An Energy-Efficient Framework for Large-Scale Parallel Storage Systems

Ziliang Zong, Matt Briggs, Nick O'Connor, and Xiao Qin*
Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro, New Mexico 87801, USA
{zzong, mbriggs, nick03d}@ nmt.edu, xqin@cs.nmt.edu

Abstract

Huge energy consumption has become a critical bottleneck for further applying large-scale cluster systems to build new data centers. Among various components of a data center, storage subsystems are one of the biggest consumers of energy. In this paper, we propose a novel buffer-disk based framework for large-scale and energy-efficient parallel storage systems. To validate the efficiency of the proposed framework, a buffer-disk scheduling algorithm is designed and implemented. Our algorithm can provide more opportunities for underlying disk power management schemes to save energy by keeping a large number of idle data disks in sleeping mode as long as possible. The trace-driven simulation results based on a revised disksim simulator show that this new framework can significantly improves the energy efficiency of large-scale parallel storage systems.

1. Introduction

With the tremendous development of human society, billions of data in the form of knowledge and information is generated everyday, which leads to huge requirements of super computers and high performance clusters. Without the help of modern computer systems, we can never imagine how scientists and engineers accomplish the great projects like human genome sequence, universe dark matter observation and Google search engine. Therefore, the last decade becomes the boosting period of data centers in the whole world. Many huge data centers were built and the future data center will be even bigger.

However, the rapid growth of data centers introduces a significant problem: huge energy consumption. According to EUN (Energy User News) [1], the power requirements of today's data centers range from 75 W/ft^2 to 150-200 W/ft^2 and will increase to 200-300 W/ft^2 in the near future. High energy consumption will cause serious economical and environmental issues. For example, a large-scale cluster may require 40TWh per year, costing over $4B per year at the price of $100 per MWh [2]. According to the data from EPA, generating 1 kWh of electricity in the United States results in an average 1.55 pounds (lb) of carbon dioxide (CO2) emissions.

Among various components of a data center, storage subsystem is one of the biggest consumers of power. A recent industry report [3] shows that storage devices account for almost 27% of the total energy in a data center. This problem will be even exacerbated when faster disks with higher power needs appear in the future. Therefore, new technologies of designing energy-aware storage systems for super computers and clusters are highly expected and have become a hot research topic in the high performance computer area.

In this paper, we present a new buffer disk based strategy to build energy efficient parallel storage systems. The basic idea of our strategy is very simple and straightforward. To most people, it is common sense that leaving a light bulb on at daytime is a waste of energy. The same thing happens if we keep the computing nodes on when it is idle. It makes no sense that we still feed those idle machines power, without producing any useful work. Huge energy can be conserved if we turn the idle machines to sleeping mode or just shut them down directly. However, new problems come out if we spin the storage system up and down so frequently. First, the life cycle of storage disks will be shortened. Second, the availability and reliability of disks will be degraded. Third, the whole system will suffer great time and energy overhead for waking the system up again. One possible solution for these problems is trying to reduce the time of spinning status of storage system. Our framework and algorithm is designed on the basis of this strategy. By using data buffer disks to temporally buffer the requests, once a disk is tuning to sleeping node, it will keep this mode as long as possible. In the view of the whole system, the number of sleeping disks is optimized as well. In this research, we construct a buffer-disk based

1-4244-0910-1/07/$20.00 ©2007 IEEE.

framework for parallel storage system and accordingly, a corresponding scheduling algorithm for parallel disk requests is also implemented.

The rest of the paper is organized as follows. In section 2, we present a brief description of related work. Next, Section 3 illustrates the buffer-disk based parallel disks framework. In Section 4, we introduce mathematical models for calculating the power of parallel storage system. Section 5 demonstrates the working of buffer disk scheduling algorithm. Experimental environment and simulation results are analysis in Section 6. Finally, Section 7 provides the concluding remarks and future research directions.

2. Related work

Most of the previous research regarding energy conservation issues focuses on single storage system like laptop and mobile devices to extend the battery life. Recently, researchers have realized that energy conservation is also important for large scale parallel disks in the cluster systems. Several techniques proposed to conserve energy in storage systems include dynamic power management schemes [4][5], power-aware cache management strategies [6], power-aware prefetching schemes [7], software-directed power management techniques [8], redundancy techniques [8], and multi-speed settings [10][11][12]. However, the research on energy-efficient parallel disk systems is still in its infancy. However, the issue of using buffer-disk frameworks to reduce energy consumption in parallel disk systems is not well investigated.

