PERFORMANCE OF SYSTEM AND APPLICATION BENCHMARKS ON PARAM FOR PREMIER INSTITUTES IN INDIA

V. C. V. Rao, Manisha Dhanke, Ravi Kumar, Subba Ramanna,
Ch. Kalyana Krishna and Imran Aziz

National PARAM Supercomputing Facility
Centre for Development of Advanced Computing, Pune
vcvrao@cdacindia.com

SYSTEM CONFIGURATION

The PARAM 10000 system for Premier Institute has peak computing power of 6.4 giga floating-point operations per second (GFLOPS). It is based on cluster architecture, network of Sun Ultra workstations. It supports different system area networks such as Fast Ethernet (100 Base T), and PARAMNet. It uses processors based on SUN Microsystems's UltraSparc architecture, each operating at a clock speed of 400 MHz running Solaris 2.8 as the Operating system. It has 3 compute nodes and 1-server node. Each node is a dual-processor SMP (shared memory, symmetric multiprocessing) system. The server node has 1 GB main memory and all other compute nodes have 512 MB each. All the nodes have 16 KB L1 cache and 2 MB L2 cache. C-DAC designed the PARAMNet switch architecture with associated network interface. PARAMNet is built around a high speed and low latency switched hub. This hub uses STC104 packet switch, which is a low latency, 32 by 32 way, non-blocking crossbar switch. The switch uses cut-through or wormhole routing of packets. Each link of the hub supports 100 + 100 Mb/s bi-directional bandwidth. The latency of the switch is less than one μs. The hub is configurable as 8 by 8 switch, with each port supporting 400 + 400 Mb/s bi-directional bandwidth. C-DAC has also developed High Performance Computing and Communication (HPCC v-1.3) software, which supports development and execution of both sequential and message passing programs. The software distribution contains KSHIPRA, which is a fast communication substrate with active messages (AM) ported on PARAM using C-DAC's optimized MPI (CMPI). HPCC software contains a rich set of high performance tools for clusters. Sun Workshop 5.0 is provided. Sun Performance Library in Sun Workshop has a set of optimized, high-speed mathematical subroutines for solving linear algebra and other numerically intensive problems.

PERFORMANCE ON ONE NODE OF THE SYSTEM

The performance of LAPACK depends on use of Sun-performance libraries for BLAS computations. LAPACK test suites are executed on one node of PARAM 10000 for Premier Institute using SMP features. LAPACK

reorganizes the algorithms to use block operations, such as matrix multiplication in the innermost loops. A sustained performance of **700** MFlops is obtained on one node (dual CPU) for single precision matrix-matrix multiplication operations (SGEMM), which performs Sample BLAS III. Highly tuned Sun-performance libraries for BLAS computations are used to extract the performance of LAPACK on PARAM 10000. The performance of several LAPACK test suites varies from **250** MFlops to **500** MFlops. LINPACK achieves nearly **200** MFlops on one processor of this system. The best NAS code gives performance of **90** MFlops on one processor of this system. NPB (NAS Parallel Benchmarks) suites are not optimised and only compiler optimisations have been used to extract the performance. Third party Computational Fluid Dynamics code gives performance of nearly **60** MFlops.

MEASURING OVERHEADS IN POINT-TO-POINT COMMUNICATION PRIMITIVES

P-COMS (PARAM - Communication Overhead Measurement Suites) is a set of programs to measure the communication overheads in MPI point-to-point communication, collective communication and computation primitives on PARAM. These suites compare the performance of point-to-point communications, including send and receive overheads for different send and receive modes for different (contiguous) message lengths to estimate the network latency. Latency, in the sense of a message passing system, refers to the cost to set up a message transmission or time taken for execution of a MPI library call. Start-up time is also called as latency, which is the time in microseconds to communicate a zero-byte or short message. The *start-up time* in μs, *asymptotic bandwidth* in MB/s, *half peak length*, and *bandwidth* for small and large message sizes that characterize the point-to-point communication overhead in microseconds *(μs*) have been calculated. A popular method for measuring a point-to-point communication (e.g., between processor 0 and processor 1) time is the *ping-pong* scheme. On PARAM 10000 for Premier Institutes, using HPCC software (with AM over PARAMNet using MPI), the latency is 33 μs. In Figure 1, we compare the performance of bandwidth numbers, and in Figure 2, performance of latency numbers using HPCC software and ANL (Argonne National Laboratory) MPI (*mpich* version 1.2.4) over Fast Ethernet (100 base T) at an application layer by executing *ping-pong* test. With HPCC software the bandwidth is 30 MB/s where as on Fast Ethernet the bandwidth is 10 MB/s.

