High-Throughput Mail Gateways for Mobile E-mail Services based on In-Memory KVS

Masafumi Kinoshita, Gen Tsuchida
Yokohama Research Laboratory
Hitachi, Ltd.
Yokohama-shi, Japan
{masafumi.kinoshita.rt, gen.tsuchida.qc}@hitachi.com

Takafumi Koike
Information & Telecommunication System Company
Hitachi, Ltd.
Kawasaki-shi, Japan
takafumi.koike.kc@hitachi.com

***Abstract*—Mobile network operators providing e-mail services require mail systems to process large volumes of e-mail traffic to and from mobile terminals. Mail gateways, particularly those that accept e-mail messages from external systems and transfer them with store-and-forward communication, require much higher throughput that conventional mail gateways cannot provide. Mail gateways are also required to preserve consistent data and to provide queued services in order. We propose a mail gateway system for mobile e-mail services based on a distributed in-memory key-value-store (KVS) to meet four requirements of high-throughput, high-speed responses, scalability, and availability. We propose KVS to achieve these requirements, which can store messages physically in a queue structure and preserve the consistency of data in respective queues. We present a method of high-throughput access to pipeline messages on an active TCP connection that is linked to a queue on mail gateways and its backup queue in KVS. The mail gateways have a management function for each backup queue in KVS to both preserve consistency and avoid problems in the system. We evaluated the performance of the KVS we propose and a mail gateway corresponding to the KVS. The results proved both the KVS and mail gateway achieved the high throughput that was required.**

Keywords-MTA (mail transfer agent); KVS (key-value-store); in-memory data grid.

I. INTRODUCTION

The growth in the number of users of mobile e-mail services has led to an explosion in the volumes of e-mail traffic encountered by mobile network operators. For example, one mobile network operator has more than 10 million active users and their e-mail system processes more than 10,000 e-mail messages per second.

A large-scale e-mail system is composed of mailbox servers storing the e-mails of their users and mail gateway servers. Mail gateway servers function as mail transfer agents (MTAs), process incoming e-mail messages from external systems, and transfer them to their destinations, such as MTAs on the Internet and mailbox servers. Architectures with mail gateways proposed in [1] are flexible and extendable, and these gateways can stabilize e-mail services by controlling traffic [2]. Mail gateways in mobile e-mail systems also serve to provide billing processes and e-mail security, transcode e-mails, and provide other processes.

Mail gateways generally relay e-mail messages with store-and-forward communication that incoming e-mails are stored in a local queue located within non-volatile storage, which are then forwarded to the destination server. The three main advantages of store-and-forward communication are quick response, control of traffic to avoid burst traffic, and guaranteed e-mail delivery. The main disadvantage of this communication is low throughput because non-volatile storage, such as that in disk and a storage system is accessed, which is a bottleneck in relaying e-mail messages.

Mail gateways in mobile e-mail systems require high throughput, high-speed responses, scalability, and availability. High-throughput and high-speed responses are particularly important for three reasons of: 1) preventing mobile terminals from failing to send e-mails, 2) minimizing connections between mail gateways and mobile terminals to avoid congestion in both e-mail systems and wireless networks, and 3) reducing the number of mail gateway servers. We set the following target values from our experience and expertise as a systems integrator of mobile e-mail service, which has more than 10 million subscribers; high-throughput means a mail gateway should process more than 1,000 e-mails per second, and high-speed response means a mail gateway should respond to a received message within 100 milliseconds.

Well-known MTA software, such as sendmail [3] and postfix [4], fail to meet the high-throughput requirement because their throughput is less than 100 e-mail messages per second [5]. We proposed a method of improving throughput where mail gateways used the method to reduce access disk I/O requests and parallelized these requests while attaching them to storage area network (SAN) storage systems [5]. A mail gateway could process 850 e-mail messages per second and have a response of 80 milliseconds by adopting this method, but this approach failed to meet the requirement for scalability because it was necessary to scale-out too many configurations and too many operations for the storage and servers of mail gateways. Moreover, its performance has recently been insufficient for the requirement for throughput.

Scalability and availability for e-mail systems have also been proposed. Christenson et al. proposed an e-mail system using a network file system (NFS) [7], and Saito et al. [8] and Behren et al. [9] proposed an e-mail system using a distributed storage system with a hash table. Koromilas et al. [11] proposed an e-mail system using Cassandra [10], which is a distributed key-value-store (KVS) software. However,

 ISBN: 978-1-61208-203-5

these proposed systems were mainly designed for the functions of mailboxes, and few considerations were given to mail gateways. These systems, therefore, failed to meet the requirements of high throughput and high-speed responses.

