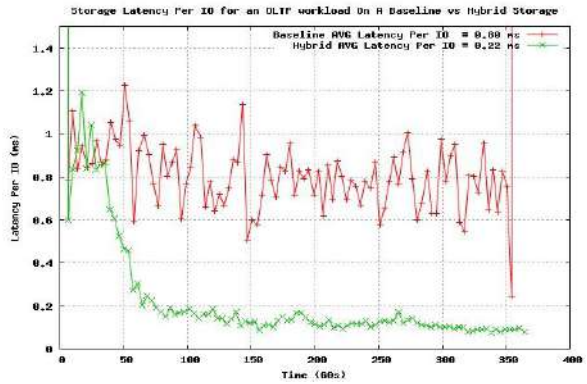

Fig. 20. Latency per I/O for OLTP workload.

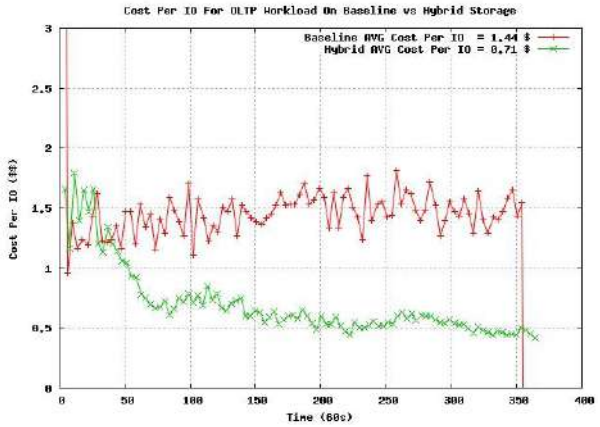

Fig. 21. Per I/O for random OLTP workload.

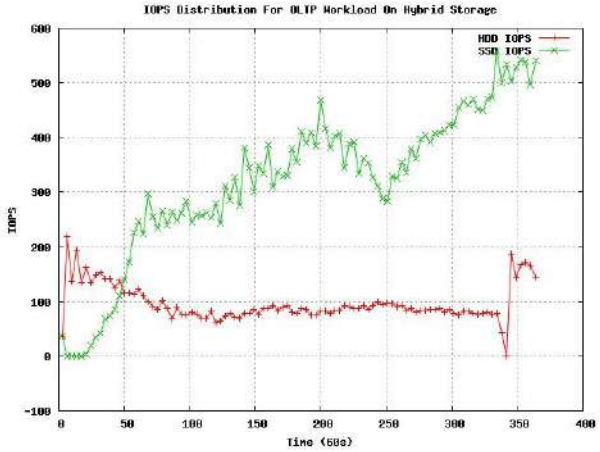

Fig. 22. I/O source for OLTP workload.

This workload might update data blocks in the SSD cache and therefore invalidate and evict them to the Disk. The tables show that the disk hit ratio is proportional to the working set size in a fixed SSD cache size since more disk hits are expected with larger working sets. Additionally, the disk hit ratio is inversely proportional to an increasing SSD cache size as more data are retrieved faster from the SSD rather than the disk. The disk hit ratio also dominates over the RAM and SSD hit ratio especially for larger working set sizes as an increasing number of data blocks are evicted and therefore retrieved from the disk due to data blocks invalidation. Furthermore, the SSD hit ratio is inversely proportional to the working set size in a fixed SSD cache size while the SSD hit ratio is proportional to an increasing SSD Cache size as expected. Finally, Table 8 shows that the IOPS for the OLTP workload is inversely proportional to the working set size in a variable RAM and a fixed SSD.

