IEICE TRANS. INF. & SYST., VOL.E96–D, NO.12 DECEMBER 2013

PAPER Special Section on Parallel and Distributed Computing and Networking

Network Interface Architecture with Scalable Low-Latency Message Receiving Mechanism

Noboru TANABE[†a)], ***Member and*** **Atsushi OHTA**[††], ***Nonmember***

SUMMARY Most of scientists except computer scientists do not want to make efforts for performance tuning with rewriting their MPI applications. In addition, the number of processing elements which can be used by them is increasing year by year. On large-scale parallel systems, the number of accumulated messages on a message buffer tends to increase in some of their applications. Since searching message queue in MPI is time-consuming, system side scalable acceleration is needed for those systems. In this paper, a support function named LHS (Limited-length Head Separation) is proposed. Its performance in searching message buffer and hardware cost are evaluated. LHS accelerates searching message buffer by means of switching location to store limited-length heads of messages. It uses the effects such as increasing hit rate of cache on host with partial offloading to hardware. Searching speed of message buffer when the order of message reception is different from the receiver's expectation is accelerated 14.3 times with LHS on FPGA-based network interface card (NIC) named DIMMnet-2. This absolute performance is 38.5 times higher than that of IBM BlueGene/P although the frequency is 8.5times slower than BlueGene/P. LHS has higher scalability than ALPU in the performance per frequency. Since these results are obtained with partially on loaded linear searching on old Pentium®4, performance gap will increase using state of art CPU. Therefore, LHS is more suitable for larger parallel systems. The discussions for adopting proposed method to state of art processors and systems are also presented.

key words: *network interface, MPI, message passing, queue management, low latency communication, scalability*

1. Introduction

Development of future-generation supercomputers whose 1Exa Flops peak performance requires hundreds of thousands of nodes or more is underway in the U.S. and Japan. In these systems, keeping scalability is the key to performance. On massively parallel computers, MPI (Message Passing Interface) is widely used for parallel programming. In typical MPI implementations, the Eager Protocol and the Rendezvous Protocol are used selectively. A sender sends messages without respect to the state of a receiver in the Eager Protocol. In this case, performance degradation occurs when the sequence of message receptions differs from the receiver's expectation. The message that reached the receiving side before calling receiving function such as MPI_Irecv( ) is called an unexpected message.

In Brightwell's work [4], a significant portion of the messages received by the NAS Parallel Benchmarks (NPB) are unexpected messages. Some of them have long maximum queue depth or average queue depth which increases with increasing number of nodes. Table 1 shows the average search depth of MPI unexpected messages queue on applications reported by Brightwell [4], [20]. The last column for 16K nodes is estimated from the reported values with linear approximation.

For traversing a long unexpected message queue on massively parallel machines, it is necessary to keep both low-latency and high-scalability. For example, if the number of unexpected messages increases in MPI over QsNET-II [1] using Tports, it is reported that delay time increases in proportion to it [3]. It consumes 100ns per the depth. Minimum MPI communication latency over QsNET-II using Tports is only 1.3 micro seconds [1]. The latency is far better than that of NIC(Network Interface Card)s for Ethernet with fat layer for TCP/IP. If MPI over QsNET-II is implemented using hashing algorithm, the minimum latency becomes over 4micro second [3]. This shows that hashing spoils the most important minimum latency like over Ethernet. Lower latency of NICs for PC cluster such as QsNET-II using Tports contributes for the scalability of many applications. However, a message at 3,837th in the depth of messages queue may be searched in NPB FT on a massively parallel machine with 16K nodes. In this case, the searching time with QsNET-II using Tports for a message will be 383.7 micro seconds. Balaji [8] measured unexpected message overhead up to 4,096th in the depth on IBM BlueGene/P which is one of brand new massively parallel computers. The result shows that it takes huge 7,000 micro seconds message matching time for only one message reception. This overhead is 17 times bigger than QsNET-II. Most of all communication latencies in these cases are time for searching message queue. Therefore, we have to prepare the long communication latency on huge massively parallel supercomputers or rewriting such applications with this feature from scratch. In order to minimize rewriting such applications, scalable low-latency MPI communication architecture is needed for the supercomputers in the future.

One of the prior works for this problem is ALPU [5].

Table 1 Average search depth of MPI unexpected messages queue on applications with bad scalability (*: estimated with linear approximation).

# of nodes	32	64	128	16,384
NPB FT [4]	5	10	27	3,837*
ITS [20]	7	13	21	2,344*

Manuscript received December 26, 2012.
Manuscript revised April 19, 2013.

[†]The author is with Toshiba Corporation, Kawasaki-shi, 212–8582 Japan.
[††]The author is with Hitachi Information and Communication Engineering, Ltd., Yokohama-shi, 220–6122 Japan.
a) E-mail: noboru.tanabe@toshiba.co.jp

DOI: 10.1587/transinf.E96.D.2536

It is a hardware acceleration unit for MPI queue processing. However, ALPU does not have enough scalability because of the needs of many logic gates.

The objective of this paper is acceleration of MPI by means of hardware supports on the NIC with higher scalability without spoiling minimum latency. NICs such as QsNET-II [1], SeaStar [2] and ALPU [5] are off-loading MPI queue processing on a slow embedded CPU or limited hardwired logic. On the other hand, we try to off-load MPI queue processing more partially than ALPU to a small accelerator on a NIC.

The contribution of this paper is showing the collaboration of a small accelerator on NIC and a cache based high-performance CPU with prefetching functions. The authors propose LHS on NIC in this paper. This is a simple hardwired message receiving function to separate key data block for fast and reasonable queue searching.

