

HMF: High-available Message-passing Framework for Cluster File System

Dong Yang, Zhuan Chen, Rongfeng Tang

National Research Center for Intelligent Computing
Systems

Institute of Computing Technology
Chinese Academy of Sciences

Graduate University of Chinese Academy of Sciences
Beijing, China

{yangdong, chenzhuan, rf_tang}@ncic.ac.cn

Jin Xiong, Dan Meng

National Research Center for Intelligent Computing
Systems

Institute of Computing Technology
Chinese Academy of Sciences

Beijing, China

{xj, md}@ncic.ac.cn

Abstract—In large-scale cluster systems, the failure rate of network connection is non-negligibly high. A cluster file system must have the ability to handle network failures in order to provide high-available data accesses service. Traditionally, network failure handling is only guaranteed by network protocol, or implemented within the file system semantic layer. We present the high-available message-passing framework which is called HMF. Based on the operation hierarchy in cluster file system, HMF guarantees the availability of each pair of network transmissions and their interaction with the file system sub-operations. It separates the network fault-tolerance design from the file system and keeps a simple interface between them. HMF could handle a lot of network failures internally, which greatly simplifies the implementation of file system semantic layer. Performance results show that HMF can increase the availability of message passing and reduce the cost of recovery from network failures. When there are two network channels, HMF also improves aggregate I/O bandwidth by 80% in normal condition while the performance degradation due to recovery is below 10%.

Keywords—cluster file system, message passing layer, high availability

I. INTRODUCTION

As the scale increases rapidly in today's cluster, the probability of failure for each component becomes more and more serious [9]. The problem of availability in large-scale file system is critical especially for the network transmission.

The operation in distributed file system can be viewed as a kind of hierarchy structure (Figure 1). File operation processed in distributed file system is different from that in local file system. In distributed environment, one file system operation usually consists of a series of sub-operations, and each sub-operation further contains two network operations, that is, request and reply. Network operation is constructed based on reliable protocol such as TCP/IP. However, a distributed file system further needs extra mechanism to guarantee the availability of the pair of request and reply in network transmission and its interaction with the file system sub-operation. Some systems embed the network fault-tolerance into upper file system [8], which nevertheless increases the complexity of system design; this approach

also has high cost since the high-availability method built in upper hierarchy inevitably increases the recovery time for bottom network transmission.

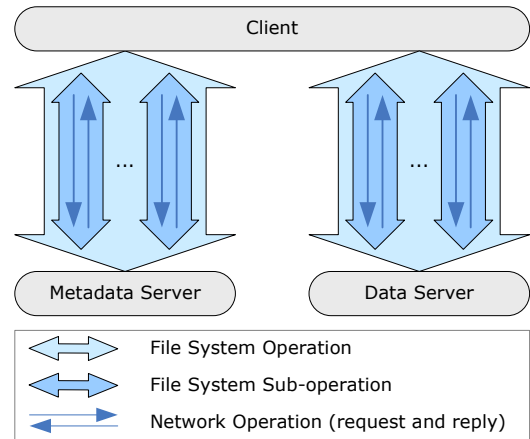


Figure 1. Operation hierarchy in cluster file system

We propose HMF, the High-available Message-passing Framework, to support the fault-tolerance of network transmission in cluster file system. HMF separates the network fault-tolerance design from the file system and keeps a simple interface between them. HMF has been adopted in DCFS3 to improve its availability. When there is only one network channel, the network fault could be recovered after failure occurs. With multiple network channels, additional channels could be used to do failover, and the multiple channels could also be adopted to balance the network load. Performance results show that when there are two network channels, HMF improves aggregate I/O bandwidth by 80% in normal condition while the performance degradation due to recovery is below 10%.

The rest of this paper is organized as follows: In section II, we briefly introduce our cluster file system and its message-passing subsystem. Section III discusses the related work. The design of HMF is introduced in section IV. Experiment and evaluation are given in Section V. Finally we conclude our work in section VI.

II. BACKGROUND

DCFS3 (Dawning Cluster File System v3) is a distributed file system designed for Dawning 5000A supercomputer. Its architecture consists of three parts: Metadata Server (MDS), Object-based Storage Device (OSD), and Client.

All components within DCFS3 are connected by Ethernet or InfiniBand. The network transmission among file system is supported by AMP subsystem (Asynchronous Message Passing). This work is carried out based on this subsystem. AMP contains three hierarchical structures. (1) Message Passing Interface Layer provides API for upper file system to do synchronous and asynchronous network transmission. (2) Message Handler Layer contains a series of queue structures and multiple service threads, all of which are used for message and data transmission. (3) Protocol Packaging Layer encapsulates the bottom physical communication protocols and provides a uniform interface for upper layers.

