Journal of Theoretical and Applied Information Technology
15th December 2022. Vol.100. No 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN ENERGY EFFICIENT APPROACH FOR BIG DATA MASS STORAGE SYSTEMS USING A SEQUENTIAL CACHE

AHMED F. MHDAWY[1], MAEN M. AL ASSAF[2]
King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan
E-mail: [1]amahdawe@gmail.com, [2] m_alassaf@ju.edu.jo

ABSTRACT

Big data storage centers became important in the recent era due to the advances in today's world cloud systems and the increased amount of the data generated by users. Mass-inexpensive storage devices are still used in order to provide storage for a lower cost. It is a fact that data retrieval from commonly used mass-inexpensive storage devices (i.e. HDDs and Tapes) consumes much power due to their cost-inefficient hardware architecture (i.e. arm moves over different tracks on magnetic disk) especially with randomly stored data blocks. Retrieving sequentially stored data blocks tends to consume less power as they require less arm moves. Hence, the need for energy efficient solutions is important. In this research, we present an energy efficient solution for big data storage system using sequential caching approach. It is practically proven that users' I/O requests for data tend to show repeating patterns. Data caching that keeps the most frequently accessed data blocks showed efficient performance improvement for many solutions over the last decades. Our Solution (EEHSC) boosts data caching and arm moves' space locality by buffering most frequently used data blocks in the middle of the storage medium. This in result reduces power consumption by increasing the percentage of data blocks that are read sequentially over those that are read randomly. We run performance evaluation for our EEHSC algorithm using a simulator created by ourselves based on real world I/O traces. Results show an EEHSC can efficiently reduce power consumption without the need for allocating big size cache due to the high data hit-rate that can be achieved in the cache. Results showed that power saving ratio can reach 67.60% when using LASR machine01 trace and 36.74% when using LASR machine06 trace. This difference goes to the variation of the repeated pattern for the most frequently used data blocks.

Keywords: *Energy Efficiency, Big Data Centers, Caching, Hard Desk Drive, Magnetic Storage*

1. INTRODUCTION

Big data centers use Hard Disk Drives (HDD) as an inexpensive mass storage device in order to store the huge amount of data that are stored by users. Such devices consume much of electrical power as they comprise an arm that keeps moving during the data read/write process. Hence, energy efficient data retrieval algorithm is important when fetching the data from HDDs. Solutions for energy saving in storage systems became very important during the recent years [1].

The use of Mass-inexpensive HDDs in today's world big data centers is inevitable due to their ability to store big data at the lowest possible cost. Indeed, their low cost came from their inexpensive hardware architecture that consists of magnetic cylinders and arms. To our best knowledge, previous research on improving HDD performance who focused on decreasing data retrieval time and power consumption suggested adding an extra hardware layer (i.e. Solid State Drives (SSDs)) to the current basic architecture due to their high performance and low energy consumption. They are used to pre-load predicted important data [2][3][5] Unfortunately, their cost per storage unit is much expensive in respect to HDDs. In addition, SSD's cells lifetime is threatened by mass data writing operations.

Our research gives a solution to reduce power consumption without adding an extra layer to the storage system. Most of users' data request tend to show repeating requests patterns. This helps data caching algorithms to efficiently increase system performance, enriches data locality, and increases sequential reads without the need of using large size caches according to possibility of caching as much of most frequently used small data blocks in a small buffer that is allocated in the HDD.

In this research, we present an energy efficient approach (EEHSC) that caches the frequently accessed data in a sequential HDD buffer in order

to reduce its power consumption by increasing the volume of sequential reads.

The following points summarize our research' motivations:

1. The inevitability of Mass-inexpensive HDDs use in today's world big data centers due to their ability to store big data in lowest possible cost.