There have been many efforts in the last few years on distributed in-memory KVS or in-memory store technology for high throughput and high-speed responses. A distributed in-memory KVS is composed of multiple nodes, and stores replicated data in the memory of multiple nodes without accessing non-volatile storage. It has been adopted in banking systems [12], stock exchange systems [14], telecommunications [15], and other fields. The systems using it meet the four requirements of high throughput, high-speed responses, scalability, and availability.

Here, we propose a distributed in-memory KVS designed for mail gateways and mail gateways based on it. Four main questions need to be answered in adopting in-memory KVS.

- What is an appropriate architecture for mail gateways for mobile e-mail services to meet all four requirements of high throughput, high-speed responses, scalability, and availability?
- How do they achieve both consistent data and preservation of order for in-order queues for several KVSs? Mail gateways also require these properties for mobile e-mail services.
- How do they efficiently store e-mail messages in KVS to achieve the requirement for high throughput? In other words, the method of storing messages requires high write throughput.
- How do they avoid system problems and recover quickly, without having impact on mobile terminals? In addition, the process for mobile terminals should be executed within several seconds.

The rest of the paper is organized as follows. The background and related work are introduced in Section II. Section III presents the system architecture and design. Section IV describes the implementation and the results obtained from evaluating performance. Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

Fig. 1 outlines an example of the system structure for a mobile e-mail service. One function of mail gateways (GWs) is that they can accept e-mail messages from external systems, such as mobile terminals and MTAs on the Internet, via messaging protocols, such as simple mail transfer protocol (SMTP) and multimedia messaging services (MMS). They can also relay e-mail messages to MTAs or internal mailboxes. Their traffic to MTAs is generally larger than the traffic to mailboxes. Mobile terminals can also access mail gateways to retrieve e-mail messages from the mailboxes, via the Internet message access protocol (IMAP), post office protocol (POP), and MMS.

Mail gateways adopt store-and-forward communication to relay messages via SMTP and MMS. They can also adopt this communication to transfer other message data, such as billing data and notification messages related to e-mails, while processing messages via IMAP and other protocols. Mail gateways with store-and-forward communication store e-mail messages on their disks while waiting for relays. Therefore, they can respond to acceptance of e-mail messages promptly after storing them. This efficiently reduces receipt errors for e-mail messages from mobile terminals, and reduces their impact in both e-mail systems and wireless networks.

Relaying messages may be instantaneous, but this may also be delayed if the destination MTA is unavailable or cannot be reached due to network error. Mail gateways will keep re-trying to make deliveries for a certain period, such as several hours or a few days.

Mail gateways control congestion in MTAs with a destination queue of e-mail messages. Mail gateways also manage queues that have billing data and other messages. Mail gateways are required to manage many queues and have consistent messages in these queues.

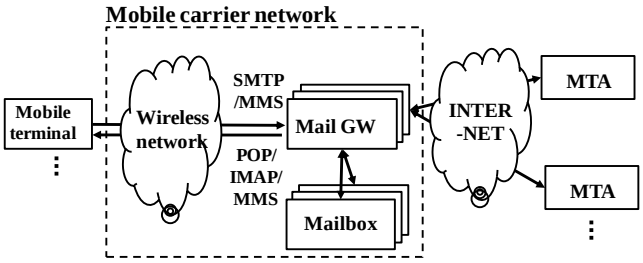

Figure 1. Example of system structure for mobile e-mail service.

We briefly overview related work in what follows. First, we explain studies on conventional mail systems and then we describe some efforts on in-memory KVS.

A. *Mail system architecture*

Jeun et al. proposed an architecture for a cluster-based e-mail system with an MTA-MDA structure [1] on conventional mail gateways, which equals the mail gateways-mailbox structure discussed in this paper. This architecture was highly scalable, highly available, inexpensive to develop, and had low maintenance costs. Since this system used sendmail [3] and postfix [4] with a network file system (NFS), it did not perform satisfactorily to satisfy the requirement for throughput.

The scalability and availability of e-mail systems have been discussed [8], [9], [11], where scalability has been achieved by using distributed storage systems. The performance of one e-mail system with distributed KVS, i.e., Cassandra [11], is superior to some others, but it is still inadequate to meet the requirement for throughput. Using distributed KVS instead of a conventional storage system provides easy scalability and availability. KVS distributes its data to multiple servers with a consistent hash method [17], [18], which is similar to the methods proposed by Saito et al. [8] and von Behren et al. [9].