III. RELATED WORK

The availability of network is an important issue. The fault-tolerant mechanisms for network fall into four levels.

Network protocol defines the management principle for network transmission. It can solve several network errors, like fragmentation, disordering, and packet loss. Network protocol like TCP/IP [7] provides fundamental fault-tolerant function. RI2N/UDP [2] adopts multi-link to improve the availability of Ethernet. It uses participating Ethernet-links to multiply the bandwidth of each communication channel when there is no failure, and keeps transferring the packets through the remaining links when failure occurs on switches or Ethernet-links. It transfers data into small packets on the source node and the packets received by target node should be combined and rebuilt; this may cause extra cost and complexity. Such methods only ensure the availability of single network operation. A highly available distributed file system further requires extra fault-tolerant mechanism.

Some file systems design the fault-tolerant mechanism on message passing level, like NFS/RPC [10, 11]. NFS v3 [10] is stateless and depends on RPC for network fault-tolerance. Timeouts and re-trying in client is adopted to solve several network faults, such as server failure and server reply loss. NFS v4 [11] is stateful, which makes the fault-tolerance more important and complex. It involves two aspects: RPC failure and lock mechanism. Such methods can solve some network faults like disconnection, but is unable to deal with the problems of network partition and performance decline.

Some systems integrate the network fault-tolerance into file system, like PVFS2/BMI [4]. When network faults occur, BMI does not do resending automatically. Network message is one part of multi-level operation in distributed file system; automatic resending on network API may cause inconsistent requests. BMI decides the appropriate re-trying point based on the library of server and client, and it is able to cancel the operation which has been submitted but not yet completed. Such solution would increase the coupling,

which may make it complicated for file system modification and scalability.

Some systems build complex network protocol, such as Lustre/LND [3, 5, 6]. Lustre uses redundant routers to support network fault-tolerance and load balance. Servers are connected with high-speed network connection while clients adopt slow-speed network connection. Servers and clients are connected by a group of redundant routers. Clients can choose different routers for transmission when there are network faults. This method depends on high redundancy of network topology model, which can solve the network fault caused by hardware, especially for network partition. Its disadvantage is the high cost and complexity of hardware and construction.

IV. DESIGN

HMF is embedded into AMP and it includes a two-level architecture, that is, the HMF is implemented in both the Message Handler Layer and the Protocol Packaging Layer. HMF can be used in distributed file system with either multiple network channels or single network channel. Such construction can also be easily transplanted to multiple bottom network protocols.

A. Detection and Selection Mechanism

HMF adopted several mechanisms to detect the failure and recovery in network: (1) by adopting heartbeat [1], HMF sends periodic ICMP packets; (2) the state of connection would be passed to the upper file system through the bottom protocol interface; (3) the timeouts mechanism in the Protocol Packaging Layer could prevent from getting into never-ending iteration, and the timeouts mechanism in Message Passing Interface Layer could ensure that the request proposed by client would finally be released.

To utilize the resources provided by multiple network channels, there are two kinds of selection policies: (1) selecting other connection channel when network failure occurs, and (2) balancing the data flow when network works normally. The balance algorithm first locates a channel randomly, and then count the weight of every channel forward from the one located. The channel with the least weight would finally be chosen. The weight of a channel is defined as $(\sum load)/capability$, where the $\sum load$ denotes the amount of data transmitted in current channel and the $capability$ represents its transmission capacity.

B. reconfiguration

The reconfiguration of connection and operation is based on two kinds of context models, the Connection Context and the Operation Context.

TABLE I. SYMBOLS FOR CONTEXT MODEL

Symbol	Description	Value & Representation	
M	equip multiple network	1 = yes	0 = no
C	whether is in client	1 = yes	0 = no
H	network connection is normal	1 = yes	0 = no
T	operation timeouts	1 = yes	0 = no

We define the symbols in TABLE I which would be used to represent the model. We define the Connection Context State R and the Operation Context State S as follows.

$$R_{x+1} = f(R_x, S_x, C, H, T)$$

$$S_{x+1} = g(S_x, R_x, M, C, H, T)$$

TABLE II. THE DESCRIPTION OF CONNECTION CONTEXT

Connection Context Name	Connection Context Value	Description
INIT	NULL	The initialization of the connection channel.
OK	$R = 0$	The network transmission service is now available.
BAD	$R = 1$	The connection has already been released.
RECOVER	$R = 2$	The connection will be or is being re-constructed.
CLOSE	$R = 3$	The connection is cut and will be finally released.

