Performance Analysis of High-speed Multi-Rail Data Transmission on SMP based Architecture

Akihiro Tsutsui*, Kazuaki Obana+, and Makoto Takizawa*
*NTT Network Innovation Laboratories +NTT Cyber Solutions Laboratories
1-1 Hikarinooka
Yokosuka, Kanagawa 239-0847 Japan

***Abstract*- We present some ideas for designing software architecture that optimizes the resource arrangement of the operating system and application programs in a PC so that Nx10Gbps near line-rate multi-rail data transmission can be achieved using most popular SMP-based high-end PC architecture. We evaluated the effectiveness of the computation resource arrangement policies we have presented by experimentation. These are necessary because current PC architecture does not make maximum use of multi-rail data links without tactful management of processors, memories and peripheral control. Multi-rail data transmission and multi-wavelength networking devices will be the key to creating next generation terabit LANs.**

Keywords: multi rail, multi lane, multi core, multi processor, terabit LAN, multi wavelength, SMP, PC

I. INTRODUCTION

The performance of computers and interface technologies has increased significantly in recent years. In local area networks (LAN), 1Gpbs transmission speeds have been achieved using Ethernet technology. Currently, 10Gbps Ethernet devises (interface cards and switches) are on the market [1]. Furthermore, the specifications for 100Gbps Ethernet are currently being standardized [2]. Wide area networks (WANs) offering gigabit access services are available for consumers. In fact, there is now a real possibility that terabit data-links will be applied to junction networks by using wavelength multiplexing. From the standpoint of network applications, ultra high speed data transmission technologies are necessary in the following fields:

1. Large-scale internet servers (cluster computers, etc.)
2. Large-volume data storage (over petabyte order)
3. High-end video processing applications, such as super high definition image transmission (multi-image video conferencing, digital cinema, etc.)

To keep up with the requirement for such high-speed data transmission, the data-transmission speeds that LANs can handle will need to be significantly increased. In addition, the data-transmission speeds that other device specific networks and data transmission systems, such as Infiniband, FiberChannel, and eSATA, can handle will also have to be increased [3][4][5].

In general, speeding up data transmission is initially achieved by applying data parallelism, which increases the amount of bit-data transmitted at once. Next, to simplify the data transport system, the speed is increased by using asynchronous serial transmission technology. The next stage is to bundle together multiple hi-speed serial data links. This method is called "Multi Rail" or "Multi Lane. The method is used to further speed up device specific data transmission systems.

The "Multi Lane/Multi Rail" technique has received a lot of attention recently. The technique consists of creating parallel "rails" through every aspect of an end-system [6][7]. For example, processing in the multiple processor cores, generating multiple application data flows, streaming over multiple-lanes, and connecting multiple NICs via a parallel interconnect. Multi-rail data transmission and multi-wavelength networking devices will be the key to creating next generation terabit LANs [8].

The specifications for 100Gbps Ethernet will also use this technique [2]. It will be part of the fundamental design of next-generation terabit-LAN systems. Research had been conducted on the "Multi Rail" method and its use had been demonstrated with multiple wave length data links [9][10]. The problem is that optimization of PC architecture is necessary because the current architecture does not make maximum use of multi-rail data links. For transmitting 10Gbps data streams, effective computation resource management is required in operating systems and application programs in PCs. We have developed and demonstrated the effectiveness of an optimum resource arrangement. Using our arrangement it is possible to achieve Nx10Gbps near line-rate multi-rail data transmission using most popular SMP-based high-end PC architecture.

This paper contains design ideas of software architecture for PCs so that multiple network interfaces (Multi Rails) are effectively used. We experimented using Intel processor based high-end PCs and multiple 10Gbit Ethernet interfaces in order to estimate the performance of transport and computation loads. Details of these experiments are included in this paper. Finally, this paper contains a discussion on effective resource (processing units) arrangements that are necessary to make maximum use of multi-rail data-links.