Facebook [19] uses distribute KVS, i.e., HBase [20], for its messaging service [21]. They selected it for its scalability, consistency, performance, and other reasons. However, they did not evaluate it.

KVS is a major approach to achieving requirements such as scalability, high performance, and availability.

 ISBN: 978-1-61208-203-5

B. *In-memory store efforts*

Many efforts have been expended on distributed in-memory KVS or in-memory technology for high throughput and high-speed responses. In–memory KVS memcached [16], which is known as high throughput KVS, is used as a cache by many companies, such as Facebook and Twitter [22]. As memcached is a single server and is not replicated in multiple servers, it cannot be a persistent data store. However, distributed in-memory KVS, such as the IBM WebSphere eXtreme Scale [12], can store replicated data in the memory of multiple nodes for persistent storage. They have higher performance than disk-based KVS, such as Cassandra and HBase. Nevertheless, they have two disadvantages compared with disk-based KVS. The first is data are lost if all nodes having replicated data are down at the same time. However, as power supplies are duplicated at the data center and data are backed up to disks periodically, there is little probability that data will be lost. The second disadvantage is that their storage capacity is smaller because memory is more expensive than disks.

As previously mentioned, mail gateways are required to manage queues and keep data consistent. There are a few KVSs storing data in a queue structure, but WebSphere provides a queue service [13]. Its function of queuing was developed as a thin layer on top of a typical KVS structure, which can store one simple set of key values. It stores a meta-data managing queue in KVS, and does not physically store data in a queue structure. However, there are no solutions to resolving the issues with mail gateways described in Section I.

III. PROPOSED ARCHITECTURE AND DESIGN

We first present the mail gateway architecture based on distributed in-memory KVS and the KVS architecture for it in this section. We then propose methods of resolving the issues described in Section I.

A. *Architecture for Mail Gateway System*

The architecture for the mail gateway system is outlined in Fig. 2. There are mail gateway (GW) modules and in-memory KVS modules in each node. These modules are independent of each other, and communicate with other modules in other nodes. The load balancer (L4 switch) dispatches incoming message to mail gateway modules and monitors a TCP port of the mail gateway modules to switch over the path to a non-responding one.

Mail gateway modules receive incoming messages and process them in their memory without accessing their disks. Mail gateway modules back them up (store) in two KVS modules on request. In addition, mail gateway modules also store them in their own queue. Thus, three replicated messages are in memory in respective nodes to avoid them from being lost. Backup in the in-memory only enables high-speed responses.

Mail gateway modules provide consistency and tolerance against partitioning in the consistency, availability and partition-tolerance (CAP) theorem [6]. In addition, availability is important for e-mail services. In this architecture, the availability of a whole system composed of many mail gateway modules is provided by the load balancer switching over a path to mail gateway modules. In addition, there is no single point of failure in the system.

Fig. 2 shows the queue in mail gateway modules are backed up in two KVS modules. KVS modules have a queue structure that physically store messages in queues in their memory. (Details on KVS are described below.) In other words, a queue in mail gateway modules synchronizes two backup queues in a KVS module. Mail gateway modules have many queues of messages to guarantee their order (first in, first out) and they manage the flow to avoid congestion. KVS modules also have the same queues of mail gateway modules for backup. These mail gateway systems attached to KVS have a scalable structure, because they make it easy to add nodes.

Mail gateway modules relate their own queue to backup queues in two KVS modules. They select KVS modules from a group of KVS modules with a set configuration, not using hash distribution on request such as that with Cassandra. Next, they send parameters such as queue length and access accounts, and create backup queues. After that, the mail gateway modules start the service to relay messages with store-and-forward communication. The mail gateway modules store a message in backup queues and its queue. Next, they forward the message to its destination and delete it from the backup queues and its own queue. They occasionally replace messages to back up intermediate states to process messages. These communications (i.e., storing, deleting, and replacing messages) achieve queue data are synchronized between mail gateway modules and KVS modules.

Mail gateway modules monitor backup queues in KVS modules with responses of synchronization and heart-beat health checks. If they detect faults in a backup queue, they isolate it and replicate another backup queue in another KVS. If a mail gateway module's service is down, it reboots, and obtains all data in the backup queue in KVS modules to continue e-mail services. If it cannot reboot, the KVS module moves messages from the backup queue to a program to continue sending them in the same node.