TABLE III. THE DESCRIPTION OF OPERATION CONTEXT

Operation Context Name	Operation Context Value	Description
INIT	NULL	The initialization state. The operation is not carried out.
SELECT	$S = 0$	The operation will select the effective connection channel.
SEND	$S = 1$	The operation will do sending through the connection channel.
WAIT	$S = 2$	The operation is waiting for the reply.
RECV	$S = 3$	The operation has successfully received reply.
RESEND	$S = 4$	The context will be kept and the operation is waiting to be activated.
RELEASE	$S = 5$	The operation has been finished.

There are five states in Connection Context as mentioned in TABLE II. R is initialized to INIT when file system is booting; and then changes to OK. The conversion of context state is through some corresponding policy. We introduce the symbol “ \neg ” which means “not”. For example, when $x \neq 0$, $\neg x = 0$, $\neg \neg x = 1$, and vice versa. We define the formula of connection context conversion policy as follows:

$$f(R, S, C, H, T) = \begin{cases} \neg H \times (\neg C + (R + 1)) & R = 0 \\ \neg H \times T + \neg T \times (R + 1) & R = 1 \\ R + 1 & R = 2 \end{cases}$$

(Note: There is no conversion when $R=3$)

When network fault is detected, the state of Connection Context will be changed and will be inserted into the repair queue. If re-connection succeeds, the state of Connection Context would be changed and the corresponding Operation Context would be notified. Otherwise, the repair would be processed after a certain time. If the re-connection can not succeed after several times, the Connection Context will be released and errors will be informed to the upper layers.

There are seven states in Operation Context as mentioned in TABLE III. S is initialized to INIT when file operation

begins to be processed, and then changes to SELECT. The formula of operation context conversion policy is as follows:

$$g(S, R, M, C, H, T) = \begin{cases} \neg R \times (S + 1) + \\ \neg M \times \neg \neg R \times (C + (S + 4)) + \\ M \times \neg \neg R \times \neg R_i \times (S + 1) & S = 0 \\ H \times T \times (S + 1) & S = 1 \\ T \times (S + 1) & S = 2 \\ H \times (S + 1) & S = 3 \\ \neg \neg R \times S & S = 4 \end{cases}$$

(Note: There is no conversion when $S=5$)

When network fault is detected, the execution of the Operation Context would be halted and put into the repair hash list. The Operation Contexts of one object belong to the same hash entry. When the corresponding Connection Context is repaired, these Operation Contexts would be notified and put into execution queue. If the Operation Contexts are not notified after a certain time, they will be released and errors will be sent to upper layers.

V. EVALUATION

This section evaluates the function and performance of HMF. The configuration of DCFS3 consists of one MDS, one OSD, and one Client. All of these are interconnected through Gigabit Ethernet. Each node is a Dawning server running SUSE10.0 (Linux-2.6.20) with two AMD Opteron Processors, 2-Gbyte memory and one 73-Gbyte SCSI disk (Seagate ST3146707LC). The size of test file is 1 GB.

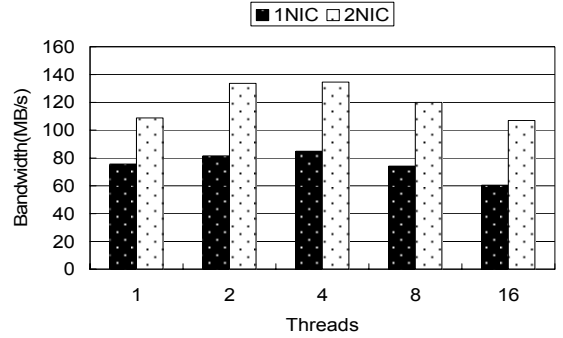


Figure 2. Write performance for different channel configurations

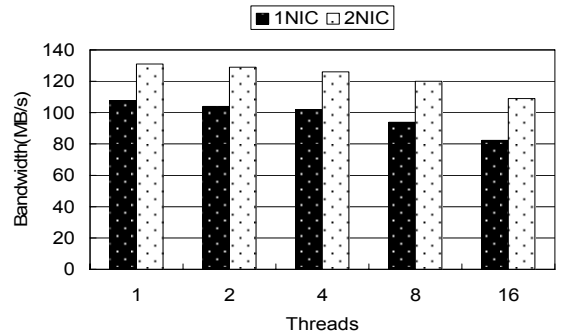


Figure 3. Read performance for different channel configurations

Figure 2 and Figure 3 show the I/O performance under different channel configurations. Compared with single channel, double channels can improve the I/O performance. Such improvement for writing is better than that for reading. This is because the sending of read requests in client is asynchronous, and the server has only one service thread to process these requests. When the number of service threads increases to 8, the read performance with double channels becomes nearly twice as that with single channel.

