Performance Evaluation of a Parallel I/O Mechanism on a Massively Parallel Processing System JUMP-1

Yasunori Osana, Noriaki Suzuki,
Tomonori Tamura, Hideharu Amano

Department of Computer Science
Keio University

Yokohama, Japan

Hironori Nakajo

Department of Computer,
Information and Communication Science
Tokyo University of
Agriculture and Technology

Tokyo, Japan

Abstract

A massively parallel processing system JUMP-1 has been developed for high performance computing with an efficient cache coherent distributed shared memory on a large system with more than 1000 processors.

Considering a total design policy, a scalable I/O mechanism is implemented on JUMP-1. Every cluster board of JUMP-1 has a serial link called "STAFF-Link" (Serial Transparent Asynchronous First-in First-out Link). Each channel of STAFF-Link is connected to an independent I/O unit, and works in parallel to obtain the scalable I/O bandwidth. In this paper, performance evaluation results of JUMP-1 I/O mechanism using real experimental systems are shown.

Keywords: Massively Parallel Processing, Parallel I/O, I/O Network, STAFF-Link, JUMP-1

1 Introduction

A Cache Coherent Non-Uniform Memory Access machine (CC-NUMA) is one of hopeful candidates for future common high performance machines. Presently, commercial CC-NUMA machines, such as SGI NUMAflex[1], has been developed and utilized for scientific computations, database and other applications. Unlike bus-connected multiprocessors, the system performance can be enhanced scalably as to the number of processors. Moreover, parallel programs developed in small multiprocessors can be ported with less efforts.

JUMP-1[2] is developed by collaboration of seven Japanese universities. The goal of the project is to build a prototype of massively parallel processing system with cache coherent DSM. The major aim of this project is to establish techniques required to build an efficient DSM on a massively parallel processor. A lot of novel technologies are innovated in the DSM management of JUMP-1 for this purpose.

An I/O architecture must be cautiously considered when we design parallel computers, since it has become apparent that the I/O performance rather than the CPU's one often limits its total performance of massively parallel computers in a large scale computation.

In the design of JUMP-1 I/O subsystem, we put emphasis on its scalability, flexibility and simplicity. Scalability is important in order to realize massively parallel processing capability. I/O bandwidth must be enhanced as increasing number of processors. Flexibility is required to connect various types of I/O devices to JUMP-1 or any other general purpose computers. It is necessary to relax restrictions on the physical distance that exists between both ends of the communication line from the distributed processing elements to the I/O system. Also, where the system utilizes many I/O devices, a number of cable connections are necessary. Simplicity is necessary to design I/O devices

and system software. It is very important to enable it to design an I/O device and system software easily.

Therefore, we have configured an I/O subsystem in which the I/O units are connected to processing elements using a fast link called STAFF-Link(Serial Transparent Asynchronous First-in First-out Link)[3].

This paper describes the design, implementation and evaluation of the I/O subsystem of JUMP-1. In Section 2, the overview of JUMP-1 is introduced. In Section 3 and Section 4, the I/O mechanism and I/O software implementation are described, and its experimental evaluation is shown in Section 5. Finally we conclude in Section 6.

2 The Structure of JUMP-1

As shown in Figure 1, JUMP-1 consists of 256 clusters connected each other with an interconnection network called RDT (Recursive Diagonal Torus). The RDT[4] includes both torus and a kind of fat tree structure with recursively overlayed two-dimensional square diagonal torus structure. Each cluster provides a high speed point to point I/O links connected with I/O units.