The following section identifies the scalability problems on MPI queue processing. The proposed hardware supports for MPI are described in Sect. 3. We also describe performance evaluation with DIMMnet-2 employing our proposed mechanisms in Sect. 4. The discussions for adopting proposed method to state of art processors and systems are presented in Sect. 5. Related works are described in Sect. 6. The paper wraps up with conclusions and future work in Sect. 7.

2. Scalable Low Latency Communication

In this section, in considering hardware supports for MPI, we describe scalability problems on MPI queue processing.

2.1 Unexpected Message Searching Latency

In MPI, data are transferred between function peers with the same rank, communicator and key for a sender and a receiver. On the other hand, in a parallel program, it is generally difficult to guarantee the order in which data are received from multiple nodes with low overhead. When a message sent by the Eager Protocol, the sequence of message receptions may differ from the receiver's expectation. If a function specifying location to receive in a receiver is not yet executed, the message is buffered in a system buffer named unexpected message queue.

Processing of MPI, such as handling of message buffer, is off-loaded into the firmware on NIC in QsNET-II [1]. In the message buffer searching with it, the penalty of 100ns per accumulated message is required. This is an excellent value at which the low-latency communication is achieved in the state with a small number of accumulated messages. On the other hand, a lot of messages comparable to the number of nodes stay in the unexpected message queue in some case. For example, FT and IS of NAS Parallel Benchmarks (NPB) is so according to the research by Brightwell [4]. He reported that in FT the average search length for the unexpected message queue at 128 nodes is 20% or more than the number of nodes, and that it increases in proportion to the number of nodes. He also reported that 50% of the receiving is unexpected receiving. Therefore, the communication delay increases in proportion to the number of nodes, because of an increase in time to search the message with the corresponding key. Many accumulated messages tend to be generated on a large-scale parallel system. In this situation, there is a problem in that the delay time of QsNET-II becomes larger than that of the system without off-loading MPI processing.

2.2 Hardware Cost

In Underwood's work [5], a hardware acceleration unit named ALPU for MPI queue processing is proposed. ALPU has high performance for searching message buffer. However, the number of ALPUs on FPGA is limited to a relatively small number such as 128 or 256. Moreover, when unexpected message queue length exceeds the number of ALPUs, the latency for traversing a long unexpected message queue becomes larger. When performing an application like NPB FT, the number of accumulated messages is expected to increase according to the number of processing nodes. Therefore, a method of preventing or reducing dramatic performance degradation on a large-scale system with a reasonable amount of hardware is desired.

3. Limited-Length Head Separation: LHS

The main target system of this paper is assumed the massively parallel systems consisting of a very large number of nodes, such as Exa FLOPS machines. Latency is very important system feature in such systems in order to keep scalability. The NICs in such system are connected to CPU with high performance channel or connected on the same chip. Therefore, those are accessed with small latency compared with the NICs plugged into the general-purpose-I/O such as PCI express. If the NIC is implemented on the same chip as CPU, NIC is a location that can be accessed with low latency from the CPU than the main memory.

We propose Limited-length Head Separation (LHS) for such system in order to reduce latency by a short message. Short messages must be executed by Eager protocol and latency for searching data in a system message buffer of a receiver. The basic concept of LHS is shown in Fig. 1. Figure 1 The basic concept of Limitedlength Head Separation (LHS).

In LHS, the head part of a message with limited length

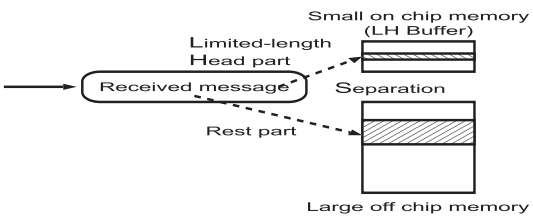

Fig. 1 The basic concept of Limited-length Head Separation (LHS).

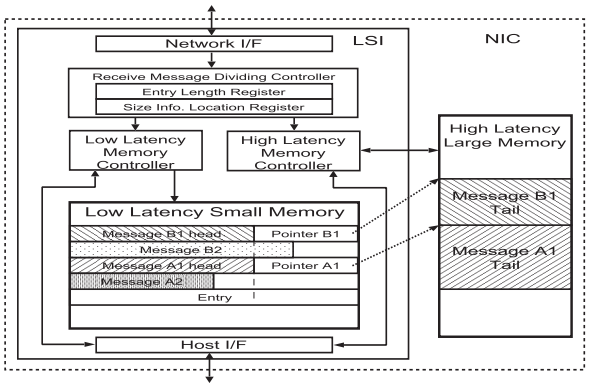

Fig. 2 Basic configuration of LHS.

is stored in a low-latency small memory (LH buffer). This can hold envelopes of MPI systems and short payloads. The rest of the message is stored in a large-capacity memory.

The basic configuration of LHS is shown in Fig. 2. In LHS, a length of a received message is checked by Receive Message Dividing Controller. When a length of a received message is shorter than the threshold length, the message is stored in an LH buffer. When a length of a received message is longer than the threshold, the message is divided into two parts. The former part of the message and a pointer to the latter part of the message are stored in an LH buffer by a low-latency memory controller. The latter part of the message is stored in a high-latency large memory by a high-latency memory controller. We recommend using PIO (Programmed I/O) for the access of the first part which is stored in LH buffer whose entry size is the same as a cache-line. The reason is the setup-overhead of DMA is larger. In the setting of the situation at the beginning of this chapter, the delay to access the LH buffer on the NIC is less than the delay to access the main memory.

The location of the size information on the message header can be varied by a register (Size Information Location Register). It is convenient for various header formats defined in various MPI implementations or future communication protocols.

