Energy-efficient Fault-tolerance Storage System

Phisan Kaewprapha*
Department of Electrical and Computer Engineering, Thammasat University, Rangsit Campus, KhlongNueng, KhlongLuang, PathumThani 12120 Thailand
Nattkan Puttarak
Department of Telecommunications Engineering, King Mongkut's Institute of Technology Ladkrabang Chalongkrung Road, Ladkrabang, Bangkok 10520 Thailand

Abstract

In this paper, we propose an energy-efficient storage system, in which the idea of massive array of idle disk (MAID) [1] is incorporated into a parity-based fault-tolerance storage system, using a class of XOR-ing error correcting codes. This system aims to serve as a massing write-once read-many system (WORM). Example applications include massive data gathering, data archiving, or surveillance system. Unlike other proposed schemes or commercially-available MAID systems, firstly, we explicitly show that the extended life-span of disks, which is a byproduct of conserving energy by turning off some disks, reduces the probability of data loss compared with the same class of system with fault-tolerance feature. We consider using a deeper energy-saving state by totally powering off storage nodes instead of just slowing down disk's spin. The deeper offline state poses a challenge to our system; it is more difficult to monitor if any disk might have failed during the offline state. We address this issue by proposing a less frequent periodic monitoring scheme that makes use of the prolonged failure rate, while minimizing the degradation of the overall system reliability, aka failure that leads to data loss. Simulation shows that we can improve the power consumption without scarifying much of neither delay time nor reliability.

Keywords : Distributed storage, Array codes, Massive array of idle disks.

1. Introduction

Digital content are exponentially growing, even faster than the advancement in the capacity of storage units such as hard disk drives. This is due to the advancement in multimedia-capturing devices such as high-definition cameras found in smart phones, or the higher bandwidth connection through different types of communication channels, home internet, mobile broadband internet, the entertainment businesses and the demand from customers themselves. These are the driving forces for better/higher capacity and yet lower cost internet-working system as a whole. Moreover, the current trend is shifting toward cloud computing where storing, processing, and transferring information is managed by cloud providers. This offloads businesses/consumers of the tedious technological barrier and instead lets them focus on the content and actual work. The cloud provides reliable services with lower cost of ownership. As much as the trend is going in this direction, private cloud is also a popular solution. In many usage scenarios, implementing a private cloud makes a perfect sense; for instance, imagine a large

*** Correspondence :** kphisan@engr.tu.ac.th

volume of data flowing through a bottle neck from a company to a cloud provider, either to upgrade to higher bandwidth or until then, the private cloud is the reasonable choice.

Total energy consumption of a cloud computing system and the cost associated with it can be high. There are investigations and various attempts to reduce the energy consumption inside data centers. Among different parts of data center that consume energy, for example, a cooling system, a networking subsystem, the computation components, the storage space is accounted for some significant energy uses. Even further, replica of data is normally used in many systems either to boost performance for higher throughput or to provide redundancy. This adds both energy consumption and space required to store data.

Trying to reduce energy consumption while maintaining the availability, response time, throughput, is a challenging problem. In this paper, we propose a distributed solution with a fault-tolerance feature, which is customized to meet logging-like applications while saving energy at the trade-off of performance. We deploy array codes [2,3], seen in RAID systems, into a large-scale storage system and a powering up-down scheme to save some energy from idle parity disks. The simple array code increases the space and power efficiency; at the same time, powering up-down is the additional energy saving from this scheme. Then we simulate the impact of turning the system up-down to the life-span or rate of failure of the system, as well as the reliability, especially when the offline parity nodes fail. We also propose the scheme to periodically check the offline disks.

The rest of the paper is organized as follows. We provide a brief review of the on-going and related work in section 1.1. In section 2, we present the architecture of our new proposed system and the details. Next, in section 3, simulation results are provided. Finally, the conclusion is provided in section 4.

