Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080657

NDM-Cache: A Network Cache for Cloud Computing System

Yunfa Li and Pengtao Wang[*]

Department of Computer and Technology, Hangzhou Dianzi University, 310018, Hangzhou, Zhejiang, China

Received: 27 Nov. 2013, Revised: 25 Feb. 2014, Accepted: 26 Feb. 2014
Published online: 1 Nov. 2014

Abstract: Block-level storage service is one of the basic services in a cloud computing system, it not only provides block-level storage volumes to the virtual machines for persistent data, but also improve the availability of data. However, with the rapid expansion of cloud computing systems and increasing number of the virtual machines, higher I/O performance of storage subsystem is demanded, when the storage subsystem under the condition of high load, its service quality is severely affected. According to this issue, this paper designs and implements the prototype of a network cache for cloud computing system, named NDM-Cache. This network cache system uses local block device as L1 cache, the block devices of other hosts in LAN as L2 cache, and proposes a data management solution for network cache system. On this basis, for the host with a spare memory space, we propose a double level cache optimization strategy to further enhance the overall system performance. The experimental results of prototype system show that in the case of high load storage subsystem, the network cache system can effectively improve the performance of storage system.

Keywords: Cloud Computing, block-level storage service, network cache

1 Introduction

In the early cloud computing [1] system, it often uses the host's own disk storage space when provides the user with a virtual machine instance service. This approach has two problems: firstly, the space of a single host is limited, it unable to meet the rapidly growing demand of user-space; Secondly, the allocation of disk space to the virtual machine instances is temporary, when the instance is terminated, this part of the space will disappear, resulting in the loss of user data. Therefore, today's popular cloud computing systems have their respective block-level storage subsystem, such as the AWS EBS [2], OpenStack Cinder [3] and other storage subsystems. They can not only provide persistence storage volume to the virtual machines, but also provide reliable storage service to the I/O intensive applications.

With the rapid expansion of cloud computing systems and increasing number of accessed applications, the concurrent access of storage subsystem rises sharply, centralized data storage and network bandwidth limitations lead to storage performance seriously decline. When the data request quantity exceeds a certain threshold, the system may be unable to handling user requests timely, resulting in the collapse of storage service. And cache technology has always been the main method and research hot spot to solve the performance problems of network storage, client cache system can effectively improve the overall performance of the storage system, however, most of the current network cache systems are based on client memory, although their performance are good, but the memory space is relatively small and the cache data is easy to lost. Although the access speed of local disk compare slowly with memory, it has larger space and reliable performance of continuity read and write, some researchers began to explore the method that use client-side local disk as a network cache, such as Stony Brook University's xCachefs [4], SUN's NFS [5], IBM's AFS [6], Carnegie Mellon University's Coda [7], UC Berkeley's xFS [8] and Pennsylvania State University's CAPFS [9] and other distributed file systems, they use the client disk as the cache of a network storage system, in order to alleviate the load pressure of the backend server and improve the overall system's performance and availability. However, since these cache systems are designed for specific file system, their versatility and flexibility are poor. In contrast, block-level cache can be transparently applied to most of the storage systems, with more versatility. But, at present in view of

[*] Corresponding author e-mail: wangpengtao.bh@gmail.com

the FC [10], NBD [11], AoE [12] and iSCSI [13] block-level protocols, only a few researchers use the client-side local disk space as a cache subsystem, among them, IBM and the university of Florida's DM-Cache [14] and Facebook's flashcache [15] is representative of the results. They both use Device Mapper mechanism [16], and map the local disk to the cache of remote block device. This scheme is applicable to iSCSI and NBD block-level storage network, it can significantly improve the performance and scalability of the storage network system. And due to the support of the Linux kernel, also shows excellent stability and availability. In addition, the university of Florida expanded the DM-Cache [17], it makes virtual machines on the same server to dynamically share a same local disk, and supports for dynamic cache replacement and write policy, this applies to different cloud computing environments.

However, DM-Cache and flashcache both only be used for a single client node, without using the correlation between client nodes in cloud computing systems. Considering the LAN, the client block device and other factors, this paper designs and implements the prototype of a network cache for cloud computing system, named NDM-Cache. NDM-Cache system uses local block device as L1 Cache, the block devices of other hosts in LAN as L2 Cache. When designing this system, this paper designs the network cache system structure based on local block device, and proposes a data management solution for network cache system. NDM-Cache system can fully utilize the advantages of local LAN and the speed difference between various levels in system, effectively extend the host's cache data access coverage, improve the overall efficiency of the cache data in client cluster, Thus effectively alleviate the high load problem of the storage subsystem, improve the availability and service quality of cloud computing storage subsystems.