2. Sequential data retrieval from HDDs consumes much less power in respect to random data retrieval.

3. The ability to localize data in the middle of the HDDs due to the user's data request repeated patterns.

The rest of this paper is organized as the following: Section 2 presents a literature review for previous power consumption research and how we can increase system performance. Section 3 present a full description of our energy efficient algorithm (EEHSC). Section 4 presents EEHSC simulation and performance evaluation. Finally, section 5 presents conclusions and a roadmap for future works

2. RELATED WORK

This section spotlights on the previous research that provide solutions for improving big data centers performance in terms of data retrieval speed and energy consumption. In addition, we discuss issues related to hard disk drives used in data centers in which our solution is oriented for.

2.1 Improving performance of magnetic mass storage systems

HDD is an in-expensive storage device used to permanently store data. HDD also provides large space and reasonable average access time. Although HDD has many advantages, but it has an annoying disadvantage which is the power consumption besides the longer data retrieval latency [1][3][5].

Retrieving data from HDD in many cases degrades system performance due to the gap of its performance in respect to CPU and Main Memory, in which they have fast data retrieving and processing speed. Many researchers provided solutions for increasing HDD's system performance and energy efficiency. There are two important concepts used to reduce the gap between CPU, Main Memory and storage system as following:

2.1.1 Caching

Data caching techniques proved their capability in reducing data access time and in reducing power consumption. Much research used either built in caches in the storage devices or in the main memory for that concern [12][13]. Cache sizes can range from small to large sizes. Many researchers proved that large caches will not significantly increase the performance when using a cache size higher than a particular threshold. [1].

2.1.2 Prefetching

Contemporary world applications request and process huge amount of data. Data centers with huge amount of storage capacity stores such data that are accessed frequently. In many applications, users tend to do data read transactions more than performing writes [1][5][6]. In addition, users tend to have a repeated data access patterns in which they could be leveraged to predict their future data need. Hence, data prefetching techniques are used to cache the data blocks that are predicted to be accessed soon in advance in caches that are near to the process level. Prediction accuracy is considered the most important dilemma in such technique. In case the prefetched data were or not needed in the near future, cache space will be polluted [4][6].

Several research in data prefetching showed that prefetching leads to a better performance in terms of data load speed when using large size data blocks. This is because a large size data block contains contiguous relative data content. Reading small data blocks from HDDs may take longer time, and this problem increases if the small data blocks are allocated randomly in the HDD. In case data blocks were small, they tend to combine them into large size data blocks [1][6]. In many applications, data blocks stored in the HDD may be small and may not be combined into larger size blocks. Hence, our research focuses on small data blocks and can cope with different data blocks sizes.

When new relative data blocks are created, they tend to be stored contiguously in the HDD especially if the disk contains a hole that can accommodate them. If the data blocks are moved into different locations, they might be fragmented [14]. In this case, prefetching such those data blocks will become in a random manner and will need more time and electrical power.

There are two types of prefetching: informed and predictive. In informed Prefetching, applications should have the ability to provide hints on future needed data blocks. On the other hand, Predictive Prefetching predicts future requests using different smart algorithms to be pre-loaded. Informed prefetching indeed provides more accuracy than predictive prefetching but for informed prefetching, however, there exists many applications that cannot give hints on their future data access and hence, predictive approach is more practical [4].

In reality, prefetching algorithms may take much time to predict future data access and provide decisions especially in case the algorithms are not enough efficient. Low accuracy in predicting the correct future data accesses (i.e. high ratio of those data blocks that are predicated to be accessed soon and the user has not actually accessed them) will decrease the system performance, increase the data fetching time, and pollute used caches. In addition, it will increase the power consumption for those systems who are using prefetching especially in case when using Hard Disk Drives.

2.2 Adding an Extra storage layer

Solid State Drive (SSD) is a modern technology of storage devices that can be used to store the data permanently. SSDs are known of their high reading speed and energy efficiency with respect to Hard Disk Drives (HDDs), but on the other hand, more cost per storage unit. Some researchers suggested using SSD as a big cache for HDD data blocks to increase system performance especially in data retrieval time [1][5].

