A High Performance and Low Cost Cluster-Based E-mail System

Woo-Chul Jeun[1], Yang-Suk Kee[1], Jin-Soo Kim[2], and Soonhoi Ha[1]

[1] School of Electrical Engineering and Computer Science,
Seoul National University,
San 56-1, Sinlim-dong, Gwanak-gu, Seoul 151-744, Korea
{wcjeun, yskee, sha}@iris.snu.ac.kr
[2] Division of Computer Science,
KAIST,
Daejeon 305-701, Korea
jinsoo@cs.kaist.ac.kr

Abstract. A large-scale e-mail service provider requests a highly scalable and available e-mail system to accommodate the increasing volume of e-mail traffic as well as the increasing number of e-mail users. To reduce the system development and maintenance cost, it is requested to make the system modular using off-the-shelf components. In this paper, we propose a cluster-based e-mail system architecture to achieve the goals of high scalability and availability, and low development and maintenance cost. We adopt the internal structure of a typical Internet e-mail system for a single server, called the MTA-MDA structure, to the proposed system architecture for the low cost requirements. We have implemented four different system configurations with the MTA-MDA structure and compare their performances. Experimental results show that the proposed system architecture achieves all the design objectives.

1 Introduction

The growth of the Internet has led to an explosion in volumes of e-mail traffic and in number of the users of e-mail service. At the same time, large-scale e-mail service providers have appeared. They have hundreds of millions of subscribers and process billions of messages: for example, in May 2001, Hotmail had over 100 million users and Yahoo! Mail, in March 1999, served 45 million users with 3.6 billion mail messages [1][2].

E-mail systems can be evaluated using various criteria [2] among which we are concerned about the following four in this paper: scalability, availability, flexibility, and extensibility. A system is highly scalable if the message throughput of the system increases linearly with the cluster size. As the cluster size increases, the probability of node failure also increases, so that making the system highly available is crucial to the service provider. An available system isolates a local failure from the system operation to avoid global outage. Considering these performance requirements, a cluster-based system architecture appears more suit-

V. Malyshkin (Ed.): PaCT 2003, LNCS 2763, pp. 482–496, 2003.

able for the large-scale e-mail systems than a single server system. Thus, our proposed e-mail system architecture is a cluster system architecture.

Flexibility and extensibility are related with the system development and maintenance cost. A system is called flexible if it has a modular structure that consists of replaceable components with only a little modification if any. An extensible system allows one to improve the system performance easily by upgrading some components. Considering these requirements, we adopt a structure that we call as "*MTA-MDA structure*" for short. MTA (Message Transfer Agent) and MDA (Message Delivery Agent) are server agents in a typical single-server e-mail system [3]: MTA receives an e-mail via standardized SMTP (Simple Mail Transfer Protocol) [4] and MDA stores it in a repository to be retrieved later by user's request. Even though the MTA-MDA structure is not a standardized structure, it has benefits of extensibility and flexibility: an e-mail system can be easily constructed by using off-the-shelf components for the MTA and the MDA.

Cluster-based e-mail systems can be classified into two approaches by their internal structure. One approach is to let each cluster node preserve the MTA-MDA structure except the modification to store an incoming mail to a remote node. Christenson et al. developed a cluster e-mail system using NFS (Network File System) for remote delivery and showed good scalability, flexibility, and extensibility [5]. However, it fails to meet the availability requirement.

The other approach is to make its own structure supporting standardized protocols for e-mail service: POP (Post Office Protocol) [6] and IMAP (Internet Message Access Protocol) [7] for e-mail retrieval and SMTP for e-mail exchange [2][8]. In this approach, they could successfully design scalable and available e-mail systems with a proprietary internal structure. But, it has serious drawbacks to avoid: lack of flexibility and extensibility. Without using existent off-the-shelf components, it takes long time and much effort to develop the system.

In this paper, we present a novel cluster-based e-mail system architecture using the MTA-MDA structure. Its modular structure provides many opportunities to improve the performance easily by using new off-the-shelf components. Moreover, this system architecture meets both scalability and availability requirements.

In section 2, we review the MTA-MDA structure of a typical e-mail system and overview some cluster-based e-mail systems classified by their internal structure. In section 3, the proposed cluster-based e-mail system architecture is explained. Section 4 presents the implementation of four different system configurations based on the MTA-MDA structure. Section 5 shows the experimental results and section 6 concludes the paper.

