Cluster Computing in Astrophysics

J.S.Bagla

Email: jasjeet@mri.ernet.in

Harish-Chandra Research Institute
Chhatnag Road, Jhusi, Allahabad 211019, India

Computing is an essential tool in almost all branches of science today and astronomy and astrophysics are no exceptions. Indeed, astronomers and astrophysicists have been leaders in deployment of high performance computing for solving complex problems, e.g. see [39, 40], and in support of major observational efforts, e.g. [38]. In this resource summary I will outline how commodity computers can be used as a platform for high performance computing.

Computing power has been increasing rapidly, the CPU performance typically doubles every 18 months. A typical desktop computer on sale today would have ranked amongst top 100 in the first list (June 1993) of fastest supercomputers in the world [23].

The increasing computing power brings a larger range of problems within reach of normal desktop computers and many problems and analyses that could be only be done on expensive workstations in the past can now be carried out on desktop computers. Easy availability of good quality software [1], operating system [2] and documentation [3] accelerated this all round development. As an example of the richness of tools and software available, we refer the reader to the *Linux Astronomy Howto* at [3].

1 Cluster Computing

Computing power can be enhanced by using a large number of computers together. This not only allows us to run several programs at once, a group of computers in a network can also be used together to solve a single problem. A group of networked computers, often of identical configuration, meant to be used together for computing is called a cluster. The first cluster was designed and made by Donald Becker and Thomas Sterling in 1993 [7]. They showed that clusters made with inexpensive desktop computers can rival very costly supercomputers in terms of performance.

Before we proceed further, we would like to describe some often used terms. We have already mentioned that a cluster is a group of computers in a network. Clusters made out of commodity, off the shelf (COTS) computers are often also called Beowulf clusters – so called after the first cluster of its kind. If workstations with RISC[1] processors are used instead of COTS then it becomes a cluster of workstations (COW). This is distinct from a typical network of workstations (NOW) which may be made of a variety of workstations and these may be used for interactive sessions as well. Workstations in a NOW may be used by a job management system when not in use for interactive sessions, thus harvesting CPU cycles that would have been wasted otherwise. Clusters are typically made of identical computers as that makes parallel programming and load balancing simpler. In parallel programming, computational load for a single problem is distributed across many processors. If all the processors are not identical then it will be difficult to keep all the processors occupied by merely dividing the computational load equally, and an imbalance will result in at least some processors waiting while other processors finish the assigned tasks. Such imbalance leads to poor improvement in the time taken for solving the problem, thus it is better to use identical computers and remove one potential reason for poor performance. Cluster nodes are not meant for interactive sessions on the console, as user interface and graphic applications can interfere with load balancing in a parallel program. Figure 1 shows picture of a 16 node cluster made using COTS computers.

Cluster computing is also called distributed parallel computing: parallel because instructions are processed concurrently by a number of processors, and as the total memory is distributed across different computers without a single addressing scheme, the name distributed parallel computing is appropriate. This is to be contrasted with shared memory parallel computing where the entire memory has a single addressing scheme on a multi-processor computer, i.e., all the processors have a direct access to the entire memory. Number of processors on largest computers of either kind can be around 10^3. Cost of shared memory computers increases very rapidly with the number of processors whereas the cost of clusters increases almost linearly with the number of processors. This gives clusters a distinct edge over multi processor computers as one can obtain very high computing performance for most applications at a low cost, e.g., it is possible to set up a cluster with a performance of 100 Giga flops[2] at a fraction of the cost of a multi processor RISC computer with the same performance. The low cost has made clusters the platform of choice for high performance computing, there are close to 300 clusters in the 500 fastest computing facilities [23] even though the idea of a cluster is just over ten years old.