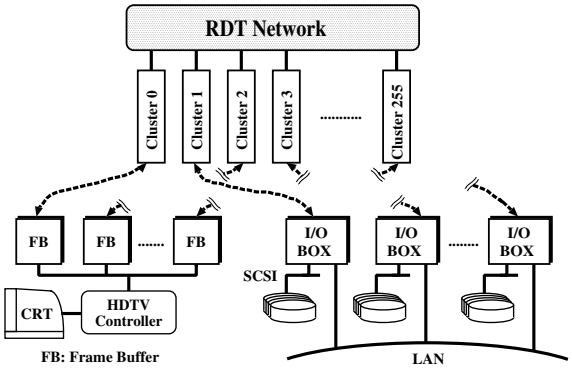

Figure 1: The Structure of JUMP-1

Each cluster is a bus-connected multiprocessor, as shown in Figure 2, including four RISC processors (SuperSPARC+), MBP-light[5] which is directly connected to a cluster memory, STAFF-Link, and RDT router chip for interconnection network[6]. MBP-light, the heart of JUMP-1 cluster, is the custom designed processor for managements of DSM and I/O.

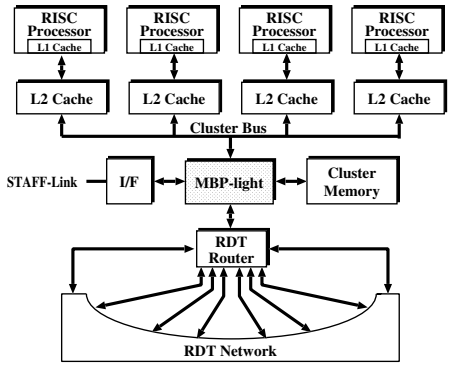

Figure 2: The Structure of JUMP-1 Cluster

Furthermore, packet multicasting and generation/collection of acknowledge packets are automatically done by the cooperation of RDT router chip and MBP-light.

Currently, a prototype with 64 processors (Figure 3) is available and the performance is being analyzed in various aspects.

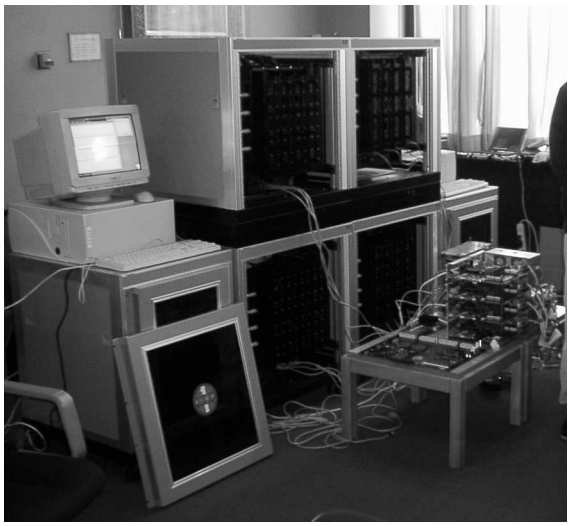

Figure 3: JUMP-1, 64 Processors System

3 I/O Mechanism of JUMP-1

3.1 Design Overview

Installing a dedicated high-speed I/O bus in a particular node and connecting various I/O equipments with that bus, is a general approach for super computers such as the CRAY, and HIPPI[7] is a typical example of that. However, when a dedicated bus is connected to massively parallel processing system comprised of many element processors and clusters, it bottlenecks the total I/O performance.

On JUMP-1, every cluster board has a STAFF-Link module which supports for every cluster to execute I/O operations simultaneously. It's also expected that total I/O bandwidth would be widened as the increasing number of processors.

We also implement a STAFF-Link interface on SBus of SUN's SPARCstation, and currently some SPARCstation5s are working as I/O units for JUMP-1. Each SPARCstation5 (SS5) has 4 channels of STAFF-Link and sharing their disks by NFS. By distributing physical disk access among multiple I/O units with NFS, I/O scalability is achieved. To evaluate this scalability is the main purpose of this paper. NFS on Ethernet can be replaced by a software system using STAFF-Link among SS5s.

3.2 Structure of STAFF-Link

High-speed signals in cables with many lines are difficult to be extended along a long distance because of electric and physical problems.

Thus, STAFF-Link adopts a high-speed serial link which has FIFO on the both ends. By using the buffer, STAFF-Link acts as communication hardware that can be accessed like FIFO memory as shown in Figure 4. Unlike bit-parallel cables, serial cables can save the area for the cables themselves and connectors, and thus, parallel data communication using multiple links between distant machines can be achieved.