Buffer management has been used to boost performance of parallel disk systems [13][14]. Previous studies showed that data buffers significantly reduce the number of disk accesses in parallel disk systems [15]. More importantly, it is observed from the previous studies that traffic of small writes becomes a performance bottleneck of disk systems, especially when RAM sizes for data buffers are increased rapidly [15]. It is expected that small writes dominate energy dissipation in parallel disk systems that support data-intensive applications like remote-sensing applications and on-line transaction processing systems [16][17].

The long-term goal of our research is to develop fundamental techniques and theories to save energy of large-scale parallel disk systems. The objective of this paper, which is paving a way towards that goal, is to design a radical buffer-disk framework in which energy-related reliability model, data partitioning, disk request processing, data movement/placement strategies, power management, and prefetching schemes are implemented and holistically integrated to reduce energy dissipation in parallel disk systems. The rationale for the proposed research is that the development of the energy-efficient buffer-disk framework accompanied with a simulation toolkit will promote more energy-efficient resource management techniques for storage systems in general and parallel disk systems in particular.

3. Buffer-disk framework

A parallel disk system is comprised of an array of independent disks connected by a high-speed network. In this paper, we proposed a novel energy efficient buffer-disk framework which consists of four major components: a RAM buffer, *m* buffer disks, *n* data disks, and an energy-aware buffer-disk controller (see Figure 1). Hereinafter, we will call this framework as BUD framework for short.

The RAM buffer with a size ranging from several megabytes to gigabytes is residing in the main memory. The buffer-disk controller carefully coordinates energy-related reliability model, data partitioning, disk request processing, data movement/placement strategies, power management, and pre-fetching schemes.

We choose to use log disks as buffer disks, because data can be written onto the log disks in a sequential manner to improve performance of disk systems. It is to be noted that in most cases, the number of buffer disks *m* is smaller than the number of data disks *n*, and values of *m* and *n* are independent of one another for workloads with inter-request parallelisms.

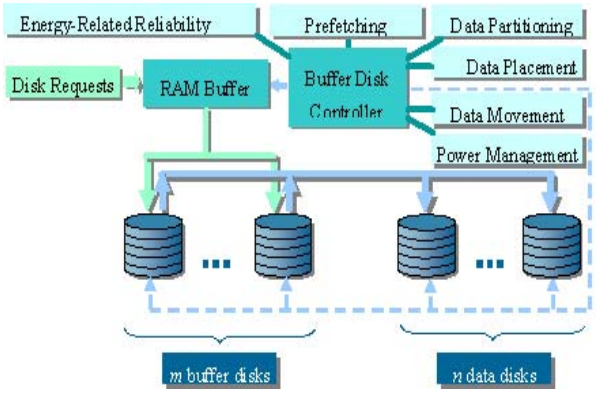

Figure1. The energy aware buffer-disk framework

The buffer disk controller, a centerpiece in our BUD framework, critically affects the overall performance and energy efficiency of parallel disk systems. Therefore, we will address several challenging issues related to the design and

implementation of an energy-aware buffer disk controller. The buffer disk controller will be implemented to achieve the following specific goals. First, the buffer disk controller will aim to minimize the number of active buffer disks while maintaining reasonably quick response times for disk requests. Second, the controller has to energy-efficiently deal with read and write (only small writes are considered in this paper) requests issued to a parallel disk system. Third, the controller must move data from buffer disks to data disks in an energy-efficient way. Fourth, the controller is intended to incorporate an energy-aware pre-fetching strategy to dynamically fetch the most popular data into buffer disks, thereby allowing most data disks to be in the sleep mode to save energy.

4. Energy consumption models

Calculating power for disk drives is complicated. As performance of disks increase, the power consumed does not increase linearly. Idle states are closely approaching run states. This means that this study has to have a specific goal of power consumption from modern disks. The basic power model for this study is a summation of all power states multiplied by the time each power state was active. The states used are start-up, idle, and read/write/seek. Read, write and seek are put together, because they share similar power consumption.