In a cluster, processors communicate over a network and the performance of the network can be a serious bottleneck in distributed parallel computing. Clearly, inter-process communication is essential and we cannot avoid the network altogether. It is however important to ensure that the communication overheads are as small as possible, typically this means working with large problem sizes so that the computation time dominates over communications. Figure 2 shows the performance of the Kabir cluster [25] with the HPL benchmark [20] as a function of the problem size. We see that the performance increases rapidly as we increase the problem size before levelling off near 290 Giga flops. There is a

[1] RISC: Reduced Instruction Set Computers. The idea is to break up complex instructions in terms of a small set of simple instructions and implement these optimally in the CPU.

[2] 1 Giga flop $= 10^9$ floating point operations per second. 1 Tera flop $= 10^3$ Giga flops.

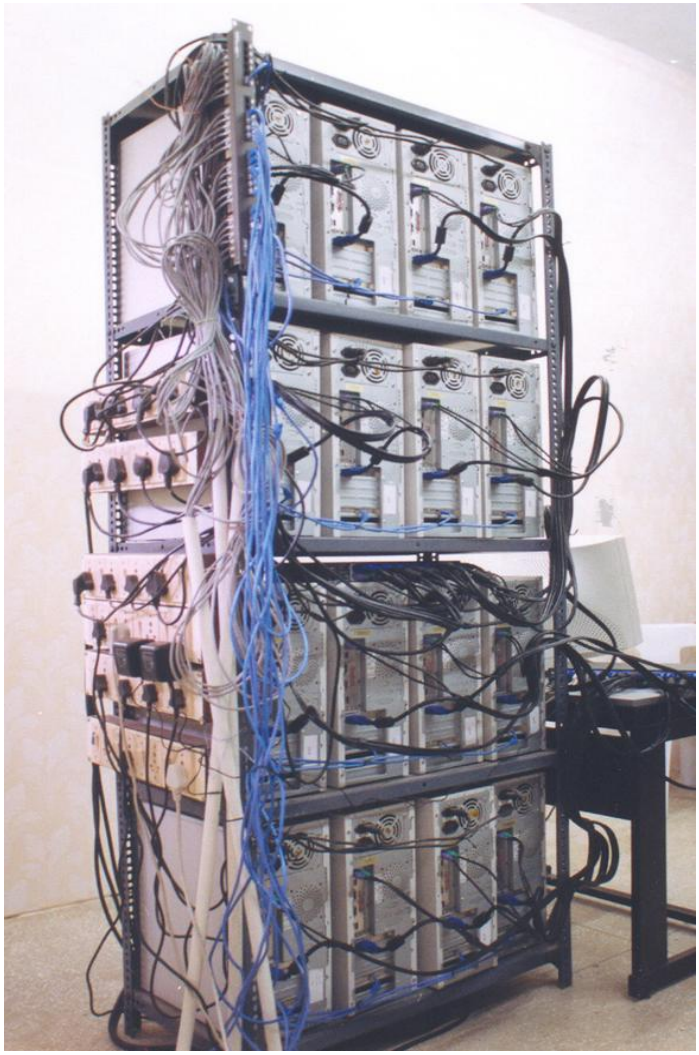

Figure 1: Picture of a 16 node cluster at the Harish-Chandra Research Institute. Note that there is just one monitor that is shared by all the computers in the cluster. Users access the cluster over the network and console access is not not required except for system administration in case of network failure. See http://cluster.mri.ernet.in/ for details.

range of problem sizes for which the cluster is a fast and efficient computing platform, using the same platform for very small problems is an inefficient use of resources. This is not a serious issue for shared memory computers.

Published performance evaluation can be used to decide the type of hardware to be procured [21], the most commonly used benchmark for this purpose is Linpack [20]. The list of 500 fastest supercomputers [23] is a useful resource for those planning to set up a major computing facility. Information can also be obtained from the pages maintained by the IEEE task force on cluster computing [4, 5, 6].