STAFF-Link as shown in Figure 4 consists of two STAFF-Link daughter cards (communication blocks), and a cable to connect between them. A STAFF-Link daughter card has two sets of FIFO and TAXI. One set is for sending, while the other is for receiving. TAXI chip converts serial/parallel signals, and controls serial flow with Xon/Xoff handshake lines.

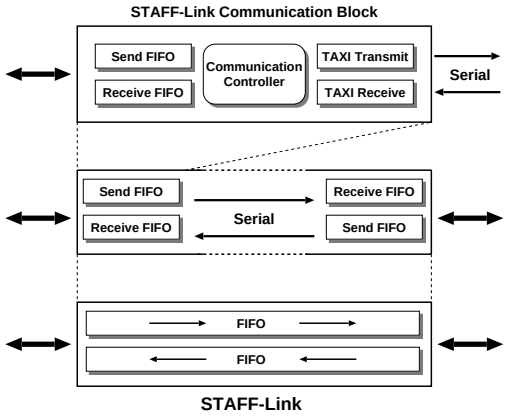

Figure 4: The Structure of STAFF-Link

3.3 STAFF-Link Interface on JUMP-1

MBP-light manages the DSM of JUMP-1, and it has a local I/O bus. A STAFF-Link daughter card is directly connected to a JUMP-1 cluster board and MBP-light can access STAFF-Link daughter card via its local I/O bus.

3.4 STAFF-Link Interface on SS5

For connecting STAFF-Link to SS5, STAFF-Link motherboard and STAFF-Link SBus card are implemented. A STAFF-Link motherboard holds 4 daughter cards. A STAFF-Link SBus card is inserted into an SBus slot on SS5, and connected to a motherboard with a cable.

Solaris7 is installed on SS5, and a STAFF-Link control program runs on it. The control program is always checking FIFO status of each channel. When an I/O request comes from a JUMP-1 cluster, SS5 receives it from STAFF-Link, processes the request, and sends back the result to the JUMP-1 cluster via STAFF-Link.

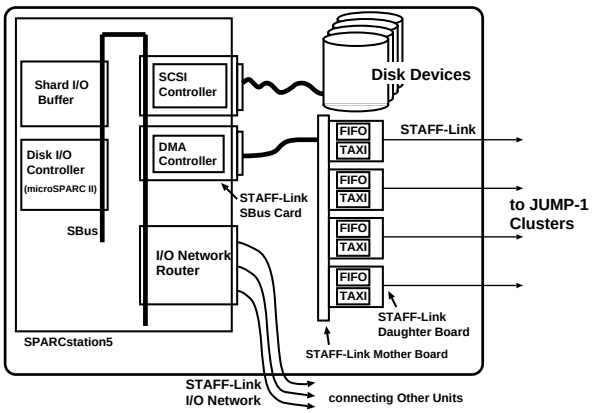

Figure 5: The Structure of I/O Unit

3.5 I/O Network

To distribute the load of I/O processing, a network is formed among the I/O units as shown in Figure 6. They can share their disks or any other I/O devices through the I/O network[8].

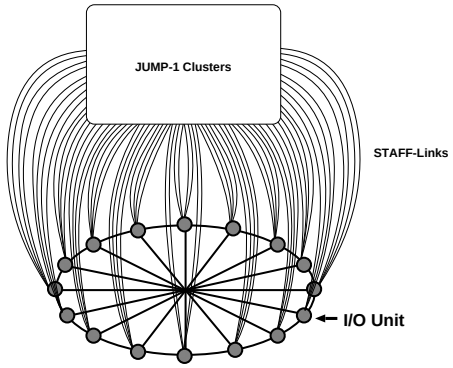

Figure 6: JUMP-1 and I/O Network

I/O network of JUMP-1 was due to be constitute from STAFF-Link. And this network itself was already implemented and evaluated. However, we need to exchange packets on STAFF-Links for it, and it is possible by using a custom board with routing controller. In this paper, we use Ethernet instead of the router board because the whole JUMP-1 system we evaluated is compact and the bandwidth of Ethernet is sufficient for it. Moreover, we can use NFS on Ethernet easily.