Let T_{tr} be the time required to enter and exit the inactive state. The power consumption of a disk when entering and exiting the inactive state is P_{tr}. Thus, energy E_{tr} consumed by the disk when it enters and exits the inactive state is expressed as $P_{tr} \cdot T_{tr}$. Similarly, let T_{active} and T_{idle} are the time intervals when the disk is in the active and inactive states, respectively. We denote the energy consumption rates of the disk when it is active and inactive by P_{active} and P_{idle}, respectively. Hence, the energy dissipation E_{active} of the disk when it is in the active state can be written as $P_{active} \cdot T_{active}$, and the energy E_{idle} of the disk when it is sitting idle can be expressed as $P_{idle} \cdot T_{idle}$. The total energy consumed by the disk system can be calculated as

$$\begin{aligned} E_{total} &= E_{tr} + E_{active} + E_{idle} \\ &= P_{tr} \cdot T_{tr} + P_{active} \cdot T_{active} + P_{idle} \cdot T_{idle} \end{aligned}. \quad (1)$$

Let T_{ai} and T_{ia} denote times a disk spends in entering and exiting the inactive state, P_{ai} and P_{ia} are the power consumption rates when the disk enters the inactive and active states. Let N_{ai} and N_{ia} be the number of times the disk enters and exits the inactive state. Then, the transition time T_{tr} and power P_{tr} can be computed as follows

$$T_{tr} = N_{ai} T_{ai} + N_{ia} T_{ia}, \quad (2)$$

$$P_{tr} = \frac{T_{ai}}{T_{ai} + T_{ia}} P_{ai} + \frac{T_{ia}}{T_{ai} + T_{ia}} P_{ia}. \quad (3)$$

In most cases where the values of N_{ai} and N_{ia} are both equal to N_{tr}, T_{tr} can be written as

$$T_{tr} = N_{tr}\left(T_{ai} + T_{ia}\right), \quad (4)$$

The time interval T_{active} when the disk is in active state is the sum of serving times of disk requests submitted to the disk system.

$$T_{active} = \sum_{i=1}^{n} T_{service}(i), \quad (5)$$

where n is the total number of submitted disk requests, and $T_{service}(i)$ is the serving time of the ith disk request. $T_{service}(i)$ can be modeled as

$$T_{service}(i) = T_{seek}(i) + T_{rot}(i) + T_{trans}(i). \quad (6)$$

where T_{seek} is the amount of time spent seeking the desired cylinder, T_{rot} is the rotational delay and T_{trans} is the amount of time spent actually reading from or writing to the disk.

Now we quantify energy saved by power management policies as below

$$\begin{aligned} E_{save} &= \left(T_{active} + T_{idle} + T_{tr}\right)P_{active} - E_{total} \\ &= \left(T_{active} + T_{idle} + T_{tr}\right)P_{active} - \\ &\quad \left(T_{active}P_{active} + T_{idle}P_{idle} + T_{tr}P_{tr}\right) \\ &= \left(P_{active} - P_{idle}\right)T_{idle} + \left(P_{active} - P_{tr}\right)T_{tr}. \end{aligned} \quad (7)$$

In this study, the transition power consumption is not considered. This is because transition power is not well defined by data presented in prior research. For this model, the real problem is deciding on what the power consumption is for each state. Values can be collected based on data sheets, physical hard disk tests, and published papers.

Trusting most of the research preformed by others should be with caution. Most of the papers concerning power of hard disks where either using old data or data sheets. Old data presents a problem in that the disks power performance is not linear. Old data can be used to prove the conceptual goal, but there is a good chance that proof is obsolete or misleading. Also, as stated before, data sheets must be checked against reasonable values and it is hard to tell from a published paper whether this was done. Consequently, it is hard to pin down a power model. The evidence would point to find a data sheet that can be confirmed with a physical electrical test. For preliminary tests, we are using a data sheet for a Seagate hard disk that is modeled by DiskSim. As long as the parameters we use are logically correct, the simulator will still be able to show performance. Without precision numbers, we will not be able to conclude accurate monetary savings. This is why disk drive research that is over a year old can be used, but only if the power consumptions are relative to modern disks. To complete this power model, one must define explicitly each state including transition periods between states.