TABLE III: DISK HITS RATIO FOR RANDOM READ WORKLOAD

Random Read Workload	10 GB Working Set			15 GB Working Set			20 GB Working Set		
	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM
4 GB SSD	57.89	45.1	43.92	73.81	62.92	60.31	80.39	72.13	70.45
6 GB SSD	44.15	36.56	35.65	63.41	56.93	54.34	72.74	68.42	67.25
8 GB SSD	24.33	16.61	23.51	51.44	37.77	47.89	64.29	53.47	64.2

TABLE IV: SSD HITS RATIO FOR RANDOM READ WORKLOAD

Random Read Workload	10 GB Working Set			15 GB Working Set			20 GB Working Set		
	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM
4 GB SSD	22.38	24.81	21.24	15.76	17.48	17.3	12.05	13.42	12.89
6 GB SSD	39.8	33.27	29.79	26.4	23.54	23.47	19.86	17.22	16.35
8 GB SSD	59.98	48.86	42.3	38.56	40.06	30.1	28.56	30.38	19.55

TABLE V: IOPS FOR RANDOM READ WORKLOAD

Random Read Workload	10 GB Working Set			15 GB Working Set			20 GB Working Set		
	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM
4 GB SSD	170.6	227.2	233.5	139.8	164	170.4	126.8	141.1	145.3
6 GB SSD	232.8	279.5	289.2	161.4	180.8	186.5	140.2	148	151.3
8 GB SSD	403.2	421.8	584.5	199.6	265.8	211.4	157	189.6	160.4

TABLE VI: DISK HITS RATIO FOR OLTP WORKLOAD

OLTP Workload	10 GB Working Set			15 GB Working Set			20 GB Working Set		
	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM
4 GB SSD	38.95	26.67	28.71	55.63	40.1	42.11	64.15	50.82	52.46
6 GB SSD	36.32	24.06	25.02	48.8	36.49	38.01	58.69	47.3	49.55
8 GB SSD	28.01	21	21.4	43.3	32.99	34.13	53.2	44.68	45.27

TABLE VII: SSD HITS RATIO OLTP WORKLOAD

OLTP Workload	10 GB Working Set			15 GB Working Set			20 GB Working Set		
	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM
4 GB SSD	23.99	16.62	17.88	20.88	14.73	15.14	18.58	12.74	12.7
6 GB SSD	28.53	22.44	23.7	26.54	20.05	20.43	24.24	16.86	17.33
8 GB SSD	33.44	27.4	27.9	34.24	25	25.44	29.63	21.65	21.92

TABLE VIII: IOPS FOR OLTP WORKLOAD AND VARIABLE RAM, SSD

OLTP Workload	10 GB Working Set			15 GB Working Set			20 GB Working Set		
	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM	2GB RAM	4GB RAM	6GB RAM
4 GB SSD	495.3	570.5	538.8	389.7	425.7	413.7	341.6	362.8	354
6 GB SSD	534.9	629.6	627.4	423.7	466.6	458.7	360.8	380.6	374.7
8 GB SSD	628.6	700.2	702.4	473.6	506.4	504.2	393.1	408.9	405.8

VII. HYBRID CACHE ARCHITECTURE SIMULATION METRICS & RESULTS

A. Performance Metrics

We have identified some specific performance metrics in order to evaluate our proposed hybrid cache architecture, some of these performance metrics relate to the hits ratio of various components of the memory hierarchy while others relate to the cost per I/O and cost reduction which is one of the primarily advantage of using SSD as an extended L2 Cache. The RAM Hits Ratio (RHR), SSD Hits Ratio (SHR), Disk Hits Ratio (DHR), Disk Response Time Average (RTA), and I/O Per Seconds (IOPS) are the main metrics measured. These three metrics are relative to the three memory hierarchy components and are calculated by dividing the total

component's hits by the total number of I/Os processed by the simulator. We have also identified other performance evaluation metrics that are needed to evaluate our hybrid architecture such as Extended Cache Response Time Average (XCRTA), Hybrid Cache Response Time Average (HCRTA), Baseline Disk Response Time Average (BDRTA) and Hybrid Cache Disk Response Time Average (HCDRTA), the Extended Cache IOPS (XCIOPS). The Cost Per I/O and Cost Reduction Ratio are measured by each experiments in order to observe the impact of each component of the memory hierarchy. The following equation (1, 2, 3, 4 and 5) are used to calculate these metrics.