The rest of this paper is organized as follows: Section 2 introduces the design and implementation of NDM-Cache. Section 3 describes the cache data management strategy and double level cache optimization strategy. Section 4 presents the theoretical analysis and the experimental evaluation of system performance. Section 5 is the conclusion of paper and future work.

2 Design and Implementation of NDM-Cache

2.1 Architecture Design of NDM-Cache

Combining with the network characteristics of the read-only shared storage system and the characteristics of local block device, this paper proposes the NDM-Cache system architecture, that is, a network cache architecture based on the local block device. NDM-Cache system is in the I/O channel between client host and remote network storage system. As shown in figure 1, in the NDM-Cache system, the client host cluster is connected to the same LAN and through the relatively slow network connection to the remote shared and read-only block storage. NDM-Cache system uses the virtualization technology provided by the Device Mapper, firstly, create the local disk cache of client host, then virtualization the cache device of each compute nodes in the client host cluster into a unified global cache that to be shared by client hosts as a global cache.

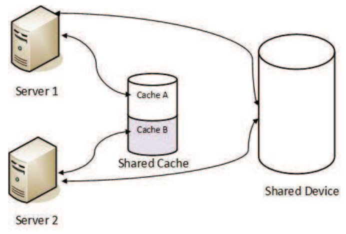


Fig. 1: System Architecture of NDM-Cache

In the above architecture, the Device Mapper is used to manage the remote block devices as well as the cache devices in local LAN. By making the corresponding mapping rule, the cache devices in local LAN are mapped into global shared cache device, this global shared cache device and the remote shared device are dynamically mapped to a virtual block device with a cache. And from the user perspective, this device is not much different from the original device, the only difference is the device descriptor, there is no difference in the use and capacity of equipment, figure 2 shows its mapping principle.

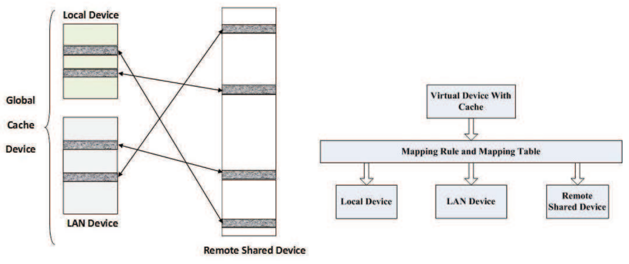


Fig. 2: Mapping Principle

Due to the advantage that the access speed of local cache device is relative faster than the device caches of other hosts in local LAN, this paper gives the local cache device first priority, the other cache devices in LAN second priority, and shared storage device third priority, when specifying mapping rule. That is, when accessing a block, the user first attempts to access the data on local host, then check whether the data is on other cache devices in LAN, finally, if the data is not on the cache device, it will access the remote shared block device. Such mapping rule can make full use of the speed difference between various levels and the advantage of local LAN, improve the overall performance of system.

2.2 Prototype Implementation of NDM-Cache

This paper uses the Device Mapper technology to realize the prototype system of NDM-Cache. For the setting of cache system in LAN between multiple nodes, it involves the complexity of I/O path, the complex scheduling of cache resources and the consistency of cache. To simplify the research, this paper only temporary realizes the caching system between two hosts in LAN and the remote shared storage system.

This paper uses the two hosts on their shared cache space is mapped into a global cache device, then by making the corresponding mapping rule, gives two devices different access priority (i.e., local block device prior to network block device), forms a hierarchical access method. This paper realizes the NDM-Cache mechanism in the form of the kernel block layer driver module. NDM-Cache module can be easily plugged into or removed from the Linux kernel by user. The implementation process of NDM-Cache module mainly involves the implementation of private data structure and mapping rule.

2.2.1 The Implementation of NDM-Cache Private Data Structure

Considering the characteristics of the prototype system architecture, in the NDM-Cache, this paper set a data structure of shared storage device named src_dev, a data structure of local host cache device named hostcache_dev, and a data structure of LAN device named lancache_dev. In order to achieve a faster cache data search, this paper sets two cache block metadata hash tables for these two kinds of devices, and places the two tables in kernel space memory. When searching the data block, operating system first searches the metadata table in kernel space, then according to the search result, determines whether the cache data exists in the cache device and exists in which position. Compared to access block device directly, this approach will has a great improvement in speed. This paper also sets the cache block size named size and the cache associativity parameter named assoc, which are used to specify the cache block size and data block storage group size.