II. MULTI PROCESSOR AND MULTI RAIL CONCEPT

Current high-speed PC architecture is based on multi processor architecture that uses a multi-core CPU. Two major CPU vendors, Intel and AMD, adopt such architecture [11][12]. Basic PC architecture for high-performance CPUs is shown in Figure 1 for Intel's Xeon and AMD's Opteron. As shown in the figure, the manufacturers' design concepts for multi processor systems are quite different. Intel's architecture is called

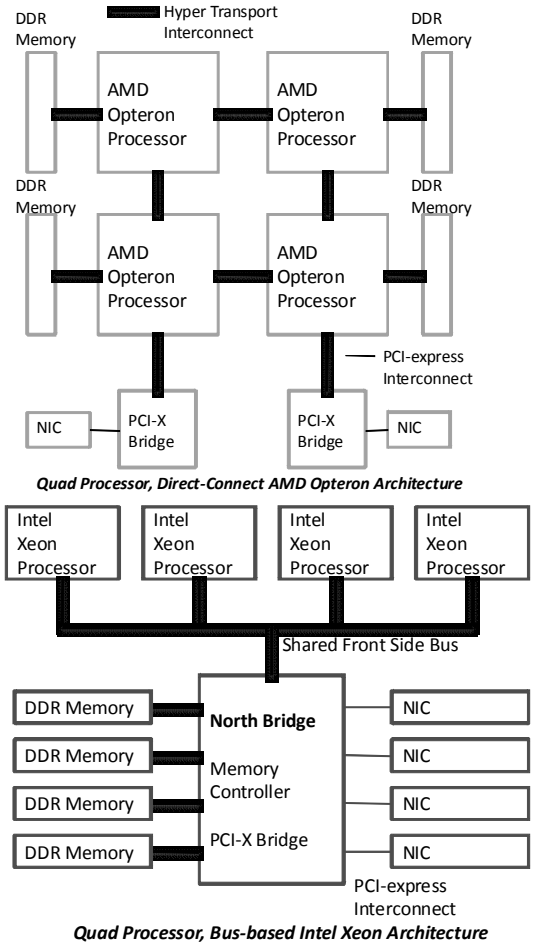

Figure 1. AMD Opteron and Intel Xeon Architecture.

SMP (Symmetric Multiple Processor). AMD's architecture is called NUMA (Non-Uniform Memory Access, Non-Uniform Memory Architecture).

In the SMP architecture, which is bus-based, a shared front-side bus can only be used by one single processor at a time. The architecture consists of a shared memory controller located on the North-Bridge (NB). The various memory banks are connected to the NB by the memory bus and the main memory access latency for every processor is identical. The architecture consists of a shared PCI-express Bridge, located on the NB, to which all the PCI-express devices are connected.

In the NUMA architecture, which is direct-connection-based, neighboring processors are directly connected by a point-to-point hyper-transport bus. This architecture features distributed memory controller architecture. Each processor has an on-chip integrated memory controller and p PCI-express Bridge to which PCI-express devices are connected. Therefore, any PCI-express based IO device is physically bound to a particular processor. Each processor has access to a local memory bank connected via a hyper-transport bus. A processor can access the memory banks of the other processors via the hyper-transport bus, though with higher access latency compared to its local memory bank.

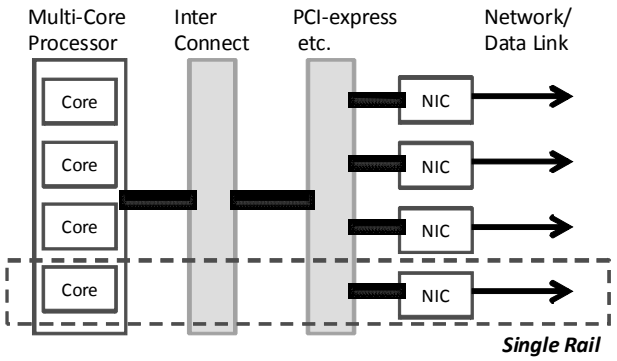

Figure 2. Multi-Rail Data Transmission Concept.

