# Performance analysis of Particle in Cell Electromagnetic code using Infiniband Interconnect

G. Singh, N. Sakthivel and S. Chaturvedi
Bhabha Atomic Research Centre, Trombay, Mumbai
{svel@ipr.res.in}

**Abstract**

Particle in Cell (PIC) codes are widely used in the simulation of many plasma related systems (E.g.: Laser plasma interactions, high power microwave sources). A detailed simulation of these systems requires parallel computing facility with faster CPUs with efficient interconnects. We have setup a 33 node Xeon (Dual socket and Dual core) cluster with double data rate Infiniband interconnect in addition to our existing 144 node Pentium-4 cluster with gigabit interconnect. The basic network performance parameters (latency and bandwidth) and the PIC code performance in 32 node Xeon cluster with double data rate Infiniband interconnect has been studied. The studies carried out using 2 cores per node and 4 cores per node are reported.

## Hardware Setup

We have setup a 33 node Intel Xeon (Dual Socket and Dual core) with Silverstorm single port double data rate Infiniband interconnect. The Infiniband Host Channel Adapater is plugged into the PCI-Express (8X) slot. The motherboard is Intel S5000PAL with Intel 5000P chipset. Each node is equipped with 4 GB of FB-DIMM memory. All machines are connected to a Silverstorm-9080 switch, which provides a theoretical uni-directional bandwidth of 20 Gigabits per second between two nodes. The uni-directional usable bandwidth becomes 16 Gigabits per second due to 8B/10B encoding. The operating system is Scientific Linux 4.0 [1] having 2.6.9-42 Linux kernel. The login node, which is also the GLUSTERFS [2] based file server for the cluster has 8 GB of main memory. The MVAPICH implementation is from Ohio State university [3] mvapich-0.9.7 compiled with gcc-3.4.6-3.x86_64.

## Infiniband Latency and Bandwidth

The Infiniband specification [4] defines a switched, high bandwidth, low latency fabric for both inter-process and I/O communication. The MPI-latency and the MPI-bandwidth achieved using the codes available in OSU web page [5] is given in figure 1 and 2. This benchmark codes uses the blocking MPI_send and MPI_receive [6] functions for measuring the latency and bandwidth.

Between two of the Xeon Compute nodes, the latency for 4096 Bytes message is 11.75 micro-seconds. The 1-byte message latency is 3.19 microseconds. A maximum bandwidth of 1019 MB/sec was achieved for 4096 Bytes message.

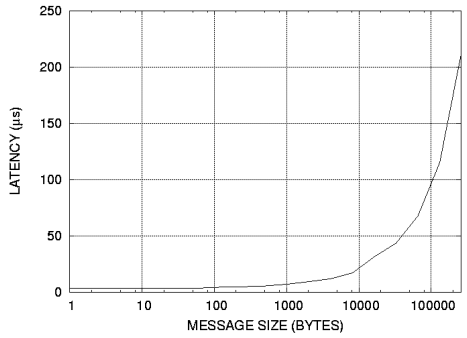


Figure 1: MPI level latency vs Message size

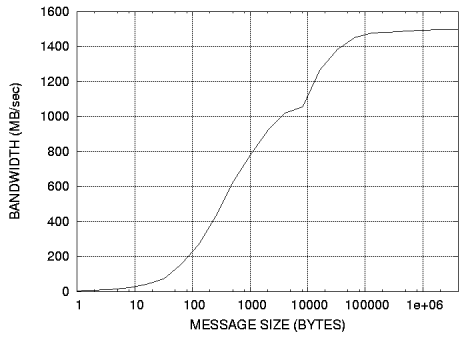


Figure 2: MPI level uni-directional bandwidth vs Message size

## Particle in Cell code Description

Particle in Cell (PIC) codes are widely used in the simulation of many plasma related systems (E.g.: Laser plasma interactions, high power microwave sources). The PIC code solves the Maxwell's equations on a predefined grid using finite difference time domain (FDTD) method. The plasma species are considered as a macro particle, which in-turn consists of a group of particles (electron or ion). The evolution of the macro particles in time is simulated using the Lorentz force equation and the self-consistent electromagnetic field.

A three dimensional PIC code has been developed and parallelized using MPICH. The code is parallelized in one-direction, however, the choice of direction (X or Y or Z) can be chosen during startup. The accuracy of the simulated system depends upon the number of simulation particles used as well as the spatial resolution used for calculating the electromagnetic fields. Hence, the simulation of realistic systems, need very large number of macro particles and better spatial resolution. This demands huge computational requirement in-terms of faster CPUs and efficient interconnects.