5. An energy-efficient disk scheduling algorithm

In this section, we will talk in detail about the scheduling algorithm which runs on the buffer disk controller based on the BUD framework presented in section 3. Basically, this algorithm will provide a solution for four most important situations in parallel storage systems and give a relatively judicious decision in each scenario.

5.1 Write requests

Write requests can be divided into small write requests and large write requests. Whenever the controller receives a write request, it will first check the size. If the request is a large write, say over 10MB or more, it is sent directly to the corresponding data disk. Otherwise, the controller will send this request to the RAM buffer which will buffer these small writes and form a log of data to be written into one of the buffer disks later. Once the data are transferred to the RAM buffer, the controller will send a "write complete" acknowledgement message to the sender. Next step, the controller will test the state of all the buffer disks. If one buffer disk is not busy with writing a previous log or reading or transferring data, the data copy will be sent to this buffer disk to ensure that a reliable copy resides on one of the buffer disks. In order to guarantee the correctness and consistency of different data version, the controller is always trying to match the data with same block to the same buffer disk unless it is known that the data block is already outdated. In other words, operations which could write the same block data into different buffer disks is forbidden if one legal copy of this block still exists in any buffer disk.

The most important scheduling strategy between RAM buffer and buffer disk is that rather than wait until the RAM is full, the data are written into the buffer disks whenever they are available. This policy has two major advantages. First, data are guaranteed to be written into one of the reliable buffer disks in the shortest time period, which is very important to ensure the reliability and availability of data. Second, the RAM buffer can have more available room to buffer a large burst of new requests because previous data are always quickly moved from the RAM to the buffer disks.

Here we should note that the total storage space of each buffer disk is divided into equal n parts (n is the number of data disks) which are used to buffer the data requests corresponding to each data disk. For example, if we have two 10GB buffer disks and ten 100GB data disks, each buffer disk will have 1GB as the buffer space for each data disk. All the small write requests to data disk 1 will be buffered in the corresponding buffer space reserved for disk 1. The reason we split the buffer disks into small pieces for each data disk is to improve the response performance of the whole system. In the case when one buffer disk is busy writing or moving data, the other disk could serve the incoming requests immediately.

5.2 Read requests

Handling read operations is kind of simple and straightforward in the BUD framework. When a read request arrives, the controller first searches the RAM buffer. If the data is still in the RAM buffer then the data is immediately sent back to the requester. Otherwise, the controller will do a seek operation in the buffer disks. If the required data can not be found in the buffer disks, a missing message will be sent back to controller and the controller will send a read request to the corresponding data disk and finally the data will be transferred to the requester by the data disk. Using this policy, the read performance should be similar to or sometimes better than that of traditional disk because most of the requests will be sent to the data disk and reading from RAM or buffer disks seldom occurs in real applications.

5.3 Data movement

The other important scheduling problem is when and how to move the data from buffer disks to target data disks. Since the data transfer process competes with the disk accesses, a good algorithm to perform data moving is critical to the overall system performance. Rather than just consider the response time performance, data moving algorithm has to ensure the correctness and consistency while moving and after movement of data.

An available space percentage or ASP for short, based scheduling algorithm was designed to complete this task. In this ASP scheduling algorithm, an up bound of available space percentage will be set as the threshold which is used to invoke the data moving operation. For example, if we set the threshold as 20%, a data moving operation request for data disk1 will be sent to the controller as long as more than 80% space allocated for data disk1 in any buffer disk has been used.

In order to guarantee the response time and also the data consistency, the controller will first send a data clean command to the requested buffer disk. This command will mark all the blocks which will be moved to the data disk as "timeout" before the actual moving data operation starts.

The ASP algorithm combined with multiple partitioned buffer disks has two distinct advantages. First, it can significantly improve the response time of the whole system because usually the moving operation will take a long time. Once one buffer disk is super busy on moving data, the other buffer disks can immediately absorb data from the RAM thus greatly increase the parallel levels of the whole systems. Second, ASP algorithm can hundred percent guarantee the data correctness and consistency by using the "timeout" marks. Whenever a write request comes, the controller will check which block will be revised and match the write operation to the buffer disk which has the only legal copy of this specific block. Since all the blocks has already been marked as "timeout" before data moving, the controller will match the same write request to a new buffer disk when the supposed buffer disk is transferring data.