2 Backgrounds and Related Work

Fig. 1 shows the structure of a typical e-mail system for a single server, focusing on the receiving process of e-mail messages. The server consists of three agent programs: MTA, MDA, and MRA (Mail Retrieval Agent) [3]. MTA is a server program that transfers e-mails between machines on the Internet via the SMTP

protocol. Three well-known examples of MTAs are '*sendmail*', '*qmail*' and '*postfix*'. When MTA receives an e-mail message, MTA invokes an appropriate MDA to store the e-mail in the repository. If the message is destined for a user that has an account on the local system, MTA calls a local MDA that writes the message to the recipient's mailbox. Otherwise, MTA calls another MDA to reroute the message to the destination MTA. The message can be filtered by mail filtering programs on the way to the mailbox. Some examples of local MDAs in UNIX systems are '*procmail*', '*/bin/mail*', and '*mail.local*'. While the mailbox format is not standardized, the most commonly used format in a single server system is '*mbox*'.

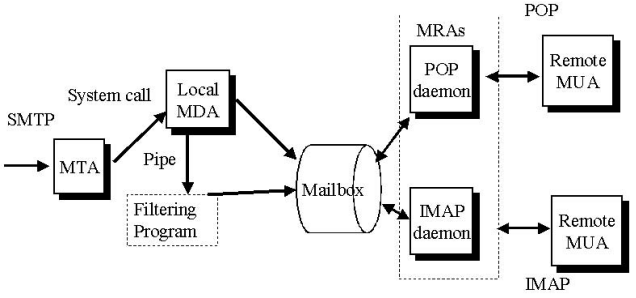

Fig. 1. The structure of a typical Internet mail system for a single server

Although accessing the mailbox directly can retrieve stored e-mail messages, MRA allows one to read the messages across the Internet. Upon request, an MRA accesses the user's mailbox. Two Internet protocols have been proposed for MRAs: they are the older and simpler POP and the newer and more complex IMAP. MUA (Mail User Agent) is a client program used by a user to send or receive e-mails. The Outlook Express of Microsoft, Inc. is an example.

This modular MTA-MDA structure allows a typical e-mail system to be constructed as a collection of loosely connected components that are developed independently. Therefore, some cluster-based e-mail systems have been developed to adopt the MTA-MDA structure on each node for reducing the development cost and preserving the benefits of flexibility and extensibility. On the other hand, others use a different proprietary architecture. Now, we briefly overview some existent cluster-based systems on each category.

2.1 E-mail Systems with the MTA-MDA Structure

Christenson et al. proposed a scalable e-mail system using the MTA-MDA structure in EarthLink Network, Inc [5]. Fig. 2 shows the message delivery and retrieval process in the case that the recipient's mailbox exists on a remote node. When a message arrives, the MTA forks a local MDA. Then, the local MDA

queries an authentication SQL DB about the recipient information. If the recipient's mailbox exists on the local node, the MDA stores the message into the mailbox. If it exists on a remote node, the local MDA transfers the message to the remote node by NFS mechanism. EarthLink system uses the '*sendmail*' as MTA with slight modification to solve a file-locking problem. They modified the '*mail.local*' to obtain the user information from SQL DB instead of the '*passwd*' file. Compared with the basic MTA-MDA structure of Fig. 1, the NFS module plays the role of an interface module between a local MDA and the remote mailbox.

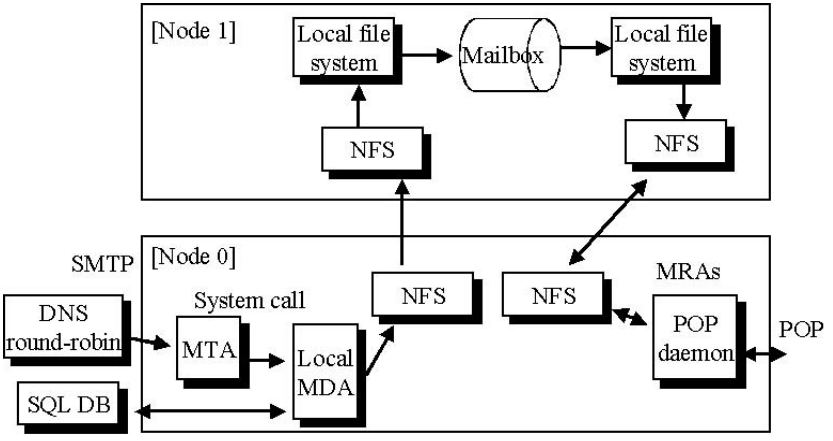

Fig. 2. The architecture of the EarthLink system

The scalability of this system depends on the performance of the NFS and the SQL DB. As processing power of each node and the network performance both increase, the system shows good scalability, flexibility, and extensibility. However, it has a serious drawback: if any node fails, the whole system stops the operation, soon. If an MDA process sends an NFS request to the failed node, the process sleeps until receiving the reply. As the number of sleeping processes increases, the load average of the node also increases. Then, high load average makes the '*sendmail*' MTA refuse all SMTP requests including the messages headed for other available nodes. This means that the system stops e-mail service until the failed node is recovered.