1.1 Related work

Massive Array of Idle Disks (MAID) is one of the early proposed systems aiming at reducing energy usage of a large storage system. It is designed to replace conventional tape archive system, which is the most energy-efficient data storage. It literally preserves the same level of energy saving as in tape system and gives better response time and throughput. MAID is built on the concept of caching a copy of frequently accessed data in a relatively small group of disks while keeping the rest of the disks at lower energy state. Up to current date, it has become a commercial product.

Popular Disk Concentration (PDC) [4] arrange data from left to right and from most frequently to least frequently accessed. It does not make a copy of data like MAID; rather, it moves data around and keeps just one copy. Both MAID and PDC storage system do not focus on fault-tolerance. Tornado MAID[5], Serria[6], and Rabbit[7] are storage systems with power saving and robustness features. Tornado MAID applies Tornado coding as an error protection measure. Serria uses replica and power-aware data layout that leave room for powering down some data servers. Rabbit assigns the first replica to a highly dense and always-active area while other replicas are distributed proportionally to the disks with lower density or utilization. It's similar to PDC but data is not uniformly distributed. Nevertheless, they share common architecture, the basic architecture consist of meta-data server and multiple chunk servers.

Our data storage scheme is also based on this common design. However, our work focuses primarily on space-efficient

redundancy where energy saving is a value-adding feature. MAID and PDC both keep small numbers of disks as hot areas while most of the other areas are kept in hibernating state. Our design keeps data disks available as they are while putting the parity disks in offline mode. This system has a similar design as the tornado-MAID system but uses simpler and smaller codes block for the sake of much simpler implementation. Although it's not as strong as Tornado-MAID, simulation shows later on that its reliability is not compromised. Our contributions include

1.1.1 Propose a distributed fault-tolerance file system that incorporates a power saving scheme onto parity disks by turning off parity disks whenever possible. The key feature of this system is fault-tolerance with space and energy efficiency.

1.1.2 Explicitly emphasize the importance of the prolonged disk life to the overall reliability and the energy saving.

1.1.3 Propose a periodic wake-up for health-checking to mitigate the reliability issue.

1.1.4 Dynamically simulate and show the impact of our scheme over the overall reliability of the system over the whole system lifetime.

2. System Design

The base structure of our networked distributed file system consists of a meta-data server, where file structure and information resides, and multiple data servers where the file contents are actually stored. The data servers are then divided into groups, each group forms an arbitrary disk array code. It can be a repetition code, a simple parity code, or other known RAID codes. Figure 1 depicts the structure of our file system. For this particular system, a code with systematic data and parity data residing on separated disks is preferred.