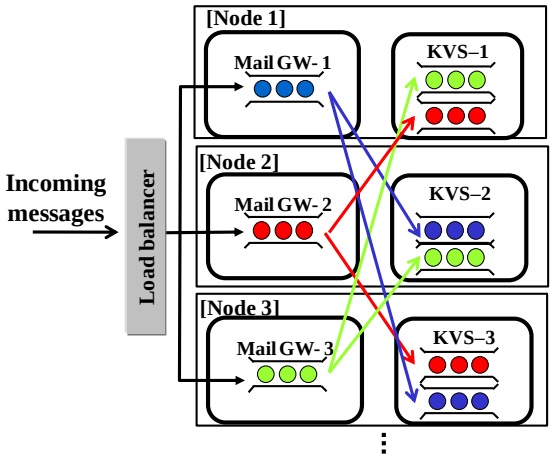

Figure 2. Architecture for mail gateway system.

B. Architecture for KVS with in-order queues structure

As previously mentioned, KVS modules have a queue structure in memory to back up queues in mail gateway modules. Mail gateway modules synchronize their own queue to two backup queues in respective KVS modules. KVS modules support that messages are stored in queues in-order and they retrieve them in-order.

Mail gateway modules have functions of distributing messages to KVS modules having backup queues. KVS modules allow them to access the queue. They can access a message in the structure with a queue's name key and message-id key. This is different from WebSphere [12] in that data in a queue are physically located in respective KVS modules.

This structure where data in queues are physically stored in respective KVS modules has the following advantages. The first advantage is that the number of communication messages is reduced while mail gateway modules and KVS modules are synchronized. That has an impact on the throughput of mail gateways systems. An order index (i.e., metadata for queues) has to be built to preserve the order for in-order queues. The metadata in general KVS, such as WebSphere, are stored as KVS data to preserve queues [12]. Therefore, they have to access a message and several metadata while synchronizing one message. The proposed KVS modules, on the other hand, physically have the metadata as a table in their own memory, and they only have to access one message while synchronizing themselves.

The second advantage is that queue data can be quickly recovered if there is trouble, such as network error or nodes are down. For example, if a mail gateway module reboots, it retrieves all data in queues from a KVS module synchronized with it. It can immediately obtain a block including many messages in queues. If a typical KVS were used instead of the proposed KVS modules, mail gateways could only obtain one message immediately and they would have to access metadata in the queue every time they obtained it. In addition, since these messages are distributed to KVS (respective nodes) with the hash table, mail gateways access many nodes with respect to each message and metadata. Thus, the queue structure can significantly reduce recovery time.

The third advantage is the availability of mail gateway systems. As previously mentioned, their availability is provided by the load balancer switching over a path to mail gateway modules. Mail gateway modules without synchronized KVS modules close their own TCP ports and suspend their own services. If a KVS module has some fault, the load balancer can definitely isolate mail gateway modules by closing their ports. If typical KVS were used instead of the proposed KVS modules, all mail gateway modules could access these KVS modules while faulty KVS modules would not be isolated, and the impact of these faults would spread to whole systems.

C. High-throughput method to access KVS

To preserve order in in-order queues and achieve the requirement for high throughput, mail gateway systems adopt the following method of communication. Fig. 3 outlines the flow for the method of communication between a mail gateway module and a KVS module. A mail gateway communicates with a KVS module with persistent transmission control protocol (TCP) connections. One queue of a mail gateway module and one queue of a KVS module are linked with two connections; these are via a respective path in different network segments. One connection is active for the communication message, and the other is on standby when network error occurs in the active path. Since one TCP connection is used to achieve communication in order, the order in the queue is preserved.

First, a mail gateway module connects to a KVS module with an authentication message, and it sends a request creating a queue whose parameters, e.g., queue length and access accounts, are requested by it. After a KVS module accepts the request, it creates a queue in its memory and this queue is related to the connection in its management table of queues. This sequence means the mail gateways have obtained ownership of the queue.

The mail gateway module sends pipelined-requests on an active connection to the KVS module. The KVS module immediately receives pipelined requests and processes them. Finally, the KVS module immediately sends pipelined responses on the connection. The KVS module obtains an internal queue lock while processing messages from the connection. The requests and the responses include a "transaction number", i.e., the number it has already processed, and a "sequence number", i.e., the number it requires for matching requests and responses. The number of transactions is checked to preserve the order in queues.

