Performance Comparison of Cray X1E, XT3, NEC SX8 and AMD/IB

Hongzhang Shan, Erich Strohmaier

Future Technology Group
Computational Research Division
Lawrence Berkeley National Laboratory
{hshan, estrohmaier@lbl.gov}

Abstract

In this paper, we compared the performance of Cray X1E, XT3 with other two platforms, NEC SX8 and an AMD Opteron cluster connected by Infiniband (AMD/IB), using both synthetic network benchmarks and a real scientific application, BeamBeam3D. In addition, we examine the relations between the synthetic benchmark results and application performance. We developed several models to predict the communication time for BeamBeam3D using the bandwidth and latency measured by synthetic benchmarks. The modeling results indicate that there is a big gap between the effective bandwidth on applications and the peak bandwidth measured by single-pair synthetic benchmarks mainly due to their inability to capture network contention. Using multi-pair communication results could significantly reduce this gap.

1. Introduction

Cray has recently introduced two new massively parallel processor systems, X1E and XT3. Cray X1E is an upgrade of the Cray X1 vector system by using advanced, faster dual-core processors to deliver better performance and greater density. XT3 is the third generation massively parallel processor system from Cray, building on the success of its predecessors, the Cray T3D™ and the Cray T3E™ systems, the Cray XT3 system is expected to bring astounding new levels of scalability and sustained application performance to high performance computing (HPC) [1]. In this paper, we are going to study its performance, together with other two HPC platforms, a vector platform from NEC (SX8) and a cluster built on AMD Opteron processors and Infiniband interconnects, using both synthetic network benchmarks and a scientific application (BeamBeam3D).

Using synthetic benchmarks instead of real applications to compare the performance has its own advantages and disadvantages. Synthetic benchmarks are usually simple and can be easily ported to different platforms while tuning and porting applications could be tedious and costly in terms of both manpower and system execution time. The behavior of synthetic programs is also not difficult to control and the results are much easier to understand. However, to relate the synthetic benchmark results to application performance is challenging and difficult as the synthetic programs may not be able to accurately reflect the complex behavior of real applications. In this paper, we examined the performance relationships between several network benchmarks and a scientific application, BeamBeam3D, using a linear modeling technology. The results indicate that the two popular network benchmarks, single-pair unidirectional benchmark (PingPong) and single-pair bi-directional benchmark (PingPing) could not precisely capture the network contention. Therefore, applying the results measured by the above two synthetic benchmarks in our model leads to significant underestimation of the application communication time. However, the multi-pair benchmarks, which allow several pair to communicate simultaneously, could capture the contention from node adapters. Using their results to predict application communication time is much more accurate.

The authors in [2] also used multi-pair benchmarks to compare the network performance of several different platforms. However, they have not tried to relate the benchmark results to application performance quantitatively. A closely related work to ours is from [3]. The authors investigated how well the simple metrics can represent the HPC application performance. The approach is to apply the synthetic

benchmark results, as the platform signature, together with application profiling data, into a framework to predict application performance. The major difference is that we depend on modeling technology and does not need any framework and application profiling.

The rest of the paper is organized as follows: The four platforms and their architectural highlights are described in Section 2. Section 3 will introduce the synthetic benchmarks and discuss their results. The performance of BeamBeam3D is analyzed in Section 4. Section 5 presents the modeling and the performance prediction results. Finally, in Section 6, we summarize our results.

2. Platforms

In addition to Cray X1E and XT3, we selected two other platforms, one is a vector platform SX8 from NEC and another is a commodity cluster built on AMD Opteron and Infiniband interconnects. Table 1 summarizes the main features of these four platforms.