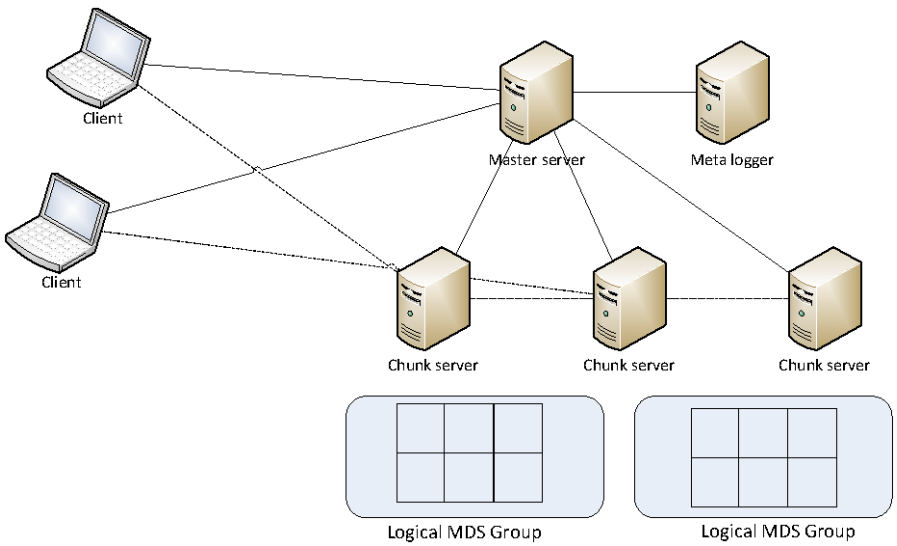

Fig.1. A system layout consisting of a meta server and groups data servers**.**

The separation is preferred because the parity disks portion can be turned off without affecting the availability of the primary data. The parity disks are turned off as soon as the group is idle for some preset time. The parity disks will remain off as long as there is no write to that particular group, the data read won't affect its state because the systematic data is available. Another case where a parity disk must be turned on is when one or more of the systematic disks fail. The parity disks will be turned on, and the systematic disks will be recovered. The recovering time may vary from data center to data center. In our simulation we use variable *d* to represent such a delay in rebuilding the code block.

When a parity disk is in the off state, it is possible that the failure may occur, and by the time we need it most, it's too late. Hence turning off the parity has a serious impact on reliability of the system. This will be shown in the simulation. To mitigate this side effect, a regular disk checking will be performed. Any error will be detected by this procedure. The benefit of this scheme will be even more significant when the failure rate depends on how long the equipments are in use (in this case, the disk drives). Equipments will eventually enter the wear-out period and eventually down with some probability. We use the Weibull distribution, which is well-known for modeling system reliability , to simulate our failure rate that is correlated to the drive's age.

The overall system health does not just depend solely on the model of single disk's failure. In particular it doesn't fail instantly when a disk fails. This is because we have a chance to recover from a single disk failure. Hence the fate of the data also depends on the recovering delay *d* . The effect of these parameters to the overall system will be shown in the next section.

3. Simulation setup and results

We study the reliability of our proposed system based on major key factors, in which the system reliability is affected. These factors are the annualized failure rate (AFR), the recovering (parity rebuilding time), and the increasing failure rate of old equipment. We simulate the dynamic nature of the whole system starting from the beginning of the service and continuing over the period of 10 years with a fine-grained time unit in hour. Disk failure can happen any time during this period with the probability of failure following the Weibull distribution where age and the failure rate are correlated. The repairing process is then simulated, and we assume that it takes some time to proceed to recovery. And this is the time during which the data are less protected. We start with a system of 100 groups of disks. Each group consists of 2 data disks and 1 parity disk, and the simplest form of the array codes, the single parity check code, is used. When the two data disks are encoded into the parity disk, the parity disk is then turned off to save energy. We measure the data loss due to the disk failure when the data cannot be recovered during the failure. That is two disks fail within the same group.

First we simulate the case with AFR being equal to 3% constant random failure rate over time[8]. We compare the case where the power saving scheme is not applied to the case where the power saving is used. However, in neither cases do we periodically wake up the offline disks to check their health. The results are shown in table 1.

Those two results can be considered as two extreme cases of the power saving scheme with periodic health checking period *p*. Without power saving is the case where its health checking is almost instantly detected. In contrast, the power saving scheme never checks the disks that are in the offline state as if the checking period *p* is set to infinity. We then plot the power

saving scheme and vary the checking interval p and plot the loss rate versus the interval as shown in figure 2. It clearly shows that just a relatively infrequent checking interval is enough to mitigate the risk of failed disks during the offline state.

The second factor we consider here is the effect of recovering time; we vary the recovery time versus the loss rate. As expected, in figure 3, the recovering time causes a higher chance of losing some data; however, within an acceptable risk, one day recovery time should provide enough time for any system to recover back with a relatively acceptable loss rate. Finally, we present the results that support our important point. When the up time plays an important factor in device's failures, the power saving scheme actually outperforms the standard settings both power saving and the probability of data loss. Table 2 is the comparison between the two. It is now that we see an opposite result; the power saving scheme outperforms the conventional in both power saving and the probability of data loss.

It is now 3 times more reliable than that of the conventional always-on scheme. It is also noted that we have to trade these advantages with the increase in the time to do a repair because parity disk has to be taken up before the parity rebuilding can be started.

Table 1. Data loss rate comparing the scenarios with and without the power saving scheme.