4 Software Implementation

To evaluate the scalability of the I/O subsystem of JUMP-1, we have implemented a file I/O library. This library consists of following three parts:

- `libjump1`: A library which includes functions to send an I/O request from a SPARC processor on JUMP-1 to MBP-light. This library is written in C language and executes on SPARC processors. Every application program running on SPARC processors of JUMP-1 must link this library.
- `INTEGRA`: A program for the RISC core of MBP-light. This program loads an application program for SPARC processors from STAFF-Link or Maintenance Bus Interface (MBIF) on the cluster. This program also processes requests from SPARC to I/O and cluster memory processors. Every functions to manage resources on JUMP-1 are *integra*ted in this program.
- `jump1d` (JUMP-1 I/O daemon): A program for SS5. This program monitors STAFF-Link, and processes I/O requests from JUMP-1.

SPARC processors on JUMP-1, MBP-light and SS5 communicate each other to execute I/O operations. An I/O operation is processed as follows:

1. A SPARC processor writes a command and data on the memory. This command and data are going to be stored in the L2 cache shown in Figure 2.
2. The SPARC processor sends a control packet (Non-cacheable Control Write Request) to MBP-light via an L2 cache controller.
3. After the MBP-light receives the Non-cacheable Control Write Request packet, it sends a Read Request packet to the L2 cache controller to get the command

and data which have been written by the SPARC. Then, the MBP-light transfers the command and data to SS5 via STAFF-Link. MBP-light waits until the SS5 replies.

4. When the SS5 receives the command and data from the MBP-light, it executes the requested I/O process. After finishing the I/O process, it sends the result to the requesting MBP-light via STAFF-Link.

5. The MBP-light sets a flag on cluster memory, and the I/O requesting SPARC gets the return value.

5 Performance Evaluation

We have evaluated total I/O bandwidth when the number of I/O units is changed from 1 to 4.

5.1 Experimental Environment

The experimental system as shown in Figure 7 is configured with the following conditions for evaluation.

- STAFF-Link
  Each link is connected with a category 5 twisted pair cable.
- JUMP-1 Clusters
  4 clusters are used for the evaluation. They are working at 16MHz.
- SPARCstation5 (SS5)
  Currently 5 SS5s are utilized for evaluations. One is NFS server, other 4 are NFS clients each has STAFF-Link connecting to a JUMP-1 cluster. They're also connected to the LAN with 10Base-T Ethernet.
  - NFS Server
    MicroSPARC II-85MHz, 160MB RAM
  - I/O Units (4 units)
    MicroSPARC II-110MHz, 96MB RAM

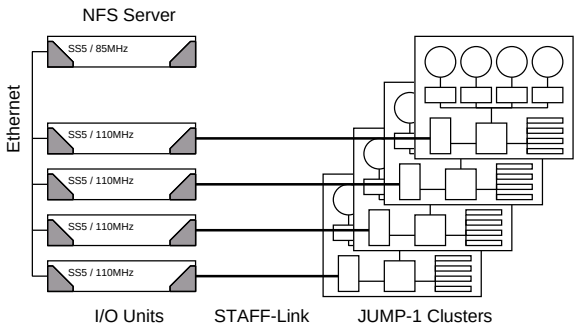

Figure 7: Experimental Environment

5.2 Performance Measurements

We measured bulk transfer rate and file transfer rate. Bulk transfer rate is the maximum transfer rate between STAFF-Link daughter card and SS5 or MBP-light.