Table 1: Architectural Highlights of Different Platforms

		CPU			Memory	Network		
Platform	**SMP**	**Type**	**Speed**	**Peak**	**Peak**	**Type**	**Topology**	**Peak[1]**
Cray X1E	4 (SMP)	X1E	1.13GHz	18GF/s	34GB/s	Custom	4D-Hyper cube	25.6 GB/s
NEC SX8	8	SX8	2GHz	16GF/s	64GB/s	IXS	Crossbar	16GB/s
Cray XT3	1	Opteron	2.4GHz	4.8GF/s	6.4GB/s	SeaStar	Torus	3.8GB/s
AMD/IB	2	Opteron	2.2GHz	4.4GF/s	6.4GB/s	Infini-Band	Fat-tree	1GB/s

3. Synthetic Benchmarks and Their Performance

We first select two popular network benchmarks for this study, the single-pair unidirectional bechmark (PingPong) and bi-directional benchmark (PingPing). The implementation of these two benchmarks in MPI is shown in Table 2. They should be very similar to other implementations in spirit. The performance of the unidirectional bandwidth on the four platforms is shown in Fig. 1. The results are obtained by selecting one processor from each of the two SMP nodes so that the benchmark measures the bandwidth on the network link between two SMP nodes. We depend on the job scheduler to assign the processes to the corresponding SMP nodes and we measure several times to avoid the case that the two nodes need to go through several network links to communicate with each other.

We can easily find that for large messages, the two vector platforms, X1E and SX8, deliver far superior performance than the two superscalar platforms, XT3 and AMD/IB. The performance order of these four platforms correlates well with the order of the peak derived from manufacture specifications, with Cray X1E the best and the AMD/IB the worst.

The ratio of the measured bi-directional bandwidth to unidirectional bandwidth is shown in Fig. 2. For networks that fully support bi-directional communication, in the ideal case, the bi-directional bandwidth is expected to double the performance of unidirectional bandwidth based on our calculation formula shown in Table 2. However, in reality, the ratio is affected by many factors and becomes much more complicated. Fully understanding the ratios need to dig into the details of network protocols and MPI implementations. Currently we are investigating this issue. For example, on the AMD/IB cluster, for smaller message sizes, the ratio is close 2. But for larger message sizes, the ratio drops to close to 1. On this cluster the Infiniband interconnect connected with the PCI bus which severely limits the available bandwidth when message size goes larger.

[1] The peak is unidirectional peak bandwidth on a network link

Table 2: The implementation of synthetic benchmarks in MPI

Single-Pair		Multi-Pair
Unidirectional:	**Bi-directional:**	

Clock (start)
For (I = 1; I < N; I++) {
If (myid == 0) {
MPI_Send();
MPI_Recv();
}
Else {
MPI_Recv()
MPI_Send(); }
}
Clock (end)
BW-Uni =
N*size/(end - start)
	Clock (start)
	For (I = 1; I < N; I++) {
	MPI_Irecv();
	MPI_Send();
	MPI_Wait();
	}
	Clock (end)
	BW-Bi =
	N *size/(end - start)
		Find pair:
		Pair.first = myid
		Pair.second = myid .XOR. (np -1)
		
		Measure:
		Clock (start)
		For (I = 1; I < N; I++) {
		Uni-directional test () or
		Bi-directional test ()
		}
		Clock (end)
		BW =
		N*message size/(end - start)

