SuperNBD: An Efficient Network Storage Software for Cluster[1]

Rongfeng Tang, Dan Meng, Jin Xiong

National Research Center for Intelligent Computing Systems
Institute of Computing Technology
Graduate School of the Chinese Academy of Sciences
{rf_tang, md, xj}@ncic.ac.cn

Abstract. Networked storage has become an increasingly common and essential component for cluster. In this environment, the network storage software through which the client nodes can directly access remote network attached storage is an important and critical requisite. There are many implementations exist with this function, such as iSCSI. However, they are not tailored for the large-scale cluster environment and cannot well satisfy its high efficiency and scalability requirements. In this paper, we present a more efficient technology for network storage in cluster and also give detailed evaluation for it through our implementation - SuperNBD.The results indicate that SuperNBD is more efficient, more scalable, and better fit for cluster environment.

1. Introduction

With the steadily increasing of data capacity produced by scientific applications and high I/O rate requirement, the networked storage has become a common but essential component of high performance cluster environment. It permits hosts to easily access remote data through network.

Most storage area networks (SAN) used to adopt Fiber Channel [1] as their private storage network, however, due to the expensive cost of the hardware, most medium and small-scale enterprises cannot afford it. Recently, with the advent of Gigabit (or even 10 Gigabit) Ethernet, the IP based storage networks (i.e. IP-SAN), which can leverage the existing LAN environment, are becoming more popular.

The network storage software for hosts to transparently access the data block on storage devices is the critical layer for network storage. iSCSI is a kind of network storage protocol currently used for IP-SAN. It encapsulates and transports the SCSI command and data across network [3], dividing the large buffer come from buffer cache of OS kernel and packetting it before sending them to the server. So every buffer transmission involves several individual messages of SCSI command and sub-buffers of data block [3]. In addition, in order to cope with the security problems when transmits across wide area network, it introduces some mechanisms which are

[1] This work was supported by the National High-Technology Research and Development Program of China (863 Program) under the grant No.2002AA1Z2102 and the grant No.2002AA104410

H. Jin et al. (Eds.): NPC 2004, LNCS 3222, pp. 244-247, 2004.

unnecessary in a close and reliable network environment such as cluster, these mechanisms will induce more overhead and lead to decrease of performance.

We present a new compact and efficient network storage implementation technology for the cluster environment named SuperNBD. The practical experiment shows that it is well efficient and scalable.

The rest of this paper is organized as follows. In the next section we will discuss in detail about design issues and implementation of SuperNBD. In section 3 we analyze its performance and scalability, and compare these metrics with other implementations. Section 4 concludes our paper.

2. Design Issues and Implementation

SuperNBD is composed of two different parts according to their functionality: SuperNBD client and SuperNBD server; both reside at the buffer cache layer of kernel, as shown by figure 1. The SuperNBD client receives the data operation requests issued by higher layer of kernel, such as VFS and directly forwarded it to SuperNBD server. We have introduced some specialized mechanisms to improve its efficiency and scalability as follows.

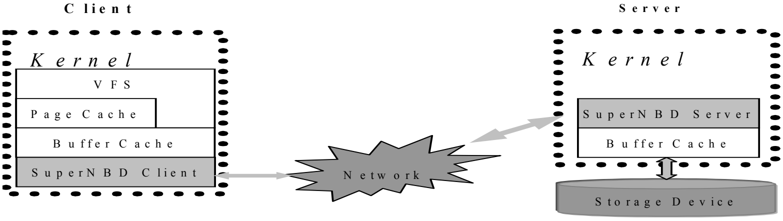

Fig. 1. The overview architecture of SuperNBD

In order to increase the total data throughput of storage system, several service threads are introduced on both sides specializing in data processing, so that the data of a file distributed across multi-devices can be read and written concurrently.

Since all the blocks within a single request are corresponding to the same device in sequence, only one simple message with the information about first block's sequence number and the total block quantity need to be sent to server before data transmission, less then those of iSCSI.

During the whole process along I/O path, all data blocks requested are transferred directly from client data block cache to that of server and vice versa, eliminates any necessary of memory copy within SuperNBD.

In order to increase the write bandwidth for highly data-intensive applications we adopt a kind of asynchronous write mechanism. As one characteristic of block I/O storage, each write operation will completely overwrite the content, so it is unnecessary to read corresponding data block from disk beforehand, only allocate a block cache from main memory and directly store the data into it, thus, as to the client issues this request, the whole operation is completed promptly. The kernel will flush the entire dirty data blocks to disk when the free memory reaches a certain low watermark, so that the process of client side writing and server side real disk writing can be parallelized and greatly improve the total data writing throughput of storage I/O path.

Things are different for data reading. Most often, when clients issue read request, the file data is just on the physical storage device. The large cache that greatly contributes to high write bandwidth has little advantage here, and the read can only be handled synchronously. In order to increase the total read performance of SuperNBD, we present a kind of adaptive data block prefetching mechanism according to the locality feature of data reading on the server end. For example, assume A and B are two requests reach server side in order, but might issued by different clients, both relate with the same device. Request A read blocks with sequence number (block number) from *a0* to *an*, and B from *b0* to *bm*, if the value of |*b0* – *an* | are within a certain reasonable interval, it can be considered as sequential operation, and when server finishes the request B, it still keep reading next several blocks, so that the next read operation can mostly be hit on cache.