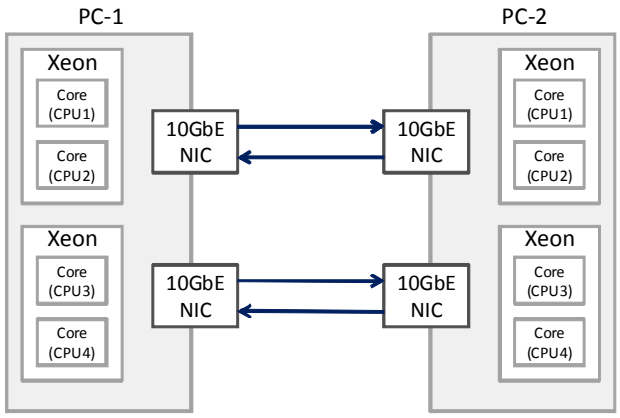

Figure 3. Experimental Model using 2 multi-multi-core PCs and 10GbE NIC

Figure 2 shows the concept of a multi-rail system. Each rail consists of a processor core that communicates with an NIC (Network Interface Card) via a dedicated interconnect, such as PCI-express. Each NIC connects to each data-link. To use a multi-rail system with current PC architecture and high-speed NICs, Intel's or AMD's high-end servers consist of multi multi-core processors and 10Gbit Ethernet NICs are available. This concept can be extended to the fundamental design of next-generation terabit LAN architecture.

III. Experimental Test bed

We used the following PC and NICs for our experiments. The overview of the test bed is shown in Fig.3.

- Two Linux PCs (IBM x3500) connected by two 10Gbps Ethernet Interfaces back-to-back
- Each PC had two Dual Core, Dual-Processor 3.0GHz Intel Xeon processors with 2GB Memory and two Myrinet 10Gbps Ethernet NICs.
- Linux OS Kernel 2.6.18-1 for SMP configuration
- Myrinet Driver Version 1.4.1 and Firmware 1.4.30

Linux OS has a function that executes an associate process (thread) and interrupts operation between peripheral devices and a dedicated CPU core in multiprocessor systems. For example, a thread can be executed by assignment to a CPU core using the "Thread Affinity" feature via the sched_setaffinity() system call. An interrupt operation can be executed by masking a

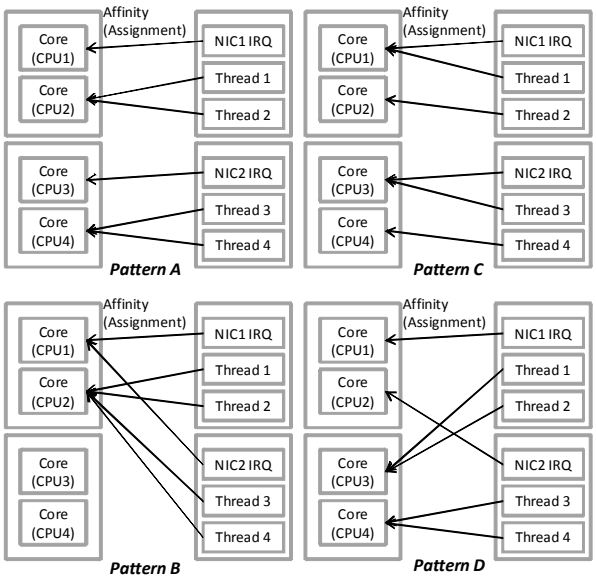

Figure 4. Resource Assignment Patterns for the Experiments.

/proc/interrupts configuration parameter in order to assign an interrupt signal from a device, such as a NIC. In addition, for NUMA architecture (AMD's CPU), there is association between a CPU core and a memory block is possible. However, SMP-based Intel's CPU cannot use this feature. SMP-based systems are designed so as to access memory space with uniform latency. These configurations affect system performance of multi-processor computing [13][14].