For large data centers, some researchers suggest adding SSDs layer on the top of HDDs layer to buffer users' frequently requested data block in the SSDs. For example, [15] suggested a scheme that adds SSD layer to work as a middle level between slow HDDs and users requests. Such solutions prefetches data from HDDs to the SSDs before the users request the data. Hence, more data will be found in the SSDs.

Another example suggested an (ECO-Storage) energy efficient algorithm that prefetches data blocks from HDDs to SSDs in order to enable HDDs to sleep as long as possible in order to reduce power consumption [6]. Our solution can reduce energy consumption of HDDs without the need of using an additional energy saving layer.

2.3 Hard Disk Drive (HDD)

Hard disk drive (HDD) magnetic storage technology was introduced long time ago. It is considered the most popular non-volatile secondary storage used to store and to retrieve data. Currently, HDDs still the most important type of secondary storage because of their huge capacity, low prices, and moderate data access and retrieval speed.

Each HDD consists of a magnetic circular platters and a head attached to an arm that moves all over the platter in order to read or write data in a particular location. Disk access time is measured by seek time and data transfer time. Seek time mainly is the time needed for the head to reach to a specific location on the disk. HDD data transfer time is mainly affected by the ability of the disk to read the data from the current head's location and differs from a type of disk to another.

When a application requests a group of relative data blocks from the HDD, those data blocks may be stored randomly in different locations. This indeed increases the total head elapsed movements and in result, increases the seek time and the disk power consumption according to mechanical HDD arm movements. On the other hand, in case the relative data blocks are founded stored sequentially, their reading time and the disk power consumption will be reduced (See: Figure 1).

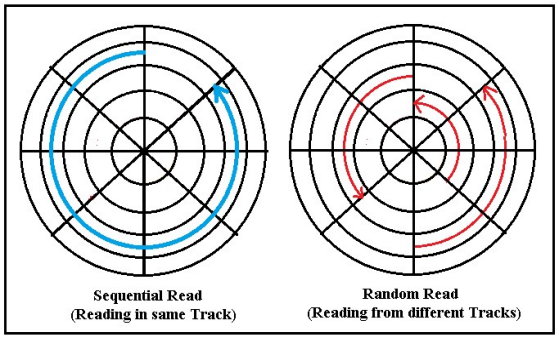

Figure 1 HDD head sequential and random read

2.4 Contemporary research on big data system

Research on big data centers is an ongoing process and new recent research provided many solutions. All of those research came to provide solutions due to the exponentially increased amount of data that are stored in contemporary big data centers for different application domains [10].

For example, [7] provided a solution based on Hadoop open-source system to manage data sets and make the access, storage, and processing of data more efficient in parallel and distributed storage environments. [8] Proposed a master/slave parallel binary-based algorithm that improves the process of finding the frequent patterns of large size datasets in big databases. This reduces the processing time when compared to binary based technique algorithm. Research in data clustering using smart algorithm plays important role for providing solutions for big data centers. [9] Introduced a big data analysis solution in real time systems based on K- mean clustering approach that provided a significant accuracy level.

The amount of stored data in today's world is estimated to exceed 44 Trillion GB which makes it need much traditional storage medium to store them. DNA molecules as information carrier are expected to be more practical in the future. [11] provided a technique that leverage the capabilities of DNA storage and to optimize the process of storing the huge amount of data.

3. ENERGY EFFICIENT HARD DISK USING SEQUENTIAL CACHE (EEHSC)

In this section, we propose our new solution called, Energy Efficient Hard Disk Drive using Sequential Cache (EEHSC).

3.1 EEHSC Introduction

Hard Disk Drive (HDD) has a mechanical motion in which a head moves all over the location insides on its platter. There exists an average head movement time for each type of HDD. As the head moves more to reach a particular data block for performing read or write operation, it consumes more electrical power. HDD vendors measure an estimated average head movement time and power consumption to be listed in the description of their HDD specifications.

When the user's application requests a particular data block stored on the HDD, the data block might be stored near or far away from the head. The worst-case (highest) power consumption in (watts) is recorded in case both of; the head and the requested data block; are allocated on the disk' different extremes. Hence, the head will need to move from the beginning to the end of the disk cylinder or vice versa.