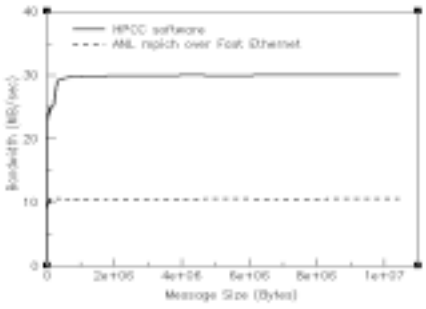

Figure 1. Performance of *ping-pong* test – Bandwidth.

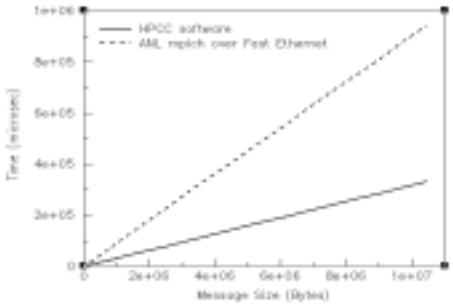

Figure 2. Performance of *ping-pong* test – Latency.

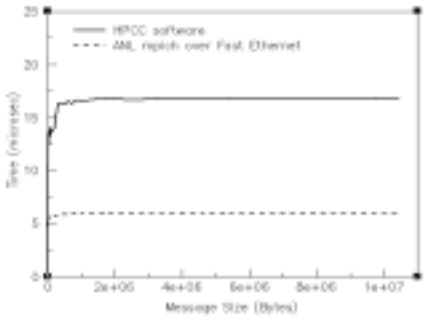

Figure 3. Performance of generalized *ping-pong* test - Bandwidth

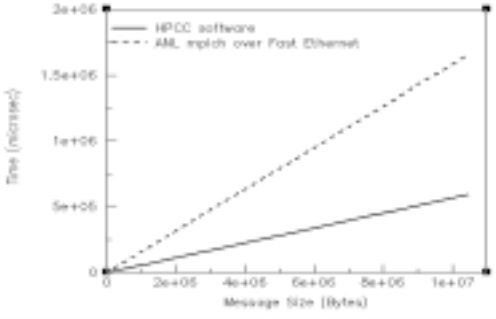

Figure 4. Performance of generalized *ping-pong* test - Latency

In the generalized ping-pong test, *p* processors are involved to identify the overheads incurred in point-to-point communication calls involving *p* processors instead of two processors (the sender and the receiver). Processor 0 sends a message length of *m* bytes to processor 1, which immediately sends the same message to processor 2, and so on. Finally, processor *p*-1 sends the message packet back to processor 0. The total time is divided by *p* to get the point-to-point communication time for *hot-potato* test. In Figure 3, we compare the performance of bandwidth numbers, and in Figure 4, performance of latency numbers using HPCC software and ANL over Fast Ethernet at an application layer by executing *generalized ping-pong* test. For message sizes above 600 KB, it is observed that there is significant improvement in performance using HPCC software in comparison to using ANL MPI over Fast Ethernet as shown in Figure 3 and Figure 4. In fact, the latency and the bandwidth vary almost by 5% when all *p* nodes have been used in hot-potato test. A permutation is simultaneous, one-to-one data redistribution in which each processor sends a packet of *m* words to a unique processor. This particular operation is quite useful in dense matrix computations. We define *circular q-shift* as the operation in which processor *i* sends a data packet to processor $(i + q) \bmod p$ in a *p*-processor ensemble ($0 < q < p$). This particular type of permutation called *circular shift*.

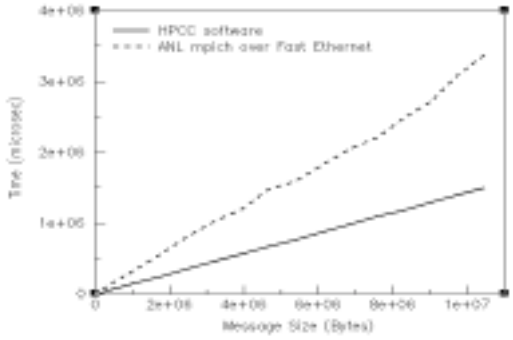

Figure 5. Performance of *Circular Shift.*

In other words, processor *p* sends an *m*-byte message to processor *p+1*, and the last processor *n* sends *m* bytes message back to processor (*n*-1). The results for HPCC software are very encouraging. It is observed that there is linear variation in communication time as message size is increased on PARAM 10000 for Premier Institutes. The overhead measurement time is less on HPCC software as shown in Figure 5.

