第 32 卷 第 6 期 2003 年 12 月 | 电 子 科 技 大 学 学 报 Journal of UEST of China | Vol.32 No.6 Dec. 2003

An Efficient Disk Queue I/O Mechanism

Wei Qingsong Lu Xianliang Ren Liyong Zhou Xu
(School of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract The storage management overload and read-write performance of traditional disk queue are analyzed. To solve the problem that disk queue I/O is the primary performance bottleneck in messaging server, an efficient disk queue I/O mechanism called FlashQ is proposed. FlashQ utilizes pre-assigned continuous disk blocks to act as disk queue and organizes data in compact layout and adopts the Lazy Write and the Ahead Read polices to elevate the performance of read-write. Experiment shows that performance of the FlashQ is much better than that of traditional disk queue.

Key words messaging server; disk queue; lazy write; ahead read

一种高效的磁盘队列I/O机制[*]

魏青松[**] 卢显良 任立勇 周 旭
(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】分析了传统磁盘队列的存储管理开销和读写性能，针对磁盘队列I/O已成为影响消息服务器性能的首要瓶颈，提出了一种高效磁盘队列I/O机制—FlashQ。FlashQ采用物理上连续的磁盘块作为磁盘队列，以紧凑方式组织数据，采用延迟写策略和预先读策略提高读写性能。实验表明，FlashQ的性能比传统磁盘队列好。

关 键 词 消息服务器; 磁盘队列; 延迟写; 预先读
中图分类号 TP316.81 **文献标识码** A

The explosive growth of messaging traffic (Email and Short Messaging Service) has put tremendous pressures on messaging system to be super-fast and reliable. Normal Mail Transfer Agent(MTA) handles 40～50 messages/s per server because large number of small file writes causes disk I/O the primary bottleneck in messaging system.

Based on SMTP protocol, messages must be saved persistently to disk before acknowledging the sender. Most email systems store message in form of a message per separate file, which brings heavy overload of disk management and file operation. In order to reduce management overload, Qmail introduces Hash directory and files are written in the directory in a round robin manner. Hash directory reduces file seek times efficiently, but it has not changed the storage manner of one message per file at all. The heavy overload of disk management still exists. On the other hand, the storage manner of a message per file causes a large amount of disk fragments.

This paper presents an efficient disk queue I/O mechanism called FlashQ. The FlashQ utilizes pre-assigned continuous disk blocks to serve as disk queue to reduce file management overload, in which adjacent messages are stored in adjacent disk block. The FlashQ adopts Lazy Write(LW) policy to accumulate multiple messages in a

Received on March 21, 2003
2003年3月21日收稿

*The project is supported by Electronic Information Industry Fund, No.: [2000]1277
信息产业部电子工业生产发展基金资助项目，编号: [2000]1277

** 男 31岁 博士生 主要从事分布式系统、网络存储、操作系统、计算机网络方面的研究

large buffer and save them into the disk queue in one large write. And multiple adjacent messages are fetched into buffer in one large read by Ahead Read(AR) policy. The LW and AR policies take full advantage of the disk bandwidth. In addition, FlashQ can reduce disk fragments efficiently.

1 File System Characteristics

In disk system, random disk access time T_{access} is composed of three parts as

$$T_{\text{access}} = T_{\text{seek}} + T_{\text{rotate}} + T_{\text{transfer}} \tag{1}$$

where T_{seek} denotes disk seek time, T_{rotate} denotes rotational latency, T_{transfer} denotes data transfer time which is the only effective time among three parts and occupies little portion of T_{access}. T_{seek} is about half of T_{access} for small file.

The characteristic of the modern disk drivers is that the relationship between seek latency and seek distance is nonlinear, which results in that reading or writing several 4 Kb or 8 Kb disk blocks costs a relatively small amount more than reading or writing only one. However, although file systems have been very successful at exploiting the disk bandwidth for large file, they have failed to do so for small file activity [1].

Small file can not obtain high performance because it can not make full use of the disk bandwidth. On the other hand, with the run of the file system, the file system gets increasingly fragmented. A fragmented file system stores a large part of its file into non-adjacent blocks. And reading and writing data from and to non-adjacent blocks causes longer seek times and can severely reduce file system throughput.