Research in storage systems uses average power consumption (in watts) to measure the amount of used power because the requested data blocks stored on the HDDs tend to show a normal distribution of their distance from the HDD' head. Therefore, we assume that every requested data block by the application requires that the head moves within the average head movement displacement in respect to the overall disk length in order to be reached.

Hence, it will consume the average power consumption (in watts) recorded for HDD's head movement. As mentioned previously, when relative data blocks (i.e. related to the same application) are stored sequentially on the HDD, the head movements will be sequential and will record the minimal displacements when those data blocks are requested. On the other hand, the head movements will be random and the displacements will be larger in case those requested data blocks are found stored randomly over the HDD. Indeed, requesting randomly stored data blocks will require more head movements, and eventually more HDD power consumption.

Our proposed solution introduces HDD energy efficient algorithm without hurting system performance. The new algorithm (EEHSC) involves allocating a relatively small data blocks cache in the middle track of the HDD in order to temporarily hold copies of accessed data blocks in the middle of the HDD. In the same time, the algorithm maintains the head allocated in the cache location when the user's application requests a particular data block. This caching and head behavior will ensure that frequently accessed data blocks will be reached sequentially (i.e. head will move sequentially to accesses them). Our suggestion of making the cache location in the middle track of the HDD (as shown in Figures 2 and 3) is for the following reasons:

1. Caching copies of frequently accessed data blocks in the middle track of the HDD will guarantee that we can cache them in one contiguous location in average head movement displacements (low power consumption).

2. Allocating the cache in the middle track of the HDD facilitates the system task of maintaining the head allocated in the cache location (in the middle of the HDD) when the application requests a particular data block. This will ensure that the

frequently accessed data blocks are reached sequentially.

3. In case a requested data block misses in the cache, the head will take less than average head movement displacements in order to reach its original location and to bring a copy to the cache.

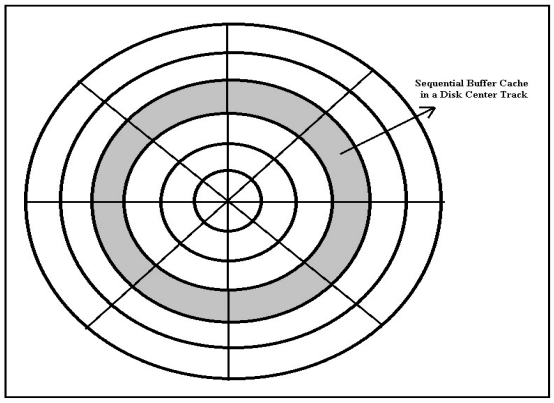

Figure 2 A new sequential cache in a middle track

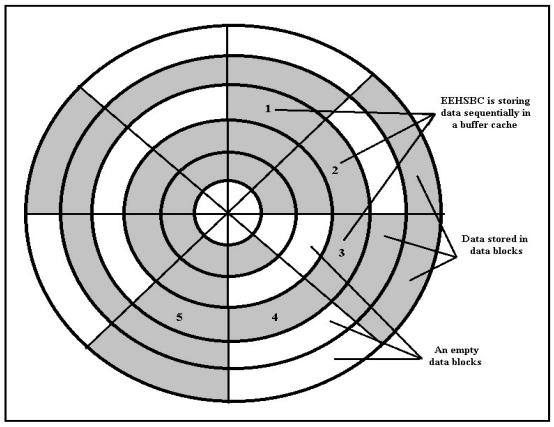

Figure 3 A data stored sequentially in a cache

3.2 EEHSBC Assumptions

Related data are stored in HDDs in a form of data blocks. Storge systems of big data centers either decide to combine data in small or larger sizes data blocks. Large data block size may consume disk space as they are liable for internal fragmentation.

Systems that combine related data into small data blocks tend to be able to manage the HDD in a better way. On the other hand, in case that the related data are stored in small data blocks, their accessibly will become harder especially if they were stored completely random on different locations on the disk. Hence, this will cause high volume of head moves, increased data reading latency, and increased power consumption.