The threshold length can be varied by a register (Entry Length Register). To accommodate a wide variety of cache line size, Entry Length Register is recommended for optional components of the LHS. If the cache line size of CPU is 128 bytes, there is no problem even if the entry length is fixed to 128 bytes.

In original MPI implementation, unexpected message buffer is searched from top to end in reading envelopes one by one. Each envelope is usually 16 bytes in length. Three integers are compared for key matching on host CPU. Before key matching, envelope is copied to main memory of host PC. Host CPU fetches envelope with the three integers to cache memory using Read-instructions, then host CPU compares specified key and key in the envelope.

In using LHS case, host CPU fetches the head part of a message with envelope from low-latency LH buffer to

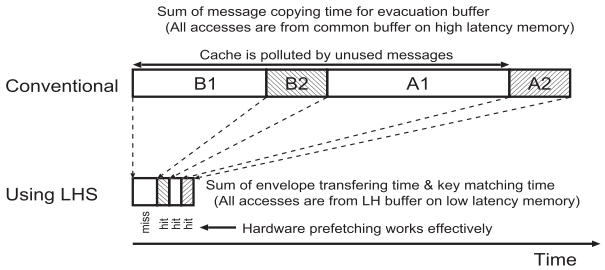

Fig. 3 Latency reduction of unexpected messages by LHS.

cache memory of host CPU. This operation is done by one of the means shown as below.

(a) Read-instructions of host CPU
(b) Hardware prefetching invoked by (a)
(c) Software prefetching using prefetching instructions

In Brightwell's work [4], a significant portion of the messages received by the NAS Parallel Benchmarks (NPB) are unexpected messages. Some of them have long maximum queue length. Latency reduction of unexpected messages by LHS is shown in Fig. 3.

Messages are sent to a receiver in the order of B1, B2, A1 and A2. If MPI_IRecv( ) is posted at first, whole messages of B1, B2 and A1 buffered in the MPI system buffer (unexpected message queue). Buffered messages are read and written from/to high-latency large memory before reading A2. However, since all transfers of envelope parts are sent from low-latency LH buffer, communication using LHS can be significantly improved. If entry length is selected appropriately, MPI envelopes are stored in the same or follow-on cache line in order of arrival. This situation makes hardware prefetching of CPU effective in searching unexpected message queue. This feature reduces searching time per stored message in unexpected message queue dramatically.

Here, large-capacity buffer can be kept in pin-down region in main memory or it can be kept in off-chip memory such as SO-DIMM on DIMMnet-2 [12], [14]. Therefore, in principle, this mechanism can be implemented by firmware such as Myrinet or QsNET-II [1]. Since fast buffer is supposed to be implemented mainly in on-chip memory in a network controller LSI, it has low latency compared with large-capacity buffer. However, its capacity is small.

4. Evaluation

This section presents the results of the evaluation for LHS and discusses the effects of it on MPI.

4.1 Evaluation Environment

The main target of this study is the Exa FLOPS machine environment. It is an environment which has a NIC near the CPU. DIMMnet is used as the experimental environment of this chapter. This is a NIC plugged into a DIMM slot which is far closer to the CPU compared with NIC

Table 2 Environment for accessing DIMMnet-2.

CPU	Pentium®4 2.6GHz, L2=512KByte
Chipset	VIA VT8751A
Memory	PC-1600 DDR-SDRAM 512MBx1
OS	RedHat8 (Kernel 2.4.27)
Compiler	gcc 3.3.5 (compile option: -Wall)

Table 3 Environment for emulating DIMMnet-3.

CPU	Pentium[1]® D 840 3.2GHz, L2=2MB
Chipset	Intel® 955X
Memory	PC2-3200 DDR2 SDRAM 512MB x2
OS	openSUSE (Kernel 2.6.22.5 with setting for Single CPU)
Compiler	GCC 4.2.1 (compile option: -Wall)

on PCI express. Therefore, DIMMnet is a better experimental environment compared with NIC on PCI express. We have developed three kinds of prototypes of DIMMnet which are NICs plugged into SDR DIMM, DDR DIMM and DDR2 DIMM. These prototypes are called DIMMnet-1 [11], DIMMnet-2 [12]–[16] and DIMMnet-3 respectively. DIMMnet can realize higher bandwidth and lower latency than NICs plugged into general I/O slot of each generation. In order to adopt newer DIMM slots with higher frequency, on-board DIMM of DIMMnet-2 and DIMMnet-3 cannot be accessed from host CPU directly. DIMMnet-2 and DIMMnet-3 have Read Window, which is a kind of vector register accessed from host CPU directly. The host CPU has to access on-board DIMM of DIMMnet-2 and DIMMnet-3 using vector load commands such as VL command before accessing the Read Window. VL command copies a data block from on-board DIMM to Read Window. DIMMnet-2 is a working prototype with LHS proposed and investigated in this paper.

Table 2 and Table 3 show the evaluation environments used for the evaluations in this section. Because of evaluations using real prototypes of FPGA-version DIMMnet-2, the frequency of operation is low, including that of the host side. Therefore, performance of the hardware portion and the software portion must be higher in the ASIC based implementation or newer FPGA based implementation. The evaluation in Sects. 4.2 and 4.3 uses environment for higher frequency DDR2 based DIMMnet-3 which is a successor of DIMMnet-2. The logic of FPGA for DIMMnet-3 is under development. Since memory part of LH buffer acts as main memory from host, a part of main memory on the environment shown in Table 3 can emulate a LHS on DIMMnet-3 accurately.

