Performance Evaluation of Instruction Set Architecture of MBP-light: a distributed memory controller for a large scale multiprocessor

Noriaki Suzuki† and Hideharu Amano†

Abstract *The instruction set architecture of MBP-light, a dedicated processor for the DSM (Distributed Shared Memory) management of JUMP-1 is analyzed with a real prototype. The Buffer-Register Architecture proposed for MBP-core improves performance with 5.64% in the home cluster and 6.27% in a remote cluster. It appears that the dominant operations in the DSM management program are handling packet queues assigned into the local cluster. Thus, common RISC instructions, especially load/store instructions, are frequently used. Separating instruction and data memory improves performance with 33%. The results suggest that another alternative which provides separate on-chip cache and instructions dedicated for packet queue management is advantageous.*

Keywords: CC-NUMA, DSM management, Instruction Set Architecture, JUMP-1

1. Introduction

A Cache Coherent Non-Uniform Memory Access machine (CC-NUMA) was one of hopeful candidates for future common high performance machines in 1990s. Unlike bus-connected multiprocessors, the system performance can be enhanced scalable as to the number of processors. Moreover, parallel programs developed in small multiprocessors can be ported easily.

JUMP-1 is a prototype of a massively parallel processor with cache coherent DSM developed by collaboration of seven Japanese universities[1]. The major goal of this project is to establish techniques required to build a cost effective DSM when a large number of processing units are connected. In order to satisfy both high degree of performance and flexibility, JUMP-1 has several distinctive structures. Interconnection Network called RDT[2] includes both torus and a kind of fat tree structure with recursively overlayed two-dimensional square diagonal tori structure. A dedicated processor called MBP(Memory Based Processor)-light[3] is proposed to manage the DSM of JUMP-1. It consists of a simple dedicated core processor and hardwired controllers which handle memory systems, bus and network packets. For efficient DSM management, a special instruction set and register architecture called "Buffer-Register Architecture" is introduced in the core processor.

In this paper, we evaluate the performance and impact of an instruction set architecture of MBP-light in JUMP-1. Using a real machine, we estimated the number of executed instructions and analyzed its efficiency.

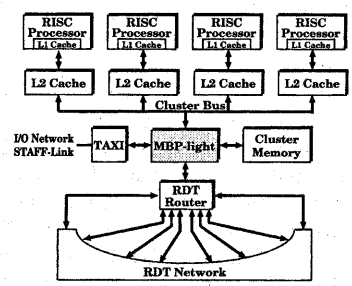

Fig. 1 The Structure of JUMP-1 Cluster

2. The Structure of JUMP-1

2.1 Overview of JUMP-1

The initial design of JUMP-1 consists of 256 clusters connected each other with a network RDT. Each cluster provides a high speed point to point I/O network connected with disks and high-definition video devices.Each cluster is a bus-connected multiprocessor, as shown in Figure 1, including four RISC processors (SuperSPARC+), MBP-light which is directly connected to a cluster memory, and RDT router chip for interconnection network[2]. MBP-light, the heart of JUMP-1 cluster, is the custom designed processor which manages DSM(Distributed Shared Memory), synchronization, and packet handling.

2.2 The DSM Management in JUMP-1

To build a cost effective DSM, the following methods are used in JUMP-1[4].

(1) Each processor (SuperSPARC+) share a

† Department of Computer Science, Graduate School of Keio University

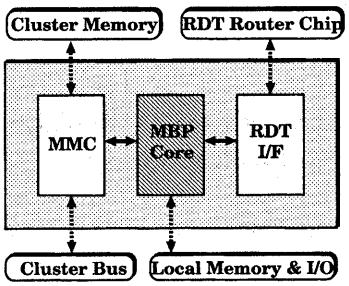

Fig. 2 The Structure of MBP-light

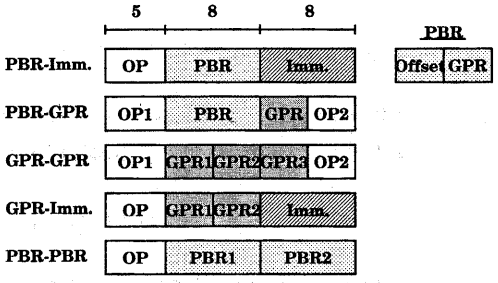

Fig. 3 The Representative Instruction Set

global virtual address space with three-stage TLB implementation. The directory is attached not to every cache line but to every page, while the data transfer is performed by a cache line. Some parts of cluster memory are available as L3 (Level-3) cache which stores the copies of other cluster memory.

(2) Various types of cache coherence protocols can be utilized dynamically, including not only an invalidate policy but also an update policy ⋆.

(3) Reduced Hierarchical Bitmap Directory schemes (RHBDs) are introduced[7] to manage directory efficiently. In the RHBD, the bitmap directory is reduced and used in the packet header for quick multicasting without directory access in each hierarchy. The hierarchical structure of RDT is suitable for an efficient implementation of the RHBD.