As mentioned previously, many previous research where not able to reach a better system performance without using large data block. For this purpose, they used special tools to group small data blocks into large ones.

In reality, data blocks may exist small, and the grouping process may create more overhead. In EEHSC, we focus on using small size data blocks as it represents the case where random data access is a big burden. On the contrary, in case the data blocks are originally large enough, sequential data access will not show much better performance than random data access.

3.3 EEHSC Algorithm

EEHSC initially allocates a buffer in the middle of the HDD (as shown in Figure 4). If the middle track is holding some data blocks, EEHSC will move them to other locations.

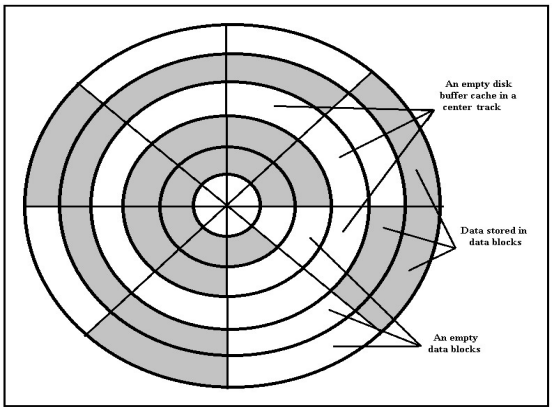

Figure 4 An empty buffer on the system startup

When a data block is requested by an application, it will miss in the cache. Therefore, EEHSC will bring copy of the requested data block from its original location to be cached in the middle cache. The head then will be allocated on the cache area This process will continue while the requested data blocks miss in the cache. Figure 5 shows the case when a data block mises in cache and the process of copying it from its original location.

Journal of Theoretical and Applied Information Technology
15th December 2022. Vol.100. No 23

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

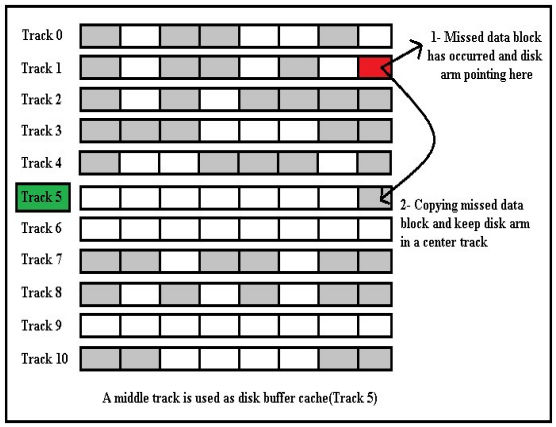

Figure 5 When a data block misses in the cache

When a requested data block hits in the cache, the HDD head will not need to move to the requested data block' original place in order to read a copy and to place it in the central cache. The data block in this case will be read sequentially from the buffer. Least recently used (LRU) algorithm is used replace newly buffered data blocks with old ones (See: Figure 6).

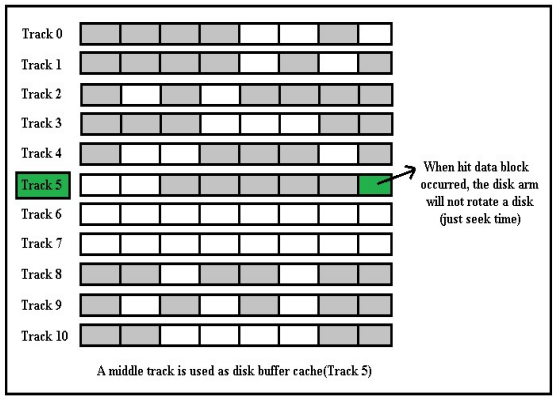

Figure 6 When a data block hits in the buffer

When the cache reaches its maximum limit, EEHSC chooses a victim block to replace it with a recently missed data block. We can use any cache replacement mechanism like: First in First out (FIFO), Most Recently Used (MRU) and Least Recently Used (LRU).

