A Memory Management Architecture for a Mobile Computing Environment

Shigemori Yokoyama[1], Takahiro Okuda[2],
Tadanori Mizuno[2] and Takashi Watanabe[2]
[1] Mitsubishi Electric Corp. [2] Faculty of Information, Shizuoka Univ.

Abstract

In recent years, the rapid progress of hardware technology has enabled people to use mobile terminals away from the office or home with the use of cellular phones. However there are not a little remaining issues which are narrow bandwidth of wireless communications, the limited battery duration of mobile terminals, and others. This paper proposes a common memory management mechanism for mobile terminals and servers called Memory Management Architecture for Mobile Computing Environment (MMM). MMM allocates a part of the memory of a mobile terminal and a part of the memory of a server as common memory and maintains the consistency of the common memory areas. We evaluate MMM using sample application program models against traditional memory. The result shows that MMM can reduce wireless traffic to maintain consistency, and that depending on the charge policy, a different prefetching scheme is efficient in minimizing communication cost.

1. Introduction

A mobile computing environment has been realized according to the rapid progress of wireless communications and computer technology and the rapid popularization of cellular phones and mobile terminals. But at present several issues still exist related to how a mobile computing environment is limited due to the narrow bandwidth of wireless communications, unstable connectivity, and the limitations of the battery duration at mobile terminals, CPU performance, memory capacity, and disk capacity. Additionally, it is hard to develop application programs that cooperate in mobile computing environments because of the complicated nature of wireless communications.

An application program on a mobile terminal can access data on a server by only accessing its own memory if part of the memory of the server could be included in part of the memory of the mobile terminal. An application program on the server also can do the same thing.

In this paper we propose a memory management architecture called MMM. It controls a part of the memory of a mobile terminal and a part of the memory of a server which are common to each other. The main purposes of MMM are to achieve the following in a mobile computing environment. They include the synchronization of common data between a mobile terminal and a sever, easy programming of applications with the cooperation of a mobile terminal and a server, the increase of processing speed, the reduction of communication cost and power reduction.

In Chapter 2 the architecture of MMM is described, in Chapter 3 the result of evaluation by simulation is described and the last chapter concludes the paper.

2. Architecture of MMM

Some of the traditional memory architectures are introduced to MMM. The memory architecture most related to our study is cache memory [1]. MMM is similar to cache memory in that the current data used are fetched from the memory at a remote location and stored in local memory.

The second aspect of MMM is similar to distributed shared memory. To enhance processing performance many kinds of systems composed of multiple processors have been proposed and researched [2-6]. MMM is a kind of distributed shared memory in which some common address spaces at physically different locations are controlled to have a shared address.

The third aspect of MMM is that it provides virtual memory space to mobile terminals. In normal virtual memory the actual memory content is in external memory, and external memory is normally disk memory. In the virtual memory system used in MMM, the external memory for virtual memory corresponds to a common memory space on the server.

2.1 Outline and features of MMM

The concept of MMM is shown in Fig. 1. In MMM a mobile terminal can access a part of a server's memory space as its own memory space, and also a server can access a part of a mobile terminal's memory space as its own memory space.

Employing MMM, programming becomes easier as ap-

0-7695-0571-6/00 $10.00 © 2000 IEEE

plications on a mobile terminal do not need to program the communication procedures to read or write data on a server, and communications can be more efficient by communicating only at the necessary time and in only a necessary amount.

We summarize the features and the differences from other systems as follows.

a. Mobile terminals and a server have different address spaces and share some of this space. (It is possible to have a single address space.)
b. Mobile terminals and a server are located at physically different locations, and fundamental communication media is wireless.
c. The bandwidth of wireless communication is narrow and not stable.
d. Applications can use data conditionally even if wireless communication is disconnected.
e. MMM can be implemented by additional hardware of conventional circuits.