4.2 Unexpected Message Searching Latency with LHS

We have measured the unexpected message searching latency. We have evaluated the following five conditions:

[1]Pentium and Intel are trademarks of Intel Corporation in US and other countries.

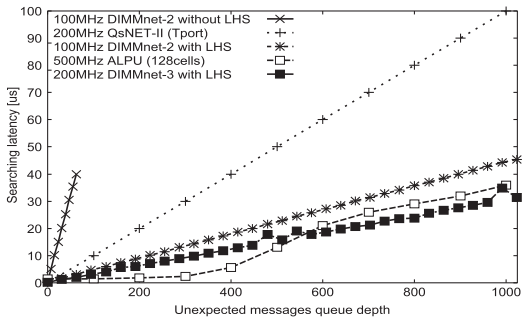

Fig. 4 Comparison of unexpected message searching latency of NICs.

(1) Without LHS on 100MHz working DIMMnet-2 prototype: Reading envelopes from SO-DIMM with Read Window and Vector Load command (VL)
(2) On 200MHz QsNET-II with Tport reported by Underwood (i.e. 100ns per message for unexpected message queue) [3]
(3) ALPU with 128 cells reported by Underwood [5]
(4) 100MHz working DIMMnet-2 prototype with LHS
(5) 200MHz virtually emulated DIMMnet-3 with LHS: Emulated by mapping virtual LH buffer on DDR2 (PC2-3200) based main memory which has the same latency of LH buffer on DIMMnet-3 plugged into a PC2-3200 slot. This is an actual time measurement on a real PC without working DIMMnet-3.

In this experiment, the number of entries on an LH buffer is 1024 on DIMMnet-2. All entries of LH buffer are 0-cleared in advance except for the location of x-th envelope. Unexpected message buffer is searched from top to end in reading envelopes one by one. Each envelope is 16 bytes in length.

Three integers are compared for emulating key matching on host PC. Before key matching, envelope is copied to main memory of host PC. Bodies of unwanted messages received earlier are not evacuated from unexpected message queue in this experiment.

The results of this evaluation are shown in Fig. 4. Horizontal axis means the buffer depth from top of unexpected message queue to envelope of specified message in MPI_Irecv( ).

Without LHS on DIMMnet-2 is the slowest. In this case, an envelope is read using a VL command from SO-DIMM via Read Window which is a kind of vector register mapped on user space. VL command is not suitable for small data block such as envelope. Using LHS on DIMMnet-2, acceleration ratio of time for searching unexpected message queue is 14.3. 44.4ns per message for searching unexpected message queue is needed on DIMMnet-2 with LHS. DIMMnet-2 with LHS operating at 100MHz can search unexpected message queue 2.3 times faster than QsNET-II operating at 200MHz. This is 38.5 times higher than that of IBM BlueGene/P [8] although the frequency is 8.5times slower than BlueGene/P. 500MHz ALPU with 128 cells always a little faster than 100MHz DIMMnet-2 with LHS. When the depth on 500MHz ALPU

with 128 cells becomes more than 600, it takes more time than 200MHz DIMMnet-3. That is, even though LHS operates at lower frequency, its scalability is higher than ALPU with respect to performance.

Memory accesses with vector load commands have high latency on DIMMnet-2. This is a reason why the improvement of latency in searching long unexpected message queue is significant with LHS. The above-mentioned performances of QsNET-II and ALPU are achieved by higher frequencies. These frequencies are twice and five times than that of DIMMnet-2 by implementation based on ASIC, respectively. Therefore, if a comparison is performed at the same frequency, the effect of LHS would be bigger than what was illustrated by Fig. 4.

Very large memory on next version hardware named DIMMnet-3 will achieve good scalability for large node number and avoid latency increase. SO-DIMM access latency using vector load command is about 600ns. Main memory and LH buffer access latency is about 100ns. Therefore, acceleration ratio caused by the special memory organization is about 6. This cannot entirely explain the acceleration ratio of 14.3. There is other reason why the acceleration ratio of 14.3 is achieved. 44.4ns is smaller than access latency for main memory. This shows that LHS increases cache hit ratio. The program used does not have explicit software prefetching. Therefore, LHS makes cache and hardware prefetching of host CPU effective.

Moreover, we can use IPUSH in addition to LHS. IPUSH can reduce the number of traverses of the unexpected messages queue by grouping. Because reduction in the number of traverses is executed at 14.3 times higher speed with IPUSH and LHS, significantly low-latency unexpected queue traversing can be achieved on DIMMnet-2.

4.3 Effect by Setting of Prefetching in Searching Unexpected Message with LHS

We have measured the unexpected message searching latency with the effect of hardware prefetcher (HWP) and software prefetching instructions (SWP). "Hardware prefetcher" means hardware-based implicit data prefetching mechanism of many CPU such as Intel Pentium 4/D. "Software prefetch instructions" mean Intel's SSE prefetch instructions (e.g. prefetchnta, prefetcht0, and so on). We have evaluated on DIMMnet-2 in the following six conditions:

(1) HWP is off, SWP is off.
(2) HWP is on, SWP is off.
(3) HWP is on, SWP is on. (Offset of Software prefetching address is 0byte. Timing of prefetching is late.)
(4) HWP is off, SWP is on. (Offset of Software prefetching address is 0byte. Timing of prefetching is late.)
(5) HWP is off, SWP is on. (Offset of Software prefetching address is 256byte.)
(6) HWP is off, SWP is on. (Offset of Software prefetching address is 512byte.)