M. Grubb presented a scalable e-mail system that is deployed in Duke University [9]. The system provides aliased mail addresses whose real addresses are mapped to the e-mail servers geographically distributed in the campus. Thus the e-mail system plays the role of distributing the incoming e-mails to the distributed single servers after translating the aliases addresses to the real addresses. The system has the MTA-MDA structure. It is considered as a distributed e-mail system rather than a cluster-based system though it may be classified as a loosely coupled cluster system.

2.2 E-mail Systems with a Proprietary Structure

Several cluster-based e-mail systems with a proprietary structure have been developed among which Porcupine [8] and NinjaMail [2] are two representative systems. Y. Saito et al. developed the Porcupine system as a scalable and highly available e-mail system. The system partitions the user information and the user mailboxes across the nodes and replicates them to achieve high availability. Since they do not preserve the MTA-MDA structure, no off-the-shelf components could be used and significant effort has been paid to build the system. Their idea of caching the user information on the main memory for high performance is adopted in our proposed system.

The UC Berkeley's NinjaMail is built on top of UC Berkeley's Ninja software infrastructure [10], which supports scalable and highly available Internet services, and OceanStore [11] wide-area data storage architecture. Thus, flexibility and extensibility is limited within the Ninja infrastructure. We do not know performance number for the system.

3 Proposed E-Mail System Architecture

In this section, we explain two versions of the proposed cluster system architecture. The differences between the proposed system and the existent systems are summarized in Table 1. Similarly to the EarthLink system, the proposed system architecture augments an interface module between an MDA and the remote mailbox in the basic MTA-MDA structure. Fig. 3 shows the first version of the proposed e-mail system architecture. Message delivery process is similar to the EarthLink system until a local MDA is forked by the MTA. We modify the local MDA, '*mail.local*', to only forward the incoming message to the interface module via a UNIX domain socket. Now, the message delivery role is delegated to the interface module that stores the input message into the local file system or transfers it to the interface module of the remote node. If the remote node fails, the interface module detects the failure immediately at the connection establishment of TCP socket for remote delivery. Then, the interface module signals an error to the caller MDA and eventually to the sender. Unlike the NFS module, the interface module can service other e-mail deliveries to make the system available.

The version 1 system, however, has a performance overhead since there is a redundant message transfer from a local MDA to the interface module for remote delivery. We make a local MDA transfer the e-mail messages directly to the interface module of the remote node without intervention of the local interface module in version 2 system as shown in Fig. 4.

In the proposed system, we use the off-the-shelf components for the MTA and the MDA: '*sendmail*' or '*postfix*' for the MTA and '*mail.local*' for the MDA in the current implementations. Therefore our main effort to build the system is confined to the design of the interface module.

The proposed cluster system is implemented as a web-mail system. Therefore, we define the following basic roles that the interface module should serve:

Table 1. E-mail system comparison. (* : Availability feature is not supported in the current implementation.)

	Scalability	Availability	Flexibility	Extensibility
EarthLink	O	X	O	O
Porcupine	O	O	X	X
NinjaMail	O	O	X	X
Version 1	O	O*	O	O
Version 2	O	O	O	O

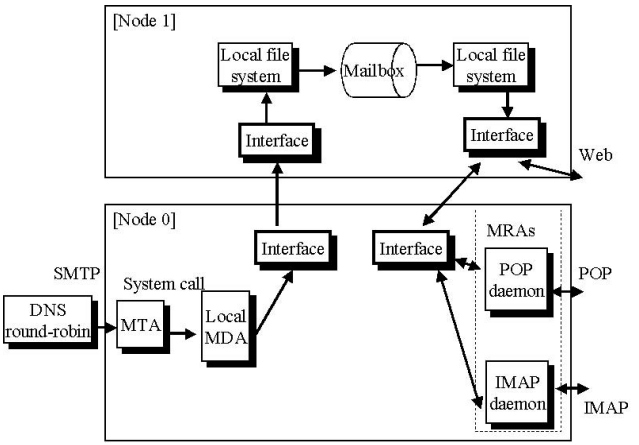

Fig. 3. The architecture of the proposed e-mail cluster system (version 1)