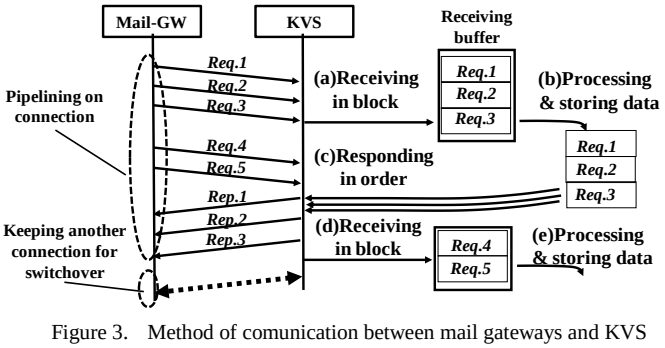

Figure 3. Method of comunication between mail gateways and KVS

Thus, this method of communication can provide high-throughput and consistent storage in queues. Typical KVS, such as memcached, Cassandra, and Dynamo are generally used in systems that have many low-performance clients that access KVS with many connections. However, the mail gateway modules in the proposed system are high performance and there are an equal number of mail gateway modules and KVS modules. There are two ways of efficiently communicating data via networks; the first is by communicating with many connections, and the second is by multi-communicating with several connections. The former in this mail gateway system can decrease performance because KVS modules control (obtain and release) internal locks by processing one message from each connection. The latter reduces the number of locks that are controlled.

 ISBN: 978-1-61208-203-5

However, the latter is susceptible to the control of TCP flows because there are large message traffic flows on the connections. There is the method of avoiding this impact, i.e., a mail gateway module can use a sufficient number of queues to adjust message traffic on the connection. Therefore, we selected the latter, which we evaluated and explain below.

The above issue is similar to those with the method of communication between parallel and serial communications. In other words, our proposed method of communication between mail gateway modules and KVS modules is similar to serial communication on TCP connections.

D. Mail Gateway's Management of KVS

Mail gateway systems require consistency for e-mail services. Mail gateway modules have a management table for each backup queue in KVS modules to preserve consistency (summarized in Table. I). They manage two states of one queue, i.e., a "network state" and a "synchronized state". The network state indicates the state of a network path. Network state (A) means the state of the active connection, and (S) means the state of the standby one. The network state is updated in these two cases, where mail gateways are connecting and network error is caused during synchronization or heart-beat health checks. Mail gateway modules switch over to standby connection in seconds when active connections are unavailable.

The synchronization state indicates whether the queue is synchronized or not. Mail gateway modules check the number of transactions to monitor the synchronization state, (which is the number KVS has already processed to preserve the order of queues), in responses from KVS. A mail gateway module uses this table in synchronization, heart-beat health checks, and other processes. If the synchronization state is no good (NG), a mail gateway deletes all data in this backup queue in KVS and re-synchronizes the data in the entire queue.

When mail gateway modules store a message in a queue, they send a message to the KVS modules indicated in this table. They succeed in storing the message after receiving all responses from KVS modules. If they cannot receive all responses before timeout, which is seconds, they retry to send the message on a standby connection and change the network state. Finally, when they fail to store it in KVS, they isolate this backup queue, and create a backup queue in another KVS by using any conditions.

TABLE I. MANAGEMENT TABLE FOR BACKUP QUEUES

Queue Name (dst.domain)	KVS	Network State (A)	Network State (S)	Sync. State	Number of Transactions
hitachi.com	NodeB	Run	Ready	Sync	18000
	NodeC	NG	NG	NG	-
xxxxxx.com	NodeD	NG	Run	Sync	3000
	NodeE	NG	Run	Sync	3000

IV. IMPLEMENTATION AND EVALUATION

We first present the implementation of the proposed system in this section and the methodology we used to evaluate it. We evaluated the throughput of one KVS module, the throughput of mail gateway modules corresponding to two KVS modules, and the process time for recovery.

A. Implementation and methodology for evalution

Mail gateway modules and KVS modules were implemented with event driven architecture [23] developed in the C language. We designed their performance, especially throughput that could be scaled up with increasing CPU frequency.

Mail gateway modules support SMTP, IMAP, and other protocols. Mail gateway modules can have 1,000 queues at maximum. Although the length of a queue can be configured for more than a million messages, it is limited by the memory size of each module. These queues are used for storing e-mail messages, billing their data, and storing and forwarding their notification messages and other messages. The queues for e-mail messages via SMTP services can also provide several functions; a control function for the e-mail traffic of each destination MTA or mailbox server, a control function for rates, which shows how many messages to send per second, a function for regulating the receipt and sending of e-mail messages, and a function for timeout to transfer them.