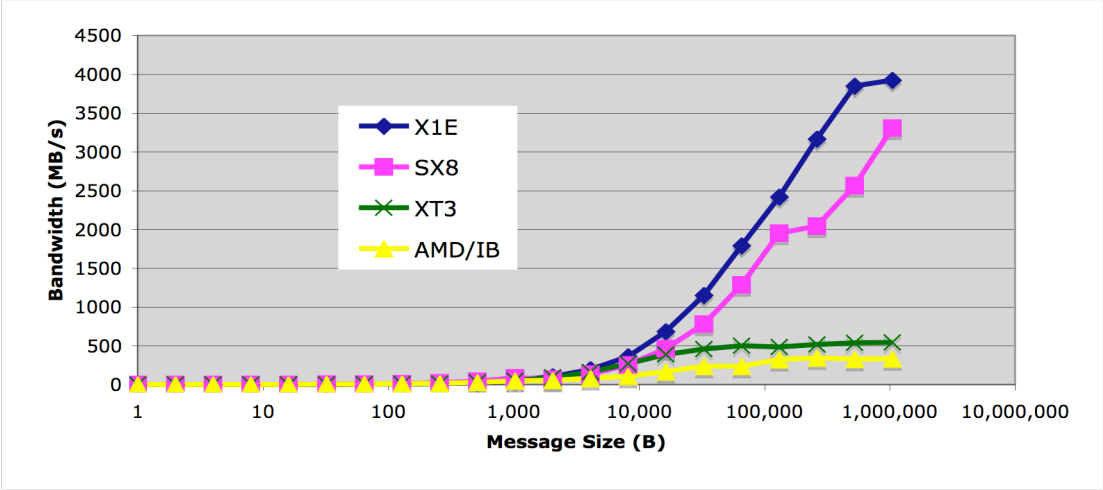

Fig. 1: The measured single-pair unidirectional bandwidth between SMP

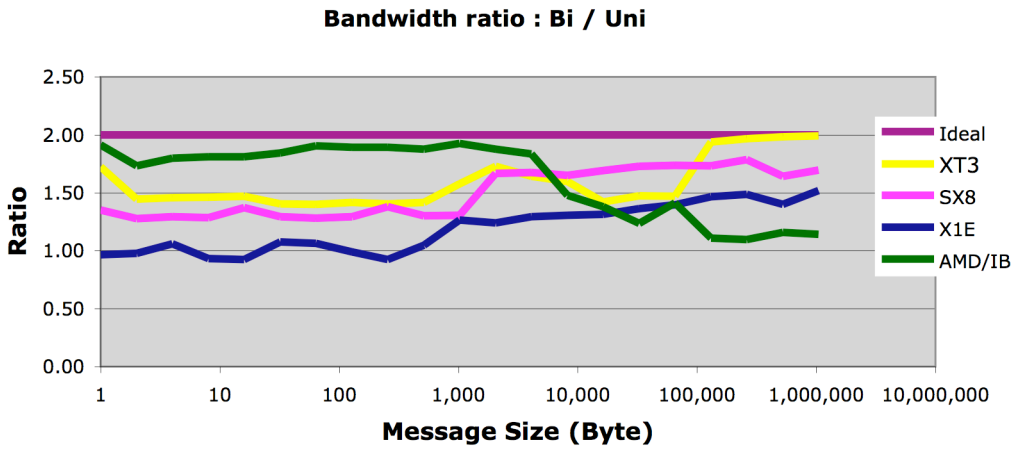

Fig. 2: The ratio of single-pair bidirectional bandwidth to unidirectional bandwidth

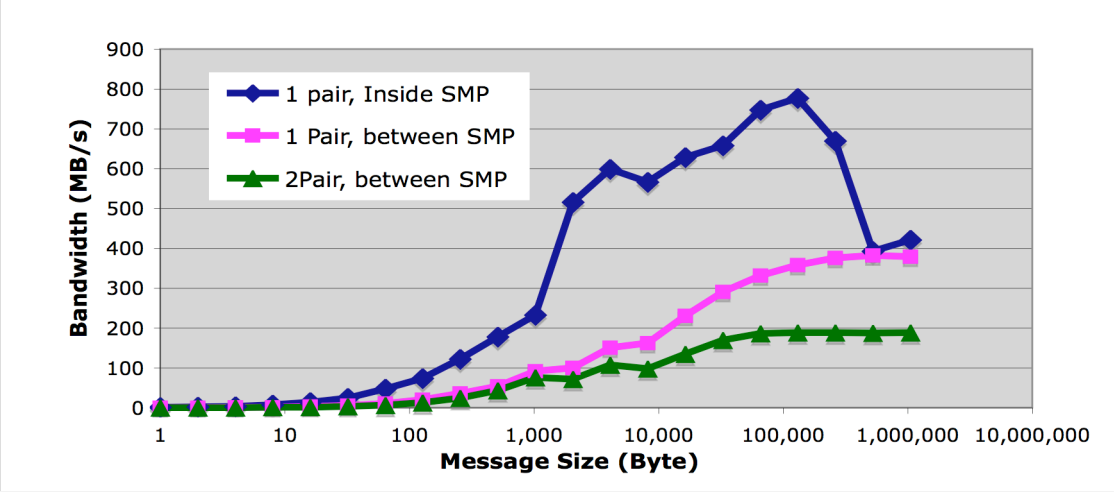

Fig. 3: The bidirectional bandwidth obtained by different benchmarks on the AMD/IB cluster