2 Conventional Disk Queue I/O Overload

Most available email systems store each message in a separate file as Fig.1 shows. There are two traditional storage management approaches of the disk queue: files reside in a single directory hierarchy or in a multiple hash directory hierarchy. The first approach obtains low efficiency because traditional UNIX directory lookup takes time linear with the directory depth. The second way by which files are written in the Hash directory in a round robin manner reduces file seek times efficiently.

In the initial stage of FSmail (a Fast and Secure Mail Server developed by us and its software copyright register number is 2003SR0075) project, we exploit the second approach as disk queue which stores each message object per file. To get a better understanding of real file storage organizations, we construct a small utility to scan the traffic of FSmail server. Due to the characteristics of email traffic, 85% of referenced message objects are smaller than 8 Kb. Fig.2 illustrates this by showing the distribution of sizes of message objects. Storing each message in a file will cause a large amount of small files, which increases storage management overload. In addition, small file can not perform well in current file system.

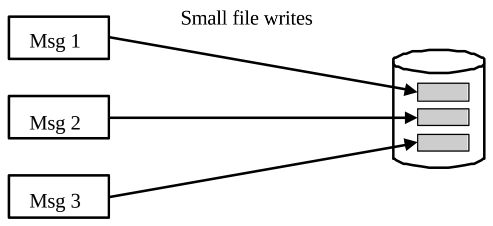

Fig.1 Traditional disk queue

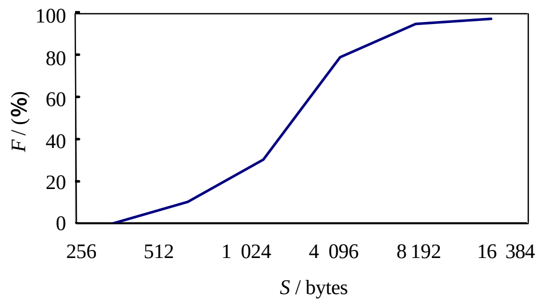

Fig.2 Distribution of message size

We test our FSmail server using the SPECMAIL 2000 under 50 000 users in the 100 Mb/s LAN. Tab.1 reports the result. We can see that the process time of the commands of SMTP DATA and DELIVER interacting with disk queue is much more than that of the other commands. The command of SMTP DATA gets a message from client, creates a new file queue and writes the message into the file, which involves network I/O, file creation and file writing. The command of DELIVER opens the file storing a message, reads and delivers the message, deletes the file after the message is delivered successfully, which involves file opening, file reading and file deletion.

Tab. 1　Messaging server command process time

SMTP Command	Helo	MAIL　FROM	RCPT　TO	DATA	DELIVER
Process Time/ms	8.40	5.37	4.47	25.12	19.80

To understand the disk queue I/O in detail, we test the overload of the disk queue based on our FSmail server which stores a message in a separate file. The result shown in Fig.3 tells us that the overload of disk I/O is much higher than that of network I/O and file creation and writing occupy a large portion of the overload of the disk queue I/O.

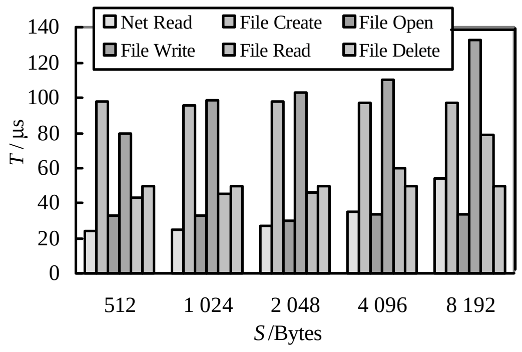

Fig.3　Conventional disk queue I/O overload

From the above experiment and analysis, the disk queue exploiting Hash directory still presents the following shortages. First, although the Hash directory reduces file seek times efficiently, it has not changed the storage manner of one message one file at all. The overload of disk management still exists. Second, with the run of messaging server, it causes a large amount of disk fragments which severely decrease the performance of small file reading and writing.

So optimizing storage management of the disk queue according to the I/O characteristics of the messaging system will reduce the management overload and improve the performance of message reading and writing, which will boost up the performance of the messaging system.

