L3C Model of High-Performance Computing Cluster for Scientific Applications

Alpana Rajan, Brijendra Kumar Joshi and Anil Rawat

Abstract High-performance computing clusters (HPCCs) are widely used for various scientific applications. In a typical scientific research environment, software applications need large but varying number of processing elements and processor cores. To maximize throughput of a computing cluster and optimum utilization of resources, one new model has been proposed. The proposed model visualizes the computing cluster as loosely coupled cluster of clusters (L3C). Execution time for scientific applications also varies in terms of lapsed time for execution and CPU time utilized. The process scheduling algorithm maintains a list of applications to be executed along with respective number of node/core required. Using the L3C model and scheduling algorithm, multiple applications are scheduled on the computing cluster for concurrent execution. Basis for proposing L3C model along with its details is discussed in the paper. Experimental results of performance evaluation of HPC clusters were published earlier by the authors and are referred at respective places. L3C model has certain inherent advantages which are also discussed in the paper.

Keywords High-performance computing cluster · Performance evaluation HPCC throughput · Scientific applications

A. Rajan (✉) · A. Rawat
Computer Division, Raja Ramanna Centre for Advanced Technology, Indore, India
e-mail: alpana@rrcat.gov.in

A. Rawat
e-mail: rawat@rrcat.gov.in

B. K. Joshi
Military College of Telecommunication Engineering, Mhow, India
e-mail: brijendrajoshi@yahoo.com

S. K. Muttoo (ed.), *System and Architecture*, Advances in Intelligent Systems and Computing 732, https://doi.org/10.1007/978-981-10-8533-8_25

1 Introduction

Use of computing technology for scientific research and development became popular with the advent of general purpose computers in the mid-sixties. As the computing technology progressed in terms of computing power and ease of programming, complexity of scientific codes also increased, leading to increase in workload of computing clusters [1]. In the endeavor to achieve high-performance computing at lower costs, computing clusters were built. Cluster-based computing deploys multiprocessor machines for executing scientific codes having parallel components. Interprocessor communication is supported by very high-speed interconnects. HPCCs are deployed worldwide for scientific applications demanding huge computing power.

Various software tools are available for measuring performance of clusters. Estimated performance of a cluster is a good indication for the capability of the cluster to execute computational tasks. High-performance Linpack (HPL) benchmark is a set of tools for performance evaluation of HPCC [2]. HPL is a de facto standard for measuring cluster performance; it solves a linear system by distributing data over a two-dimensional grid of processes. Peak computing power is quantified as GFlop/s.

Optimum utilization of the available resources is ensured by the architects of the computing clusters to achieve high throughput. When a cluster is built to run heterogeneous types of applications, the task becomes more complex and parameters affecting the performance are to be balanced for maximizing throughput.

2 Performance of HPCC

Performance of a computing cluster depends on components used to build it and also on the type application deployment on it. There are various software tools available for measuring the performance. Estimation of performance indicates the strength, capability, and power of a computing cluster for processing a computation-intensive task. With the advancements in technology, introduction of new system architectures, and also renewed user requirement, it is important that performance evaluation tools are also updated. In this section, the standard basic formula used for estimation is briefly described.

Let us say R_{peak} is the maximum theoretically calculated computing power of system, and R_{max} is the maximum actual computing power of system. R_{max} can be obtained by using software tools like HPL [2–4], high-performance computing challenge benchmark. R_{peak} can be calculated using the following formula:

$$R_{\text{peak}} = \text{PCS} \times \text{NC} \times \text{FPOC}$$

where PCS is processor clock speed in GHz, NC is total number of Cores X, FPOC is floating point operations/cycle.

3 Scientific Applications on HPCC

Clusters have provided a platform for execution of a wide range of applications, including supercomputing (highly processor-intensive jobs). While focusing on scientific applications, use of cluster computing is widely seen for running codes for weather forecast modeling, life sciences, computational fluid dynamics, simulation of beam tracking for accelerators, image processing, aerodynamics, astrophysics, etc. Following subsections briefly cover some popular legacy application and also include certain specific details on sequential applications and parallel applications.

Legacy Applications: Very large number of scientific codes has been written over the years, and these codes have existed for decades. They have been altered and expanded to improve them, and experiments have been conducted using these codes. Scientific community has been hesitant to replace these codes by rewritten codes, since the risk of introducing a logical error is high. Even a subtle change may result in serious consequences.