$$XCRTA = SSD\ RTA\ \_\ SHR \quad (1)$$
$$HCRTA = RHR\ \_\ (DRAM\ Latency) + SHR\ \_\ XCRTA \quad (2)$$
$$XCIOPS = SSD\ IOPS\ \_\ SHR \quad (3)$$
$$BDRTA = Baseline\ DHR\ \_\ Disk\ RTA) \quad (4)$$
$$HCDRTA = Hybrid\ DHR\ \_\ Disk\ (RTA) \quad (5)$$

We believe that the hits ratio of each component of the memory hierarchy is a crucial factor in measuring these performance metrics as they indicate the contribution of each component in the system. The results in the next section substantiate the impact of the hits ratios on each metrics.

B. Hybrid Cache Architecture Results

Two I/O intensive real trace workloads have been used to evaluate our hybrid cache architecture. Fig. 23 and Fig. 24 show the hits ratios for the two workloads with a transient buffer size fixed to (64K). They show how RHR, DHR for baseline and RHR, SHR and DHR for the hybrid cache are varying with variable RAM/SSD size ratios. Fig. 26 clearly shows that the disk hits ratio is decreasing for various RAM/SSD size ratio as evicted pages are migrating from L1 RAM cache to L2 SSD cache. Furthermore, it shows that the RAM hits ratio has also increased slightly because we count the page hit in the transient buffer as a RAM hit since they both use the same media.

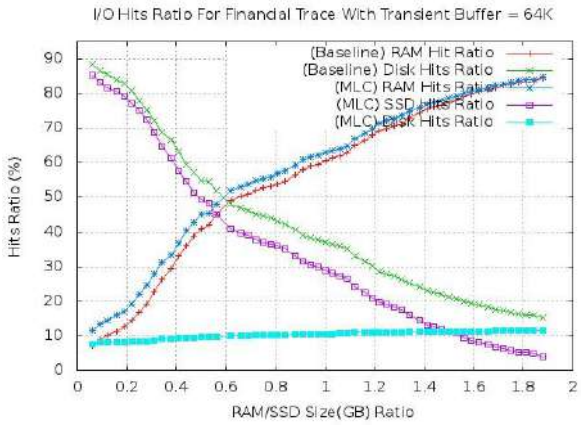

Fig. 23. Financial trace hits ratio for baseline & hca.

The figure also shows that the SSD hits ratio is increasing over time as it is being filled by excessive pages from L1 Cache. It is also observable from the figure that the RAM hits ratio is increasing as the size of the RAM grows and exceeds the size of the SSD cache, this increase is expected since more pages are found and retrieved from L1 RAM Cache. Fig. 27 shows better SSD L2 hits ratio since the Web Server trace is predominately read I/O request and less update is required on the SSD cache. Fig. 25 and Fig. 26 show the hybrid cache response time average (HCRTA) for both Financial and Web Server Traces, it is shown from Fig. 25 that the HCRTA is fluctuating when the L1 RAM cache size is relatively small and more I/O requests are being satisfied from L2 cache which has unstable response time average due to the nature of the I/O trace and impact of the SSD updates on the performance. The figure interestingly shows that the transient buffer has improved the HCRTA by absorbing some I/O requests as we have mentioned before. Fig. 26 Shows stable HCRTA for the Web Server Trace since it is primarily a read trace and therefore does not generate more SSD update to negatively impact the performance of the L2 Cache. The transient buffer has almost no impact in this trace as the majority of I/O request are reads.

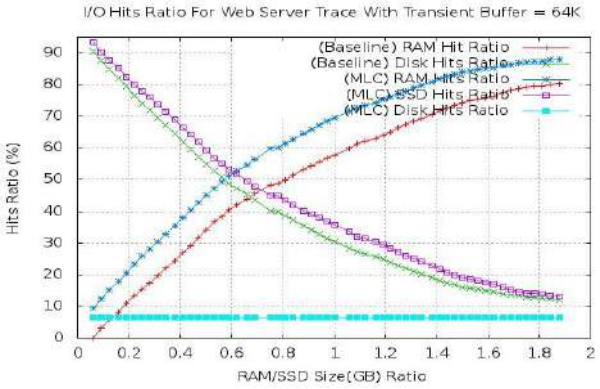

Fig. 24. Web trace hits ratio for baseline & hca.