3　The Design of FlashQ

3.1　Basic Idea

Considering the I/O character of messaging server, we present an efficient disk queue I/O mechanism called FlashQ whose basic structure is as Fig.4 shows. The FlashQ optimizes the disk queue at the following three respects:

1) Pre-creating a large continuous disk space as disk queue to reduce file management overload of file creation and file deletion[2].

2) Introducing LW policy to reduce disk machine movement overload and make full use of disk bandwidth by gathering multiple messages in a big buffer and saving them to disk queue in one large write[3].

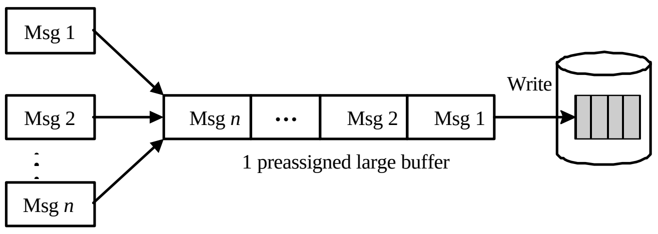

Fig.4　Flash Queue structure

3) Exploiting AR policy to reduce the times of reading and make full use of disk bandwidth by reading multiple continuous messages in one big read operation.

3.2　FlashQ Disk Layout

In traditional disk queue such as Qmail, message head and message context are saved in two separate files which are organized by current file system and always scatter in discontinuous disk blocks.

The FlashQ pre-occupies a large continuous disk space from a NAS or SAN as disk queue at initialization. The FlashQ is organized in compact storage management which stores the message head and message context in continuous disk blocks as a whole data object and saves adjacent message objects in continuous disk blocks [4].

Fig.5 shows the disk layouts that occur in the traditional disk queue and the FlashQ after storing three messages.

To keep the compact disk layout, the FlashQ copies the delivered failing message to the deferral queue and sets the delete flag on

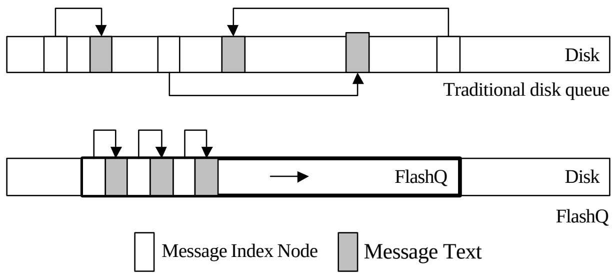

Fig.5 Disk layout of the FlashQ

3.3 Message Index Node

To improve the read performance, the FlashQ keeps a Message Index Node (MIN) table in memory, which indexes the messages waiting for being delivered. The MIN in memory including message ID, offset, the MIN size and message context size is a subset of the MIN in the disk, which provides quick location of the message by the map between message ID and message location.

According to the character of First Come First Serve, the MIN table is arranged in form of queue in memory. When a message comes, the FlashQ allocates a new MIN in memory, fills in all the fields of the MIN and appends the MIN into the MIN table. When a message is delivered successfully, the FlashQ deletes and frees the MIN.

In case that something happens to the server and it goes down, we should be able to restart and read MIN table from the disk queue on the NAS or SAN. Queue recovery should take very short time, typically less than 50 ms.

3.4 FlashQ Writing

The FlashQ adopts LW policy to reduce the times of synchronized write and improve the message writing performance [5].

The SMTP front-end doesn’t use any local storage for incoming messages. The SMTP front-end receives messages from client and performs some processing in a process pipeline such as anti-virus, anti-spamming and then sends it to the Queue Server. The SMTP front-end waits for the Queue Server to write this mail to the disk. The Queue Server accumulates multiple messages in a large buffer. Once the buffer is full or a certain time period has elapsed, the Queue server saves all the messages in the buffer into the disk queue in one large write and sends a response to the SMTP front-ends. The SMTP front-end acknowledges the sender of the message that the message has been stored and the client does not fell obvious latency. From this point, our MTA will take all the responsibility to route this message to its destination address.

Thus, the LW policy insures the impact disk layout of the FlashQ by putting multiple adjacent message objects in continuous disk blocks. It makes full use of disk bandwidth and advances the write performance by reducing the disk seek and rotation latency.

3.5 FlashQ Reading

Because the message transferring abides the principle of First Come First Serve, adjacent message objects can are placed in continuous disks block and it is possible to equip large cheap memory in special large scale messaging server, the FlashQ uses AR policy to boost up read performance [6].