It is observed that the communication overhead time is reduced to more than half of the overhead time obtained using Fast Ethernet (100 Base T).

MEASURING OVERHEADS IN COLLECTIVE COMMUNICATIONS AND COMPUTATION PRIMITIVES

Overhead measurement time for MPI Collective communication and computation library calls in P-COMS suites is supported. Programs based on various collective communication calls such as *Broadcast, Scatter, Scatterv, Gather, Gatherv, Allgather, Allgatherv, Alltoall,* and *Alltoallv* have been used to measure the communication overhead time. In all the collective communication and computation library calls, the communication overhead is a function of both message size and number of processors used. However, the start-up latency depends only on *p* and the asymptotic bandwidth varies with the machine size *p*. In S*catter* operation, processor 0 sends a distinct *m*-byte message to each of the *p* processors. Figure 6 shows the performance of S*catter* communication primitive using ANL MPI over Fast Ethernet and using HPCC software. It is observed that there are minor oscillations present in the graph and the graphical behavior increases linearly.

In *Scatter* operation processor 0 sends a distinct *m*-byte message to each of the *p* processors. Figure 7 illustrates the performance of *Scatterv* communication primitive on 4 nodes of PARAM 10000 for Premier Institutes. MPI_Scatterv extends the functionality of MPI_Scatter by allowing a varying count of data from each process. In broadcast operation, processor 0 sends an *m*-byte message to all *p* processors. For large message sizes i.e., above 600 KB, the overhead measurement time using HPCC software drastically reduces.

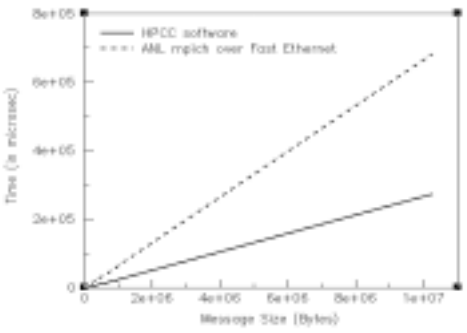

Figure 6. Performance of *Scatter*.

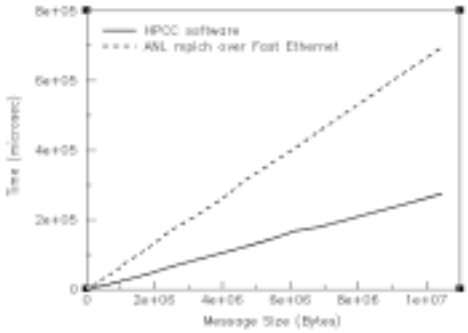

Figure 7. Performance of *Scatterv*.

Also, a *Gatherv* operation on a significantly different sized data, will take more time than a *Gather* operation of equal-sized data. This is because unequally sized data may lead to contention; thus, increasing the overall time. The result for broadcast communication primitive on 4 nodes of PARAM 10000 for Premier Institutes is shown in Figure 8. It is observed that the communication time behavior is same as that of *scatter* communication primitive. In a *Gather* operation, processor 0 receives an m-byte message from each of the *p* processors, so in the end, processor 0 receives mp bytes. MPI_Gatherv extends the functionality of MPI_Gather by allowing a varying count of data from each process. MPI_AllGatherv can be thought of as MPI_Gatherv but where all processes receive the result, instead of just the root. The results for *AllGatherv* communication primitive are shown in Figure 9. It is clear from the figure that using HPCC software, the measured communication time increases linearly as the message length increases.

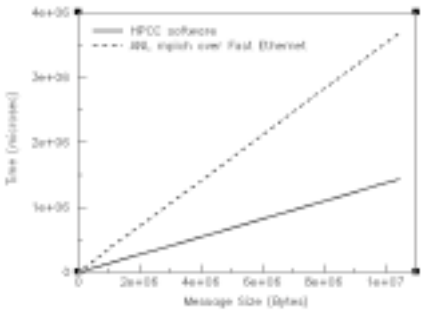

Figure 8. Performance of *Broadcast.*

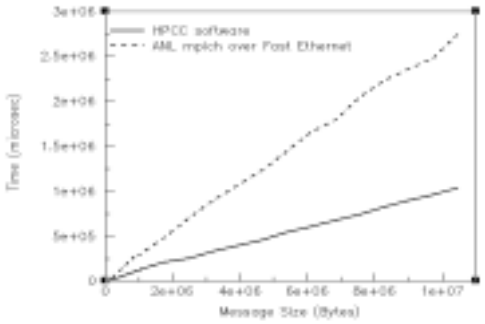

Figure 9. Performance of *allgatherv.*

Figure 10 illustrates the performance of *Alltoall* communication primitive on PARAM 10000 for Premier Institutes for both ANL MPI over Fast Ethernet and using HPCC software. In *Alltoall* (or transpose) collective communication primitive, every processor sends a distinct *m*-byte message to each of the *p* processors, so in the end, mp^2 bytes are communicated. The results indicate that the overheads in communication time are less for HPCC software in comparison to ANL MPI over Fast Ethernet. Figure 11 illustrates the overhead measurement time for *AllReduce* communication primitive.