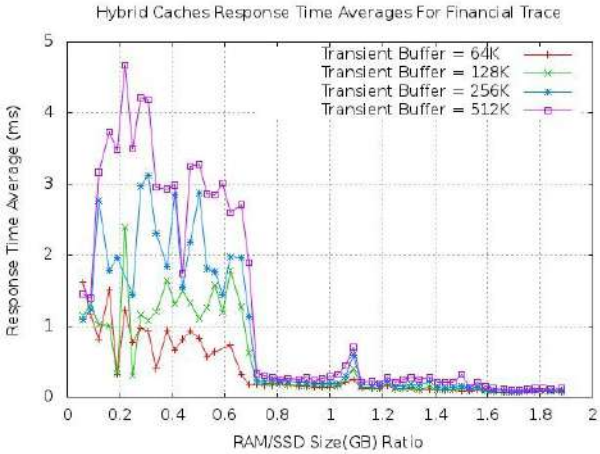

Fig. 25. HCRTA for financial trace.

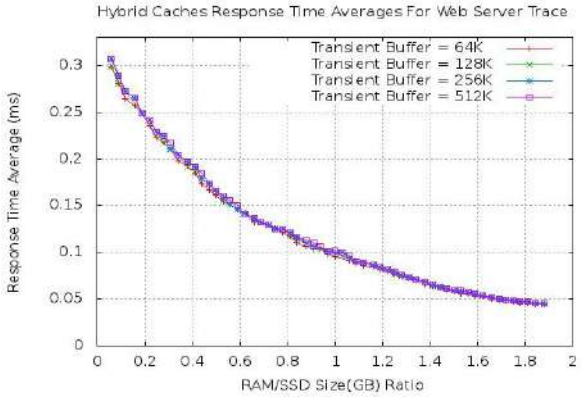

Fig. 26. HCRTA for web trace.

Fig. 27 and (Fig. 28) show that extended cache IOPS

achieved by the L2 Cache for variable transient buffer sizes, they clearly see that larger transient buffer size has decreased the IOPS achieved by the cache, we believe that the time needed to process the content of the transient buffer impacted the total achieved IOPS. We also observed that the financial trace IOPS for variable transient buffer size are almost the same when using larger RAM cache, we also believe that larger RAM cache size will always lead to less I/O on the L2 cache and therefore less extended cache IOPS. It is also observable that the IOPS for the Web Server trace (14000 IOPS) is much higher that the Financial trace (3000 IOPS), again, the nature of the Web Server trace as being majority of read requests has perfectly utilized the L2 Cache and produced the best performance of the cache.

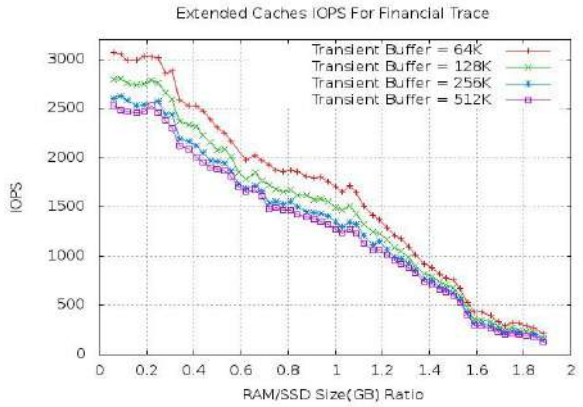

Fig. 27. XCIOPS for financial trace.

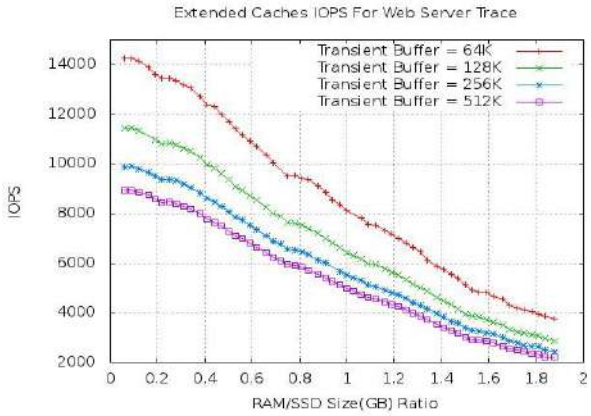

Fig. 28. XCIOPS for web trace.