We have evaluated on environment shown in Table 3 for DIMMnet-3 in the following four conditions:

(1) Cache attribute of LH buffer is Uncacheable.
(2) HWP is off, SWP is off. Cache attribute of LH buffer is Writeback.
(3) HWP is on. Cache attribute of LH buffer is Writeback.
(4) HWP is on. Cache attribute of LH buffer is Writeback. All access for LH buffer is forced on cache by deleting cache-line flushing instructions.

The results of this evaluation for DIMMnet-2 and DIMMnet-3 are shown in Fig. 5 and Fig. 6 respectively.

For DIMMnet-2, condition (2) which uses only hardware prefetcher was best. The performances vary according to the setting of prefetching.

For DIMMnet-3, performances in condition (2) and (3) do not have big difference from that in condition (4) in which all access for LH buffer is forced on cache. LHS acts very effectively in order to enhance cache hit ratio of host CPU.

On the environments using DIMMnet-2 and DIMMnet-3, positive effects on cache of host processor by LHS are confirmed with the experiments in this section. Since DIMMnet-2 does not have enough host interface bandwidth, sensitivity of the setting of prefetching is bigger.

On the other hand, since DIMMnet-3 has enough host interface bandwidth, sensitivity of the setting of prefetching is smaller. The difference between cache-enabling or not is very big. The bandwidth of dual channel 200MHz DDR2 host interface is 6.4GB/s for DIMMnet-3 with dual channel DIMM interfaces. This is enough for MPI queue processing with LHS and a single core of Pentium D 840

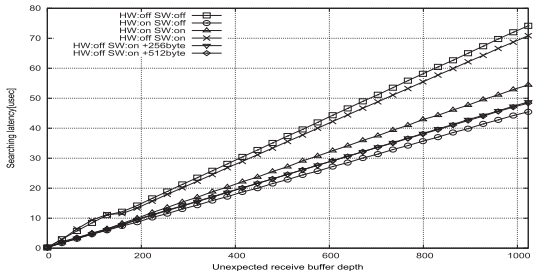

Fig. 5 Effect by several settings of prefetching in searching unexpected message on DIMMnet-2.

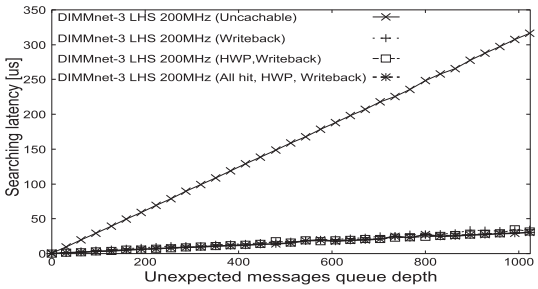

Fig. 6 Effect by several settings of prefetching in searching unexpected message on DIMMnet-3 (emulated by main memory of PC shown in Table 3).

(3.2GHz). This shows that farther acceleration on frequency of LHS does not contribute on performance well. The performance bottleneck on DIMMnet-2 environment was DIMMnet-2. On the other hand, the performance bottleneck on DIMMnet-3 environment becomes host CPU. Using other commercial compiler or multi CPU cores for the acceleration of software part is a subject for future work.

5. Adopting to State of Art Systems

Since main parts of this research are done in the middle of the decade of the 2000s, the environments of evaluations are a little old. The circumstances surrounding have been changed. In this section, the discussions for adopting proposed method to state of art processors and systems are presented.

5.1 Multi-Core or Many-Core Environments

Multi-core or many-core environments are widely used recently and in near future. Multi-core or many-core environments are better than single core environments on our partial on loading strategy for queue processing. Figure 7 shows concept of an enhanced NIC for host CPU with multicore.

Multi-cores or many-cores can execute liner searching for queue with shorter depth in parallel, if NIC can execute grouping messages. Grouping -using the rank which is bound to a core and a sub-queue make the depth shorter. We can implement simple hardwired hashing functions to select a sub-queue based on the rank of MPI message.

In addition, production cycle of CPU is shorter than that of NIC. Therefore, replacing by newer CPU can improve performance of queue processing easier than improving hardware or firmware on NIC.

5.2 One Sided Communication and PGAS Environments

One sided communication and PGAS (Partitioned Global Address Space) environments will be used more widely in near future. In some cases, the problem to be solved with proposed method can be avoided with one sided communication semantics. However, changing semantics means rewriting and re-optimizing applications. MPI is widely used because it is standardized and foremost de-fact standard on HPC environments. Though many PGAS languages have been proposed, there is no de-fact standard such as MPI. Most of scientists do not want to concentrate into rewriting and re-optimizing their MPI applications but their scientific matters. In these circumstances surrounding, MPI applications are going to remain as de-fact standard for at least a few years.

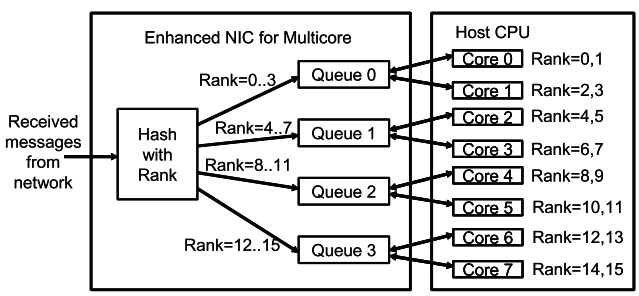

Fig. 7 The concept of an enhanced NIC for host CPU with multicore.

Some of one sided communication and PGAS environments are implemented as a translator to MPI program, since MPI can be used on most of all HPC platforms. For example, ARMCI (Aggregate Remote Memory Copy Interface) [21], [22] and Global Arrays (GA) [23]–[25] are ported based on such strategy. In such case, user cannot optimize translated MPI programs. Fine grained MPI communications are easily generated on such system with large number of processors. Therefore, there are some situations which need proposed method even if it is one sided communication and PGAS based environment.