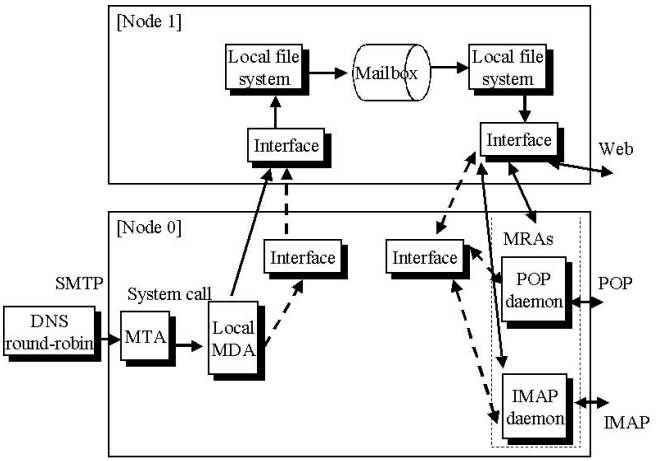

Fig. 4. The architecture of the proposed e-mail cluster system (version 2). For comparison, we also display the message delivery path of the version 1 system (*dotted lines*)

- E-mail message delivery to the local mailboxes
- User authentication
- Web-mail service for compact message summary
- Web-mail service for user information handling
- Web-mail service for user log-on request

Therefore, we define five different subsystems as displayed at the bottom of Fig. 5 and Fig. 6. A subsystem is a collection of functions. An '*auth subsystem*' manages user authentication information using an SQL DB. We cache the user authentication information to the memory as a hash table, borrowing the idea of the Porcupine's work for faster user authentication [8]. A '*logon info subsystem*' keeps a status of user log-on information for checking an illegal logon trial. A '*user info subsystem*' manages additional information of users such as name, address, phone number, and so on for the web-mail service. A '*mailbox subsystem*' manipulates the mailboxes of users in the directory structure as described earlier. A '*message info subsystem*' keeps additional information of messages such as sender, recipient, date, size, and so on. We expect that these modular subsystems make us easy to replace and improve them.

3.1 Interface Module, Version I

Fig. 5 shows the structure of the interface module, version 1. Four kinds of threads compose the interface module, among which the '*operation threads*' are the central threads that process the e-mail messages and the web-service requests using the subsystems. A '*node thread*' is a thread that receives a message from the local MDA or from the other nodes. The '*node thread*' encapsulates the message in an '*op-entry*' structure and puts it into the central '*work-list*' queue. A '*data thread*' is a thread that receives a request from the web, a POP daemon, or an IMAP daemon and puts the request in an '*op-entry*' structure into the '*work-list*' queue. An '*auth thread*' is a thread that receives and processes a user authentication request, from the web, a POP daemon, or an IMAP daemon. This thread puts the request in the '*auth list queue*' and processes it by calling functions in the '*auth subsystem*'. After the request completes, this thread replies to the requesting MRA. An available '*operation thread*' fetches an '*op-entry*' out of the '*work-list queue*' and examines whether the recipient is a valid user and where the recipient's mailbox is located by calling functions in the '*auth subsystem*'. If the recipient's mailbox exists on this node, the '*operation thread*' stores the message into the recipient's mailbox through function calls in the '*mailbox subsystem*'. At the same time, it stores the compact summary of the e-mail message in the '*message info subsystem*'. If the mailbox exists on a remote node, it sends the op-entry structure to the remote node where a '*node thread*' receives and puts into the '*work-list queue*'. After completing the message saving, the correspondent '*operation thread*' of the remote node acknowledges to the requesting node.

The number of available '*operation thread*'s is determined a priori considering the trade-offs between the parallel processing benefits and thread scheduling

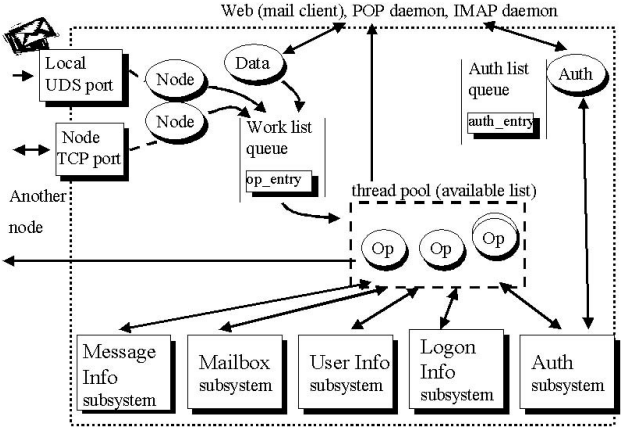

Fig. 5. The interface module of the proposed e-mail cluster, version 1. Interaction between four kinds of threads and five subsystems is depicted.