2.2 Memory management systems

(1) Memory lines and memory control status field

The common memory area of a mobile terminal and a server is divided into fixed areas called lines. The memory mapping is handled by line units. A line is divided into several blocks, and memory write back for memory consistency is handled by the blocks. The memory control status memory is introduced to control the memory line status. It contains memory control status fields corresponding to each line and is used to ensure memory consistency, i.e., lines on a mobile terminal and on a server having the same line number are controlled to have the same contents by memory control status fields. Fig. 2 shows the correspondence of memory lines and memory control status fields on a mobile terminal and a server. Line size and block size are assumed to be from 16 bytes to 4096 bytes and from 16 bytes to 1024 bytes, respectively.

(2) Memory control status field and memory line status

Memory line status is determined by the memory control status field. Each memory control status field is composed of V bit, C bit and plural D bits.

V_M bit in a memory control status field on a mobile terminal is a valid bit which indicates whether the corresponding line is valid or invalid. C_M bit is a copy bit which indicates whether the line has been copied to a server or not. Similarly, V_S bit in a memory control status field on a server is a valid bit, and C_S bit is a copy bit which indicates whether the line has been copied to a mobile terminal or not. D_M and D_S bits on a mobile terminal and a server are dirty bits that show whether the write operation has been performed in those blocks. Table 1 shows the meaning of the status of the memory line determined by the memory control status fields.

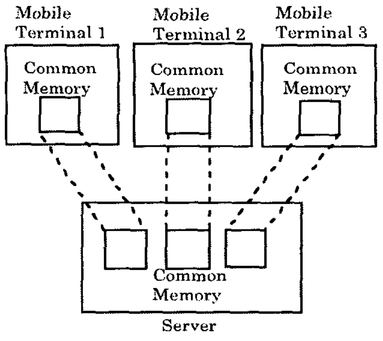

Fig. 1 Concept of MMM

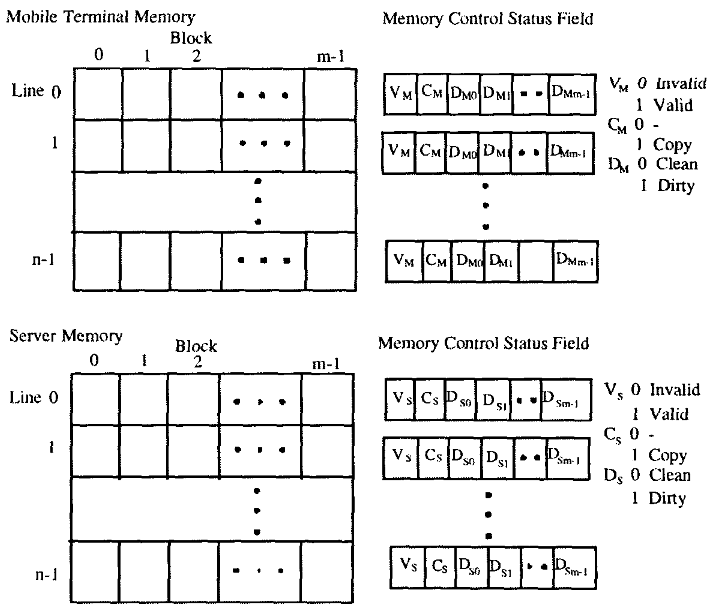

Fig. 2 Memory mapping and memory control status field

Table 1 Status of memory line

VCD	Memory Line Status
100	Latest, Not Altered
101	Latest, Altered
11-	Possibly Old, Copied to Remote
0--	Invalid, Initial State