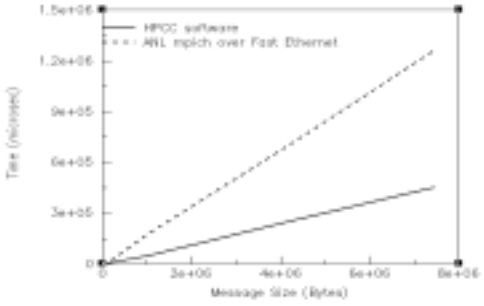

Figure 10. Performance of *alltoall.*

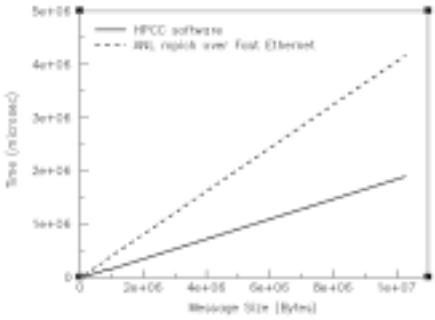

Figure 11. Performance of *allreduce.*

In *AllReduce* collective communication primitive, MPI includes variants of each of the reduce operations where the result is returned to all processes in the group. MPI requires that all processes participating in these operations receive identical results. Same as *Reduce* collective communication primitive except that the result appears in the receive buffer of all the group members. *Reduce* combines the elements provided in the input buffer of each process in the group, using the operation op, and returns the combined value in the output buffer of the process with rank root. The computational time for execution of *Barrier* communication primitive on 4 nodes is approximately 90 μs using HPCC software and 180-200 μs using ANL MPI over Fast Ethernet.

Consequently, the performance of large-scale scientific applications on PARAM 10000 for Premier Institutes will significantly improve if the application developer uses HPCC software.

ScaLAPACK

ScaLAPACK, or Scalable LAPACK, (Distributed Version of LAPACK) is a library of high performance linear algebra routines for distributed memory message-passing MIMD computers and networks of workstations supporting MPI. Highly tuned Sun Performance computing libraries have been used on each node of PARAM 10000 for Premier Institutes with HPCC software. Results for selective ScaLAPACK test suites have been published. The LU test program (**XSLU)** calls the ScaLAPACK routines to factorize and solve the system of linear equations. XSLU achieves sustained performance of **1.07** GFlops on **8** processors. The performance can be further improved with a proper choice of Matrix size, Block size, and the processes grid. The other LU test program (**XSGBLU**), which uses banded test matrices, achieves performance of nearly **2.5** GFlops on **8** processors. The Cholesky factorization program (**XSLLT**) calls the ScaLAPACK routines to factorise and solve the system of linear equations, achieves performance of **1.05** GFlops on **8** processors of this system. The Single Precision Banded Cholesky factorization (**XSPBLLT**) benchmark gave a sustained performance of **1.06** GFlops using all the **8** processors. The Inverse test program (**XSINV**) achieves performance of **1.3** GFlops on **8** processors of this system. The sustained performance of several ScaLAPACK benchmark family suites on PARAM 10000 for Premier Institutes with HPCC software indicate that there is an performance improvement of 15% to 30% in comparison to ANL MPI over Fast Ethernet.

LINPACK

LINPACK subroutines analyze and solve linear system of matrix equations by LU factorisation. It is simple and easy to use, yet a good indicator of the numerical computing capability of a Parallel system.

Table 1. Performance of LINPACK Benchmark.