Though it's common to use single-pair benchmark results to compare performance, it is hard for the single-pair communication to capture the network contention. Therefore, we also developed the multi-pair benchmarks in which multi pairs communicate simultaneously. The implementation is shown in Table 2. It has two stages; the first stage is for each processor to find its partner. The algorithm to use could be different, but the partner should not stay in the same SMP node. The second stage is for multi pairs to communicate at the same time. In Fig. 3, we present the results on the AMD/IB cluster. The two-pair benchmark only achieves half of the bandwidth obtained by single-par benchmark. Recall that each SMP has two processors inside on this platform. The two processors inside a SMP node have to compete the communication resources and therefore cause significant contention on the node adapter, leading to the performance degradation. The results indicate that the multi-pair benchmarks could capture the network contention much better than the single-pair benchmarks. In addition, we also show the results for single-pair benchmarks inside SMP. Due to the higher memory bandwidth than network bandwidth, the performance inside a SMP node is also much better than between SMP nodes. Fig. 4 exhibits the performance difference between single-pair and eight pairs on all four platforms. We can see the significant bandwidth drop on X1E, SX8, and AMD/IB. However, XT3 is different. Its eight-pair performance is very close to single-pair performance. This is probably because the peak network link bandwidth is much higher than the node to network injection rate. At this scale, the network contention seems not an issue.

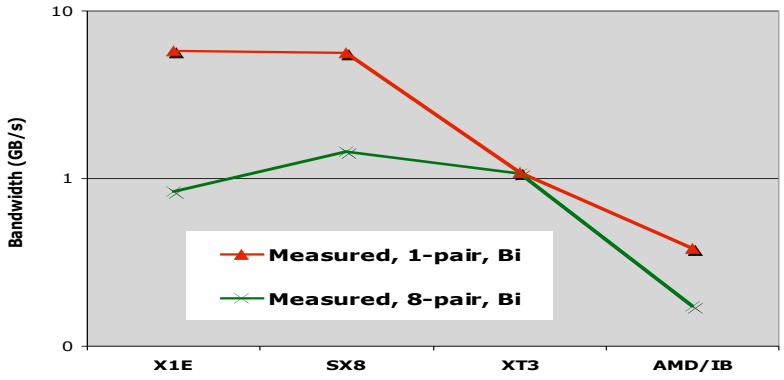

Fig. 4: The single and multi pair bi-directional bandwidth on all four platforms

4. BeamBeam3D and its Performance

BeamBeam3D models the colliding process of two counter-rotating charged particle beams moving at close to the speed of light in a circular accelerator. Accurate modeling of the beam-beam interaction is essential to maximizing the luminosity in existing colliders and critical for building the next generation colliders such as Large Hadron Collider (LHC). Under the paraxial approximation, for the colliding beams, which move in the opposite directions, the electric forces and the magnetic forces add up. The resulting beam-beam force produces a strongly nonlinear interaction that can significantly affect the motion of the charged particles. We use a multiple slice model to calculate the electromagnetic forces. In this model, each beam bunch is divided into a number of slices along the longitudinal direction in the moving frame of reference. Each slice contains nearly the same number of particles at different longitudinal locations z.