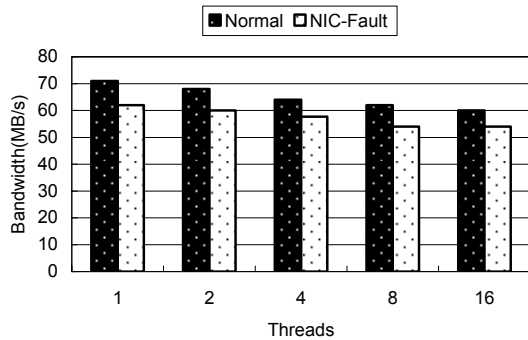


Figure 4. Write performance under multiple channels with HMF

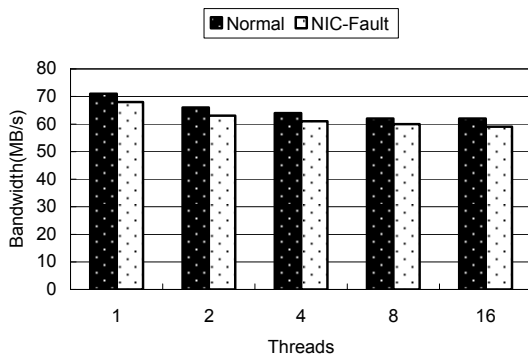


Figure 5. Read performance under multiple channels with HMF

We also use the I/O throughput to measure the recovery and performance loss for reading and writing when network failure occurs. The simulation of failure is by taking out the network wire; after a certain time we plug back the network wire to trigger the recovery process. Figure 4 and Figure 5 present the I/O performance when network failure occurs. HMF is equipped with multiple network channels. The system can recover after the failure and the performance degradation is less than 10%.

VI. CONCLUSION

This paper describes HMF, a high-available message-passing framework for cluster file system. HMF guarantees the availability of each pair of network transmissions and their interaction with the file system sub-operation. It

separates the network fault-tolerance design from the file system. Evaluation results show that HMF can increase the availability of transmission and reduce the cost of failover for network failures. The next step is to integrate HMF with the high-availability mechanism of the upper file system to efficiently improve the availability of the whole file system.

ACKNOWLEDGMENT

This work is supported by the National High-Tech Research and Development Program of China under grant numbered 2006AA01A102.

REFERENCES

- [1] M. K. Aguilera, W. Chen, and S. Toueg. "Heartbeat: A timeout-free failure detector for quiescent reliable communication", In *Workshop on Distributed Algorithms*, pages 126--140, 1997.
- [2] Okamoto, T., Miura, S., Boku, T., Sato, M., and Takahashi, D., "R12N/UDP: High bandwidth and fault-tolerant network for a PC-cluster based on multi-link Ethernet", *International Parallel and Distributed Processing Symposium (IPDPS)*, 2007.
- [3] Eric Barton, "Lustre Networking over OpenFabrics", www.clusterfs.com, www.lustre.org.
- [4] Carns, P., Ligon, W., III, Ross, R., and Wyckoff, P., "BMI: a network abstraction layer for parallel I/O", *the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2005.
- [5] P. LeMahieu, V. Bohossian, and J. Bruck, "Fault-Tolerant Switched Local Area Networks", in *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pp.747-751, 1998.
- [6] Peter J. Braam, "Lustre Networking", *A White Paper from Cluster File System, Inc.*, July 2007.
- [7] W. Richard Stevens, "TCP/IP Illustrated Volume 1: The Protocols", *Pearson Education, Inc.*, 1994.
- [8] M. Treaster, "A survey of fault-tolerance and fault-recovery techniques in parallel systems", *Technical Report cs.DC/0501002, ACM Computing Research Repository (CoRR)*, January 2005.
- [9] Patterson, D. A., A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, and N. Treuhaft, "Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies", *UC Berkeley Computer Science Technical Report UCB//CSD-02-1175*, March 15, 2002.
- [10] B Callaghan, B Pawlowski, and P Staubach, "NFS version 3 protocol specification", *RFC 1813, Network Working Group*, 1995.
- [11] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, D. Noveck, "NFS Version 4 Protocol Specification", *RFC 3010, Network Working Group*, December 2000.
- [12] W.D. Norcott, "Iozone File System Benchmark," 2005, http://www.iozone.org/docs/IOzone_msword_98.pdf