For evaluating file transfer rate, we used the following measures: "Actual" and "Total". "Actual" transfer rate, shown in Table 2 is a real rate of STAFF-Link. It is smaller than bulk transfer rate, because of the latency for read/write operations on cluster memory on JUMP-1 cluster and memory on SS5. The other is "Total"transfer rate in Table 3. This is the sum of effective I/O bandwidth on each cluster which contains the latency of disk operations in SS5. Both of them are measured by using local disks and shared disks to clarify the effect of using multiple I/O units.

- Bulk Transfer Rate
  - From SS5 to JUMP-1
  - From JUMP-1 to SS5
- File Transfer Rate
  - With local disk
    * 1 Cluster, 1 I/O Unit
    * 2 Clusters, 1 I/O Unit
  - With NFS shared disk
    * 1 Cluster, 1 I/O Unit
    * 2 Clusters, 2 I/O Units
    * 3 Clusters, 3 I/O Units
    * 4 Clusters, 4 I/O Units

5.3 Results

Bulk transfer rate is shown in Table 1. Table 3 shows average transfer rate from JUMP-1 to SS5s. Table 2 shows actual transfer rate on STAFF-Link.

These three sets of data are measured on the same conditions.

Table 1: Bulk Transfer Rate

SS5→JUMP-1	SS5→SendFIFO	25.24
	RecvFIFO→JUMP-1	15.32
JUMP-1→SS5	JUMP-1→SendFIFO	45.17
	RecvFIFO→SS5	25.24

(Mbps)

Table 2: Actual Transfer Rate on STAFF-Link

Disk	Cluster–SS5	`read`	`write`
Local	1 – 1	3.86	12.29
	2 – 1	2.50	12.29
Shared	1 – 1	3.86	12.29
	2 – 2	3.86	12.29
	3 – 3	3.86	12.29
	4 – 4	3.86	12.29

(Mbps/cluster)

Table 3: Total Transfer Rate on File I/O

Disk	Cluster–SS5	`read`	`write`
Local	1 – 1	2.10	3.50
	2 – 1	1.64	0.64
Shared	1 – 1	2.10	3.51
	2 – 2	4.22	7.02
	3 – 3	6.41	10.93
	4 – 4	8.49	14.50

(Mbps)

Although the maximum transfer rate of STAFF-Link is 140Mbps, the results show that only 15–45Mbps transfer rate is available (Table 1). It is supposed that there exists bottlenecks in the both ends of STAFF-Link. Moreover, it would suffer from overhead concerned with accessing cluster memory from MBP-light when executing I/O operations. Since overhead comes from read and write cluster memory, actual transfer rate becomes slower as shown in Table 2.

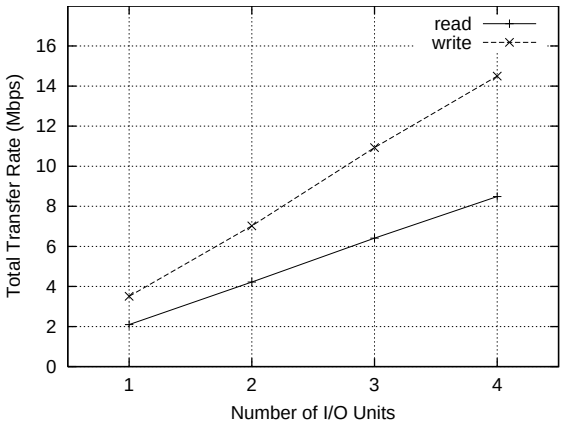

Figure 8: Total Transfer Rate on File I/O

In cases with a single SS5, the I/O bandwidth falls down seriously when two clusters access simultaneously. Table 2 shows that the transfer rate of STAFF-Link itself is degrading by simultaneous accesses from two clusters. This performance degradation is caused by unbalanced transfer rate between MBP-light and SS5 when FIFOs are not full.

Total bandwidth of each cluster in Table 3 is slower than actual transfer rate. This is caused by the latency of SS5 program.

From Figure 8, we can recognize clearly that the system has scalability on I/O bandwidth. When a SS5 with a NFS server is connected to a JUMP-1 cluster, its transfer rate is the same that of a SS5 with local disk connected to a JUMP-1 cluster. This result shows there is no performance degradation by sharing disks, and I/O bandwidth can extend almost linearly by the increasing number of I/O units.