EEHSC shows it is best performance when using LRU to evict useless data blocks (i.e. maintaining most frequently used (MRU) data blocks). For example, if the size of the HDD cache is equal to 8 slots and all of them were used (as shown in Figure 7), EEHSC will evict the least recently used cached data block (i.e. 1) whenever a newly requested data block misses in the cache and the later one will occupy the location (i.e. 9).

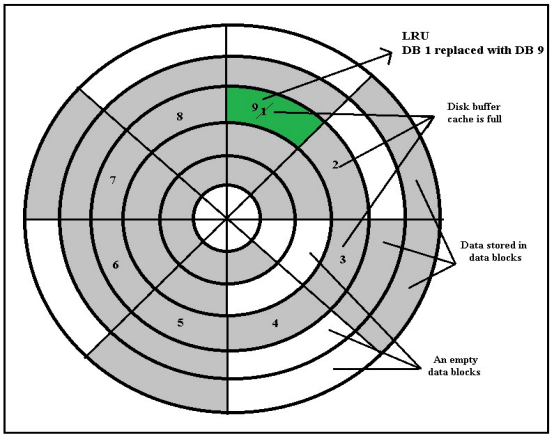

Figure 7 When using LRU replacement algorithm

Figure 8 shows EEHSC Algorithm Pseudo-code

```
IsFound = FALSE
BufferSize = 0
HDD arm currently pointes at first block of buffer;
When a Block ReqBlock is requested
    Loop (Search for ReqBlock in all buffer blocks sequentially)
        If ( ReqBlock is Exist ) Then
            IsFound = TRUE;
            Resort Buffer Blocks using replacement algorithm;
            Return ReqBlock;
            Break;
        End If;
    End Loop;
    If (! IsFound) Then
        Fetch ReqBlock from HDD;
        If (BufferSize <  MaxBuffereSize) Then
            BufferSize++;
            Determine a buffer location for the new Block
            Copy ReqBlock in a buffer location and keeps HDD arm pointers to
            buffer;
        Else
            Apply replacement Block replacement algorithm
            Copy ReqBlock in a buffer location and keeps HDD arm pointers to
            buffer;
        End If;
    End If;
```

Figure 8 EEHSC Algorithm Pseudo-code

4. PERFORMANCE EVALUATION.

EEHSC uses small data block size stored in HDD as of the fact that small data block requires more rotations when seeking for a particular block, and hence, consumes more energy in respect to the case when using large data blocks. We also assume that the HDD has enough space to implement the

cache in the middle, as the cache does not require a large space.

We implement EEHSC using a trace-driven simulator written in C++. We extract results when using different cache sizes in the HDD from one to 200 cached data blocks. HDD Power consumption was tested in terms of (watt) with and without implementing EEHSC.

We used real world traces from SANIA: LASR machine01 and LASR machine06 [17][18]. Each trace has number of HDD reading requests and a particular behavior of repeated patterns. In LASR machine01, the trace consists of 11676 reading requests on 886 distinct data blocks. In LASR machine06, the trace consists of 51206 reading requests on 3142 distinct data blocks.

For simulation purpose, we did our best to find power consumption in watts when moving HDD arm and reading a data block sequentially and randomly. We inferred from [16] that when using HDD of ATOM D525 Disk for small size data blocks, it requires about 0.078688119 Watts to read a data block that is stored in the furthest random location. We also inferred that a sequential move for the HDD head and reading the data block that is stored near the current position of the it requires 0.009020619 Watts, which is significantly less than the random case. We assumed using small sizes data block where each block does not exceed 100 KB.

We simulated EEHSC using the two traces when using different cache size implemented in the middle of the HDD from 1 to 200 block. The size of each cache block is relative to the size of the original blocks. Without EEHSC, data blocks were read randomly, and the power consumption stays always constant. Figures 9 and 10 show when using LASR machine01 and LASR machine06 respectively.