First, we evaluated the transaction throughput for a single KVS module. Second, we evaluated throughput via the SMTP of mail gateway modules corresponding to KVS modules. Next, we evaluated how quickly the system recovered from server incidents.

B. Throughput of proposed KVS

There were two nodes that had dual processor dual cores and 4 GB of RAM, and they were connected to two Gigabit Ethernet networks. One node includes a KVS module, and the other included a test program. The test program generated the workload to the KVS modules, received responses, and evaluated throughput, i.e., the number of transactions per second. One transaction was a set where a message was stored and deleted because the transaction of relaying an e-mail message with storage and forwarding includes a set.

We evaluated the KVS module for different-sized messages from 0.4 to 20 KB. The reason we evaluated these sizes for messages is that they represent the greatest volume of e-mail traffic in mobile e-mail services from our experience. The KVS module had one queue and the test program sent messages to the queue with one active connection.

We also evaluated memcached [16] to compare it with the evaluation of the proposed KVS module. Memcached has a very simple KVS that does not have lock control, a queue structure, or other functions; therefore, its throughput is known to be high.

Fig. 4 plots the transaction throughput for a single KVS for different sized messages. The throughput for the proposed KVS modules for 0.4 KB is 200,000 transactions per second, which is twice the throughput of memcached. The throughput for the proposed KVS for 1 KB is 100,000 transactions per second, which is 1.4 times the throughput of

 ISBN: 978-1-61208-203-5

memcached. The throughput for the proposed KVS for more than 2 KB is limited by the 1-Gbps network, just like it is for memcached. Thus, the experimental results proved the proposed KVS modules met the requirement for high throughput to store messages. In addition, it proved the method of communicating messages with pipelining on one active connection was efficient.

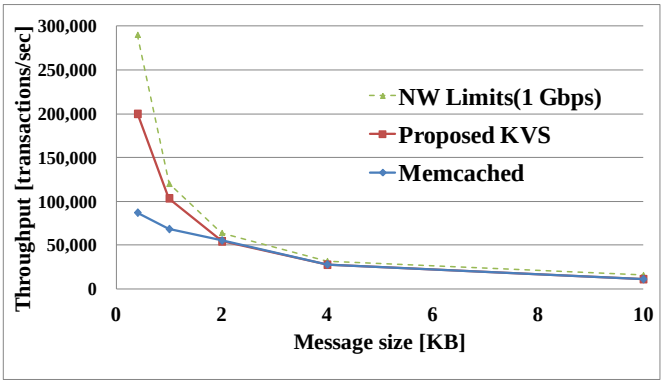

Figure 4. Transaction throughput of KVS for different sized messages

C. *Throughput of mail gateways attached to KVS*

There were two nodes that had six processor dual cores and 32 GB of RAM, and they were connected to two Gigabit Ethernet networks. One node included a mail gateway module, and the other included two KVS modules and two kinds of test programs. All programs are connected via Gigabit Ethernet network. The first was a test client program that generated the workload of e-mail messages to the mail gateway modules via SMTP, received responses, and evaluated throughput, i.e., the number of messages per second. The second was a test server program that received e-mail messages from the mail gateway module. The mail gateway module received an e-mail message from the test client program and stored it in backup queues in two KVS modules. After that, the mails gateways sent a response to the message to the test client program and concurrently transferred it to the test server program.

We evaluated a mail gateway with a workload composed of 70 percent 1KB-messages and 30 percent 10KB-messages, to simulate realistic message traffic for mobile e-mail services.

We compared our evaluations of the proposed method using KVS modules with a conventional method where mail gateways used a redundant array of independent disks (RAID) storage with a method of streamlining disk I/O requests [5]. Accessing disks to store messages is a bottleneck for throughput when the conventional method is adopted. A past experiment by Kinoshita et al. [5] adopted the conventional method to compare its performance with well-known MTA software; the throughput for sendmail [3] was less than 20 messages per second, and the throughput for postfix [4] was 80 messages per second.

Fig. 5 compares the throughput and average response times of the proposed and conventional methods. The throughput for the proposed mail gateway module is 3,600 e-mail messages per second, which is 4.4 times the throughput of the conventional method. The throughput is less than that of a single KVS, i.e., 11,000 transactions per second. This means accessing KVS modules no longer creates bottlenecks in throughput. The average response time for the proposed mail gateway module is 14 milliseconds, which is an fifth of the response with the conventional method. Thus, the experimental results prove that the proposed mail gateway module meets both the requirements of high throughput and high-speed response described in Section I.