2.2.2 The Implementation of NDM-Cache Mapping Rule

On the basis of setting two kinds of tables, this paper realizes the mapping rule of NDM-Cache in the mapping method cache_map, the specific process is described by the later algorithm. This paper uses the kernel replication thread kcopyd to realizes the asynchronous replication of cache data. The cache data replication occurs mainly in the first cache data loading and the cache data replacement. For the replacement strategy of data cache, this paper uses LRU (Least Resent Used) algorithm. This algorithm has a simple mechanism, and a high cache hit ratio.

3 Cache Data Management and Double Layer Optimization Strategy

This paper designs the NDM-Cache oriented system block level cache data management solution. This solution mainly includes the cache data storage and organization scheme and cache data mapping algorithm.

3.1 Analysis of Influencing Factors

Because of working in the local area network (LAN) environment, NDM-Cache system has obvious difference compared with the traditional caching system built in the memory in the cache media characteristics and processing level of operating system. So in the design of the cache data management solution, the two differences are mainly considered.

3.1.1 The Influence of Local Block Device

This paper refers the local block device as the device within the local area network (LAN), which is mainly normal disk medium of its physical layer. Compared with SDRAM and flash disk medium, normal disk medium has three differences as follows: firstly, the response speed is slow; secondly, data continuity requirement is higher; Thirdly, data is non-volatile.

Because normal disk media response speed is slow, the operating system's I/O waiting time will be longer accordingly. When the network block device in a local area network (LAN) used as a cache system, the I/O processing speed is also affected by network bandwidth and protocol. Therefore, in the process of cache handling, we must fully consider the impact of this feature, especially to estimate whether the additional I/O time brought by cache operation will lead to decline in overall speed I/O. If it is necessary, cache operation can be skipped. Because requirement of normal disk medium for data continuity is high, local characteristics of data can be combined with disk storage location, which stores data blocks of high correlation in the neighboring physical storage position. Because the disk data has characteristic of non-volatile, so in the case of a power outage or network suddenly disconnected, the data on the local block device also will not be lost, and therefore the cached data do not need to be reload when restart.

3.1.2 The Influence of Operating System Features

In general, operating system processes the request from general block layer and normal file layer in a different way. Operating system allow writing for normal file level requests on reading and writing, but the writing time for request from general block layer is short, otherwise the operating system may be deadlock for the I/O without treatment for a long time. So comprehensive consideration is need on this operating system properties as well as the local block device I/O processing speed to make reasonable I/O processing rules.

3.2 Cache Data Organization and Storage

In the NDM-Cache system, the cached data mainly includes two aspects, namely the actual cache data and the metadata associated with cached data. Metadata is mainly used to find the corresponding cache data, which records basic information about cache data, including the cache block size, the offset set in the shared storage device of original data corresponding to the cache data , associative parameters of set as well as status information and so on. In order to distinguish metadata from ache data in storage, this paper stores metadata in the tail of the device. And in order to increase the search speed, in this article these metadata will be loaded into kernel memory space.

This cached data processing granularity of data blocks, which consist of a number of sectors, and the user can customize the data block size. On one hand the response of disk media is slow relative to the SDRAM, on the other hand performance of disk media on processing consecutive read and write is higher than the random read and write, so the cache block size should be not too large nor too small. Too big cache block will result in long waiting time for the operating system, affecting the normal operation of the operating system; if the cache block is too small, it can not make good use of continuous nature of cache data, and will generate more metadata , declining the overall system performance. Considering Linux operating systems typically use 4KB page size as the default, this paper use eight sectors (each sector 512B, 8 sectors is 4KB) as the default cache block size.

In terms of cache data storage, this paper adopts the way of set associated, that multiple sectors of data make up a single data block, multiple of data blocks make up a single data set. In this paper, hash algorithm is introduced to map a data block to its set data position. The hash key is got by calculating the position offset on a remote storage device of cached data block, the hash value is then divided by the correlation parameters to get the final set offset. However, the original hash algorithm can't use the local characteristics of the data, which may lead to near the position original data be mapped to relatively distant position on cache device. Since normal disk has a better performance when processing consecutive data than random data, we need to improve the original hash algorithm so as to makes the consecutive data on the shared storage devices can store in close position. This paper uses a simple but effective improved hash algorithm, to obtain data block local characteristics. In this hash algorithm , the article use the parameters of continuity as cache_consecutive to set the number of stored continuously data blocks and before the calculation of hash value the original offset is divided by the continuity parameters. An example about cache data stored on the cache device is shown in figure 3.