The bandwidth of NFS on Ethernet is 10Mbps at the best condition, but total I/O bandwidth in this experiment is 14.5Mbps. In the case of 4 sets of cluster and I/O unit, I/O scalability is increased linearly as well as the case of less than 3 sets of cluster and I/O unit. However, Table 4 and Figure 9 show that the time to complete `open/close` operation of a file increasing intensively. This means that the cache mechanism on I/O units is working ef-

Table 4: Time for `open/close` Operations

Cluster –SS5	read		write	
	`open`	`close`	`open`	`close`
1–1	3.91	1.47	28.73	42.20
2–2	4.37	1.34	28.48	54.50
3–3	3.06	1.22	23.66	42.29
4–4	3.78	1.26	947.28	2269.52

(msec.)

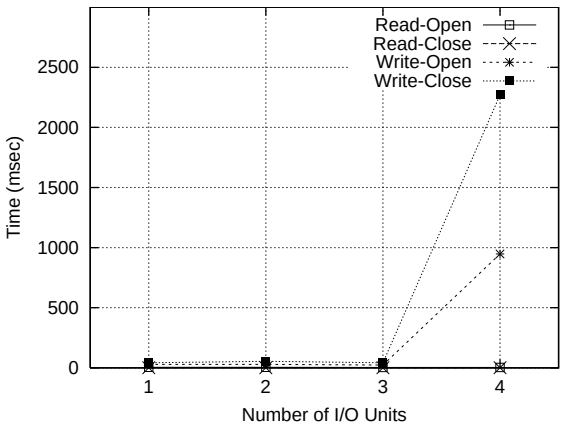

Figure 9: Time for `open/close` Operations

fectively, and it enables I/O units to delay accesses to NFS server.

This shows that cache operation in I/O unit is important, and works effectively. However, it is a result with only four clusters and I/O units. Improved cache method and sophisticated migration of files to specific I/O units will be required to realize I/O scalability for massively parallel processing.

6 Conclusion

In this paper, the I/O subsystem for JUMP-1 is implemented and evaluated. From experiment results, we have confirmed that bundles of serial links are effective for an I/O subsystem of massively parallel computers. We also recognized that the cache mechanism is effective for scalability of total I/O performance.

We are now going to evaluate this I/O system with some I/O intensive applications like ray tracing. Software implementation to access devices other than a disk from JUMP-1 is also considered.

References

[1] John R. Mashey. NUMAflex Modular Design Approach. *News*, 2000.

[2] Hidehiko Tanaka *et al.*, editors. *The Massively Parallel Processing System JUMP-1.* Ohmsha, 1996. ISBN4-274-90083-5.

[3] Hironori Nakajo *et al.* High speed serial communication in a future parallel computer architecture. In *Proceedings of Innovative Architecture for Future Generation High-Performance Processors and Systems*, pages 125–132, 1997 , 1998.

[4] Yule L. Yang *et al.* Recursive Diagonal Torus: An interconnection network for massively parallel computers. In *Proceedings of the 5th IEEE Symposium on Parallel and Distributed P rocessing*, pages 591–594, December 1993.

[5] Inoue Hiroaki *et al.* MBP-light: A Processor for Management of Distributed Shared Memory. In *Proceedings of the 3rd International Conference on ASIC*, pages 199–202, October 1998.

[6] Hiroaki Nishi *et al.* The RDT Router Chip: A Versatile Router for Supporting a Distributed Shared Memory. *IEICE transaction on Information and Systems*, pages 854–862, 197.

[7] *ANSI document: High Performance Parallel Interface, document #X3T9.3*, 1987.

[8] Hironori Nakajo *et al.* An I/O Network Architecture of the Distributed Shared-MemoryMassively Parallel Computer JUMP-1. In *Procedings of 11th International Conference On Supercomputing (ICS97)*, pages 253–260, 1997.