5.4 Power management

The ultimate objective of our research is to conserve as more energy as possible without sacrificing performance. To reduce energy consumption, modern disks use multiple power modes that include active, idle, sleep and shut down modes. In active mode, the platters are spinning and the head is seeking or actively reading or writing. In idle mode, a disk is spinning at its full speed but no disk activity is taking place. Therefore, staying in the idle mode when there is no disk request provides the best-possible access time since the disk can immediately service requests, but it consumes the most energy. In sleep mode, the disk consumes much less energy, but in order to service a request, the disk has to incur significant energy and time overheads to spin up to active mode.

In order to fully utilize the gap of energy consumption rate under different modes, the controller will be always trying to keep as more data disks in sleep mode as possible and also keep the sleep mode as long as possible. However, once a data disk is waken up, it will be keeping busy for a while because a large trunk of data coming from RAM buffer directly or coming from buffer disks will be written to it. At the same time, the controller will set an idle time threshold for the wakened up data disks. These disks will be tuned back to sleep mode to save power if the idle time exceeds this threshold. By using this power management strategy, we can farthest conserve energy and reduce potential damage to disks caused by frequently tune the states of disks.

6. Simulation Results

Our simulation results consist of first developing a simulator which meets all project specifications and running this simulator with a trace to get some preliminary results.

So far the simulator completed all the main functions that are necessary in order to model our distributed system. That is the program takes data from a trace. Then the program moves data to appropriate virtual disks that use disksim to derive there timing information. These virtual disks use a simple model to calculate total energy. Finally, both timing and energy data are reported to the user in the form of the two respective totals.

Our results from the simulator consist of two parallel disk systems. To simulate with these two systems we used a simple trace that came with disksim. The first system that was simulated was a simple disk system which is used today by many storage systems. It is basically a RAID 1 system consisting of 31 disks. This is basically a baseline system in which to compare our results. The second simulated disk system is similar to section 3 with 6 buffer disks each one acting as a buffer for a group of approximately 5 disks.

The simulator runs 25.55 minutes for each simulated system. Table 1 shows the energy consumption of the disk system without employing buffer disks and the disk system using buffer disks to conserve energy. Specifically, the traditional system

without buffer disks consumed 189279.78 J (0.05 KWH) with all disks starting from off and being turned on when needed. In contrast, the parallel disk system with buffer disks only used 117345.99 J (0.03 KWH), which is substantially less. This resulted in overall power savings of 38%.

Table 1. Energy consumption of the two simulated parallel disk systems

Simulated Disk System	Energy Consumption
Disk system without buffer disks	189279.78 J
Disk system with buffer disks	117345.99 J
Energy saving	71933.79 J
Energy consumption reduced by	38%

Our results have shown substantial gains can be made by using buffer disks for very specific workload conditions. In order to further develop our storage algorithm we need to test our results in many more simulations and with more varied workloads. The simulator also needs to check with analytic calculations and the code needs to include the transition time and power from switching from mode to mode.

7. Conclusions and future work

In this paper, we proposed a novel buffer-disk based framework for large-scale and energy-efficient parallel storage system. In light of this novel buffer disk framework that are energy-efficient in nature, we designed a power consumption model, a disk scheduling algorithm, read/write quests handling strategies, data moving and power management policies. The preliminary results have shown substantial gains that the proposed energy-efficient disk framework can conserve more than 38% power compared with traditional parallel systems without employing the buffer disks.

Future studies in this research include the following points. First, we will conduct extensive simulation experiments by running disk traces from real-world applications. Second, we will extend our scheme to deal with intra-request parallelisms. For now, we simply consider inter-request parallelisms. Thus, I/O activities with intra-request parallelisms will be considered in the future. Third, we intend to further extend the disk scheduling algorithm to handle real-time disk requests, where hard deadlines must be guaranteed.

Acknowledgements

The work reported in this paper was supported in part by the New Mexico Institute of Mining and Technology under Grant 103295 and by Intel Corporation under Grant 2005-04-070.

References

[1] B. Moore. Taking the data center power and cooling challenge. Energy User News, August 27th, 2002.

[2] J. Chase and Ron Doyle, "Energy Management for Server Clusters," Proc. the 8th Workshop Hot Topics in Operating Systems (HotOS-VIII), pp. 165, May 2001.