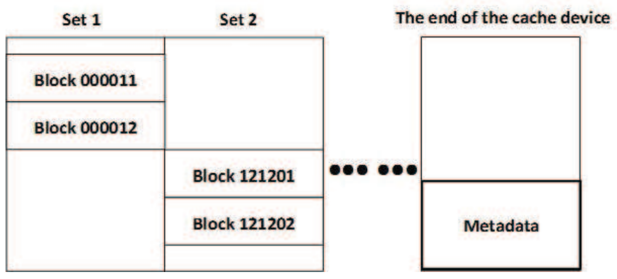


Fig. 3: Schematic Diagram of Cache Data Storage

3.3 The Design of Mapping Algorithm for Cache Data

In this paper, Device Mapper is used to achieve the appropriate mapping rules and virtualize the client on each node in the cluster into a unified cache device. Each node accesses to the cached data block via the global metadata table. The local cache device, LAN cache device on other hosts and remote block device were given three different priorities from high to low in the rule-making. And in order to achieve the level of relations, this paper designs corresponding mapping algorithm for cache data. The algorithm disposes the generic block layer request bio from the top of the file system or VFS layer based on the priority of plot devices and shared storage devices.

Table 1: Related Data Structures and Parameter List

Type	Domain
struct	Bio
struct block_device *	bio→bdev
struct dm_dev *	src_dev
struct block_device *	src_dev→bdev
struct dm_dev *	hostcache_dev
struct block_device *	hostcache_dev→bdev
struct dm_dev *	lancache_dev

Figure 4 depicts the flow of processing algorithms about a bio sent from the node in file system from the upper layer or I/O control layer, which involves the relevant data structure and the parameters in Table 1. NDM-Cache system processing bio specific process is as follows:

Step 1 The node query metadata belong to their compute nodes in global metadata tables. If hits, the block device descriptor (bio→bdev) requested by generic block layer will be modified cache block device descriptor (hostcache_dev→bdev). After doing some appropriate treatment to the bio, it will be handed over to core, the bio is processed after this; If not hits, proceed to step 2.

Step 2 The node continues to research metadata belong to other local computing nodes in global metadata tables. If hits, bio→bdev compute nodes will be modified into device descriptor lancache_dev→dev of cache block devices. After doing some appropriate treatment to the bio, it is taken to the kernel and then goes to step 3; If not hits, proceed to step 4.

Step 3 When kernel copies block data, it quires the status of local cache device and finds the available replacement block according to the replacement algorithm, and the data block is replaced by bio required. Then updates the global metadata table, thus ends the process for this bio.

Step 4 bio→bdev is modified into device descriptor src_dev→bdev of a remote block device. Next, the bio is delivered to the kernel, then goes to step 3.

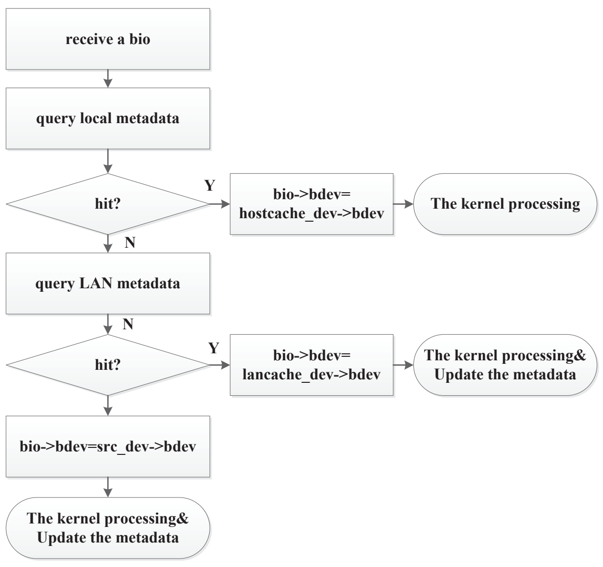


Fig. 4: Flowchart of Cache Data Mapping Algorithm

3.4 Double Layer Cache Optimization Strategy

Considering the memory relative to the disk on the speed advantage and in order to make the NDM-Cache for a further improvement in performance. This paper proposes double layer cache optimization strategy in view of the spare memory space in the client. That divide spare memory space and plus a layer of memory cache on the basis of the original global cache equipment. This optimization program has two preconditions:

Firstly, the client must have free memory space, and must be at the level of GB. The default size of a single block data is 8 sectors, that is 4KB. Memory block device must provide sufficient buffer space to make the effect of acceleration in cache mechanisms, which often need to divide dozens or even hundreds of MB for cache space. Coupled with the interventions and scheduling of operating system, only when the free space reaches at the level of a few hundred MB or even GB , can the normal operation of the client do not been affected, but also to provide a stable buffer space.

Secondly, the client power outages and other fault conditions are not taken into account. Because memory medium is volatile media, in the case of failure or power failure may cause loss of data. Thus the internal storage could be considered into two-story cache memory in the case of the loss of data in the cache has little effect on the system.

Double layer cache optimization strategy is still using Device Mapper mapping mechanism, its structure and principle are shown in figure 5.

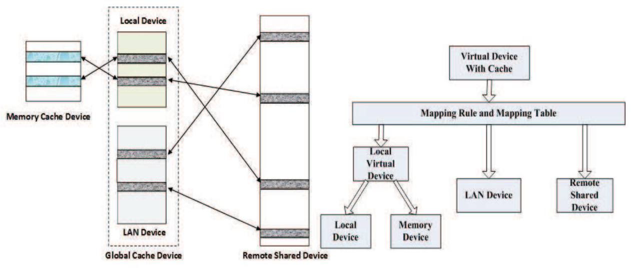


Fig. 5: Schematic Diagram of Double Layer Architecture Mappings

4 Performance Evaluations

The NDM-Cache prototype system mounts the LAN cache devices and remote storage device through the iSCSI technology. The prototype system uses two client servers, six assisted client servers and one storage server. All the servers are in a LAN. These servers' configurations in experiment are as shown in Table 2.

Table 2: DNM-Cache Prototype System's Configurations

Node name	Quantity	Memory	OS
Client Node	2	32G	Ubuntu 11.10
Added Node	6	32G	Ubuntu 11.10
Server Node	1	32G	Ubuntu 11.10

4.1 Experiments of the NDM-Cache Prototype System

This section designs the speed of data reading experiments for NDM-Cache prototype system. Experiments are divided into two parts, the first part uses the normal disk as cache devices and the second part uses the SSD as cache devices.

4.1.1 The Read Speed Experiment Based on the Normal Disk

In the read speed experiment based on the normal disk, specific experimental settings are as follows: the size of remote storage device is 30GB; on each client server, the normal disk cache space is 2GB; the size of cache data block is 32 sectors, namely 16KB; set-associative parameter is 1024, the 1024 data blocks are a single storage group; the continuity parameter is 512, that is 512 consecutive data blocks are to be processed simultaneously. Firstly, this paper uses only two client servers and a storage server for testing. The test respectively do the data read on original shared storage devices without cache, the virtual devices uses DM-Cache mechanism and NDM-Cache system. During the test, two client nodes simultaneously do data read operations on a shared storage device. In the test, the data block size is divided into 4KB, 8KB, 16KB, 32KB , 64KB and 128KB. Then, we add six assisted client servers, these assisted servers connect to a same storage device provided by the storage server. Eight client servers simultaneously access to the shared storage devices and the same with the above experiment process, we do the test on data of different block size under different mechanisms. The specific experimental results are as shown in figure 6.

As shown in figure 6, figure (a) represents that when the client-side has only two servers, NDM-Cache system and DM-Cache system's read performance are worse than the original shared storage device. Figure (b) represents that when the number of servers in the client-side increased to 8, The NDM-Cache system's performance relative to the original shared storage device has improved by an average of 11.31% and relative to the DM-Cache system, has improved by an average of 2.34%. The reasons for the above results are as the following three aspects:

Firstly, when the storage server's load is relatively low, the speed that client servers access to the shared storage device and access to the local normal disk does not differ much. At this time, the operating system's scheduling delay and Device Mapper's processing delay mainly influences the data read performance. So, the NDM-Cache system and DM-Cache system's performance relative to the original shared storage device's performance has declined.