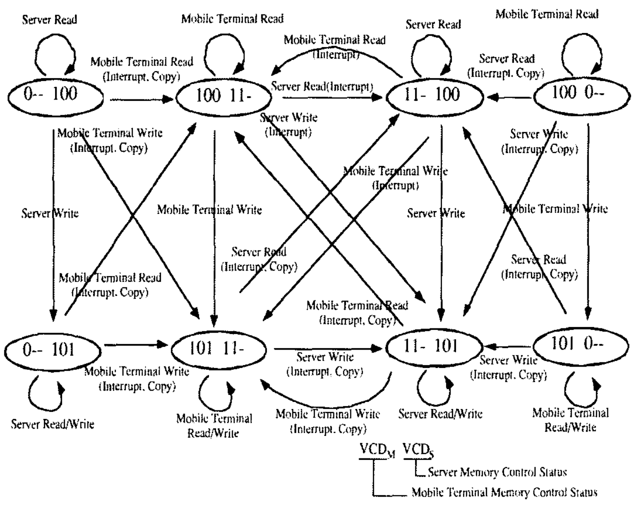
Server Read
Mobile Terminal Read
Server Read
Mobile Terminal Read
Mobile Terminal Read (Interrupt)
Mobile Terminal Read (Interrupt. Copy)
Server Read (Interrupt. Copy)
0-- 100
100 11-
Server Read(Interrupt)
11- 100
100 0--
Server Write (Interrupt)
Mobile Terminal Write (Interrupt. Copy)
Server Write (Interrupt. Copy)
Mobile Terminal Write (Interrupt)
Server Write
Mobile Terminal Write
Server Write
Mobile Terminal Write
Server Read (Interrupt. Copy)
Server Read (Interrupt. Copy)
Mobile Terminal Read (Interrupt. Copy)
Mobile Terminal Read (Interrupt. Copy)
0-- 101
101 11-
Server Write (Interrupt. Copy)
11- 101
101 0--
Mobile Terminal Write (Interrupt. Copy)
Server Write (Interrupt. Copy)
Server Read/Write
Mobile Terminal Read/Write
Mobile Terminal Write (Interrupt. Copy)
Server Read/Write
Mobile Terminal Read/Write
VCD_M VCD_S
Server Memory Control Status
Mobile Terminal Memory Control Status

Fig. 3 Transition of memory control status

(3) Memory access and line transfer control

When a memory access occurs, the kind of memory operation and the succeeding memory line status are determined by the kind of memory access and the current memory line status. The line transfer control and related memory line status control are performed in a memory exception interrupt which is invoked by a memory access. Communications between a mobile terminal and a server to transfer the lines are executed in a memory exception interrupt. The transition diagram of the memory control status is shown in Fig. 3.

Here, we explain some sample memory accesses and line transfer controls. If an application accesses a line with the latest status (VC_M=10) on a mobile terminal, the content of the memory is valid. Thus, the read or write operation on a mobile terminal is performed successfully. If it is a write operation, D_M bit is set. If the line status is invalid (V_M=0) or possibly old (VC_M=11), the memory exception interrupt occurs to achieve memory consistency. Then the routines of memory exception interrupt on the mobile terminal and the server communicate to transfer the designated line. By receiving the line request from the mobile terminal, the remote interrupt occurs on the server. In its interrupt routine if the line status of the mobile terminal is invalid (V_M=0), the content of the line is transferred to the mobile terminal. If the line status is possibly old (VC_M=11) then the memory control status field on the server is inspected further. If one of the D_S bits is set, the corresponding blocks are transferred. And if all D_S bits are reset none of the blocks is transferred because the line has not been altered. Then the status on the server is set to possibly old (VC_S=11) and the remote interrupt routine is terminated. In the interrupt routine on the mobile terminal when the data from the server is transferred, the line on the mobile terminal becomes the newest, the status is set to the latest (VCD_M=100), and the memory exception interrupt is terminated. After the control returns to the interrupted instruction, the instruction reads or writes, and if it is a write operation the dirty bit (D_M) is set.

The operation of memory access on a server is fundamentally the same as the above-mentioned access on a mobile terminal.