## PIC code parallelization method

The PIC code solves two types of data types, one is related to field components and the other related to macro particle parameters. Based on data types, two types of decomposition are required viz., the field decomposition and the particle decomposition. In field decomposition the computational box is divided into non-overlapping sub-domains, with approximately equal number of grid points in each domain. In particle decomposition, the Eulerian decomposition technique is used. In this, each processor manages only those particles, which lie within its sub-domain. Each sub-domain may have different number of macro-particles during the time-evolution.

The parameters related to grid in the neighboring layers of each sub-domain and the parameters related to macro-particles, which crosses their sub-domain during each time step form the MPI Message. Hence the message size between any two processors may vary.

**PIC code performance**

In this study, the number of cells in the X-direction, Y-direction and Z-direction are 4000 and 2000 and 1 respectively. One cell along the Z-direction makes the code as two-dimensional. The parallelization is done along X-direction.

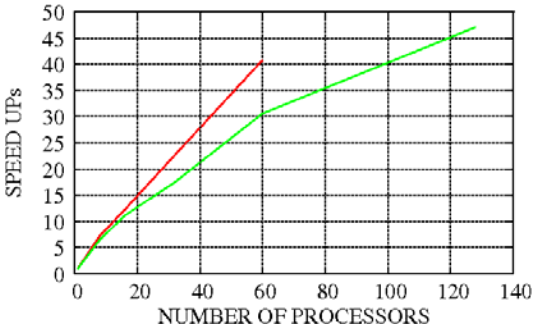


Figure 3 Speed ups vs Number of processors with 2 cores per node and 4 cores per node used.

Figure 3 shows the speed ups while using 2 cores per node and 4 cores per node. The speed up is same for both the cases up to 16 processors. However, when the number of processors is increased to 64, use of 2 cores per node is faster by a factor of 1.3. This may be due to more contention for cache and memory bandwidth.

A total of around seven MPI_Send, one MPI_Sendrecv, one MPI_Allreduce, calls are used during each time step and total of six MPI_Send and one MPI_Sendrecv calls are used initially at tome 0 for message exchange. The total number of bytes exchanged between processes per time step is around 192 kB to 188 kB depending on the number of processors. This variation is due to variations in the number of macro-particles that crosses each sub-domain at each time step as explained above. The total number of bytes exchanged at every time step vs time is shown in Figure 3.

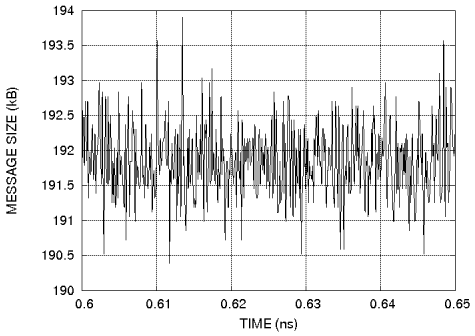


Figure 4: Message Size vs Time in PIC simulation

Table 1: Total network latency and run time

| # Procs. | TBytes | Total latency (milli-second) | Run time (min.) |
|---|---|---|---|
| 8 | 196244 | 7.4 | 61 |
| 16 | 194136 | 6.4 | 33 |
| 64 | 192436 | 4.5 | 10 |

***TBytes - Total bytes transferred per time step***

Table 1 presents the network latency and run time of our PIC code when 2 cores per node is used. The total latency refers to the total time taken for a process, which communicates with the neighboring process in the other node while using six MPI_Send calls. The toal number of bytes that got transferred is given by TBytes. This shows that the total latency decreases with the decrease in the number of bytes transfer. However, a detailed analysis of each MPI_send with its message size and latency is required to have a comparison with latency number presented in Figure 1.

**Summary**

A Particle in Cell (PIC) code has been parallelized using MPICH and its performance using Infiniband interconnect has been presented. The basic network performance parameters (latency and bandwidth) and the PIC code performance in 32 node Xeon cluster with double data rate Infiniband interconnect are reported.

**REFERENCES**

1.https://www.scientificlinux.org

2.http://www.gluster.org – GlusterFS

3.http://mvapich.cse.ohiostate.edu

4. Infiniband Trade Association, Infiniband Architecture Specification, Release 1.1, November 6th, 2002.

5.http://mvapich.cse.ohiostate.edu/benchmarks/ - Benchmark codes

6.Message Passing Interface Standard and Forum. http://www-unix.mcs.anl.gov/mpi