Sequential Scientific Code: Many scientific codes have been parallelized to gain on execution time using various programming tools and techniques. Tools have been developed to auto-parallelize sequential codes, although with limited success. Certain standard sequential codes are still being used by scientists and researchers.

Parallel Scientific Codes: Parallel scientific codes are either written afresh or are developed by identifying parts of the code which can be parallelized. Major improvement in performance of these codes has been achieved by computer scientists and programmers while working in tandem with scientist having required domain knowledge of the scientific codes [1, 5, 6].

4 Factors Governing Performance for Scientific Applications

Scientific codes have different characteristics as compared to codes in other application areas, as they are computationally intensive programs requiring huge resources in terms of memory, storage, etc. A large number of CPU cycles are required for completing the computations as the problems being solved are very complex in nature [7]. Typical characteristics in addition to computational requirements are computations on large data sets and large number of iterations to converge to the result. Large numbers of iterations are required before the program converges to the solution.

Impact of Number of PEs and Cores: For most of the scientific codes, parallelization is done to optimally distribute the computational part on multiple processors in such a way that the overall computational time reduces. The speedup achieved by parallelization is governed by Amdahl's law. Theoretically speaking, very large number of processing elements (PEs) and cores can be used for solving any complex problem, but there is a limitation due to other limiting factors like interconnect, interprocess communication overheads, inherent serial part of the code, etc.

Impact of Interconnect: Interconnect used for building HPCC is an important factor governing performance of the cluster. Typically, internode communication takes place using the interconnect network. Parallelized parts of code have to interact with each other for transferring the input data sets (which are generally large for scientific computing codes), for combining the output data sets for preparation of program output and at times for transferring intermediate result sets. Thus, this interprocess communication becomes one of the major factors governing performance of cluster. To provide high bandwidth and low latency, many high-end interconnects like Myrinet and Infiniband are used in HPCCs.

Process Distribution Patterns: Scientific codes being resource hungry applications, when execute on a computing cluster, will tend to exploit all the available resources. The performance governing factors like number of processors/cores, size of memory, type of interconnect, etc., depend on the budget available for building any computing cluster. After the hardware components are bought and cluster is built, optimization of algorithm and granularity of parallelism play important role in improving performance of the application. Many a times source codes of scientific applications are not available due to various reasons. It may be old legacy application or buying source code for commercially available application may be prohibitive in terms of high cost. This leaves little scope for performance improvement by changing these parameters. There is lot of scope for performance improvement of HPCC running multiple scientific codes simultaneously by optimally distributing the processes on available nodes/processors/cores. Process distribution among available nodes and cores to maximize throughput and in turn improve overall cluster performance is an area to be explored and investigated.

5 Models for Understanding HPCC Performance for Scientific Applications

Process Distribution Model: When computation in form of a software application is to be executed on a cluster, its resource requirement is assessed in terms of number of cores required, memory required, scratch area requirement, etc. Based on the assessment of requirement, applicability of available resources is carried out. Figure 1 shows a cluster having six nodes and each node is having four cores, thus total 24 cores are available for carrying out computational tasks.

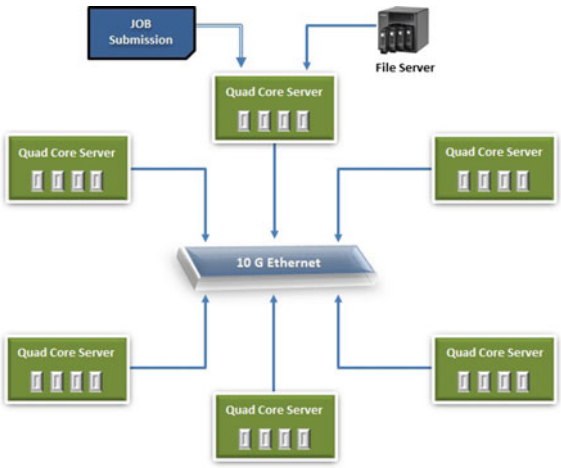

Fig. 1 Process distribution model—six node cluster with 24 cores

In this basic model to understand process distribution, processes are created increasingly from 6 to 36, while keeping the task size constant [8]. The cluster peak computing power increases as we deploy more cores and processors for the computational task [9]. The peak computing power deliverable by the cluster will be achieved when maximum number of available cores matches number of processes created for the computational task. Practically achieved peak computing power in each case is recorded for analysis and further explorations [8]. Using the 'process distribution model,' complexity or the size of the scientific application can be fixed for execution on the available resources.