The above DSM management mechanism is executed by the software on MBP-light developed in Kyoto University[5].

3. MBP-light

As shown in Figure 2, MBP-light consists of three modules: RDT interface to treat network packets, MMC (Main Memory Controller) to control cluster memory and cluster bus, and MBP-core which is the core processor. RDT Interface and MMC provide their own hardware mechanisms, and work independently from MBP-core.

3.1 MBP-core

3.1.1 Buffer-Register Architecture

Since jobs which must be quickly processed are mostly managed by the hardware mechanisms in the RDT Interface and MMC, the MBP-core only processes a complicated part of the DSM protocol.

MBP-core provides 16 GPRs (General Purpose Registers) of 16-bit width and 112 PBRs (Packet Buffer Registers) of 68-bit width. The PBR is indicated by the content of the GPR, and accessed in the processor pipeline like a common register. While operations and data transfer between PBR and GPR or PBR and PBR are allowed, the content of the PBRs is transferred directly as a packet from/to MMC or RDT Interface. As operations are mainly done between such a packet buffer and a register, we call this structure the *Buffer-Register Architecture*.

3.1.2 Instruction Set

Figure 3 shows the instruction set and representative instruction formats of the MBP-core. The PBR is byte-addressed by the content of the GPR with 4-bit displacement. Operation target of the PBR is mainly a certain part of 8-bit or 16-bit in the PBR addressed by the GPR. This comes from that almost all the fields of a packet header are less than 8 bit and they are treated independently. While various operations are provided between GPR and GPR or GPR and PBR, operations between PBR and PBR are limited (mainly data transfer operations).

Table 1 The Classification of MBP-core Instructions

Class	Type
WGG	operate between GPR and GPR
LMA	access to local memory
BRANCH	branch
WGI	operate between GPR and Imm.
BPI	operate between PBR and Imm.
RDT	control RDT Interface
MPP	transmit from PBR to PBR
WPG	operate between PBR and GPR
MMC	control of MMC
TJ	table jump
INT	control interrupt
IMA	access to internal memory
SPE	special instructions
NOP	no operation

The classification of instructions on MBP-core is shown in Table 1. In addition to the instructions

⋆ The update policy could not be implemented with a hardware problem in the real machine.

described above, MBP-core has some instructions which control RDT Interface or MMC. It also has a dozen special instructions to process a protocol transaction quickly.

3.1.2.1 Common RISC style instructions:

WGG, WGI, LM and BRANCH instructions are common 3-operand RISC style instructions. WGG includes arithmetic (add/sub) and logical operation between GPR and GPR, while WGI for GPR and immediate data. There are no instructions to copy the date between registers because number 0 GPR (R0) is always zero. BRANCH instructions use simple flags.

3.1.2.2 PBR manipulating instructions:

PBR manipulating instructions are classified into WPI, WPG and MPP instructions. By using WPI instructions, 8 bits immediate data and a 8-bit field of PBR can be calculated. It is convenient to generate the field of a packet header. WPG instructions are used for operation between a 16-bit field of PBR and GPR. Both the GPR and the target field of PBR can be a destination register. Finally, 8-bit/16-bit field and the whole field (68-bit) copy between PBRs can be done using MPP instructions. MPP instructions are used for copy of packet header or body.

4. Evaluation Results

4.1 Evaluation Environment

Now, the DSM management program is available, and composed as a firmware of JUMP-1[5]. In order to evaluate the instruction set architecture of MBP-core, an application program, DSM based parallel matrix multiplication, is executed on JUMP-1. The product of 256 × 256 matrices consisting of single precision floating point numbers is calculated in parallel. The number of instructions in the following sections is average number for processing 100 receiving packets. Note that, the application program itself runs on node processors (SuperSPARC+), and MBP-core executes the DSM management program composed as a firmware.

4.2 The Instruction Mix

Here, the instruction mix of MBP-light is analyzed. Although the same DSM management program is executed, the executed processes are different in the home node and remote nodes. Thus, the cluster 0 (home cluster) and the cluster 1 (remote cluster) are evaluated. These results are shown in Figure 4.

These results show that common RISC instructions are dominant compared with PBR manipulating instructions and special instructions. Only MVPG (MV PBR from/to GPR) is frequently used

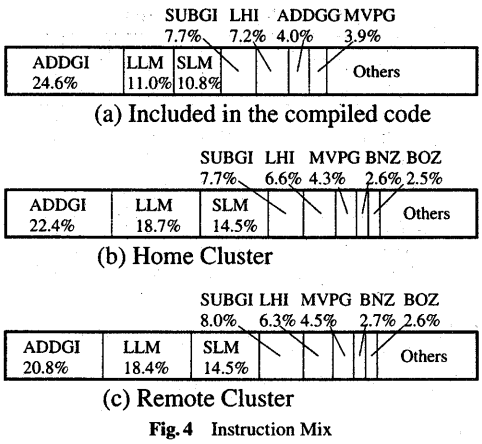

Fig. 4 Instruction Mix

special instructions of MBP-core, and others are negligible.