For multi-rail data transmission, NUMA architecture has an advantage over SMP-based systems. This is because it can construct a rail that consists of tightly associated CPU cores, memory, and an NIC using its direct-connected-based architecture. Therefore, the latency between a CPU core, memory and a NIC can be small without disturbance, such as by a bus arbiter. Huge band data transmission in excess of 10Gbps using the multi rail method is suitable for server-class high-end PCs based on AMD architecture [8]. However, if actual applications on a PC requiring not only data transmission but also data processing are an important job, then the total performance of the PC must be taken into consideration and the choice of architecture must be made on a case-by-case basis. We have attempted to effectively assign a CPU core to threads and NICs using SMP-based architecture.

IV. Experimental Results and Discussions

Our experiment methodology is as follows.

- Interface Bonding by Linux OS
- One NIC One Thread 10Gbps data transmission
- Two NICs Two Thread 20Gbps data transmission total

A. Interface Bonding by Linux OS

Current Linux OS has a function that enables interface bonding. It can bundle multiple physical network interfaces and provide them as one virtual interface for users. The purposes of interface bonding are to enable fault tolerance and load sharing of the network interfaces. In Linux, seven bonding modes are defined and implemented in a kernel. In this experiment, we bound two 10Gbit NICs and made one virtual interface using this function. Then, we measured UDP and TCP throughput of one and two data streams through the virtual interface. The experimental results show that the modes designed mainly for fault tolerance did not have a high throughput: single flow did not exceed 10Gbps. Therefore, to achieve near 20Gbps with 2 NICs, we think that a multi-rail technique is necessary.

TABLE I
THROUGHPUTS USING INTERFACE BONDING

Mode/Transport		Throughput 1-flow (MB/sec)	Throughput 2-flows(MB/sec)
Mode 0	UDP	5847	7946
	TCP	7927	9786
Mode 1	UDP	7568	9571
	TCP	8117	9515
Mode 2	UDP	7385	9527
	TCP	8005	9518
Mode 3	UDP	4696	4958
	TCP	6704	8898
Mode 5	UDP	7540	9441
	TCP	7902	8930

*Mode4/6 could not be available, 9KB fram

B. One NIC One Thread 10Gbps data transmission

On the testbed shown in Figure 3, using one 10Gbps NIC, TCP and UDP data was transmitted between sender and receiver PCs. We measured the transmission performance and load average of each CPU. The experimental results are shown in Tables 2 and 3. In these tables, CPU1, CPU2 and CPU3 stand for the CPU core in a Xeon Processor. CPU1 and CPU2 are on the same processor. CPU3 is the core of the other processor. The Interrupt signal from the NIC is always handled by CPU1.

TABLE II
UDP/TCP PERFORMANCE 2-NICS 2-THREADS

Message Size / Tx (thread affinity)		Rx (thread affinity) CPU1	CPU2	CPU3
Tx 64Bytes	CPU1	281.3 MB/sec	344.4	299.1
		0.63 Load av. Tx	0.78	0.81
		0.56 Load av. Rx	0.55	1.35
	CPU2	261.3	363.1	301.9
		0.83	0.87	0.82
		1.00	0.86	1.56
	CPU3	196.6	197.6	197.5
		0.83	0.86	0.85
		0.95	0.59	0.67
Tx 1500B	CPU1	2040.4	4811.6	4606.6
		0.87	0.92	0.87
		0.65	0.80	0.83
	CPU2	2018.7	4739.7	4741.1
		1.50	0.88	0.88
		0.84	0.68	0.81
	CPU3	2057.3	3710.0	3646.3
		0.88	0.85	0.87
		0.60	0.52	0.67
Tx 4KB	CPU1	5614.3	8297.9	8336.7
		0.93	0.89	0.88
		0.66	0.69	0.70
	CPU2	5609.2	8451.0	8408.7
		0.85	0.87	0.92

		0.68	0.74	0.92
	CPU3	5547.4	7141.0	7189.8
		0.87	0.87	0.85
		0.54	0.49	0.68
Tx	CPU1	8427.3	8700.8	8764.5
		0.42	0.92	0.95
9KB		0.61	0.71	0.83
	CPU2	8451.9	9309.8	9175.3
		0.90	0.86	0.87
		0.86	0.69	0.66
	CPU3	7714.0	7778.7	7701.7
		0.92	0.87	0.87
		0.67	0.56	0.67