[3] Power, heat, and sledgehammer. White paper, Maximum Institution Inc., http://www.max-t.com/ downloads/whitepapers/ SledgehammerPowerHeat20411.pdf, 2002.

[4] F. Douglis, P. Krishnan, and B. Marsh, "Thwarting the Power-Hunger Disk," Proc. Winter USENIX Conf., pp.292-306, 1994.

[5] K. Li, R. Kumpf, P. Horton, and T. E. Anderson, "A Quantitative Analysis of Disk Drive Power Management in Portable Computers," Proc. Winter USENIX Conf., pp.279-292, 1994.

[6] Q. Zhu, F. M. David, C. F. Devaaraj, Z. Li, Y. Zhou, and P. Cao, "Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management," Proc. High-Performance Computer Framework, 2004.

[7] S.W. Son and M. Kandemir, "Energy-aware data prefetching for multi-speed disks," Proc. ACM International Conference on Computing Frontiers, Ischia, Italy, May 2006.

[8] S.W. Son, M. Kandemir, and A. Choudhary, "Software-directed disk power management for scientific applications," Proc. Int'l Symp. Parallel and Distributed Processing, April, 2005.

[9] E.Pinheiro, R. Bianchini, C. Dubnicki, "Exploiting redundancy to conserve energy in storage systems," Proc. Sigmetrics and Performance, Saint Malo, France, June 2006.

[10] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Fanke, "DRPM: Dynamic Speed Control for Power Management in Server Class Disks," Proc. Int'l Symp. of Computer Framework, pp. 169-179, June 2003.

[11] D. P. Helmbold, D. D. E. Long, T. L. Sconyers, and B. Sherrod, "Adaptive Disk Spin-Down for Mobile Computers," Mobile Networks and Applications, Vol. 5, No.4, pp.285-297, 2000.

[12] P. Krishnan, P. Long, J. Vitter, "Adaptive Disk Spindown via Optimal Rent-to-buy in

Probabilistic Environments," Proc. Int'l Conf. on Machine Learning, pp. 322-330, July 1995.

[13] J.-H Kim, S.-W. Eom, S.H. Noh, and Y.-H. Won, "Striping and buffer caching for software RAID file systems in workstation clusters," Proc. 19th IEEE Int'l Conf. Distributed Computing Systems, pp. 544 – 551, 1999.

[14] P.J., Varman, R.M Verma, "Tight bounds for prefetching and buffer management algorithms for parallel I/O systems," IEEE Trans. Parallel and Distributed Systems, vol. 10, no. 12, pp. 1262 – 1275.

[15] Y. Hu and Q. Yang, "DCD-Disk Caching Disk: A New Approach for Boosting I/O Performance," Proc. Int'l Symp. Computer Framework, 1996.

[16] S.-K. Lee, C.-S. Hwang, and M. Kitsuregawa, "Efficient, Energy Conserving Transaction Processing in Wireless Data Broadcast," IEEE Trans. Knowledge and Data Engineering, no. 18, no. 9, pp. 1225 – 1238, Sept. 2006.

[17] D. Stodolsky, M. Holland, W.V. Courtright II, and G.A. Gibson, "Parity Logging Disk Arrays," ACM Trans. Computer Systems, pp. 206-235, Agu. 1994.

[18] John Zedlewski, Sumeet Sobti, Nitin Garg, Fengzhou Zheng,Arvind Krishnamurthy, and Randolph Wang. Modeling Hard-Disk Power Consumption. Proc. Second Conference on File and Storage Technologies. March 2003.

[19] Qingbo Zhu, Francis M. David, Christo F. Devaraj, Zhenmin Li,Yuanyuan Zhou and Pei Cao, "Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management", 10th International Symposium on High Performance Computer Framework (HPCA'04), 2004

[20] Hogil Kim, E. J. Kim and Rabi N Mahapatra, "Power Management in RAID Server Disk System Using Multiple Idle States", in the Proceedings of International Workshop on Unique Chips and Systems (UCAS) 2005, pp.53-59.

[21] HDD Diet: Power Consumption and Heat Dissipation, Alex Karabuto July 11, 2005. http://www.digitlife.com/articles2/storage/hddpower.htm