overheads. An '*operation thread*' is assigned to each outstanding e-mail message or each web-service request. If there are multiple messages destined for the same recipient, there can be more than one outstanding '*operation thread*'s that try to access the same mailbox. To minimize the mailbox synchronization overhead, we allow only one '*operation thread*' to be active for each user by locking mechanism. Extensive experiments reveal that separation of the '*node thread*' and the '*operation thread*' incurs non-negligible message copy overhead. In addition, too many '*operation thread*'s may degrade the performance due to the thread scheduling overhead. The second version of the interface module overcomes these drawbacks.

3.2 Interface Module, Version 2

Fig. 6 shows the structure of our improved interface module. To remove an additional message copy overhead, the local MDA sends the message to the remote node directly without intervention of the interface module of the local node. To make such decision, however, the MDA should inquire the interface module where the recipient's mailbox is located. Therefore, the interface module of the proposed e-mail cluster version 2 has a new thread named '*location query thread*' as shown in Fig. 6. With the information on the recipient's ID, the '*location query thread*' obtains the destination node using the '*auth subsystem*'. If the recipient's mailbox exists on the local node, MDA transfer the message to a '*node thread*' in the interface module. The '*node-thread*' of the second version implementation takes the roles of both a '*node-thread*' and an '*operation-thread*' in the first version. We create as many '*node-thread*'s as the number of nodes in the cluster. In fact, a '*node-thread*' is dedicated to each node including the local node. The '*node thread*' dedicated to the local node receives a message by a UNIX domain socket while a '*node thread*' assigned to a remote node, receives a message by a TCP socket. Such coalescing of two threads implies

that the messages or requests for a certain node are served in sequence in the second version of implementation. Therefore, we do not need the central '*work-list queue*' in the second version. As a result, the second version greatly reduces the implementation complexity while improving the performance.

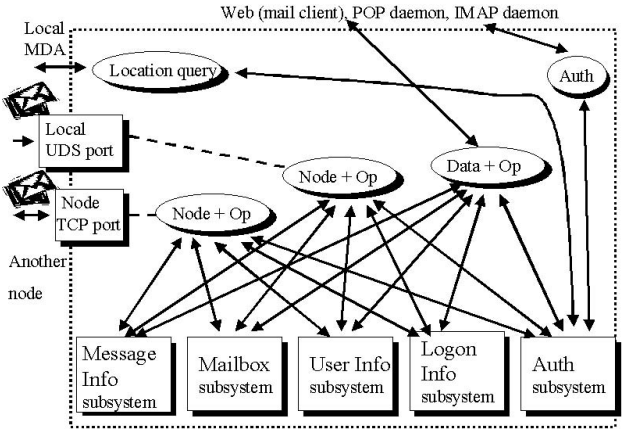

Fig. 6. The interface module of the proposed e-mail cluster, version 2. Interaction between four kinds of threads and five subsystems is depicted.

4 Implementation

The modular structure of the proposed system architecture allows one to change the system configuration easily by replacing a component with another. We have implemented three different configurations of the proposed system architecture. And, we have also implemented a simple e-mail cluster system based on the NFS mechanism, which is similar to the EarthLink system, for comparison purpose. We choose the EarthLink system for performance comparison because it is the only known e-mail cluster to us with an MTA-MDA structure. In this section, we explain some implementation details of those systems that are used for experiments in the next section. Four system configurations including the EarthLink configuration are summarized in Table 2.

We could easily implement an EarthLink system using off-the-shelf components such as '*sendmail*' [12], '*mail.local*', and the NFS. We used the NFS version 3 with the options of hard mount, asynchronous I/O, and 8KB read/write size. Notwithstanding our best efforts to replicate the described system in the paper, we admit that there may be discrepancy between the implemented one and the original one [5]. We use the '*mail.local*' as the local MDA, and make slight modification. We replace the code calling '*getpwnam()*', '*getpwuid()*' functions with a code querying the authentication SQL DB. We use MySQL 3.23.41 as the authentication SQL DB. And, we increase the "*max connection*" value of MySQL

Table 2. Experiment configurations

Configuration	MTA	Interface Module
EarthLink	Sendmail	NFS
Version1 (Sendmail)	Sendmail	Version 1
Version2 (Sendmail)	Sendmail	Version 2
Version2 (Postfix)	Postfix	Version 2

daemon from 50 as default value to 500 to avoid "*too many connections*" error [13].