TABLE III
TCP PERFORMANCE 1-NIC 1-THREAD

Message Size / Tx (thread affinity)		Rx (thread affinity) CPU1	CPU2	CPU3
Tx	CPU1	86.4	398.7	230.4
		0.00	0.00	0.04
104Bytes		0.20	0.47	1.02
	CPU2	85.9	399.8	230.8
		0.01	0.10	0.03
		0.57	0.72	1.12
	CPU3	82.8	399.7	193.4
		0.09	0.00	0.00
		0.77	0.40	0.65
Tx	CPU1	268.2	3168.4	1544.1
		0.00	0.04	0.01
512B		0.75	0.91	1.94
	CPU2	260.2	3106.9	1477.9
		0.05	0.18	0.10
		1.74	1.50	2.01
	CPU3	268.9	3182.4	1575.8
		0.10	0.13	0.12
		1.44	1.12	1.43
Tx	CPU1	268.2	3168.4	1544.1
		0.00	0.04	0.01
1500B		0.75	0.91	1.94
	CPU2	260.2	3160.9	1477.9
		0.05	0.18	0.10
		1.74	1.50	2.01
	CPU3	268.9	3182.4	1575.8
		0.10	0.13	0.12
		1.44	1.12	1.43
Tx	CPU1	7793.6	7631.2	7999.6
		0.65	0.87	0.92
4KB		0.55	0.68	0.85
	CPU2	8520.8	8302.7	8533.68
		0.94	0.98	0.96
		0.95	0.91	0.87
	CPU3	5991.8	5973.1	6001.1
		0.98	0.96	0.98
		0.76	0.75	0.91
Tx	CPU1	8586.3	8578.4	8703.4
		0.64	0.86	0.92
9KB		0.46	0.62	0.70
	CPU2	9139.0	9167.7	9046.9
		0.97	0.96	0.95
		0.67	0.57	0.63
	CPU3	7586.2	7567.9	7530.6
		0.98	0.96	0.98
		0.50	0.54	0.69

The experimental results revealed the following.

- On both sender and receiver PCs, assigning a data communicating thread and interrupts from NIC to the different cores on the same processor achieve maximal throughput.
- When transmitted message size is small, assigning the thread and interrupts to the same core causes significant degradation of throughput on the receiver side.
- When transmitted message size is large, assigning the thread and interrupts to the same core causes significant degradation of throughput on the sender side.

C. Two NICs Two Threads 2x10Gbps data transmission

For two Xeon processors (four cores total), we assigned interrupt signal handling and thread execution in several patterns. Figure 4 shows the model patterns of this experiment. In this experiment, we used four data streams and the corresponding four threads in order to make clear the influence of the thread and interrupt signal assignment patterns. Each two streams were assigned one NIC on the sender and receiver PCs. Similar experiments using two data streams with each NIC handling one stream showed the same trend. However, transmission throughput is better than with four streams because of the extra load arisen by executing threads.

TABLE IV
UDP/TCP PERFORMANCE 2-NICS 2-THREADS

Msg. Size	Pattern / Transport		TH-1	TH-2	TH-3	TH-4	Total
64B	A	UDP	93	94	94	90	371
	B	UDP	89	90	96	89	365
	C	UDP	87	140	84	140	452
	D	UDP	82	79	77	80	319
104B	A	TCP	138	138	139	142	559
	B	TCP	135	136	139	139	551
	C	TCP	138	137	140	140	556
	D	TCP	170	173	171	171	686
512B	A	TCP	821	599	702	510	2633
	B	TCP	686	546	782	956	2972
	C	TCP	992	778	476	733	2980
	D	TCP	510	686	610	617	2424
1500B	A	UDP	1372	1375	1583	1581	5912
		TCP	1732	1899	1236	1050	5919
	B	UDP	1318	1396	1163	1184	5062
		TCP	1646	1255	1973	1297	6174
	C	UDP	1809	2552	1784	747	6893
		TCP	1643	869	2033	1661	6208
	D	UDP	1367	1344	1323	1339	5374
		TCP	1920	2102	964	1084	6072
4KB	A	UDP	2904	2899	2801	2887	11492
		TCP	5001	4167	896	1126	11192
	B	UDP	1329	3898	591	2674	8493
		TCP	2100	2348	2163	2046	8659
	C	UDP	2529	3517	2445	3426	11917
		TCP	4489	5113	712	696	11012
	D	UDP	2583	2575	2551	2625	10336
		TCP	2305	2374	2345	2327	9352
9KB	A	UDP	3345	3319	3259	3516	13442
		TCP	3246	2991	4363	4198	14800
	B	UDP	2507	2304	2301	2348	9460
		TCP	2228	1985	2487	2630	9332
	C	UDP	3572	4108	3103	1557	12340
		TCP	4479	4896	2315	2241	13933
	D	UDP	2948	2870	2904	2899	11622
		TCP	3149	3022	3085	2963	12221