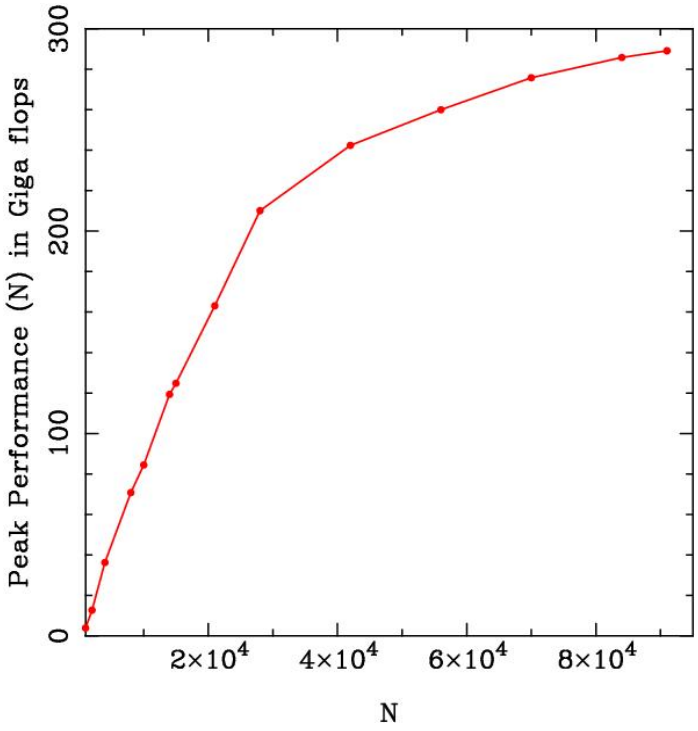

Figure 2: Performance of the Kabir cluster [25] is shown as a function of the problem size. This performance for the high performance linpack (HPL) [20] benchmark has been obtained using the Goto libraries [24]. Performance is better for larger problem sizes as the relative importance of communication overheads decreases when we run larger problems. Peak performance on this cluster is close to 290 Gigaflops.

1.1 Management

Management of a cluster involves installation, system administration and allocation of resources to users. We are concerned here with installation of operating system and/or applications. Typically each of the computers is set up with its own copy of the operating system and applications, and has its own host name and network address. Installation on a large number of computers can be a challenging task and there are several tools available to simplify this task [9], with some tools more specific for clusters [10, 11]. An interesting alternative is to use an operating system that treats the entire cluster as a single computer, e.g., see [8].

The standard for distributed parallel programming is the Message Passing Interface (MPI) [28]. MPI has been implemented on a variety of platforms, e.g. see [29, 30]. On shared memory computers, an alternative is available in form of OpenMP [31]. It is worth mentioning Parallel Virtual Machine (PVM) [32], a very different approach to distributed parallel computing.

Installation of scientific applications is typically done by the friendly system administrator of the local facility. This requires close coordination between users and the administrators as high performance computing requires careful optimisation and tuning of applications. Users participate in the system administration process in most facilities, indeed many small and medium sized facilities are managed by users themselves. Benchmarking tools