For the proposed system architecture, we have implemented three different configurations as listed in Table 2. For the systems using the '*sendmail*', we had to adjust some configuration parameters in the '*sendmail.cf*' file. First, we remove the "*w*" flag in the entry for the local delivery agent, which prevents the '*sendmail 8.11.1*' from using the '*passwd*' file for user authentication [5]. For as large as hundreds of thousands of users, linear search of the '*passwd*' file takes prohibitively long. And we increase the values of '*QueueLA*' and '*RefuseLA*' from 8 and 12 as default values to 46 and 50 in the '*sendmail.cf*' file. This makes the '*sendmail*' receive SMTP requests at high load average value until saturated [14]. The default values make the '*sendmail*' reject new incoming mails too early before saturated.

For the last configuration with the '*postfix*', we set the '*mailbox_command*' parameter to '*/usr/bin/mail.local "$USER"*' in the '*main.cf*' file. This makes '*postfix 1.1.11*' call '*mail.local*' to deliver received message. Next, we replace the code calling '*mypwnam()*' function with a code passing recipient's ID to the '*mail.local*'.

For load balancing of the cluster nodes, the DNS round-robin mechanism is used to determine which cluster node receives a new SMTP request. And user mailboxes are distributed randomly and uniformly across the nodes. In each node, user mailboxes are grouped and stored in directories whose locations are defined as "*/home/node#/{a serial number of a user ID mod 300}/user ID*": for example, if a user ID is "*s0003123*" and his mailbox exists on node-0, the mailbox location becomes "*/home/00/123/s0003123/mbox*". Such grouping reduces the time to find and access a user mailbox from a given user ID [5].

5 Experiments

We compare the four system configurations of Table 2 in terms of the message throughput, the message latency, cluster scalability, and availability. Even though SPECmail2001 is recently released as an industrial standard benchmark [15], it is not suitable to measure the raw performance of the system such as the peak message throughput of the system. Therefore, we created our own experimental method.

The experimental environment consists of a test cluster, workload generators, a DNS server, and a SQL DB server as shown in Fig. 7. Our test cluster has four nodes of Red Hat 7.3 Linux 2.4.18-3 kernel with the following hardware attributes: 550MHz Pentium III CPU, UDMA66 40GB IDE disk, 512MB main memory, two 100Mb Ethernet interface cards, and ext3 file system. We need at least as many as the workload generators as the number of cluster nodes to generate e-mail messages enough to saturate the cluster. We use four Linux machines of a similar kind but with 600MHz Pentium III CPU as the workload generators. The DNS server is a Red Hat 6.2 Linux 2.2.14-5.0 server machine with 166MHz Pentium MMX CPU, 2GB IDE disk, and 128MB main memory. The SQL DB server is a Red Hat 7.2 Linux 2.4.7-10 machine with 1GHz Pentium III CPU, 40GB IDE disk, and 256MB main memory. The test cluster nodes and a SQL DB server are interconnected via a switched 100Mb Ethernet network. The cluster, workload generators, and the DNS server are connected by another switched 100Mb Ethernet network. The version of BIND is 8.2.2-P5. Each node has the mailboxes of 50,000 users. We use a constant message size of 8KB for a workload to evaluate the system. This size is known to be the mean or the median value on the workload characterization of mail servers [16].

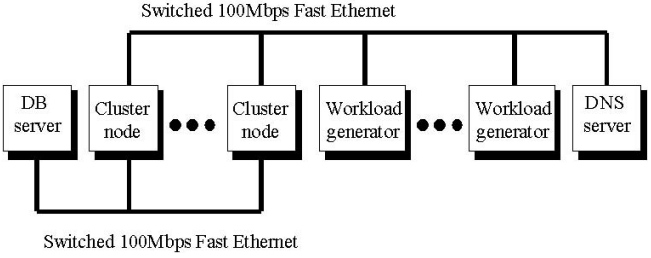

Fig. 7. Experimental environment that includes the e-mail cluster system and workload generators

5.1 Message Throughput

We define the message throughput of an e-mail cluster system as the number of messages that the system can process maximally in a second. The workload generators generate a certain number of e-mail messages in 60 seconds at a constant rate. And the total elapsed time for the system to process all messages is measured. Then, the number of messages divided by the measured time becomes the average number of messages, which a system processes for a second. For example, if we send 1200 messages to a system in 60 seconds and the system takes 180 seconds to process all the messages, the average number of messages processed for a second becomes 1200/180=6.7 (messages/second). Increasing the number of generated messages by 60 messages (one message per second), we

repeat these experiments until finding the maximum average number of messages processed for a second. Then, the value is regarded as the message throughput of the system. We used the mean value of 3 sets of experiments for the same number of messages in 60 seconds.