A pipeline mechanism has been introduced, which can parallelize the device reading with data transmission, two major time-consuming operations during reading process. In SuperNBD, when some of blocks are ready (hit in buffer cache or just be read from device), and others are during handling by local kernel, the ready data is firstly transferred to the clients. After finishing sending the previously ready data, most of the other blocks are ready in memory now. In this way, it can greatly reduce the response time for read request and completely exert the potential performance of hardware resource.

In addition, the feature of SuperNBD server's combining with the local buffer cache can also benefit for shared operations, i.e. multi-clients simultaneously access the same data blocks. In this situation, only one physical device operation needed for each block, so that most requests can be serviced directly from the server buffer cache. This enable SuperNBD to scale well to large number of clients.

3. Performance Comparison and Analysis

In this section, we evaluate the efficiency and scalability of SuperNBD and present performance comparison with unh-iscsi[3]. There are two sets of environments for our experiments:

（1）Set1. Consists of 33 nodes, each has four AMD opteron(tm) processor (2.2GHZ), 8GB memory, SuSE Linux 8.0 with kernel 2.4.19SMP. All nodes connected by Gigabit Ethernet. This environment is used to test efficiency and scalability of SuperNBD.

（2）Set2. Consists of 9 nodes, each with dual 2.4GHz Intel Xeon Processors , 1GB memory, Red Hat 7.2 with kernel 2.4.18-3smp. Both connected by 100bit Ethernet. This environment is used for performance comparison between SuperNBD and unh-ISCSI, for unh-iscsi cannot support x86_64 architecture.

3.1. Efficiency and Scalability Evaluation

In this test, we use one SuperNBD server and continually increments the number of clients, each read or write 1GB data with 1MB record size. Figure 2 shows SuperNBD has well efficiency and scalability and can always keep peek performance along with scale increasing. The asynchronous writing and adaptive read-ahead mechanism greatly contribute to this achievement.

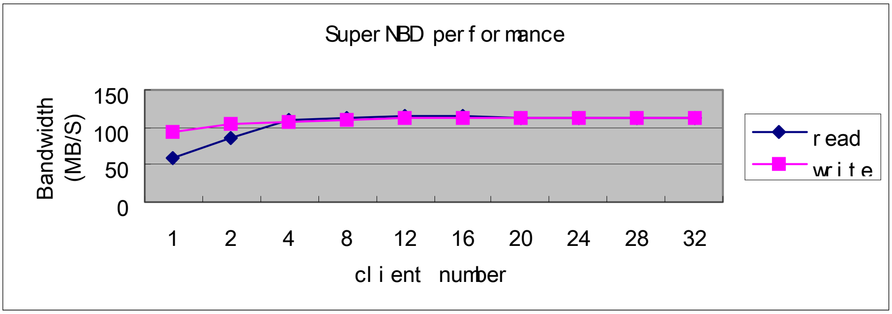

Fig. 2. Efficiency and Scalability of SuperNBD

3.2. Performance Comparison with Other Implementations

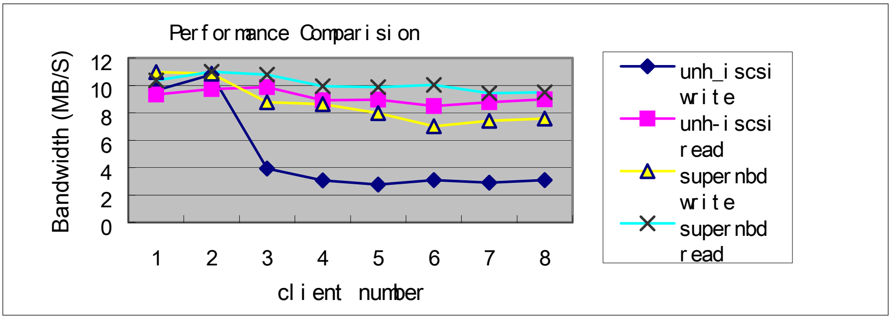

Fig. 3. Performance comparison between SuperNBD and unh-iscsi

Figure 3 shows the performance comparison between SuperNBD and unh-iscsi. SuperNBD outperforms unh-iscsi on both writing and reading, mainly due to its compact protocol and optimized implementation mentioned in section 2.

4. Conclusion and Future Work

In this paper, we present a compact but efficient technique to construct network storage for cluster and also evaluate the performance of our implementation named SuperNBD. As the result shows, SuperNBD is efficient and scalable, and better fit for the cluster environment.

References

1. Prasenjit Sarkar, Sandeep M.Uttamchandani, Kaladhar Voruganti, Storage over IP: A Performance Study. IBM research report, 12/2002
2. http://www.iol.unh.edu/
3. Julian Satran, Kalman Meth, IBM. iSCSI,draft-ietf-ips-iscsi-20. .