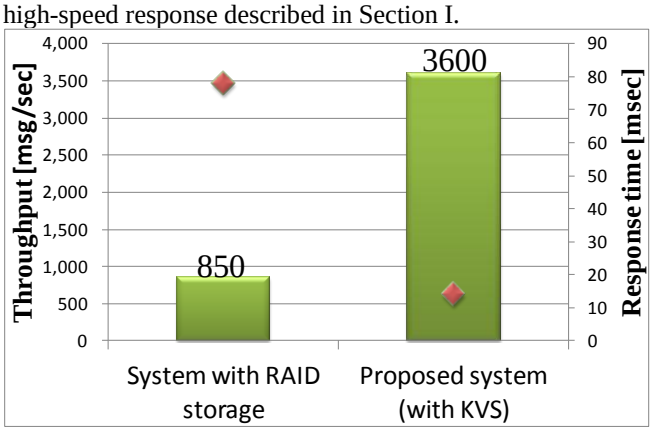

Figure 5. Throughput and average response time.

D. *Time for Recovery*

As previously mentioned, one advantage of the proposed system is its quick recovery of queue data. A mail gateway module or a KVS module can reduce the time for recovery to immediately obtain a block that includes many backup messages for queues. We evaluated the time for recovery for different numbers of messages in the block.

There were two nodes that had six processor dual cores and 32 GB of RAM, and they were connected to two Gigabit Ethernet networks. One node included a mail gateway module, the other included KVS modules and a stored e-mail program. First, the program stored messages in the mail gateway module. The mail gateway module stored a number of messages in the backup queues in two KVS modules and its own local queue. We then shut down the mail gateway module to simulate the server down, and rebooted it. After that, the mail gateway module obtained all its messages from a backup queue in a KVS module. This process meant recovery from trouble with mail gateway modules.

We evaluated the time it would take for this recovery process where the mail gateway module obtained all messages from a backup queue in a KVS module. The message size was 1 KB and the queue had 500,000 messages.

The experimental results are given in Fig. 6, which prove that the more messages there are in a block, the shorter the recovery time is. The recovery time for 1,000 messages was 5.9 seconds, which is a fourteenth of the recovery time for one message. The case of one message in a block was the same as the method of recovery for a typical KVS, such as memcached or WebSphere (with a thin layer of queue service implemented) [12]. In addition, if a typical KVS is used instead of the proposed KVS modules, there is more than double the access to the metadata of queues in KVS.

 ISBN: 978-1-61208-203-5

Therefore, the recovery time for using typical KVS is more than 160 seconds; which cannot be adopted for mobile e-mail services. Thus, the experimental results proved the proposed system met the requirement for rapid recovery described in Section I.

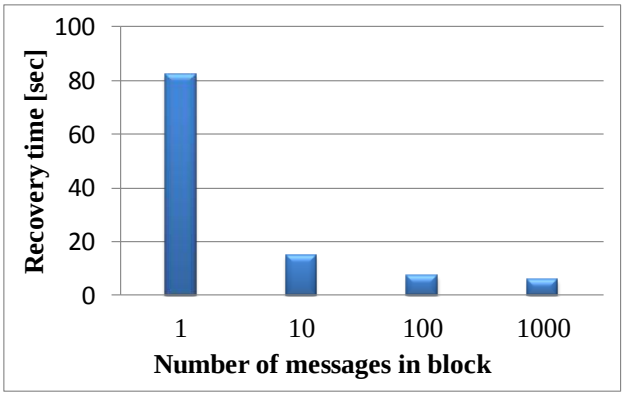

Figure 6. Recovery time for different numbers of messages in block

E. Potential for Lost Messages

This system stores messages only in the memory of multiple nodes. If all nodes having replicated data are down at the same time, this system loses messages. Therefore, it has to be located in a data center that has a stable power supply. The power supplies of all nodes are generally duplicated in the data centers of mobile network operators. In addition, mail gateway modules have to store messages for several hours, or days at maximum. Thus, there is a very low probability of messages being lost in this system.

V. CONCLUSION

We presented a mail gateway system based on distributed in-memory KVS to satisfy the four requirements of high throughput, high-speed responses, scalability, and availability for mobile e-mail services.

We proposed KVS, which can store messages physically in a queue structure and provide consistent data in respective queues. To preserve order in in-order queues and achieve the requirement for high throughput of backup to KVS, we proposed a method of pipelining messages on an active TCP connection that linked a queue in mail gateway modules and its backup queue in KVS modules. In addition, mail gateway modules switched over to standby connection in different network segments in seconds when active connection was unavailable. The mail gateway modules had a management table for each backup queue in KVS to avoid system problems. The system recovered almost immediately within seconds to obtain a block including many backup messages for queues.