*TH=Thread, Value is MB/sec

The experimental results revealed the following.

- When the message size is large, the pattern-B degrades the total throughput.
- When the message size is small, the effect of the interrupts becomes significant.

Our results show that in order to achieve high throughput, it is better to assign data streaming thread and handling interrupts from an NIC to a different core on the same processer. Small messages increase the number of interrupts from an NIC. These trends represent the general case. When the number of data streaming threads increases, the load on the CPU handling both interrupts and the threads becomes high. Therefore, to achieve high throughput on a multi-rail system, optimal load balancing and resource allocation is required. Such balancing and allocation must take into account he number of threads, interrupt signals, and rails

V. Conclusion and Future Work

In this paper, we have presented a basic idea for designing software architecture for PCs so that the PC makes effective use of multi-rail data links in terms of computation resource assignment. We mainly investigated SMP-based high-end PC architecture running on Intel processors. We found that when handling multiple 10Gbps (and over) streams, we must consider effective resource management of PCs, such as thread/process assignment, interrupt handling, memory assignment, etc. Moreover, this kind of management is required for both the sender and receiver side of data streaming. Therefore, some kind of signaling protocol that exchanges the status of the sender and the receiver will be required before each data streaming begins. Developing and testing such a protocol will form part of our future work. Session Initiation Protocol (SIP) is a candidate protocol.

References

[1] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[2] http://www.ieee802.org/3/

[3] http://www.infinibandta.org/home

[4] http://www.fibrechannel.org/

[5] http://www.serialata.org/

[6] J.Liu, A. Vishnu and D.K. Panda, "Building Multi-rail InfiniBand Clusters: MPI Level Design and Performance Evaluation," in Super Computing 2004.

[7] Y. Qian and A.Afsahi, "Efficient RDMA-based Multi-port Collectives on Multi-rail QsNet Clusters," in Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium, 2006.

[8] V. Vishwanath, T. Shimizu, M. Takizawa, K. Obana and J. Leigh, "Towards Terabit/s Systems: Performance Evaluation of Multi-Rail Systems,", in Proceedings of High-speed Networks Workshop 2007, 2007.

[9] M. Tomizawa, J. Yamawaku, Y.Takigawa, M. Koga, Y. Miyamoto, T. Morioka and K. Hagimoto, "Terabit-LAN with optical virtual concatenation for grid applications with super computers," in Proceedings of OFC2005, paper OthG6, 2005.

[10] A. Hirano, L. Renambot et al., "The first functional demonstration of optical virtual concatenation as a technique for achieving Terabit networking," in Future Generation Computing Systems Vol. 22 pp.876-883, 2006.

[11] http://www.intel.com/

[12] http://www.amd.com/

[13] L. Chai, Q. Gao and D.K. Panda, "Understanding the Impact of Multi-Core Architecture in Cluster Computing and the Grid," in CCGrid2007.

[14] A. Foong, J. Fung, D. Newell, S. Abraham, P. Irelan and A. Lopez-Estrada, "Architectural Characterization of Processor Affinity in Network Processing Performance Analysis of Systems and Software," in Proceedings of ISPAS2000