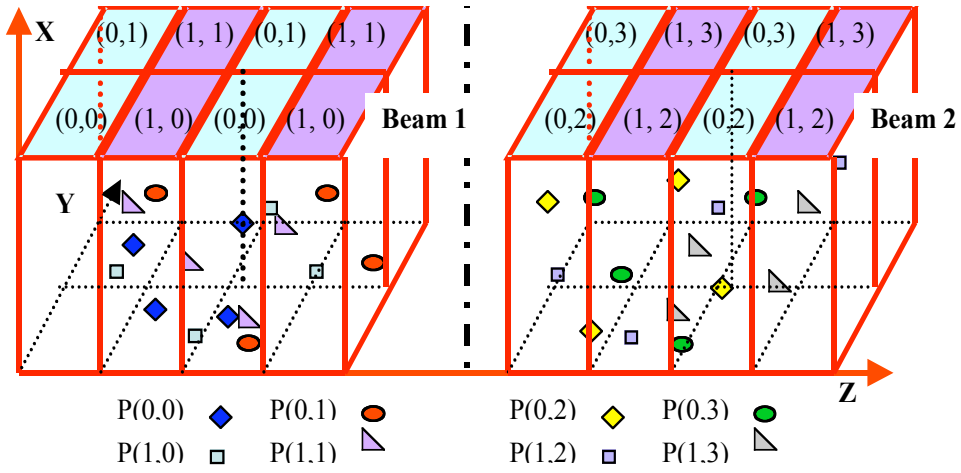

Fig. 5: Illustration of the particle-field decomposition for eight processors. Each beam has four slices. The eight processors are divided into two groups and each group forms a 2x2 processor grid.

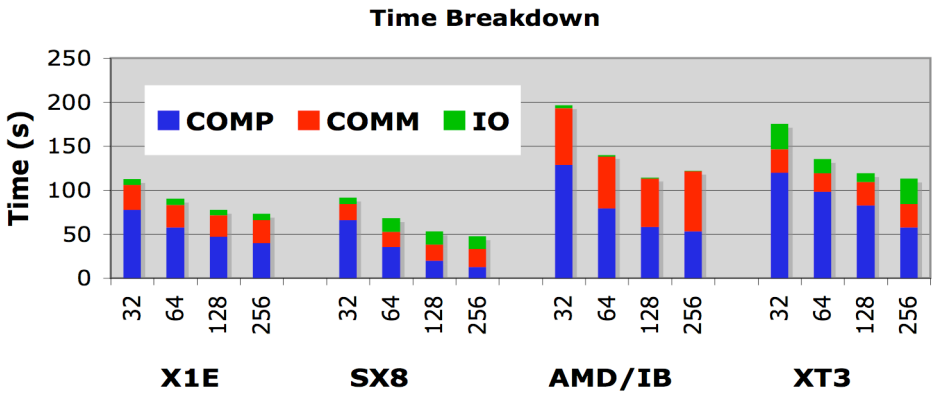

Fig. 6: The BeamBeam3D performance and time breakdown on all four platforms

There are two important domains in Beambeam3D, particle domain and field domain. The particle domain is the configuration space containing the charged particles, and the field domain is the space where the electric field is generated by the charged particles. BeamBeam3D adopts a novel particle-field decomposition approach to combine the advantages of both domain decomposition and particle decomposition, which has been demonstrated to deliver better performance than either particle decomposition or domain decomposition alone [4]. In this approach, each processor possesses the same number of particles and the same number of computational grid points, i.e., a spatial subdomain of the same size. Fig. 5 shows a schematic plot of the particle-field decomposition among eight processors. The total number of processors is divided into two groups, with each group responsible for one beam. We

furthermore divide each beam longitudinally (z-direction) into a specified number of slices (Nslice=4 in Fig. 5). The processors in each group are arranged logically into a two-dimensional array P_z*P_y to partition the computational domain, with each column (P_y) of the array containing a number of slices which are assigned to this column of processors cyclically along the longitudinal direction. This gives a good load balance of slices among different column processors. Within each column, the computational grid associated with each slice is decomposed uniformly among all the column processors. This allows us to parallelize the solution of the Poisson equation.