Without Power saving	With Power Saving
8.33E-07	2.08E-03

Table 2. Comparing the power saving scheme with health check scheduling versus a conventional standard system without the power saving scheme.

Without Power saving	With Power Saving & scheduled checking
0.036	0.012

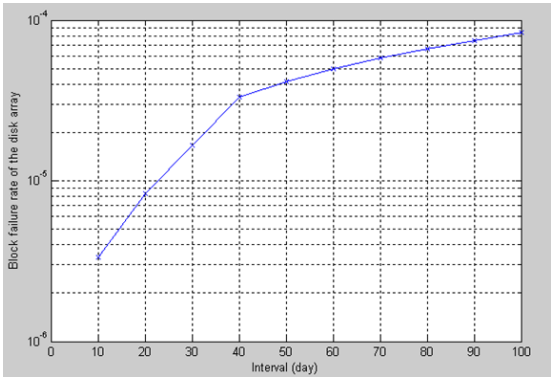

Fig.2. Probability of data loss versus the time health checking interval.

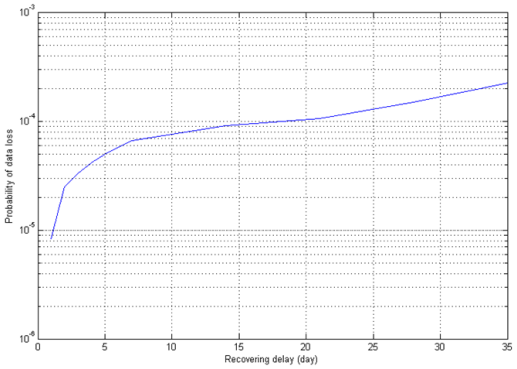

Fig .3. The effect of recovering delay over the data loss rate.

4. Conclusion

A simple yet efficient network storage system with a power saving scheme is proposed. The energy saving from this scheme is twofold: the saving as a direct result from the array codes deployment, and the saving from powering off the parity nodes. The effect is significant when the disk drives suffer from higher failure rates as they get older. In such case, our main finding is that the power saving scheme has a positive impact with respect to the probability of data loss due to the longer life span of the drives.

5. Future work

The actual implementation of this project is being developed based on an open-source network file system, MooseFS [9]. The array codes and power saving scheme presented in this paper will be the new features added. Once the system is fully implemented, an extensive analysis based on real usage data will be presented.

6. References

[1] Colarelli, D. and Grunwald, D., In Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, Baltimore, Maryland, pp. 1-11, 2002.

[2] Blaum, M., Brady, J., Bruck, J., Menon, and J., EVENODD: An efficient scheme for tolerating double disk failures, IEEE Trans. on Computing, Vol. 44, No. 2, pp. 192-202, 1995.

[3] Plank, J. S., The RAID-6 Liber8Tion Code, Intl. J. High Performance Computing Applications, Vol. 23, No. 3, pp. 242-251, 2009.

[4] Pinheiro, E. and Bianchini, R., Energy conservation techniques for disk array-based servers, ACM Proceedings of the 18th Annual International Conference on Supercomputing (ICS '04), Malo, France, pp. 68-78, 2004.

[5] Woitaszek, M. and Tufo, H.M., Tornado Codes for MAID Archival Storage, 24th IEEE Conference on Mass Storage Systems and Technologies (MSST'07), pp. 221-226, San Diego, CA, USA, 2007.

[6] Thereska, E., Donnelly, A. and Narayanan, D.,Sierra: a power-proportional, distributed storage system, Microsoft Research, Technical Report, MSR-TR-2009-153, 2009.

[7] Amur, H., Cipar, J., Gupta, V., Ganger, G. R., Kozuch, M. A. and Schwan, K., Robust and flexible power-proportional storage, In Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC'10), Indianapolis, IN, USA, pp. 217-228, 2010.

[8] E. Pinheiro, W. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In Proceedings of the 5th USENIX Conference on File and Storage Technologies,San Jose, CA, USA, pp. 2, 2007.

[9] http://www.moosefs.org