VIII. Conclusion and Future Work

Our work in this paper is based on Sun 3-tier Hybrid storage model; we have conducted an empirical and comparative study to investigate the behavior of the model when running various I/O intensive workloads along with variable RAM, SSD configurations. Our objective was to monitor and measure the hits ratio of L1 cache, L2 cache and the disk. We have measured other performance metrics such as latency and dollar cost per IO. Our experiments have shown the performance of this hybrid storage model is sensitive to the workload I/O access patterns. Our work shows that off-the-shelf cheap SSD can lead to substantial performance gain for some workloads and decent gain for others.

We have also proposed an SSD-Aware Hybrid Caching Architecture (HCA) and a replacement policy that uses SSD as an extended read cache to main memory. We have developed a simulator to evaluate our proposed architecture with two I/O intensive workloads. We have also conducted a cost analysis in order to evaluate the cost reduction for both real system experiments and simulated architecture. Our work in both real system and simulated architecture has shown that SSD can effectively be used as an extended cache only when an efficient SSD-Aware replacement policy is used to migrate pages between the cache levels. We intend to continue this work by following three major enhancements. First, We will use the SSD as an extended cache to the magnetic disk in order to further minimize the disk hits ratio, second, we will enhance our SSD-Aware algorithm by exploiting the locality among pages which we believe will lead to better RAM and SSD hits ratio and better utilization of the hybrid cache architecture. Third, we are also interested in optimizing this hybrid storage model to find out the right balance between different components within the memory hierarchy in order to reach the best I/O performance given a specific budget.

References

[1] A. Amer and D. D. L. Noah, "Low-cost file access prediction through pairs," in *Proceedings of 20th International Performance, Computing and Communications Conference*, 2001.

[2] T. M. Kroeger and D. D. Long, "The case for e_cient file access pattern modeling," in *Proceedings of the 7thWorkshop on Hot Topics in Operating Systems*, 1999.

[3] X. Ding, S. Jiang, F. Chen, K. Davis, and X. Zhang, "Exploiting disk layout and access history to enhance i/o prefetch," in *USENIX Annual Technical Conference*, pp. 261–274, 2007.

[4] J. Gri and R. Appleton, "Reducing file system latency Using a predictive approach," in *USENIX Summer Technical Conference*, pp. 197–207, 1994.

[5] S. W. D. Bruce, and J. Ng, "Memory Systems, Cache, DRAM, Disk, 1st Edition," *Morgan Kaufman Press*, 2008.

[6] A. L. Holloway, "Adapting database storage for a new hardware," Ph.D. thesis, *University of Wisconsin-Madison*, 2009.

[7] J. T. Poole, Preliminary survey of i/o intensive applications, Tech. rep., Caltech Concurrent Supercomputing Facilities, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.9241

[8] R. S. M. A. R. J. A. Ailamaki, "valuating and repairing write performance on flash devices," in: *Fifth International Workshop on Data Management on New Hardware*, pp. 9–14, 2009.

[9] E. N. S. Min, "Current trends in flash memory technology," in: *Fifth International Workshop on Data Management on New Hardware*, pp. 9–14, 2009.

[10] Embedded ssd in HDD, Ever wonder what happens when a Hard Drive goes bad? REALLY bad [Online].Available:http://www.samsung.com/global/business/semiconductor/support/ brochures/downloads/hdd/hdd_datasheet_200708.pdf

[11] Fusion drive, fusionio [Online]. Available: http://www.fusionio.com.

[12] Windows hardware engineering conference, Samsung printer convention [Online]. Available: http://download.microsoft.com/download/9/8/f/98f3fe47-dfc3-4e74-92a3-088782200fe7/TWST05002_WinHEC05.ppt.