(4) Resolution of line access conflict

When a mobile terminal and a server request the same line at the same time, there is a possibility of memory deadlock. Preventing the deadlock, to accept the remote interrupt is inhibited between the acceptance of memory exception interrupt and the completion of the first instruction execution after the end of the interruption. By this method when the same line is used, the memory line is accessed alternately between the mobile terminal and the server.

(5) Control of communication failure

If a memory exception interrupt occurs, the connection of wireless communication should be established. And if the connection fails, the control program reports the status of memory to the application. The application can use the data conditionally.

2.3 Expansion of memory address space

By means that a common memory area on a mobile terminal is controlled by virtual memory, memory on a mobile terminal can be expanded.

Virtual memory of the common memory area on a mobile terminal is mapped to real memory, and the necessary memory for the time being is located in real memory. Virtual memory and real memory are divided into pages. Virtual memory uses pages as a base of control. Memory control status memory of MMM corresponds to a common memory area of real memory. If the line size is different from the page size and is smaller than the page size, the page size should be equal to the integer times of the line size. Similarly, if the line size is larger than the page size, the line size should be equal to the integer times of the page size.

Introducing a virtual memory system in MMM, the common memory area on a mobile terminal can be expanded. Additionally, conventional virtual memory systems can be used so the architecture need not to be changed.

2.4 Hardware structure of MMM

Fig. 4 shows a block diagram of the hardware structure of MMM. An MMM system can be realized as a processor which has standard architecture and need not be modified, but only has appended RAM for memory control status memory and related control circuits.

2.5 Prefetching of line

Memory control of MMM is a demand basis. If the line that is transferred is used in succeeding instructions, the efficiency of communication increases. Therefore, it is better that the line size is larger to some extent. But if the line size is too much larger, the unused memory area will increase and the efficiency will decrease. Therefore, it is expected that the proper line size exists. In MMM, little effect occurs when expanding the line size to a size too large for increasing execution speed because of the interruption of program execution during line transfer. Increasing the line size increases wait time and reduces usability, which is irritating for users. Giving usability precedence over others and waiting time being less than a second, the maximum line size is less than 1024 bytes in the case of 9600 bps communication speed. But if the program execution and the line transfer are performed in parallel, it is possible to enhance the execution speed. This can be realized to prefetch the lines that are probably used next after the requested line is transferred. It is possible to prefetch multiple lines. But if too many lines are prefetched, unused data increase and the overhead increases. The effect of prefetching is proportional to concurrency of program execution and line transfer. It is supposed that many applications on mobile terminals are interactive. Thus, waiting time for responding from a user can be used for prefetching to enhance the processing speed.

3 Evaluation of MMM

3.1 Simulation

The MMM system was evaluated by simulation of various application program models. Application models running solely on a mobile terminal are used first, and a model running alternatively on a mobile terminal and a server is used last. Initially, applications are stored in the memory on the mobile terminal, and data are stored in the common memory on the server.

Communication speed between the mobile terminal and the server is 9600 bps, and the communication overhead (command, line address, count and check code) is 10 bytes.

Application models used are as follows.

(1) Schedule Book

The data amount per an item is 48 bytes, the data amount per a day is 1152 bytes, the amount of the days is 50 days, and the total amount of the data is 60Kbytes. The

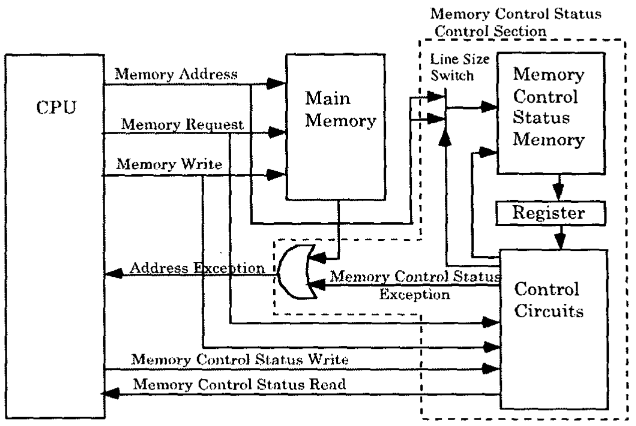
Memory Control Status Control Section
CPU
Memory Address
Memory Request
Memory Write
Main Memory
Line Size Switch
Memory Control Status Memory
Register
Address Exception
Memory Control Status Exception
Control Circuits
Memory Control Status Write
Memory Control Status Read