The time breakdowns of BeamBeam3D for different number of processors on these four platforms are displayed in Fig 6. The total running time has been divided into three categories: computation time, communication time, and I/O time. In terms of absolute performance, the SX8 performs best, followed by X1E. The two vector platforms deliver significantly better performance than the other two superscalar platforms. The I/O time on the AMD/IB cluster is the best. On other platforms, the I/O time is significantly worse. For example, the I/O time on SX8 takes almost 30% of the total running time when using 256 processors. This is mainly because the I/O frequency is very high. At the end of every turn, BeamBeam3D will output some data to multiple files. One way to reduce the I/O cost is to aggregate the data to reduce the output frequency. The experimental results show that this optimization works well on all platforms. Regarding the communication time, the AMD/IB cluster has the highest time. This is mainly due to its lowest network bandwidth. With the increase of the number of the processors, the communication time never decreases. This is also true on other platforms. The communication becomes scaling performance bottleneck and may take over 50% of the running time.

Table 3: Six most important communication phases and their main characteristics. Nx*Ny is the grid field size. Prow*Pcol is the processor grid as P_z*P_y mentioned earlier. Nslice is the number of slices a beam has been divided into (Prow <= Nslice).

Phase	Name	Pattern	Direction	Beam	Size [Byte]	# messages per turn
1:	Greenf2D	FFT Transpose	Column	Same	(Nx/Pcol+1)* (Ny/Pcol)*16*2	(Pcol-1)*(Nslice*2-1)
2a:	Guardsum2D	All-to-All Reduce	Column	Same	Nx*Ny/Pcol*8	(Pcol-1)*Nslice*Nslice
2b:	Guardsum2Drow	All-to-All Reduce	Row	Same	Nx*Ny/Pcol*8*I I = 1, Nslice/Prow	(Prow-1)*MIN(2*Prow, CEILING(Nslice/I, 1)*2-1)
3:	Fieldsolver2D	FFT Transpose	Column	Same	(Nx/Pcol+1)* (Ny/Pcol)*16	(Pcol-1)*Nslice* (Nslice+Prow-1)/Prow*2
4a:	Guardexch2Drow	All-to-All Broadcast	Row	Same	Nx*Ny/Pcol*8*I I = 1, Nslice/Prow	(Prow-1)*MIN(2*Prow, CEILING(Nslice/I, 1)*2-1)
4b:	Guardexch2D	All-to-All Broadcast	Column	Other	Nx*Ny/Pcol*8	Pcol*Nslice*Nslice

The BeamBeam3D's dominant communication phases include a parallel grid reduction, during which each processor accumulates its local portion of a global, discretized charge density through a reduction of all its local grid elements from all other processors, a broadcast of electro-magnetic field to all other processors, and a forward-backward 2D FFT. All these phases represent different types of all-to-all personalized communication (AAPC). Furthermore, for most of these phases the communication volume per process stays constant in a strong scaling scenario, which results very fast in a communication bound execution with flat execution times at best. Table 3 lists the six dominant communication phases along with their major characteristics. The transposes of the 2D FFTs (phases 1 and 3) take place within processors columns only, which in a typical case might contain 16 processors each. The parallel global grid reductions involve only processors within one beam and are organized in two phases communicating in column (phase 2a) or row (phase 2b) direction only. During the parallel global grid broadcast each processors has to send its part of the electromagnetic field to all members of the other beam. This is again organized in two

phases, one in row direction within one beam (phase 4a) and the second one within column direction between beams (phase 4b). The communication is implemented by point-to-point communication in MPI.

5. Modeling

In this section, we are going to examine whether the synthetic benchmark results can be used to predict the communication performance for BeamBeam3D. Note that the approach we used does not limit to BeamBeam3D and can be directly applied to other codes. We chose a simple latency (L) and bandwidth (B) model for the time needed to exchange a single message of size s: $t = L + s/B$ and decided to measure the effective values of latency and bandwidth using the synthetic micro-benchmarks discussed in Section 3. We then estimate the communication time for each phase by summing up the individual message transfer times along its critical path, which is determined by the processor with the maximal volume of data and number of messages to send. We investigate a series of four performance models, which differ from each other by the type of micro-benchmark used to determine L and B, by separating different levels in the network hierarchy, and finally by replacing the linear timing model with the actual message transfer times.