When reading message, the AR policy reads not only the current message object from disk but also its next multiple messages objects stored in adjacent disk blocks in one large read. The buffer size and the MIN table in memory decide how many messages to be read one time.

As a result, the AR policy improves the read performance and makes full use of the disk bandwidth by lessening the disk machine movement cost which elevates the speed of message delivery.

3.6 FlashQ Deletion

As the message being delivered successfully, the traditional one message per file disk queue deletes the file storing that message, which not only greatly aggravates the overload of disk I/O but also produces a mass of disk fragments which accelerates the file system fragmented. And the fragmented file system reduces the file read performance to a great extent.

When a message has been delivered, the FlashQ deletes the MIN of the message in the memory and sets on the deletion flag of the message in disk by one byte writing, which greatly decreases the overload of disk I/O and disk fragments.

4 Experiment Results

We test the performance of our FSmail server with the FlashQ using the SPECMAIL 2000 under 50 000 users in the 100 Mb/s LAN. Tab.2 shows the comparison of performance between the traditional disk queue and the FlashQ. From the Tab.2, we can see that the process time of the commands of SMTP DATA and DELIVER with the FlashQ decreases greatly.

Tab.2 Messaging server performance under flashQ and traditional hash queue

SMTP Command	Helo	MAIL FROM	RCPT TO	DATA	DELIVER
Process Time(Hash Queue)/ms	8.40	5.37	4.47	25.12	19.80
Process Time(FlashQ)/ms	7.41	4.85	4.50	10.11	7.67

Going deep into the commands of the SMTP DATA and the DELIVER, we test the performance of message reading and writing with FlashQ. Fig.6 plots the read and write performance of the FlashQ in comparison to the traditional disk queue. From Fig.6, we can see that the read and write performance of the FlashQ is much better than that of traditional disk queue.

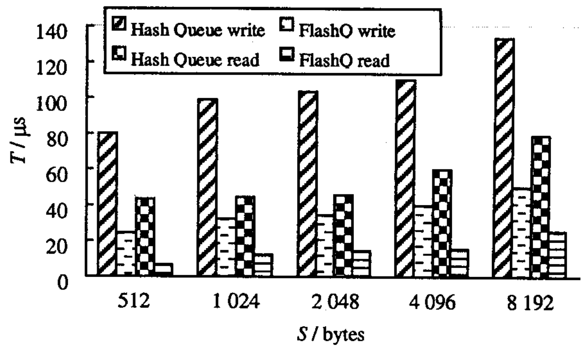

Fig. 6 I/O performance comparison of FlashQ with traditional Hash Queue

5 Conclusions

In this paper, we analyses and finds that the disk contention is the chief bottleneck of the traditional mail system. According to the I/O characteristic of messaging system, we design and implement an efficient disk queue I/O mechanism called FlashQ. The FlashQ achieves high efficiency by reducing file management overload and adopting comfortable rdad write policy to make full advantage of the disk bandwidth.

References

[1]Gregory R G, Kaashoek M F. Embedded Inodes and Explicit Grouping:Exploiting Disk BandWidth for Small Files[C]. Proceedings of the 1997 USENIX Annual Technical Conference Anaheim, CA, 1997. 1-17

[2] Maltzahn C, Kathy J R. Reducing the Disk I/O of Web Proxy Server Caches[C]. Proceedings of the 1999 USENIX Annual Technical Conference, Califorma, USA, 1999. 225-238

[3] Evangelos P M, Manolis G H K, *et al.* Secondary Storage Management for Web Proxies[C]. Proc. 2nd USENIX Symposium on Internet Technologies & Systems, Colorado, USA, 1999. 93-104

[4] Rosenblum M, John K O. The design and implementation of a log-structured file system[J]. ACM Transaction on Computer Systems,1992, 10(1):26-52

[5] McCormick M, Ledlie J T. A Fast File System for Caching Web Objects[C]. First Annual Symposium on Operating Systems Design, Implementation, and Evaluation, California, USA, 2000. 21-27

[6] McKusick K M, William N J, Samual J L, *et al.* A fast file system for UNIX[J]. ACM Transaction on Computer Systems, 1984, 2(3): 181-197

编 辑 王 燕