Fig. 4 Hardware architecture of MMM

mobile terminal can display a day of data per a display frame at a time. Several operation patterns were prepared and simulated. They are the different combinations of designation of days, referencing, and updating schedule data.

(2) Address Book

The data amount per a record is 160 bytes, the amount of the records is 500, and the total amount of the data is 86 Kbytes. For displaying and altering, Initial Frame, Select Frame and Alter & Display Frame are prepared. Initial Frame displays the table of initial letters to select the name to inquire. Select Frame displays name lists selected by the initial letter. Alter & Display Frame displays the address and related data selected by the name on Select Frame. Reference update and create are done on Alter & Display Frame. The operation patterns having several combinations of references and updates were prepared.

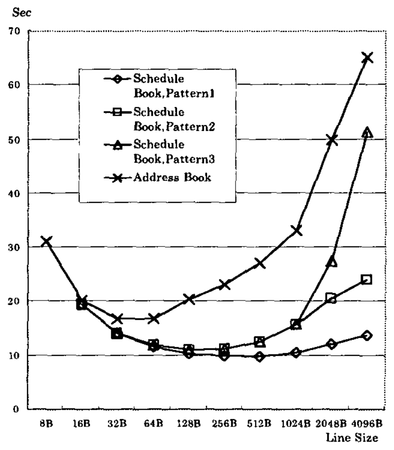

Figure 5 Line size and data transfer time -Schedule book and address book-

3.2 Evaluation for optimal line size

The relationship between the line size and the data transfer time is evaluated. Figure 5 shows the relationship for the Schedule Book and the Address Book simulations. The Schedule Book simulation uses three kinds of operation patterns. The Address Book simulation uses the operation pattern to search 10 records. The Schedule Book simulation shows the optimal line size in which the data transfer time becomes minimum is from 64 bytes to 512 bytes and the Address Book simulations shows the minimum data transfer time is about from 16 bytes to 128 bytes. The smaller line size gets the bigger protocol overhead. The more data transfer time according to increasing line size is for the increasing ratio of unused data.

3.3 Comparison of MMM and a conventional memory system on a mobile terminal and a server

MMM is compared with the conventional memory system using the mutual operation scenario of the Schedule