For our first model we choose latency and bandwidth values measured with single-pair benchmarks between two processors on different nodes (Model 1a: unidirectional, Model 1b: bi-directional). In Fig. 7, we show the ratios of measured and predicted total communication times based on our different models. Both Model 1a and Model 1b substantially underestimate the application communication time. In the worst case on XT3, the predicted time is ten times faster than the measured communication time of BeamBeam3D. For any model based on values measured with micro-benchmarks it is crucial that its parameters are chosen and measured in a fashion appropriate for the communication pattern in question. For all our communication phases (**Error! Reference source not found.**) all processors are communicating in pairs simultaneously. This implies that we cannot use latency and bandwidth numbers measured with single-pair benchmarks, but that we have to use benchmarks, which replicate this pattern by using a sufficient number of communicating pairs of processors. For SMP based systems we typically have all processors of one SMP communicate with processors in a second SMP, which represent the largest load on the network between them possible and actually generated during the execution of BeamBeam3D. If we use parameters based on multi-pair benchmark for long-range communication (inter-SMP), the model prediction in Fig. 7 (Model 2) improves substantially (XT3 has only one processor in a node, so Model 2 and 3 do not apply there), but the errors of the model are still large. Using the multi-layer model (Model 3) to differentiate inside SMP and between SMP performances does not help at all (on some other platforms not used in this study, Model 3 actually works quite well).

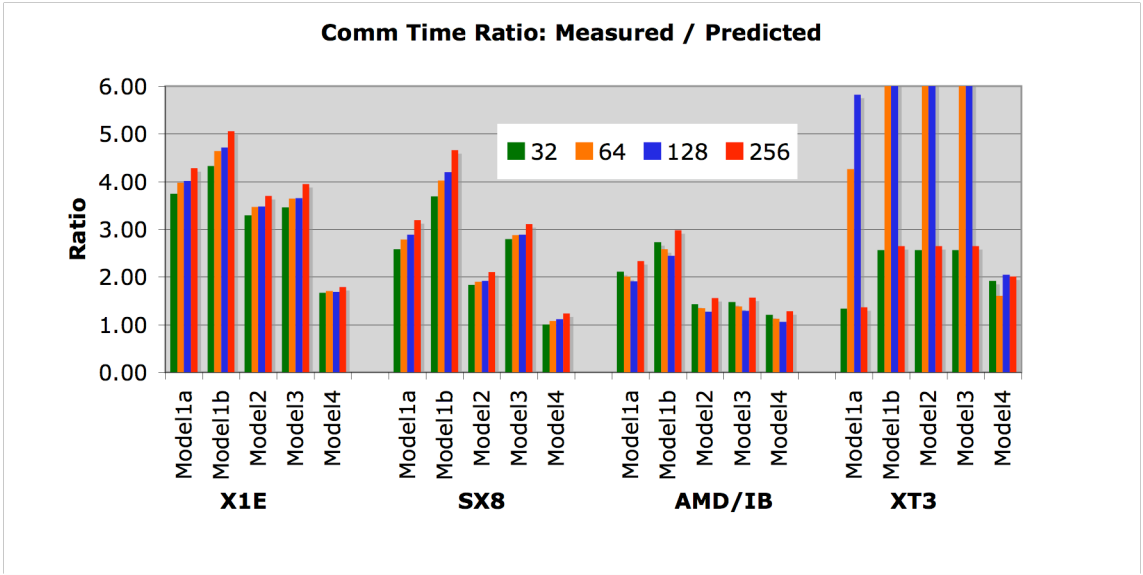

Fig. 7: Ratio of measured to predicted total communication times of our performance models