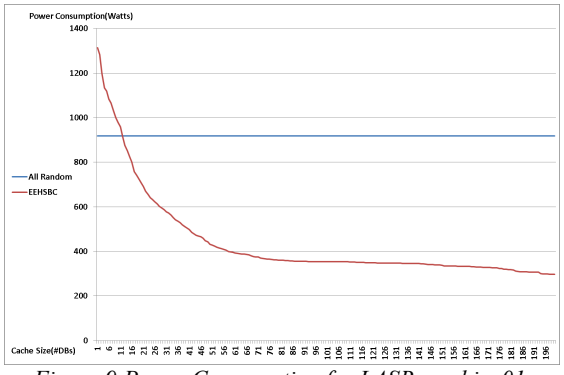

Figure 9 Power Consumption for LASR machine01.

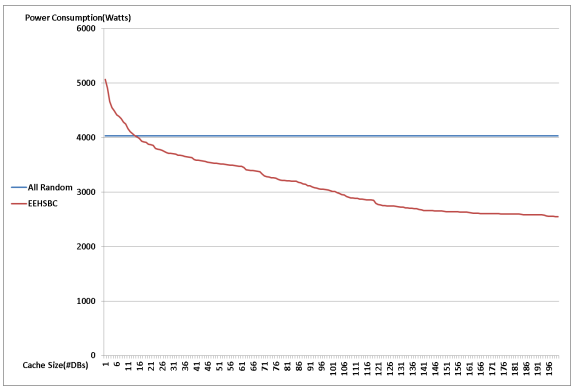

Figure 10 Power Consumption for LASR machine06.

The experimental results show that EEHSC reduces the total energy consumption when the cache size is at least 12 blocks or larger when using LASR machine01 and 15 blocks or larger when using LASR machine06. In non-EEHSBC case, the system's power consumption is always constant and equals to 918.762 watts when using LASR machine01 and 4020 watts when using when using LASR machine06.

Depending on the behavior of the users recorded in the traces and their repeated patterns of data requests, EEHSC starts giving good performance when the cache size is 12 blocks in LASR machine01 case and when it is 15 blocks in LASR machine06 case. Performance improvement has reached to 67.60% when using LASR machine01 trace and to 36.74% when using LASR machine06 trace with respect of non-EEHSC scenario.

It is also visible that LASR machine01 shows faster reduction in power consumption as the cache size increases. This is also due to the higher repeated requests and locality of reference.

In addition, the two solutions give high power consumption when using small size cache size as it will increases the disk moves for data reads without much benefit from the disk caching solution.

It is also noted from the above results on the two traces that after a particular cache size, there will be no significant reduction in the power consumption, this is due to the caching limited power after reaching a particular threshold, and hence, this shows that EEHSBC does not need a big cache size in the HDD to provide a significant performance improvement.

One final note is that our solution was able to reduce power consumption with the mentioned ratios (67.60% and 36.74%) without the need of using an energy efficient device in the storage system as other research did (For example: [6]) in which an performance improvement in-terms of power consumption reduction was achieved by 75% when using a SSD layer.

5. CONCLUSIONS AND FUTURE WORK

This paper introduced EEHSC approach that reduces the power consumption of the Hard Disk Drivers HDDs used in big data centers by caching the frequently accessed data blocks in a central cache. Such cache does not consume the disk space. Randomly accessed data blocks in the HDDs tends to consume more power as the arm needs to move longer especially with small size data blocks. Caching them in the center will increase the amount of the data blocks that are reached sequentially, and hence, reducing the power consumption. Our simulation results on real world traces shows that EEHSC power consumption reduction may reach to 67.60% in many cases.

Our solution was able to reduce power consumption without the need of adding an extra energy efficient layer of storage devices such as SSDs. Adding extra layer of SSDs might provide higher reduction of energy consumption, but indeed, it will be will extra cost.

In our future work, we will be including AI algorithms that predict the user access pattern for performing the caching process in the HDD.

REFERENCES:

[1] Al Assaf, Maen et al (2018). Informed Prefetching in Distributed Multi-Level Storage Systems. Journal of Signal Processing Systems. 90. 10.1007/s11265-017-1277-z.

