File System Write-Optimization on High-Performance Computing Support Platform

Jiye Wang[1, a], Nan Zeng[1,b], Jun Yu[2,c],

Guangxin Zhu[2,d], Hengmao Pang[2,e], Jian Lin[2,f]

[1]State Grid Corporation of China, China

[2]NARI Group Corporation State Grid Electric Power Research Institute, China

[a]jiyewang@sgcc.com.cn, [b]nzeng@sgcc.com.cn, [c]yujun@sgepri.sgcc.com.cn,

[d]zhuguangxin@sgepri.sgcc.com.cn, [e]panghengmao@sgepri.sgcc.com.cn, [f]linjian@sgepri.sgcc.com.cn

Keywords: Write Optimization; Distributed File System; Write-Ahead Log; HCSP

Abstract. The development of energy grid results in explosive data increasing. Handling the large volume of data not only requires vast storage space, but also requires the power to process them. Based on various write-optimization technologies, we proposed the high-performance computing support platform (HCSP) as a demonstration of write-optimization performance. HCSP integrates both hot/cold data layering and full-stack sense optimization, utilizes both SSD and SATA disk to accelerate the data storage and access. The experimental results show that compare to the small servers, HCSP not only provides a high-performance data access but also have a relatively short response time in various typical scenarios.

Introduction

Along with the development of energy grid and fast movement of smart grid, the interaction of power grid and user side is greatly enhanced. The storage and processing of massive data become an important research topic. More and more cloud services are built on the Unix-based operating systems, and the big data processing problem becomes a major challenge that Unix file system facing, the file system's write performance has attracted more and more attention.

Since the EXT file system is the core of Linux file management and data management, when the EXT file system access a directory with a lot of files, the efficiency will become very low due to the use of linear search; therefore it is vital to establish the write-optimization on the file system.

Background and Motivations

Optimizing with Non-volatile Storage and Checkpoint

Wan Hu et al. [3, 7] found that in the user data and metadata persistence process of file system, if there is abnormal power-down or system crash, it may lead to data inconsistencies in the file system. The existing Ext4 file system combines the transaction mechanism with the pre-log technology to ensure the consistency of the persistence operation. Pre-log technology writes the file system metadata to the disk twice. The granularity of the metadata is small, the quantity is large and the repeatability is high, affecting the performance of the program, and also shorten the life of Flash storage media. To solve the problem, they proposed a method that uses the new non-volatile memory (NVM) as a separate external device to store the log, and accessing directly(to file system?) by accessing command interface; at the same time using reverse scan technology to optimize the checkpoint process to reduce the repeated write operation of the same data block.

Shortcut JFS

The current log file system is not efficient for phase-shifting memory, and especially when writing large amounts of data, the current log file system can degrade the performance of phase-changing memory significantly. Therefore, Eunji et al. devised a new Shortcut log file system[4,6], which can reduce the log write blocking by 50%. In order to achieve the goal, they developed two novel plans: (1) only modifying the log of part of the blocking point slightly; (2) setting local checkpoint, and eliminates the overhead of replication. They implemented the Shortcut log file system on Linux 2.6.32, and measured the performance of the Shortcut log file system, and compared the measurement data with the existing log and log structure file system.

The results show that Shortcut-JFS file system's performance was improved by 54% and 96% compared to Ext4 and LFS.

When facing the system crash, keeping the data stability is the most important thing in storage systems. Most of the current file system uses the log method. As the log is usually stored in the file system, a fixed location and log writing is likely to pre-empt the disk bandwidth, and these easily lead to a series of storage system problems. Solving these problems will make the file system allow external storage of logs; explore the parallelism and the increasing spatial location of disk access mode.

Pedro et al. evaluated the external log file system [2,5,8] and found that the command and write-back logging modes would grow by 40 percent compared to the traditional file system. In addition, the data logging mode could outperform the other mode which workloads are under specific workloads mode significantly.

Regular EXT File System Optimization

The EXT file system's overall performance has been well optimized. Therefore, it was selected for standard of almost all Linux file system type. However, current write performance in EXT file system is still need to be enhanced. For example, when they deal with a directory file including a large number of files, if they index a file directory entry, the performance degradation of file system is obvious due to the time cost of linear search. Therefore, the performance will be significantly degraded. The main idea of Phillip's H-tree[1] is to generate red-black tree with the weight which is the hash value generated from directory entry in the file name, and to search quickly with red-black tree instead of linear search, so it will speed up the file retrieval speed. However, this method will change the original file system disk directory format and add an index block. Since the index block cannot be identified by the original file system, a fake directory entry is introduced to hide the index block in order eliminate the impact on the use of the original file system. This method has another drawback that the difficulty of reduction to restore the file system will increase when data inconsistencies come up in the file system when facing power failure or other reasons.

Write-Optimization Technologies for HCSP

Layer technology of Hot/Cold Data

Since current storage method cannot meet the demand of large volume of data storage, the layer of hot/cold data solve the problem significantly. We split the data into hot or cold by its accessing frequency, and store it into different levels of storage devices. Since the device is in different level and different performance, it is possible to achieve the high-performance and low-cost of the storage device. The virtualization technology will push the auto-layering from physical tier to application tier, and also provides the uniform activity management for users, as well as the automatic management of the layering. The layer storage technology usually supports the three-layer storage; the first one is high-performance layer, composed by solid state drives (SSD), since the SSD have no mechanical movements compares to the traditional disks, so its addressing speed is greatly enhanced. In the environment of random read/writes, the total performance will be greatly improved. However, those high-performance storage devices are costly compared to the traditional tape or SATA disks. The second layer is the performance layer, composed by the SAS disks. The bottom layer is the SATA high-volume disks. The Hot/Cold layer technology puts the low-frequent access data into the high-volume, low-performance media like the SATA in bottom layer, and moves the important, high-frequently accessed data to SSD in top layer. Since the data temperature difference indicates the data purpose, the access rate of hot data ensures the total performance of the system.