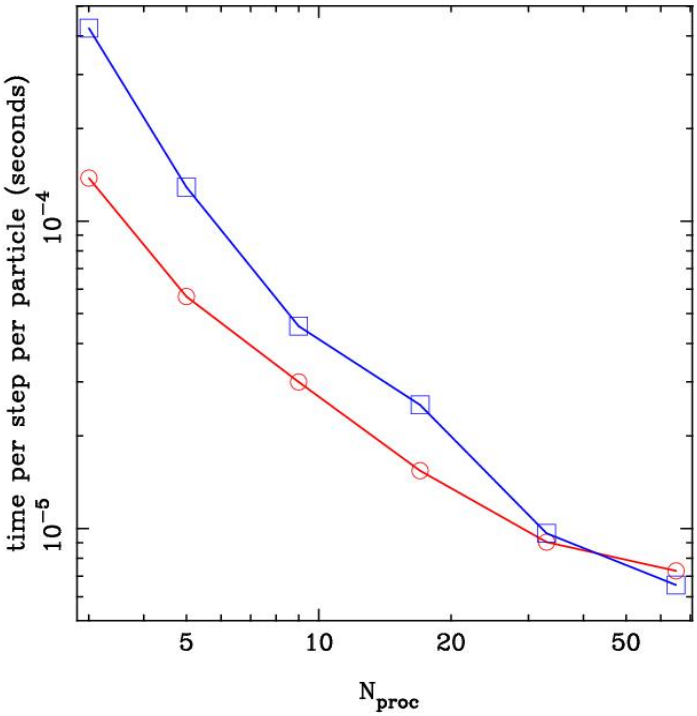

Figure 3: This graph shows the performance of the parallel TreePM code [41]. We have plotted the time taken per particle per time step as a function of number of processors used. The red curve is for a simulation with 2 million particles and the blue curve is for a simulation with 16 million particles. Programs that require larger computing resources are more efficient when run over a large number of CPUs, this is why the larger simulation continues to scale well up to the largest number of processors used.

are essential for tuning and testing the computing facility. Programs for testing nearly every aspect of computers are available [18, 20, 19, 22].

Most operating systems provide easy to use packages for maintaining user accounts on a large network. These tools set up configuration files for network services such as the Network Information Service (NIS) and the Network File System (NFS) [3]. Managing user accounts on a large number of computers without these network services is an extremely tedious task. Cluster command and control (C3) tools [12] are a useful alternative in such situations. Rationale for not using network services is to reduce non-computational load on the network, this makes sense if the same network is used for computation related communications. Many clusters use high performance networks for computation related communications in addition to an Ethernet network for all other communications, and tools such as C3 are not essential for such clusters.

1.2 Job Schedulers and Load Balancing

Clusters and other high performance computing facilities that are used by multiple users require job scheduling systems or other mechanisms for balancing computational load across the available CPUs. Such mechanisms are not needed for facilities that are dedicated for a single application or user.

Job schedulers [14, 13] are used to schedule jobs with different requirements in terms of number of CPUs, memory, CPU time, etc. using a simple algorithm to ensure equitable distribution of resources amongst different users and jobs while maximising utilisation of computing resources. A large number of schedulers with a variety of scheduling algorithms have been implemented.

Load balancing on a cluster is a more complex task in that it requires migration of jobs from one computer to another in a seamless manner. The most challenging aspect of this is the transfer of Input/Output and communications related to a program from one computer to another. Load balancers are more convenient than job schedulers for a situation where most of the jobs are sequential in nature. Mosix [16], OpenMosix [15] and Condor [13] are examples of load balancing software. These also have many other useful features and are often used instead of job schedulers on small clusters. Recently an OpenMosix cluster has been set up in IUCAA.

A concept called the grid has been proposed to harness the available computing power. A large number of computing facilities accessible on the Internet can, by mutual agreement, allow requests for resources to be made available to the set of all the users. Thus jobs will get scheduled at whichever facility is underused at the given moment. Software implementations of the grid [17] must incorporate features for secure transactions between different computing facilities that may be of a completely different type and located on different continents. This idea and the software being written for the purpose can also be used to efficiently manage sites with multiple computing facilities where the grid software serves as a coherent job management system.

2 Astronomy and Astrophysics

Computations of interest in astronomy are similar to those required in physics and chemistry. Efficient subroutines for most computational requirements can be found on many repositories, e.g. see [34]. Parallel versions of these subroutines can be use to enhance performance of programs without making significant changes to the sequential programs [35, 36, 37]. A comprehensive list of software packages and subroutine libraries of interest to astrophysicists is maintained at the astrophysics source code library (ASCL) [33].

Astrophysics applications of many diverse types have been parallelised. Several parallel algorithms for N-Body simulations have been proposed and implemented, e.g. see [41, 42, 43, 44, 45]. Figure 3 shows the performance of the Parallel-TreePM code as an example of how the performance of an N-Body code scales on a cluster. Some of the codes mentioned above also solve for gas dynamical effects. Mesh based codes for MHD have also been parallelised, e.g. see [46, 47]. CMBfast [48] is a popular software package for computing the temperature anisotropy power spectrum for the cosmic microwave background radiation. Many familiar packages have been rewritten to incorporate support for MPI, e.g. [49]. The number of astrophysics applications that have been parallelised is very large and we refer the reader to ASCL [33] and other web resources for a more exhaustive listing.