In some PGAS environments such as XcalableMP [26], [27], MPI is used with PGAS environments in order to optimize performance. In such environment, performance centric parts will be written in MPI preferably. There is a possibility that the proposed method helps performance of PGAS based environment.

5.3 Exa FLOPS Class Systems

Exa FLOPS class systems in the end of the decade of the 2010s have been actively discussed for a few years in US and Japan. In such systems, they say that hundreds of thousand socket processors will be used. The need of strong scaling situation and low latency communication between many processors will be increasing. In US, IAA (Institute for Advanced Architectures and Algorithms) is one of the developers of Exa FLOPS machine supported by Sandia National Laboratory and OakRedge National Laboratory. It lists the improving MPI latency and scalability of MPI message throughput as a focused area [28] in their web page and presentation [29]. ALPU for MPI queue processing is written as a starting point of this focused area. The problems to be solved by ALPU and our LHS are the same. In the context of Exa FLOPS machine, accelerating scalable MPI queue processing with these technologies is important.

5.4 Application Trends

Not only the conventional HPC applications, but also the graph processing is having higher attention recently. Although Top500 lists have been had high attention from HPC users, character of graph applications is far different from Linpack used for Top500. Graph500 lists are established several years ago in order to cover such new kind of demands for HPC. This kind of application easily generates fine grained communications on systems with large number of processors. Communication patterns of them are not uniform because of the character of each graph. In the real graph processing, for example PageRank for web sites, some sites such as Google's toppage have huge number of

connections. This means the existence of communication hot-spots which need help of proposed method.

6. Related Work

In order to reduce latency of sending short messages, several trials have been performed with commercial NICs such as QsNET-II and Infiniband. They use the firmware running on on-chip CPU with lower performance than host.

QsNET-II [1] supports a quick sending mechanism for short messages called STEN, which is similar to BOTF [8], [14], [16] in DIMMnet-2. However, in such systems, receiver has no performance enhancement techniques such as LHS proposed here. The only exception is the firmware of on-chip CPU called "Thread Processor" for off-loading the matching process on QsNET-II, but the improvement is limited. If the number of unexpected messages increases in MPI for QsNET-II using Tports, it is reported that delay time increases in proportion to it. About 100ns per message for searching unexpected message queue is needed on QsNET and QsNET-II even if their processing speeds are different [3]. This operation may need external memory access per message on them. On the other hand, LHS increases cache hit ratio for searching unexpected message queue. DIMMnet-2 with LHS operating at 100MHz can search unexpected message queue 2.3 times faster than QsNET-II operating at 200MHz.

Hashing is also an effective method in the case only to implement scaleable searching unexpected message queue with less hardware. However, hashing degrades communication latency under almost all situations in MPI communications. Minimum communication latency is one of the most important items in a catalogue of an NIC, thus the degradation of the latency may lead to commercial damage. On the other hand, applications and situations in which time of searching unexpected message queue causes troubles are limited. Therefore, we suppose that implementation of low latency under any situation should take highest priority. The improvement of time of searching unexpected message queue should take the second priority.

As regards research on acceleration for MPI queue processing with hardwired logic, two approaches can be distinguished. In Underwood's work [5], a hardware acceleration unit named ALPU for MPI queue processing is proposed. However, ALPUs consume many logic gates by using many hardware comparators. The number of ALPUs on FPGA is limited to a relatively small number such as 128 or 256. Moreover, when unexpected queue length exceeds the number of ALPUs, the latency for traversing long unexpected message queue becomes larger. The K-computer and Exa scale machine require about tens of thousands of nodes or more. In some applications, queue length grows to 20% of the number of nodes. Therefore, ALPU does not have enough scalability. In this approach the searching process is done by NIC. On the other hand, searching process with LHS is done by host computer. Superior performance is expected for our approach using high performance host CPU based on multi-core technology. Our LHS-based method is different from theirs because envelope needed for searching is sent to host at low latency without processing on NIC. ALPU consists of random logic gates. Main part of LHS (i.e. LH buffer) does not consist of them but a simple on-chip memory. Hardware cost of ALPU [30] is far higher than LHS. For a large system, LHS can enlarge the number of entries or the number of groups in unexpected message queues more easily than with ALPU.

Ternary Content Addressable Memory (TCAM) is a hardware that is mainly used for the Table look-up in TCP / IP based communication. The structure of TCAM is similar to that of ALPU. If this is used for matching in MPI, there is a possibility that problems caused by insufficient capacity of ALPU may be relaxed. However, the strategy to achieve higher capacity by replacing the ALPU to the TCAM is not for massively parallel machines. The reason is high cost, power consumption (Example: 23W [31] for 100K words, 128bit key length) and an increase of latency for off-chip access even on good situation. In addition, since the structure is simpler than ALPU, the configuration using TCAM can not guarantee FIFO feature between pair of RANK by itself. Therefore, to achieve the semantics of MPI in the configuration using TCAM, it is necessary to add a software tasks. As a result, the overhead such as present in TCP / IP is added.

In Tanabe's work [15], a hardware acceleration unit named IPUSH for MPI queue processing is proposed. IPUSH realizes multiple unexpected queues associated with the key information such as RANK. This will reduce the length of each unexpected message queue to be traversed. However, in the DIMMnet-2 case, the latency for each access to traverse unexpected message queue on SO-DIMM seemed larger than that for conventional -PCI-based NICs. Our LHS-based method is different from theirs because envelope needed for searching is sent to host not from SO-DIMM but from low latency on-chip LH buffer.