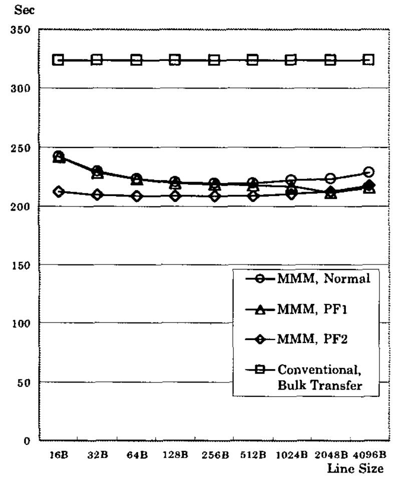

Fig.6 Schedule book on the mobile terminal Line size and total execution time

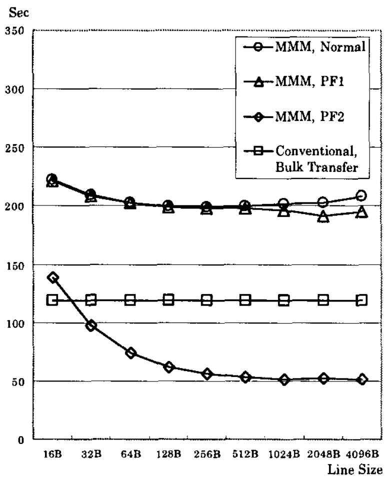

Fig.7 Schedule book on the mobile terminal Line size and connection time

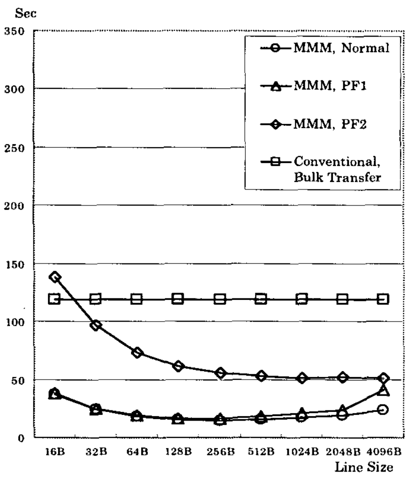

Fig.8 Schedule book on the mobile terminal Line size and data transfer time

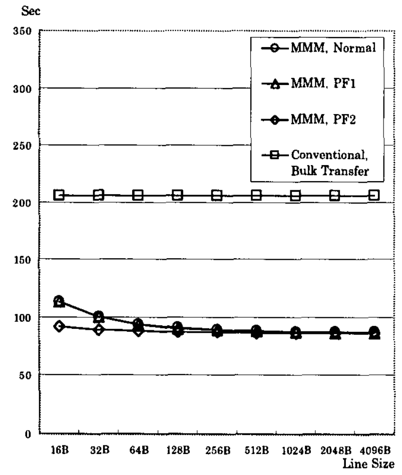

Fig.9 Schedule book on the server Line size and total execution time

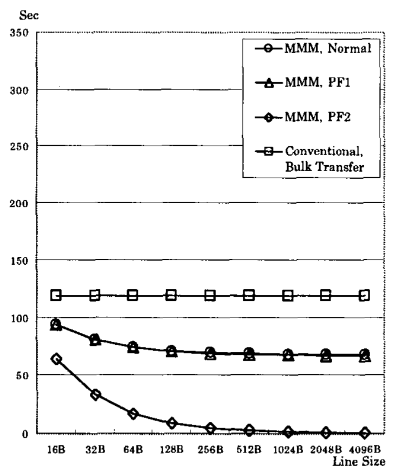

Fig.10 Schedule book on the server Line size and connection time

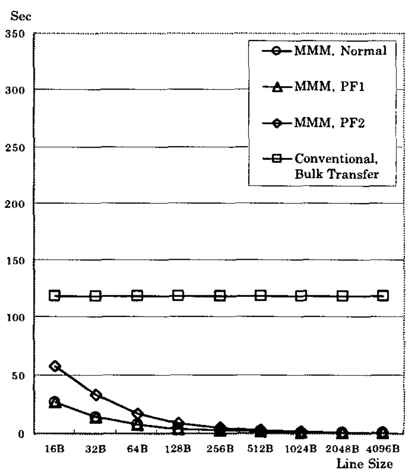

Fig.11 Schedule book on the server Line size and data transfer time

Book model on a mobile terminal and on a server. In a conventional memory system it is possible to only transfer the necessary data, but composing applications is complicated because applications on a mobile terminal should handle communication procedures and also applications on a server should be prepared to handle the data. Therefore, for comparison the conventional memory system model transfers all data in the beginning of program execution from the server to the mobile terminal, and at the end of program execution all data are transferred back. We call this case the Bulk transfer method. The scenario is as follows. First, on the mobile terminal ten days of schedule data are referenced and five days are updated. Second, on the server ten days of schedule data that were referenced and updated on the mobile terminal are referenced. Third, on the mobile terminal the same ten days of schedule data are referenced and five days are updated, and finally, on the server the same ten days of data are referenced. After a day of data is displayed, 3 seconds are necessary for referencing, and for updating, an additional 12 seconds are needed for input of 24 letters. Total execution time includes execution time of the program, referencing time, updating time and 10 seconds of establishing time for the connection. MMM and the Bulk transfer method are compared. MMM includes three cases, the normal case and two prefetch cases. Prefetch consists of fetching the succeeding lines using waiting time during connection time after the fetch of the targeted line. Prefetch 1 is to fetch one line (PF1 in Fig6-11). Prefetch 2 is to fetch multiple lines during waiting time (PF2 in Fig6-11). In MMM, if all data are transferred during the execution, the connection is disconnected at that time and reconnected at the beginning of last data transfer to the server. If all data are not transferred during the execution, the disconnection does not happen.