Further investigation reveals, that the approximation of message transfer times by a linear function in message sizes in not accurate enough to provide acceptable model predictions. Fig. 8 shows that the actual transfer times for message sizes of interest between 4kB and 128kB are substantially different due to message protocol changes. If we replace our linear latency-bandwidth model of transfer times by the actual achieved bandwidth values (Model 4), prediction on SX8 and the AMD/IB cluster improves to within a few percent of measurement for concurrencies up to 128 processors and around 20% for 256 processors; prediction on X1E and XT3 improves from 5 – 10 times difference to within two times difference. The Cray X1E uses a 4D hypercube network topology and XT3 uses a 3D torus topology that are quite different from the fat-tree networks on AND/IB cluster and the crossbar on SX8. Networks with such topology are more sensitive to network contention as long-range messages traverse multiple links and increase network load over-proportional. Capturing contention effects in these networks is difficult and requires more sophisticated models and/or benchmarks, which are sensitive to the average distance of processor in the typical communication patterns.

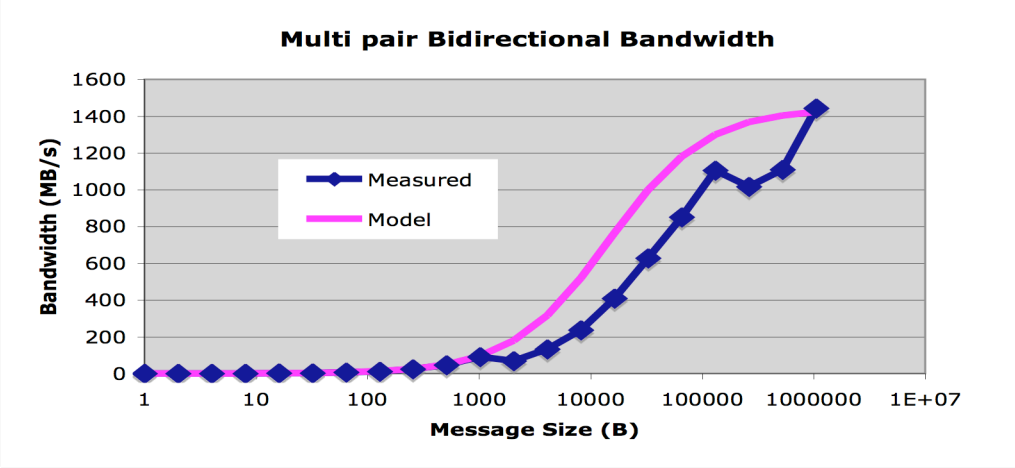

Fig. 8: Message transfer times for the multi-pair (8) benchmark on SX8

6. Summary

In this paper, we examined the performance of Cray X1E and XT3, together with NEC SX8 and the AMD/IB cluster. The two vector platforms delivered much superior performance than the two superscalar platforms for both the synthetic network benchmarks used and the BeamBeam3D application. Furthermore, we examined the methodology to correlate the synthetic network benchmark results to application communication performance. We find that the single pair benchmarks are often failed to capture the network contention and therefore may significantly underestimate the communication time. The multi-pair benchmarks perform substantially better, especially for the networks with either crossbar or fat-tree topology. On network, such as torus or 4D hypercube, which are more sensitive to network contention as long-range messages traverse multiple links and increase network load over-proportional, more sophisticated model or benchmarks are needed.

References

1. XT3 Overview, http://www.cray.com/products/xt3/
2. P. H. Worley, S. Alam, T.H.Dunigan, Jr M.R. Fahey, and J.S. Vetter, “Comparative Analysis of Interprocess Communication on the X1, XD1, and XT3”, in Proceedings of the 47th Cray User Group Conference, Albuquerque, NM, May, 2005.
3. Laura C. Carrington, Roy L. Campbell Jr., Lary P. Davis, “How Well Can Simple Netrics Represent the Performance of HPC Applications? ”, Proceedings of SC05, 2005.

4. J. Qiang, M.A. Furman, R.D.Ryne, “A parallel particle-in-cell model for beam-beam interaction in high energy ring colliders”, J. Comput. Phys. 198 (2004) 278-294.

Acknowledgements

We would like to thank NERSC, Oak Ridge National Laboratory, and te High Performance Computing Center Stuttgart (HLRS) of the University of Stuttgart to provide the platforms.