Table 2 Comparison with Local Memory Access

	1 Clock Stall	Without Stall	Impr.
Home	64471	48394	33.2%
CL1	43130	32448	32.9%

The ratio of the local memory access instructions (LLM and SLM) is also high, that is, 33.2% in the home cluster and 32.9% in the cluster 1. This is because the DSM management program frequently manipulates software managed packet queues in the local memory.

An instruction memory and a data memory are not separated in MBP-light by the pin limitation problem of the package, and these instructions cause 1 clock stall. Table 2 compares the number of executing clock cycles with and without stall. This table shows that separation of instruction and data memory improves the performance 33.2% in the home cluster and 32.9% in the cluster 1. Thus, the separation should be done if the package with enough pin number can be used.

4.3 The Effect of PBR Manipulating Instructions

Table 3 shows the number of PBR manipulating instructions included in the compiled code and executed in home/remote clusters. Since special function calls are needed, a large part of PBR instructions are not included in compiled code, and never executed. Only move instructions and bit manipulation instructions are frequently used, while numerical/logic operation instructions are rarely used.

Table 4 assumes the following conditions and shows the number of cycles in the case of replacing PBR manipulating instructions with common

Table 3 PBR Manipulating Instructions

Ins.	Compiled code	Home	Remote
ANDPI	10	0	38
LPI	40	52	3
ADDPG	9	56	42
ANDPG	41	454	330
ORPG	10	0	39
MVBPG	252	185	319
MVPG	909	2099	1476
MVBPP	46	2	0
MVPP	144	134	166
MVLPP	158	298	205

Table 4 Replacement of PBR Manipulating Instructions with Other Instructions

Ins.	Cycles	Ins.	Cycles	Ins.	Cycles
ANDPI	3	ORPG	2.5	MVPP	2
LPI	2	MVBPG	1	MVLPP	10
ADDPG	2.5	MVPG	1		
ANDPG	2.5	MVBPP	2		

Table 5 Compare Required Cycles: PBR Manipulating Instructions

	With PBR	Without PBR	Impr.
home	64471	68106	5.64%
CL1	43130	45836	6.27%

instructions.

- The packet buffers are mapped in an internal memory.
- A byte accessing to the internal memory is allowed.
- The access of the internal memory is completed in 1 cycle.

The numbers of required cycles for ADDPG, ANDPG and ORPG differ whether the destination is PBR or GPR. It requires 2 cycles when GPR is destination, while 3 cycles are necessary when PBR is destination. Here, it is considered to require 2.5 cycles for convenience.

Table 5 shows the required clock cycles with and without PBR manipulating instructions. In this case, 5.64% and 6.27% performance improvement are attained in the home and a remote cluster, respectively. Although the cost of Buffer-Register Architecture approach is small compared with common on-chip memory with DMA mechanism, it does not contribute to the performance improvement of the DSM management program so much.

5. Conclusion

In this paper, we evaluate the instruction set architecture of MBP-light executing the DSM management program on a JUMP-1 prototype.

The Buffer-Register Architecture proposed for MBP-core improves performance with 5.64% in the home cluster and 6.27% in a remote cluster. It appears that the dominant operations in the DSM management program were handling packet queues assigned into the local cluster. Thus, common RISC instructions, especially load/store instructions, are frequently used. Thus, separating instruction and data memory improves performance with 33%.

The results suggest that another architecture which provides separate on-chip cache and instructions dedicated for packet queue management is advantageous.

References

1) K. Hiraki et. al., "Overview of the jump-1, an mpp prototype for general-purpose parallel computations," IEEE International Symposium on Parallel Architectures, Algorithms and Networks, pp.427–434, 1994.
2) Y.Yang et. al., "Recursive Diagonal Torus: An Interconnection Network for Massively Parallel Computers," In IEEE Transactions on Parallel and Distributed Systems, Vol.12, No.7, pp 701–715, July 2001
3) Inoue Hiroaki, et. al., "MBP-light: A Processor for Management of Distributed Shared Memory," 3rd International Conference on ASIC, pp.199–202, Oct. 1998.
4) Hieharu Tanaka et. al., "The Massively Parallel Processing System JUMP-1," Ohmsha, ISBN4-274-90083-5, 1996.
5) M. Konishi et. al., "Implementation of Distributed Shared Memory Management of the JUMP-1 Multiprocessor," IPSJ Transactions, Vol. 42, No. 4, pp. 674 − 682, 2001.
6) N. Suzuki et. al., "Performance Evaluation of a Multicast Mechanism of RDT Network for a Massively Parallel Processor JUMP-1" The Transactions of the Institute of Electronics, Information and Communication Engineers, Vol J85-D-I, No.12, pp.1114-1125, Dec. 2002.
7) T. Kudoh et. al., "Hierarchical bit-map directory schemes on the RDT interconnection network for a massively parallel processor JUMP-1," International Conference on Parallel Processing, August, pp.I-186–I-193, 1995.