MX [32] supports a variable threshold message length to switch between the Rendezvous protocol and the Eager protocol. In addition, if the Rendezvous protocol is selected, the information stored in unexpected queue is only matching information of MPI. For this reason, readers might think that LHS with optional registers to implement the variable threshold is similar to the MX. On the other hand, LHS is aimed for further acceleration of short messages transported with Eager protocol which is chosen by MX etc. MX's variable-threshold function and Rendezvous protocol which sends only matching information at first can not achieve the objective of speeding up of short messages. In addition, the behavior is different from each other in many ways. For example, the splitting place, the storing location of the second part of the message, and the ability to include payload in the first part of the message are different.

Header splitting is an existing method that is easily confused with LHS. It is used in TCP Offload Engine (TOE) [6] and Intel I/O Acceleration Technology (IOAT) [7]. Differences between LHS and Header Splitting

are the following five points.

(1) Header Splitting just divides a header and its payload. On the other hand, LHS can accelerate short messaging with dividing the message including a part of its payload.
(2) LHS creatively uses fast memory and slow memory according to the location in the message.
(3) In LHS, dividing packets are conducted with hardwired logic based on a simple rule instead of software in a CPU on an NIC.
(4) Header Splitting is an off-loading approach. On the other hand, LHS is an on-loading approach. Comparing with LHS is processed in a host CPU instead of a CPU on an NIC.
(5) Header splitting involves splitting and storing the header and payload of a transport layer. LHS accelerates handling of envelope in MPI, which is a kind of header of the application layer processed by a message passing library.

There have been no previous reports of research on the effect of separation of an envelope in MPI and a payload for short message which is used with Eager Protocol. This research clarified for the first time the effect of the separation in searching a buffer of MPI is very high. Our paper first describes the right way of using hardware or software, an NIC or a host and kinds of memory. It first describes dividing packet in a different layer of Header Splitting. Moreover, our paper first describes its quantitative evaluations about scalability of searching queue by separating a packet into two parts.

7. Conclusions and Future Work

In this paper, network interface architecture for scalable message queue processing named LHS is presented. LHS can be applied to NICs without extended memory such as DIMMnet.

The acceleration ratio for searching message queue by LHS on DIMMnet-2 is 14.3. The acceleration ratio caused by the special memory organization of DIMMnet-2 is about 6. The remaining acceleration is caused by the effects of caching and prefetching. The absolute searching performance of DIMMnet-2 with LHS is 2.3 times higher than that of QsNET-II [1]. It is 38.5 times higher than that of IBM BlueGene/P [8] although the frequency is 8.5times slower than BlueGene/P.

ALPU [5] which is a hardware accelerator for searching message buffer has better searching performance than QsNET-II. Even though LHS operates on at lower frequency, its scalability is higher than ALPU in with respect of a performance. We confirmed the best setting of prefetching for LHS is using hardware prefetcher without software prefetching.

LHS has higher scalability than ALPU in the performance per frequency. However, the searching latency for unexpected messages queue with 1000 messages is about 30 microseconds using LHS without grouping. This latency is too large to be hidden by overlapping computation and communication. Therefore, one order of magnitude improvement from this latency is desired in order to keep better application scalability. Further improvement of this latency using several LH buffers which hold parts of unexpected messages classified properly with hardware supports is promising. For example, coupling LHS and a mechanism of grouping messages into multiple queues with hardware such as IPUSH [14] can overcome defects of linear searching. The investigation about the combination of these mechanisms is the future work.

It is needed to evaluate with a large number of nodes in order to realize such situations with real applications. Therefore, benchmarking overall application speedup by LHS with NPB etc. on a small system is not very meaningful for evaluating scalability. Meaningful evaluations of scalability with real applications can not be executed on our small experimental environment. It must be a future work.

Acknowledgments

This work is supported by the Ministry of Internal Affairs and Communications (Soumu-sho).

References

[1] J. Beecroft, D. Addison, D. Hewson, M. McLaren, F. Petrini, and D. Roweth, “Quadrics QsNet II: pushing the limit of the design of high-performance networks for supercomputers,” IEEE Micro, pp.34–47, July 2005.
[2] R. Brightwell, K.T. Pedretti, K.D. Underwood, and T. Hudson, “SeaStar interconnect: balanced bandwidth for scalable performance,” IEEE Micro, vol.26, no.3, pp.41–57, May 2006.
[3] K.D. Underwood and R. Brightwell, “The impact of MPI queue usage on message latency,” Int. Conf. Parallel Process. (ICPP’04), pp.152–160, 2004.
[4] R. Brightwell and K.D. Underwood, “An analysis of NIC resource usage for offloading MPI,” 18th International Parallel and Distributed Processing Symposium (IPDPS’04), 2004.
[5] K.D. Underwood, K.S. Hemmert, A. Rodrigues, R. Murphy, and R. Brightwell, “A hardware acceleration unit for MPI queue processing,” 19th International Parallel and Distributed Processing Symposium (IPDPS’05), 2005.
[6] Chelsio Communications, “Time for TOE - The Benefits of 10 Gbps TCP Offload,” A Chelsio Communications White Paper http://www.chelsio.com/toe_value_prop.html
[7] K. Lauritzen, T. Sawicki, T. Stachura, and C.E. Wilson, “Intel(R) I/O acceleration technology improves network performance, reliability and efficiently,” Technology@Intel Magazine, 2005.
[8] P. Balaji, A. Chan, W. Gropp, R. Thakur, and E. Lusk, “Non-data-communication overheads in MPI: analysis on bluegene/P,” EuroPVM/MPI2008, LNCS 5205, pp.13–22, Sept. 2008.
[9] N. Tanabe, J. Yamamoto, H. Nishi, T. Kudoh, Y. Hamada, H. Nakajo, and H. Amano, “MEMOnet: Network interface plugged into a memory slot,” IEEE Int. Conf. Cluster Computing (CLUSTER2000), pp.17–26, Nov. 2000.
[10] N. Tanabe, Y. Hamada, H. Nakajo, H. Imashiro, J. Yamamoto, T. Kudoh, and H. Amano, “A low latency high bandwidth network interface prototype for PC cluster,” Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA’02), pp.87–94, 2002.