Table 3 presents the message throughput of four e-mail system configurations we have implemented. All configurations, except for '*Version2 (postfix)*', scale well up to 4-node cases. For a single node case, both version 1 system and version 2 system have a slightly higher throughput than EarthLink system. For 2 and 4 nodes, '*Version2 (Sendmail)*' has the highest performance among all four system configurations even though the difference is not significant.

Table 3. Message throughput of e-mail system configurations (messages/second)

Configuration	1-node	2-nodes	4-nodes
EarthLink	7.8	16.9	30.9
Version1 (Sendmail)	9.0	17.7	33.6
Version2 (Sendmail)	9.0	19.4	35.0
Version2 (Postfix)	10.8	21.4	30.1

Also, we have set up another experimental environment to examine the scalability with a larger cluster system before implementing the second version of the proposed system with '*sendmail*'. The cluster consists of 16 nodes of Red Hat 7.2 Linux 2.4.7-10 kernel with 1.7GHz Pentium4 CPU, UDMA66, 40GB IDE disk, 256MB main memory and 2 100Mb Ethernet interface cards. We compared the scalability of the proposed system version 1 and the EarthLink system and show the result in Table 4. Both systems are fairly scalable up to 16 nodes while the version 1 system degrades its performance a little bit more. In short, the proposed system and the EarthLink system both possess good scalability.

Table 4. Message throughput of the EarthLink system and the proposed sytem, version 1 (messages/second)

Configuration	1-node	2-nodes	4-nodes	8-nodes	16-nodes
EarthLink	9.8	19.9	37.6	80.4	178.1
Version1 (Sendmail)	11.5	19.2	42.5	79.5	175.0

5.2 Message Latency

We define the message latency of a system as a time interval from receiving a SMTP request to storing the message on the recipient's mailbox. We compute

the average value out of 100 experimental results excluding both upper 10% results and lower 10% results to compensate the run-time variances of the system behavior.

Fig. 8 and Fig. 9 show the message latencies that are divided into sections for a single node and for two nodes clusters respectively. The '*MTA*' section (*MTA*) means a time interval for MTA to process a message. The '*setup*' section (*setup*) means a time interval for MDA to prepare store the received message. The '*store*' section (*store*) means a time interval for MDA to store the received message into temporary file. The '*transfer*' section (*transfer*) means a time interval for the interface to complete the message delivery process.

The message latency of each system is nearly constant except for the '*transfer*' section, independently of the cluster size. The version 1 system has longer latency than the version 2 system because it adds additional message copy and thread switch overhead for message storage as explained in the previous section. For remote delivery case, the performance degradation of the version 1 system becomes significant. On the other hand, the version 2 system with '*postfix*' has the shortest latency among all four systems.

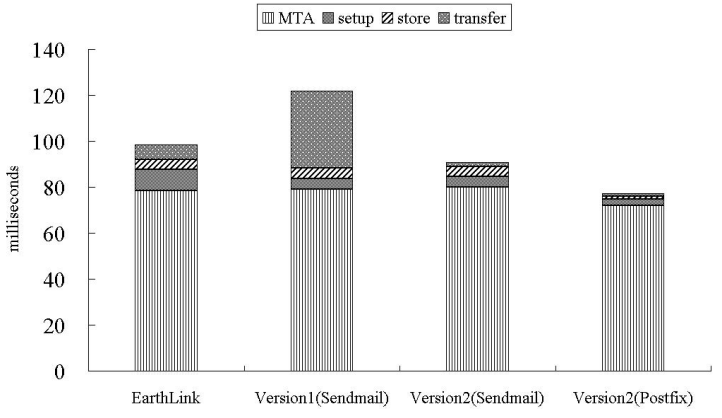

Fig. 8. The message latencies of single-node e-mail cluster systems.

5.3 Availability

To test the availability, we disconnect a cable from a node or turn the power off a node suddenly at run time. Then, we examine whether the system operates properly, until all messages are received. The EarthLink system has not been able to accept any SMTP connection in a few seconds after a fault occurs. However, '*version 2*' system survives by making a local failure cause a local, not global outage.