We evaluated the performance of one KVS module, the performance of a mail gateway module that corresponded to two KVS modules, and the process time for recovery. The results proved both the KVS modules and the proposed system met the requirement for high throughput. The throughput for the proposed KVS module was 200,000 transactions per second with 0.4-KB messages, which is superior to memcached. The throughput for a proposed mail gateway module was 3,600 e-mail messages per second, which is 4.4 times the performance of the conventional method. Since the proposed KVS was adopted for mail gateways instead of conventional storage, the process of persistently storing messages was no longer a bottleneck to throughput.

REFERENCES

[1] W.-C. Jeun, Y.-S. Kee1, J.-S. Kim, and S. Ha, " High Performance and Low Cost Cluster-Based E-mail System", Parallel Computing Technologies, pp. 482–496, 2003.

[2] M. 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, pp. 131–138, 1996.

[3] sendmail: http://www.sendmail.com/sm/open_source/ April.6.2012

[4] postfix: http://www.postfix.org/ April.6.2012

[5] M. Kinoshita, M. Nakahara, and T. Sagara, "An Implementation and Evaluation of Multiprotocol Message Gateway", The 71th National Convention of IPSJ , March 2009.

[6] E. A. Brewer, "Towards robust distributed systems", In Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing, p. 7, 2000.

[7] N. Christenson, T. Bosserman, and D. Beckemeyer, EarthLink Network, Inc., "Highly Scalable Electronic Mail Service Using Open Systems", Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, December 1997.

[8] Y. Saito, B. N. Bershad, and H. M. Levy, "Manageability, availability and performance in Porcupine: A highly scalable, cluster-based mail service", 17th ACM Symposium on Operating System Review, 34 (5) pp. 1–15, 1999.

[9] J. R. von Behren, S. Czerwinski, A. D. Joseph, E. A. Brewer, and J. Kubiatowicz, "NinjaMail: the Design of a High-Performance Clustered, Distributed E-mail System", In Proceeding of International Workshops on Parallel. Processing 2000, pp. 151–158, 2000.

[10] apach cassandra: http://cassandra.apache.org/ April.6.2012

[11] L. Koromilas and K. Magoutis, "CassMail: A Scalable, Highly-Available, and Rapidly-Prototyped E-Mail Service", Lecture Notes in Computer Science, Volume 6723/2011, pp. 278–291, 2011.

[12] IBM WebSphere eXtreme Scale: http://www-01.ibm.com/software/webservers/appserv/extremescale/ April.6.2012

[13] Y. Wang, H. Chen, B. Wang, J. M. Xu, and H. Lei, "Scalable Queuing Service Based on an In-Memory Data Grid", IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), pp. 236–243, November 2010.

[14] Y. Hashidume, K. Takasaki, T. Yamazaki, and S. Yamamoto, "Ultra-high-speed In-memory Data Management Software Achieving High-speed Response and High Throughput", Fujitsu Scientific & Technical Journal, Vol. 62, pp. 57–64, January 2011.

[15] S. Kondoh, Y. Miyagi, M. Kaneko, T. Fukumoto, and K. Ueda, "A Study of Data Arrangement for Various Retrieval and Effective Redundancy", IECE Technical Report, NS2011-63, pp. 11–16, September 2011.

[16] memcached: http://memcached.org/ April.6.2012

 ISBN: 978-1-61208-203-5

[17] G. DeCandia et al. ,"Dynamo: Amazon's Highly Available Key-value Store", Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP'07), pp. 205–220, October 2007.

[18] A. Lakshman and P. Malik, "Cassandra -A Decentralized Structured Storage System", Cornell, 2009.

[19] Facebook: http://www.facebook.com/ April.6.2012

[20] HBase: http://hbase.apache.org/ April.6.2012

[21] D. Borthakur et al., "Apache hadoop goes realtime at Facebook", Proceedings of the 2011 International Conference on Management of Data, SIGMOD '11, pp. 1071–1080, 2011.

[22] Twitter: http://www.twitter.com/ April.6.2012

[23] M. Welsh, D. Culler, and E. Brewer, "SEDA: An architecture for well-conditioned, scalable Internet services.", In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01), ACM Press, pp. 230–243, October 2001.

 ISBN: 978-1-61208-203-5