3 Discussion

Large scale computations of complex nature are the main reason why high performance computing facilities are needed by astrophysicists. High performance computing is also relevant whenever large amounts of data is to be processed, e.g., the Sloan Digital Sky Survey [38]. Cluster computing being the least expensive option, is the most popular one as well. Computers that form part of a cluster can serve as desktop computers for several years after they cease to be useful as part of a high performance computing platform.

Several clusters with performance up to 1 Tera flop have been set up at less than Rs.$25,000$ per Gigaflop [25, 26, 27]. Multiprocessor RISC workstations cost at least 4 times, this can still be worth the cost if a large number of applications are sequential. But in such a case it is important to ensure that each processor in the RISC workstation is much faster than commodity processors.

There are only a few high performance computing facilities available to academic users in India, though the number is increasing rapidly. A list of such facilities is maintained at http://cluster.mri.ernet.in, please refer to this with caution as participation is voluntary and sometimes it is difficult to get the relevant details.

Setting up a small cluster with a target performance of 50 Gflops is well within the scope of individual departments and can be set up and managed by one or two dedicated users. Happy (cluster) computing.

References

[1] http://www.gnu.org/

[2] http://www.linux.org/

[3] http://www.tldp.org/

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

[5] http://www.clustercomputing.org/cluster_white_paper.pdf

[6] http://www.clustercomputing.org/

[7] http://www.beowulf.org/beowulf/history.html

[8] http://www.scyld.com/

[9] http://www.systemimager.org

[10] http://www.rocksclusters.org/Rocks/

[11] http://sourceforge.net/project/showfiles.php?group_id=9368

[12] http://www.csm.ornl.gov/torc/C3/

[13] http://www.cs.wisc.edu/condor/

[14] http://www.openpbs.org/

[15] http://openmosix.sourceforge.net/

[16] http://www.mosix.org/

[17] http://www.globus.org/

[18] http://www.netlib.org/benchmark/

[19] http://liinwww.ira.uka.de/ skampi/

[20] http://www.netlib.org/benchmark/hpl/index.html

[21] http://www.netlib.org/benchmark/performance.ps

[22] http://www.cs.inf.ethz.ch/CoPs/ECT/

[23] http://www.top500.org/

[24] http://www.cs.utexas.edu/users/kgoto

[25] http://cluster.mri.ernet.in/

[26] http://www.imsc.res.in/ kabru/

[27] John Dubinski, Robin Humble, Ue-Li Pen, Chris Loken and Peter Martin, astro-ph/0305109

[28] http://www.mpi-forum.org/

[29] http://www−unix.mcs.anl.gov/mpi/mpich/index.html

[30] http://www.mpi.nd.edu/lam/

[31] http://www.openmp.org/

[32] http://www.csm.ornl.gov/pvm/pvm_home.html

[33] http://ascl.net/

[34] http://www.netlib.org/

[35] http://www.nhse.org/

[36] http://www.mcs.anl.gov/petsc

[37] http://www.fftw.org/

[38] http://www.sdss.org/

[39] http://zeus.ncsa.uiuc.edu/GC3_Home_Page.html

[40] http://www.virgo.dur.ac.uk/

[41] Suryadeep Ray and J.S.Bagla, astro-ph/0405220, Submitted to New Astronomy

[42] Bode P. and Ostriker J.P. 2003, ApJS 145, 1

[43] John Dubinski, 2004, New Astronomy **9**, 111

[44] V.Springel, N.Yoshida and S.D.M. White, 2001, New Astronomy **6**, 79.

[45] Brian W. O'Shea et al. astro-ph/0403044

[46] Michael L. Norman, astro-ph/0005109

[47] Ue-Li Pen, Phil Arras, ShingKwong Wong, 2003, ApJS 149, 447

[48] http://www.cmbfast.org/

[49] B.T.Draine and P.J.Flatau, astro-ph/0309069