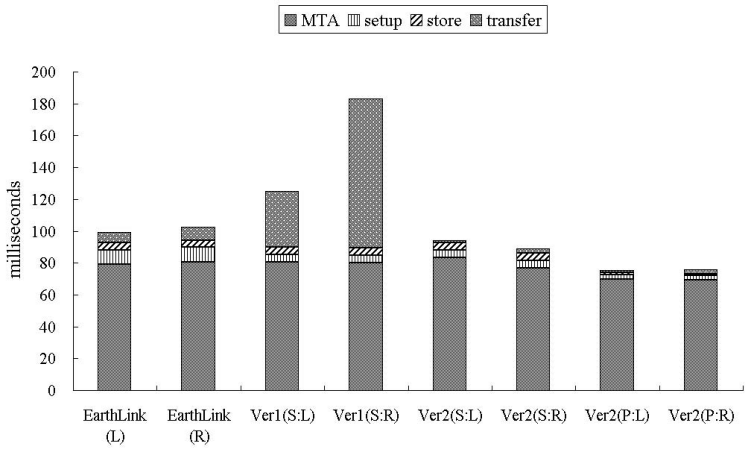

Fig. 9. The message latency of two-node e-mail cluster systems. Local delivery (L) means that recipient's mailbox exists on local node. Remote delivery (R) means that recipient's mailbox exists on other node (L: local delivery, R: remote delivery, S: sendmail, P: postfix, *Ver1*: version1, *Ver2*: version2)

6 Conclusion

We have presented a novel architecture for cluster-based e-mail systems with the MTA-MDA structure to achieve the goals of high scalability and availability, and the low development and maintenance cost. To demonstrate the flexibility and the extensibility of the proposed architecture, we implemented four systems with the MTA-MDA structure. Experimental results show that all systems are scalable in the sense the peak message throughput. Preliminary experiments show that one of our implementations (*version 2*) makes a local failure cause a local, not global outage.

Although we use our own experimental method to size a system, standardized benchmark is neccessary to compare mail systems formally. Then, we plan to evaluate our system using the new SPECmail benchmark. Because the benchmark uses POP and IMAP, we extend the functionalities of 'version 2' system for POP and IMAP services.

Acknowledgement. This work was supported by National Research Laboratory Program (number M1-0104-00-0015) and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

1. Microsoft.: MSN Hotmail Tops 100 Million User Milestone, REDMOND, Washington, 2001.

2. J.Robert von Behren, Steven Czerwinski, Anthony D. Joseph, Eric A. Brewer, and Jonh Kubiatowicz.: NinjaMail: the Design of a High-Performance Clustered, Distributed E-mail System, In Proceeding of International Workshops on Parallel Processing 2000, Toronto, Canada, August 21–24, 2000, pp. 151–158.
3. David Wood.: Programming Internet Email, CA:O'Reilly & Associates, Inc., Sebastopol, 1999.
4. Postel, Jonathan.: RFC 821: Simple Mail Transfer Protocol, 1982.
5. Nick Christenson, Tim Bosserman, David Beckemeyer, EarthLink Network, Inc.: A Highly Scalable Electronic Mail Service Using Open Systems, Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, Dec 1997.
6. J. Myers and M. Rose.: RFC 1939: Post Office Protocol – Version 3, May 1996.
7. M. Crispin.: RFC 2060: Internet Message Access Protocol – Version 4revl, Dec 1996.
8. Yasushi Saito, Brian N. Bershad, and Henry M. Levy.: Manageability, availability and performance in Porcupine: a highly scalable, cluster-based mail service, 17th ACM symposium on Operating System Review, Dec 1999, 34(5) pp. 1–15.
9. Michael Grubb.: How to Get There From Here: Scaling the Enterprise – Wide Mail Infrastructure, In the proceedings of the Tenth USENIX Systems Administration Conference (LISA '96), Chicago, IL, 1996, pp.131–138.
10. U.C. Berkeley.: Ninja project, `http://ninja.cs.berkeley.edu`
11. U.C. Berkeley.: OceanStore project, `http://oceanstore.cs.berkeley.edu`
12. Eric Allman.: SENDMAIL – An Internetwork Mail Router, BSD UNIX documentation Set , University of California, Berkeley, CA, 1986.
13. Randy Jay Yarger, George Reese, Tim King.: MySQL & mSQL, CA: O'Reilly & Associates, Inc., Sebastopol, 1999.
14. Bryan Costales with Eric Allman.: sendmail ; Second Edition, CA:O'Reilly & Associates, Inc., Sebastopol, 1997.
15. SPECmail2001.: `http://www.spec.org/osg/mail2001`
16. Laura Bertolotti – Maria Carla Calzarossa.: WORKLOAD CHARACTERIZATION OF MAIL SERVERS , In the proceedings of SPECT'2000, Vancouver, Canada, July 16–20, 2000.