[2] Niu, Junpeng & Xu, Jun & Xie, Lihua. (2018). Hybrid Storage Systems: A Survey of Architectures and Algorithms. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2803302.

[3] Al Assaf, Maen. (2020). Performance Optimization for Distributed Hybrid Storage Systems Using a Predictive Approach. International Journal of Advanced Trends in Computer Science and Engineering. 9. 4819. 10.30534/ijatcse/2020/91942020.

[4] Al Assaf, Maen & Rodan, Ali & Qatawneh, Mohammad & Abid, Mohamed. (2015). A Comparison Study between Informed and Predictive Prefetching Mechanisms for I/O Storage Systems. Int. J. Communications, Network and System Sciences. 8. 181-186. 10.4236/ijcns.2015.85019.

[5] Al Assaf Maen, Alghamdi Mohammed, Jiang Xunfei, Zhang Ji and Qin Xiao (2012), A Pipelining Approach to Informed Prefetching in Distributed Multi-Level Storage Systems. Proceedings of the IEEE 11th International Symposium on Network Computing and Applications.

[6] Al Assaf Maen M., Jiang Xunfei, Abid Mohamed Riduan and Qin Xiao (2013), Eco-Storage: A Hybrid Storage System with Energy-Efficient Informed Prefetching. Journal of Signal Processing Systems, Volume 72, (Issue 3), 165-180.

[7] Qureshi, Nawab Muhammad Faseeh & Shin, Dong & Siddiqui, Isma & Abbas, Asad. (2017). Hit-ratio storage-tier partition strategy of tempeature data over Hadoop. Journal of Theoretical and Applied Information Technology. 95. 2466.

[8] Amir, A.N. & Alhussian, Hitham & Fageeri, Sallam & Ahmad, R.. (2021). P-BBA: A master/slave parallel binary-based algorithm for mining frequent itemsets in big data. Journal of Theoretical and Applied Information Technology. 99. 3517-3526.

[9] El-Barbeer M. & Rezk A. & Abu Elfetouh S. (2021). Real-time big data clustering using spark: uber case study. Journal of Theoretical and Applied Information Technology. 99. 1414-1425.

[10] Ennassiri, A., Elouahbi, R., Alhammadi, G., Boudallaa, I., Zrigui, I., & Khoulji, S. (2022). The contribution of data mining to the education sector. journal of theoretical and applied information technology, 100(6).

[11] SAIS, M. & Najat, Rafalia & Abouchabaka, Jaafar. (2021). Synthetic dna as a solution to the big data storage problem. Journal of Theoretical and Applied Information Technology. 99. 3912-3922.

[12] Manzanares Adam, Qin Xiao, Ruan Xiaojun and Yin Shu (2011), PRE-BUD: Prefetching for energy-efficient parallel I/O systems with buffer disks. ACM Transactions on Storage (TOS), Volume 7, (Issue 1).

[13] Ruan Xiaojun, Manzanares Adam, Yin Shu, Zong Ziliang and Qin Xiao (2009), Performance Evaluation of Energy-Efficient Parallel I/O Systems with Write Buffer Disks. International Conference on Parallel Processing.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[14] Silberschatz Abraham, Galvin peter Baer and Gagne Greg (2013), Operating System Concepts, (Ninth edition). USA: Wiley.

[15] Zhang Xuechen, Davis Kei and Jiang Song (2012), iTransformer: Using SSD to Improve Disk Scheduling for High-performance I/O. Proceedings of the 26th IEEE International Parallel & Distributed Processing Symposium, 21-25 May 2012, Shanghai, China, 715-726.

[16] Issa Joseph and Figueira Silvia (2012), Hadoop and memcached: Performance and power characterization and analysis. Journal of Cloud Computing: Advances, Systems and Applications, Volume 1, (Issue 1), 1-20.

[17] LASR trace machine01. http://iotta.snia.org/traces/system-call/4926

[18] LASR trace machine06. http://iotta.snia.org/traces/system-call/4926