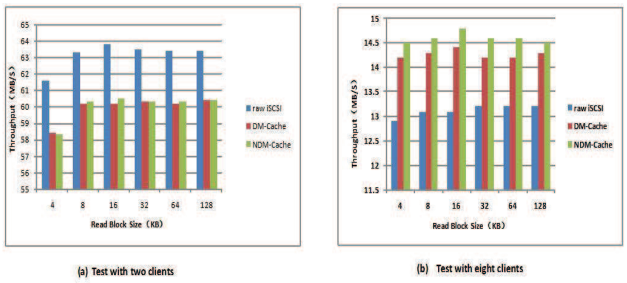


Fig. 6: The Normal Disk-based Cache Experiment Results Comparison

Secondly, when the storage server's load is high, the speed that client servers access to the shared storage device and access to the local normal disk differ much. At this time, the operating system and Device Mapper's impact relative to the impact of access latency is small. So, using disk cache mechanism can improve the overall access speed.

Thirdly, because NDM-Cache system takes full advantage of the local area network, so it has better performance than DM-Cache system.

4.1.2 The Read Speed Experiment Based on the SSD Disk

In the read speed experiment based on the SSD disk, in addition to the cache device types, other experimental configurations are same with the above experiment that based on the normal disk. And in this experiment, we also respectively test on the original shared storage device without cache, the virtual devices use DM-Cache mechanism and NDM-Cache system. The specific experimental results are as shown in figure 7.

As shown in figure 7, when the client-side has only two servers, the NDM-Cache system and DM-Cache system's read performance are worse than the original shared storage device; when the number of servers in the client-side increased to 8, the NDM-Cache system's performance improves 18.42%; and relative to the DM-Cache system, improves 2.76%. The main reasons for the above results are similar to the case of using normal disk: Firstly, when the storage server's load is relatively low, the operating system and Device Mapper will have a great impact on the I/O; Secondly, when the load increased to a certain extent, the cache effects of SSD begins to show; Thirdly, NDM-Cache system takes better use of the LAN, therefore, the overall performance is better.

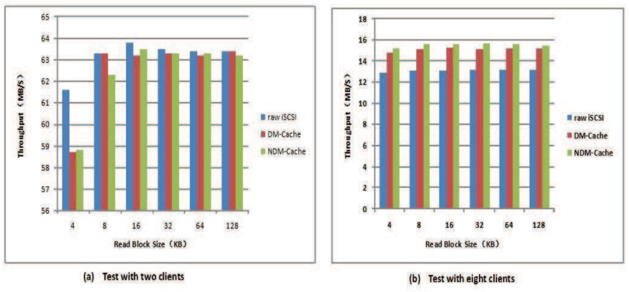


Fig. 7: The SSD Disk-based Cache Experiment Results Comparison

4.2 Experiments of Double Layer Cache Optimization

In view of the double layer cache optimization strategy proposed in this paper, this section respectively evaluates the data read performance on NDM-Cache prototype system with the double layer cache optimization strategy based on normal disk and SSD. The specific experimental settings are still same with the above experiments.

In this experiment, 2 client servers and 6 assisted client servers simultaneously access to the shared storage device provided by the storage server and the size of data block is 4KB, 8KB, 16KB, 32KB, 64KB and 128KB. The results are shown in figure 8.

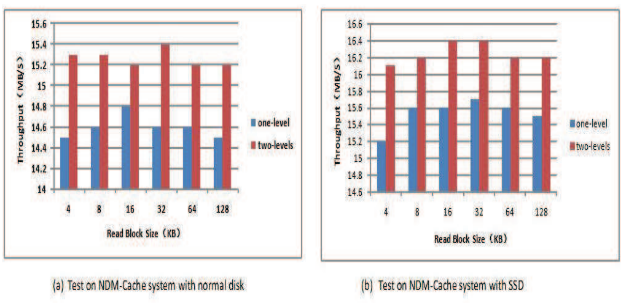


Fig. 8: The SSD Disk-based Cache Experiment Results Comparison

As shown in figure 8, compared to the single layer cache strategy, the NDM-Cache prototype system's read performance with the double layer cache optimization strategy based on normal disk and SSD has improved by an average of 4.57% and 4.61%. The main reason for the above results is that the NDM-Cache system with double layer cache optimization strategy uses the memory block device as a two-levels cache and it's performance is better than the normal disk and SSD. This experiment demonstrates that in the case of which the client server has a spare memory space, using double layer cache optimization strategy can improve the overall performance of the shared storage system.