Processors	**Matrix Size** N_{max}	**Fast EtherNet (100 Base T)** R_{max} **(GFlops)**	**C-DAC HPCC Software** R_{max} **(GFlops)**
1	5000	0.55	0.55
4	10000	1.62	1.92
8	14000	2.96	3.45

We have performed different experiments to extract maximum performance on PARAM 10000 for Premier Institutes using different system area networks and the HPCC system software i.e., AM with MPI on PARAMNet and the results are shown in Table 1. Performance of **550** MFlops is achieved on one processor and **3.4** GFlops is achieved on **8** processors of PARAM 10000 for Premier Institutes using HPCC software. Sun Performance libraries and best compiler options are used to obtain this performance. In all the computations, *Matsize* gives the matrix size and *BlockSize* gives the Block size. The processes are also divided into process matrix of size $P \times Q$. The choice of Matrix Size, Block Size / Band Size, Processes (Process Grid (P, Q)) matrix and system parameters play a vital role for performance. It is possible to get further improvement in sustained performance by proper choice of matrix size, block size, the process grid (P, Q), memory available on each node and use of the best options provided in Sun performance libraries.

NAS: COMPUTATIONAL FLUID DYNAMICS

NAS benchmark set consists of two major components: five parallel kernel benchmarks and three simulated application benchmarks. The simulated application benchmarks combine several computations in a manner that resembles the actual order of execution in certain important Computational Fluid Dynamics (CFD) application codes. The NPB (NAS Parallel Benchmarks) suite consists of five kernels (EP, MG, CG, FT, IS) and three simulated applications (LU, SP, BT) programs. The performance, in terms of MFlops for all benchmarks is shown in Tables 2 and 3.

Table 2. Performance of NAS Benchmark.

Routine/ Problem Size	**Fast EtherNet (100 Base T) (in MFlops)**	**C-DAC HPCC Software (in MFlops)**
	8 processors	**8 processors**
MG A	350.50	584.32
B	374.07	629.40
LU A	648.81	864.16
B	668.92	801.42
CG A	93.57	165.15
B	108.40	178.54
FT A	140.15	261.32
B	14.85	263.90
EP A	26.51	21.06
B	26.37	21.08
IS A	7.55	7.58
B	7.52	7.60

Table 3. Performance of NAS Benchmark.

Routine/ Problem Size	**Fast EtherNet (100 Base T) (in MFlops)**	C-DAC HPCC **Software (in MFlops)**
	9 processes	**9 processes**
BT A	257.65	268.34
B	273.28	264.61
SP A	184.74	187.17
B	185.45	178.91

The NAS codes can be further optimized using Sun-performance libraries and code restructuring. The further level of optimization i.e., tuning to PARAM 10000 architecture may improve the performance by almost 10% to 15% for LU benchmark with HPCC software on **8** processors of this system. LU benchmark for class A achieves **864** MFlops and class B achieves **801** MFlops on **8** processors.

P-CFD: COMPUTATIONAL FLUID DYNAMICS

P-CFD (PARAM - Computational Fluid Dynamics) software on PARAM 10000 is based on the solution of Unsteady State Navier-Stokes equation. The algorithm is based on explicit time marching and three dimensional structure grids are employed for computation. One dimensional grid partitioning is performed to obtain the concurrency. After discretization of given domain into required structured grid cells, finite difference method is employed to obtain matrix system of linear equations. These equations have been solved over number of iterations till a specific error criterion is met. The program is re-structured and efficient complier optimizations are used to extract performance on PARAM 10000. A sustained performance of 900 MFlops is achieved on 8 processors of PARAM 10000 for Premier Institutes with HPCC software.

PERFORMANCE OF BENCHMARKS

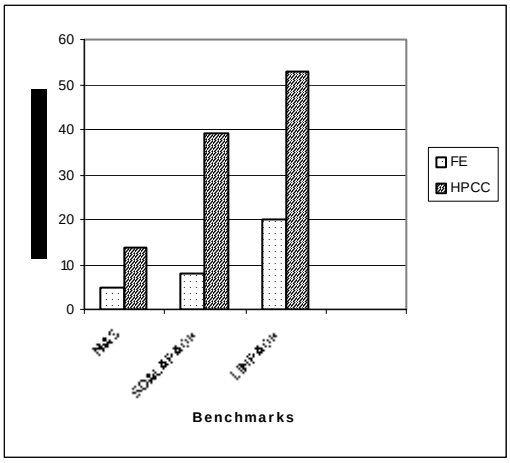

Figure 12. Performance of benchmarks.

The efficiency rate of all the benchmarks on Fast Ethernet (100 Base T) and using HPCC software (AM over PARAMNet using MPI) on PARAM 10000 for Premier Institutes in terms of percentage is depicted in figure 12. The study shows that the efficiency rate for certain test suites in ScaLAPACK gives nearly 39%. For Top-500 LINPACK benchmark, efficiency is 53%. In case of NAS, code restructuring and further optimization process may give substantial improvement in the performance using HPCC software on PARAM 10000 for Premier Institutes.