[13] I. Koltsidas and S. D. Viglas, "The case for ash-aware multi-level caching." Tech. rep, University of Edinburgh, EDI-INF-RR-1939. [Online]. Available: http://homepages.inf.ed.ac.uk/s0679010/mfcache-TR.pdf.

[14] T. Kgil, D. Roberts, and T. Mudge, "Improving nand flash based disk caches," in *35th Annual International Symposium on Computer Architecture (ISCA)*, pp. 327–338, 2008.

[15] T. Kgil and T. Mudge, "Flashcache: a nand flash memory. File cache for low power web servers," in *International Conference on compilers, Architecture and Synthesis for Embedded Systems*, pp. 103–112, 2006.

[16] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, Migrating enterprise storage to ssds: Analysis of tradeo_s, Tech. rep.,

Microsoft Research, IP Going Mannering [Online]. Available: http://research.microsoft.com/apps/pubs/?id=78895.
[17] S. V. I. Koltsidas, "Flashing up the storage layer," in *Proceedings of the VLDB Endowment*, pp. 514–525, 2008.
[18] A. G. B. U. Y. Kim, "Mix-store, an enterprise-scale storage system combining solid-state and hard disk drives," Tech. rep., Computer Systems Laboratory, Penn. State University, Technical Report CSE-08-017, MixedStore: An Enterprise-scale Storage System Combining, Solid-state and Hard Disk Drives [Online]. Available: http://csl.cse.psu.edu/publications/mixedstore_tr.pdf
[19] T. Bisson and S. A. Brandt, "Reducing hybrid disk write latency with flash-backed i/o requests," in *Proceedings of the 2007 15th International Symposium on Modeling*, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 402–409, 2007.
[20] G. Graefe, "The five-minute rule twenty years later, and how flash memory changes the rules, in P*roceedings of the 3rd international workshop on Data management on new hardware*, 2007.
[21] D. S. M. N. Megiddo, "Arc: A self-tuning, low overhead replacement cache," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pp. 115–130, 2003.
[22] A. Leventhal, "Flash storage today," *ACM Queue*, vol. 6, no. 4, pp. 24–30, 2008.
[23] Zfs file system, WiKipedia [Online]. Available: http://en.wikipedia.org/wiki/ZFS.
[24] Filebench micro benchmark, SiWiKi [Online]. Available: http://www.solarisinternals.com/wiki/index.php/FileBench
[25] Disksim storage simulator, The DiskSim Simulation Environment [Online]. Available: http://www.pdl.cmu.edu/DiskSim
[26] Storage Performance Council. [Online]. Available: http://www.storageperformance.org.

Abdullah Aldahlawi is a PhD candidate at The george Washington University, Washington DC, USA. He is a member of the High Performance Computing LAB (HPCL) at GWU. His research intrest are HPC, Information Retrieval, Data Mining, and SSD-Aware Multi-Level Caching.

Esam El-Araby received his M.Sc. and Ph.D. degrees in Computer Engineering from the George Washington University (GWU), USA, in 2005, and 2010 respectively. Dr. El-Araby joined the Catholic University of America (CUA) as an Assistant Professor in the Department of Electrical Engineering and Computer Science in 2010. He is the founder and director of the Heterogeneous and Biologically-inspired Architectures (HEBA) laboratory at CUA.

Suboh A. Suboh received his PhD degrees in Computer Engineering from the George Washington University, USA, in 2010. During his study, he was instructor for computer organization, microprocessor-based design classes. He is currently a research affiliate at the High Performance Computing Laboratory (HPCL) at the George Washington University.

Tarek El-Ghazawi received the PhD degree in electrical and computer engineering from New Mexico State University in 1988. He is a professor in the Department of Electrical and Computer Engineering at the George Washington University (GWU), where he also directs the High-Performance Computing Laboratory (HPCL). He is a co-founder of the NSF Center for High-Performance Reconfigurable Computing (CHREC), and the founding director of IMPACT: the Institute for Massively Parallel Applications and Computing Technologies. He is a fellow of the IEEE

.