[11] N. Tanabe, Y. Hamada, A. Mitsuhashi, J. Yamamoto, H. Imashiro, T. Kudoh, H. Nakajo, and H. Amano, "Prototyping on using a DIMM slot as a high-performance I/O interface," Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'03), pp.108–116, 2003.
[12] N. Tanabe, M. Nakatake, H. Hakozaki, Y. Dohi, H. Nakajo, and H. Amano, "A new memory module for COTS-based personal supercomputing," Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04), pp.40–48, 2004.
[13] T. Araki, T. Mori, J. Kanai, N. Tanabe, H. Amano, M. Namiki, and H. Nakajo, "Design and implementation of MPI-2 communication library on DIMMnet-2," IPSJ-SIG 2006-ARC-167, pp.49–54 (In Japanese), Feb. 2006.
[14] N. Tanabe, A. Kitamura, T. Miyashiro, Y. Miyabe, T. Izawa, Y. Hamada, H. Nakajo, and H. Amano, "Preliminary evaluations of a FPGA-based-prototype of DIMMnet-2 network interface," Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'05), pp.119–127, 2005.
[15] N. Tanabe, A. Kitamura, T. Miyashiro, Y. Miyabe, T. Araki, Z. Luo, H. Nakajo, and H. Amano, "Hardware support for MPI in DIMMnet-2 network interface," Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'06), pp.73–80, 2006.
[16] A. Kitamura, Y. Miyabe, T. Miyashiro, N. Tanabe, H. Nakajo, and H. Amano, "Performance evaluation on low-latency communication mechanism of DIMMnet-2," The IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN'07), #551-067, Feb. 2007.
[17] InfiniBand Trade Association, http://www.infinibandta.org/
[18] Mellanox Technologies, http://www.mellanox.com/
[19] M. Koop, S. Sur, Q. Gao, and D.K. Panda, "High performance MPI design using unreliable datagram for ultra-scale infiniBand clusters," 21st Int. ACM Conf. Supercomput., June 2007.
[20] R. Brightwell, S. Goudy, and K.D. Underwood, "An preliminary analysis of MPI queue characteristics of several applications," Int. Conf. Parallel Process. (ICPP'05), pp.175–183, 2005.
[21] J. Nieplocha, V. Tipparaju, M. Krishnan, and D. Panda, "High performance remote memory access comunications: the ARMCI approach," International J. High Performance Comput. Appl., vol.20(2), pp.233–253, 2006.
[22] ARMCI Webpage. http://www.emsl.pnl.gov/docs/parsoft/armci/.
[23] J. Nieplocha, B. Palmer, V. Tipparaju, M. Krishnan, H. Trease, and E. Apra, "Advances, applications and performance of the global arrays shared memory programming toolkit," Int. J. High Performance Comput. Appl., vol.20, no.2, pp.203–231, 2006.
[24] J. Nieplocha, M. Krishnan, B. Palmer, V. Tipparaju, and J. Ju, "The global arrays user's manual," 2006.
[25] Global Arrays Webpage. http://www.emsl.pnl.gov/docs/global/
[26] J. Lee and M. Sato, "Implementation and performance evaluation of xcalableMP: A parallel programming language for distributed memory systems," 39th Int. Conf. Parallel Process. Workshops (ICPPW10), pp.413–420, 2010.
[27] XcalableMP webpage. http://www.xcalablemp.org/
[28] IAA, "Focus area," http://iaa.sandia.gov/focus-areas/index.html
[29] IAA, "Exascale computing and the institute for advanced architectures and algorithms (IAA)," http://www.hpcuserforum.com/presentations/Norfolk/Sandia%20IAA.hpcuser.ppt
[30] N. Tanabe, A. Kitamura, Y. Miyabe, T. Miyashiro, H. Amano, A. Ohta, and H. Nakajo, "Improvement of scalability of message passing system with hardware acceleration," IPSJ-SIG 2008-ARC-177, pp.181–186 (In Japanese), 2008.
[31] B. Vamanan, G. Voskuilen, and T.N. Vijaykumar, "EffiCuts: optimizing packet classification for memory and throughput," ACM SIGCOMM 2010, pp.207–218, 2010.
[32] Myricom, "Myrinet express (MX): A high-performance, low-level, message-passing interface for myrinet," http://www.myricom.com/scs/MX/doc/mx.pdf, 2006.

Noboru Tanabe received the B.S. and M.S. degrees in Electrical Information Engineering from Yokohama National University in 1985 and 1987, respectively. During 1998-2001, he stayed in RWCP Tsukuba Research Center. He is with Toshiba corporation since 1987. He received the best paper award in HPC Asia'09 and Yamashita Memorial Research Award. He is a member of IPSJ. (Doctor of Engineering).

Atsushi Ohta was received the B.E. and M.E. degree in Computer Science from Tokyo University of Agriculture and Technology in 2007 and 2008. He is now working in Hitachi Information and Communication Engineering, Ltd. His research interests are embedded systems and mobile computing. He received the best paper award in IPSJ Embedded Systems Symposium. He is a member of IPSJ.