5 Conclusions and Future Work

Based on the problem of the decline of the storage subsystem's service quality under the condition of high load, this paper designs and implements a network cache system for cloud computing system, named NDM-Cache. This network cache system uses local block device as L1 cache, the block devices of other hosts in LAN as L2 cache and proposes a data management solution for network cache system. On this basis, for the host with a spare memory space, we propose a double level cache optimization strategy to further enhance the overall system performance. Proved by experiments that eight client nodes simultaneously access the shared storage device, as opposed to the original shared storage devices and NDM-Cache system, data reading performance of the NDM-Cache prototype system based on local normal disk has increased by an average of 11.31% and 2.34%, while the NDM-Cache prototype system based on the SSD has increased by an average of 18.42% and 2.76%. After adopting a double cache optimization strategy, the NDM-Cache prototype system with the double layer cache optimization strategy based on normal disk and SSD has improved by an average of 4.57% and 4.61%.

In this paper, some aspects still need further research and improvement. In our future work, we will consider more data management solutions for network cache system. We will study automatic mounting mechanism for network cache system and pluggable network cache devices. We will also implement various cache replacement algorithms and study the effect of cache strategy on cache efficiency under various experimental conditions.

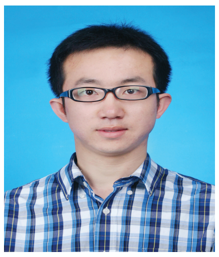
Acknowledgement

This paper is supported by the National Key Technology Support Program under Grant No.2012BAH24B04, the National Natural Science Foundation of China under Grant No.61202094 and the major science and technology plan of zhejiang province(No. 2011C11038).

References

[1] P. Mell and T. Grance, NIST special publication, **800**, 7-7 (2011).
[2] Amazon EBS Service, http://aws.amazon.com/ebs/.
[3] OpenStack Cinder, http://wiki.openstack.org/wiki/Cinder.
[4] G. Sivathanu and E. Zadok, Stony Brook University, Technical Report FSL-, **05**, 5-5 (2005).
[5] R. Sandberg, D. Goldberg, S. Kleiman, et al, Design and implementation of the Sun network filesystem, Proceedings of the Summer USENIX conference, 119-130 (1985).
[6] J. H. Howard, An overview of the andrew file system, Carnegie Mellon University, Information Technology Center, (1988).
[7] M. Satyanarayanan, J. J. Kistler, P. Kumar, et al, Computers, IEEE Transactions on, **39**, 447-459 (1990).
[8] T. E. Anderson, M. D. Dahlin, J. M. Neefe, et al, ACM Transactions on Computer Systems (TOCS), **14**, 41-79 (1996).
[9] M. Vilayannur, P. Nath, A. Sivasubramaniam, FAST, **5**, 2-2 (2005).
[10] Z. Meggyesi, Fibre channel overview, Research Institute for Particle and Nuclear Physics, (1994).
[11] M. Lopez, P. T. A. Arturo Garcia Ares, Linux Journal, **2000**, 40-40 (2000).
[12] B. Coile, S. Hopkins, The ATA over Ethernet Protocol, Technical Paper from Coraid Inc, (2005).
[13] K. Z. Meth, J. Satran, Design of the iSCSI Protocol, Mass Storage Systems and Technologies, Proceedings. 20th IEEE/11th NASA Goddard Conference on. IEEE, 116-122 (2003).
[14] E. Van Hensbergen, M. Zhao, Dynamic policy disk caching for storage networking, http://visa.cs.fiu.edu/ming/dmcache, (1997).
[15] Flashcache, http://github.com/facebook/flashcache/.
[16] Device-mapper Resource Page, http://sourceware.org/dm/.
[17] D. Arteaga, D. Otstott, M. Zhao, Dynamic Block-level Cache Management for Cloud Computing Systems, Conference on File and Storage Technologies, (2012).

Yunfa Li received the PhD degree in Computer System Organization from Huazhong University of Science and Technology, Hubei, China, in 2008. He is currently a vice professor in software engineering in Hangzhou Dianzi University, China. His research interests include Cloud Computing and Virtual Machine Technique.



Pengtao Wang reveived Bachelor degree in Communication Engineering from Yanshan University, Hebei, China, in 2010. He is now studying the Master of computer software and Theory in Hangzhou Dianzi University, China. His research interest is Cloud Computing.