We utilize the technology in the high-performance computing support platform (HCSP) to increase the total performance while lower the cost at the same time. The high performance platform is supported by SSD card, SSD disks and SAS/SATA disks and its distributed file system. The platform can also be configured according the requirements of application and devices. In actual workloads, over 90 percent of the I/O operations concentrate in high-speed storage devices, and it eliminates the bottleneck of large OLTP and batch operations.

Full-Stack Sense Optimization

The high-performance computing support platform is also optimized for the Infiniband network communication, and provides multiple optimizations in software aspects. Under small stream of communications, the platform uses the internal send/receive operations of Infiniband. When facing the large volume of accesses, the platform uses the RDMA operations in order to increase the network usage, as well as the total system performance. We optimized the cache size and the cache replacing algorithm to provide the self-adaption of network traffic. Also, we modified the LRU algorithm in "Memcached" system, replacing it with the storage-cost based cache replacing algorithm.

Experimental Results

We perform our experiments on the high-performance computing support platform (HCSP) which is optimized for high-workload scenarios. The environments and hardware configurations are shown in Table 1.

Table 1 Experiment Configurations

	Type	Quantity	Misc
Server	E7 v3 CPU, 2TB Memory	N/A	Two servers
Operating System	Red Hat Linux 6.0	N/A	-
Network Card	Infiniband 40Gbps	2	Use IB Switch
Switch	36 ports IB switch	1	-
SSD	1.6TB	1	For each server
SAS disk	1.2TB, 10000RPM	20	For each server

In experiments, we use the standard test utilities to simulate the data pressure, and configure the utility to the State Grid Alert Management System (SGAMS) scenarios. In order to verify the effect on optimization, we compare the small server and HCSP with five typical scenarios: the alert query, the user type query, the regular data access, terminal access response, and large bulk transfer. The results were shown in Figure 2.

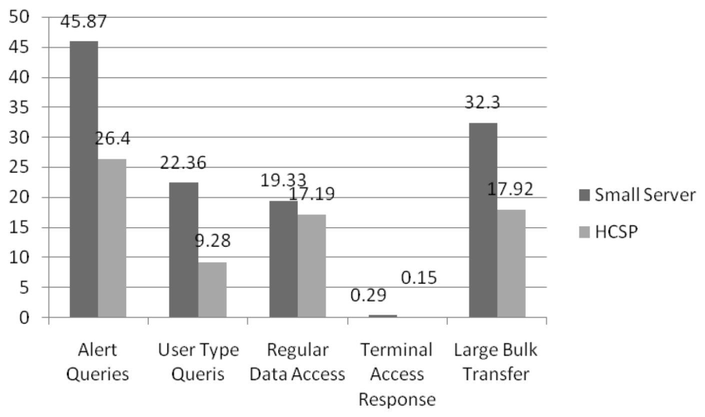

Figure 1 Performance Comparison in Typical Scenarios

According to Figure 1, the high-performance computing support platform have lower response time in all five scenarios, provided the reduction of response time in 30 percent on average. Compares to the traditional small server, the HCSP can provide a lower cost while maintaining the high performance in various scenarios.

Conclusions

The file system optimization is vital to the performance of the total system since the storage is usually the bottleneck of the server. Our optimization on HCSP platform includes the data layering, IB internal optimization, as well as cache optimization. HCSP provides not only the high I/O throughput but also the low response time, and maintain a relatively low cost. We are expecting the HCSP will solve the problems on other application scenarios in the future.

Acknowledgement

This work was financially supported by the State Grid Information System Performance Optimization Architecture and Critical Technology Research and Applications.

References

[1] Phillips D. A directory index for EXT2[C],//5th Annual Linux Showcase and Conference. 2001: 173-182

[2] Pedro and Luis. Analyzing the Performance of an Externally Journaled Filesystem[J]// Brazilian Symposium on Computing System Engineering: 2012.

[3] Wan Hu , Xu Yuanchao ,Yan Junfeng, Mitigating Log Cost through Non-Volatile Memory and Checkpoint Optimization[J]// Journal of Computer Research and Development, Seattle ,2015,52(6)..

[4] Eunji Lee, Seung Hoon Yoo, and Hyokyung Bahn, Design and Implementation of a Journaling File System for Phase-Change Memory, [J]// IEEE TRANSACTIONS ON COMPUTERS: 2015,64(5).

[5] M. Tim Jones, Anatomy of Linux journaling file systems, January 04, 2008

[6] YIN Jia-bin，TONG Xin，SUN Zhi—gang, et al. Research and Implementation of Baseboard Management in InfiniBand Switch [J]. Computer Engineering and Science , 2011, 33(1):20-24

[7] Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., & Schwarz, P. (1989). Aries: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. Acm Transactions on Database Systems, 17(1), 94-162.

[8] Baker, M. G., Hartman, J. H., Kupfer, M. D., Shirriff, K. W., & Ousterhout, J. K. (2000). Measurements of a distributed file system. Acm Sigops Operating Systems Review, 25(5), 198-212.