Fig. 6, Fig. 7 and Fig. 8 show total execution time, connection time and data transfer time on the mobile terminal. Fig. 9, Fig. 10 and Fig. 11 show total execution time, connection time and data transfer time on the server. In Fig. 7 the connection time of the normal and PF1 cases in MMM are larger than that of the Bulk transfer method. This is because the disconnection does not happen in the execution. In the case of PF2 the reason for the decrease of the connection time according to the expansion of the line size is that in the execution all data are transferred and at that time the disconnection occurs. If the communication charge is determined per connection time, it is better to prefetch multiple lines during connection time.

In contrast on the server the connection time in Fig. 10 and the data transfer time in Fig. 11 are different from those on the mobile terminal. They decrease monotonously by an increase of line size. This is because initially the original content is on the server. The data transfer to the server is only updated blocks and does not depend on the line size, and most of the communication between the mobile terminal and the server consists of transferring the control information. The connection time and data transfer time of the control information increase by a decrease of line size for protocol overhead.

Fig. 6 and Fig. 9 show that the total execution times of MMM are smaller than those of the Bulk transfer method. The difference nearly corresponds to the bulk transfer time. Fig. 7 and Fig. 10 show that the connection time of the normal and PF1 cases in MMM are not always smaller than those of the Bulk transfer method. If the communication charge is determined per connection time, it is better to prefetch multiple lines in MMM. Fig. 8 and Fig. 11 show that in all the MMM cases, excluding the 16 bytes lines of PF2, the data transfer times are smaller than those of the Bulk transfer method. The normal case in which the prefetch is not performed is the minimum communication time. Therefore, if the communication charge is determined per amount data, it is better not to prefetch.

4 Conclusion

MMM (Memory Management architecture for Mobile computing environment) is the most preferable memory architecture for mobile terminals and for servers. It incorporates cache, virtual memory and shared memory architecture.

We described and evaluated MMM and demonstrated that by introducing MMM the synchronization of common data on a mobile terminal and on a server could be maintained, the applications could be composed easily because they did not need to consider the complicated communication procedures, the applications could execute faster and the efficiency of communications could increase by decreasing the communication cost and by saving power. Moreover, we showed that the efficiency of the communications could increase by prefetching multiple lines.

References

[1] Alan Jay Smith "Cache Memories," Computing Surveys, Vol. 14, No. 13, Sept. 1982.

[2] A.S.Tanenbaum, Translated by Mizuno et al., "Distributed Operating System", Prentice Hall in Japan, 1996.

[3] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta and J. Hennessy, "The Directory-Based Cache Coherence Protocol for DASH Multiprocessor,"17th ISCA, 1990.

[4] J. Kuskin et al "The Stanford FLASH Multiprocessor," Proc. 21st ISCA, 1994.

[5] T. Lovet and R. Clapp, "STiNG: A CC-NUMA Computer System for the Commercial Marketplace," Proc. 23rd ISCA, 1986.

[6] T. Matsumoto, K. Nishimura, T. Kudoh, K. Hiraki, H. Amano and H. Tanaka, "Distributed Shared Memory Architecture for JUMP-1 a general-purpose MPP prototype," IEEE 1996.