Processor Core Usage Model: The 'processor core usage model' has been conceptualized for a finer understanding of core usage by computational tasks on clusters. The model can be viewed as two virtual models built over two different clusters differing basically in interconnect speed and also processor performance. This approach has been followed for comparative analysis purpose [10]. The model presented in previous section uses 10 G Ethernet connectivity for internode communication. To carryout comparative performance evaluation, another model built using Infiniband connectivity for internode communication is proposed. Physical realization of the model has been achieved by configuring hardware components, and the model is presented in Fig. 2.

The interconnect used to build this model is 20 Gbps Infiniband. To carryout comparative analysis, these two models are loaded with same task size and identical process creation is done for one-to-one comparative analysis [10]. The task size N in the HPL is varied in moderate range of 10,000–40,000 in steps of 10,000. A number of processes are created as a set of 1, 2, or 4 for each comparison. By using the advanced technique of HPL for distributing created multiprocesses among cores and processors, true results are obtained. For comparative analysis purpose, identical distribution of tasks on nodes and cores has been carried out on the two cluster models.

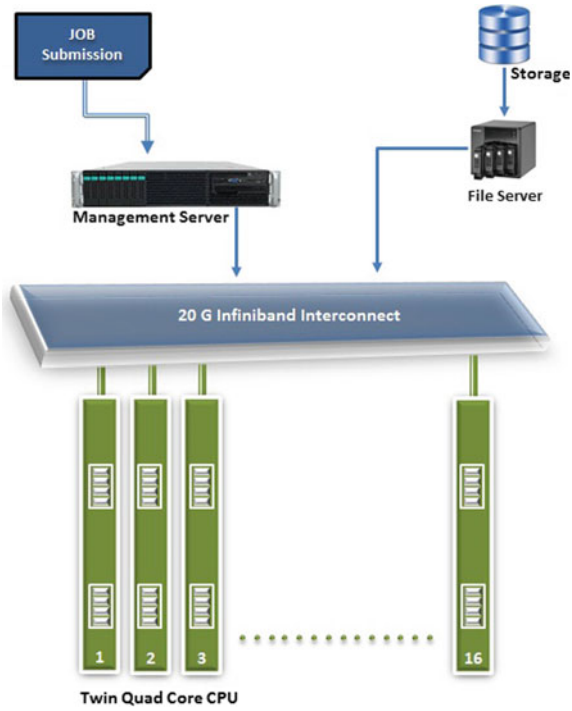

Fig. 2 Processor core usage model

In these experiments, performance is measured in terms of computing power delivered, and focus is also drawn on execution time. In both the options of this model, the created processes are forced for execution on a single processor, thus enabling interprocess communication bound to be within the processor itself. The approach uses all the cores available in the processor. The model is also investigated by distributing same size task on cores on different processor.

Figure 3 shown below represents how interprocess communication takes place using QPI when four processes are distributed on four cores of same processor. C1, C2, C3, C4 are cores of processor P1, and Proc1, Proc2, Proc3, and Proc4 are processes of an application spawned on four cores of same processor, thus the interprocess communication domain is confined to the processor itself.

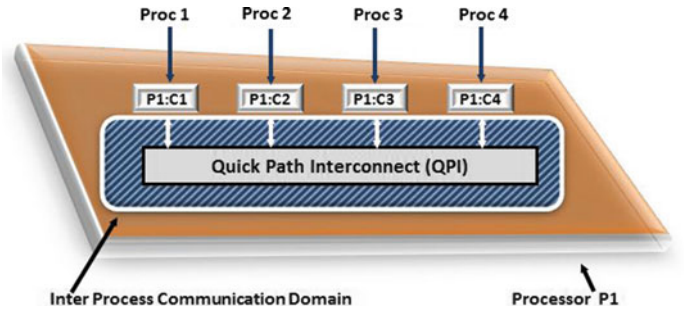

Fig. 3 Interprocess communication through QPI

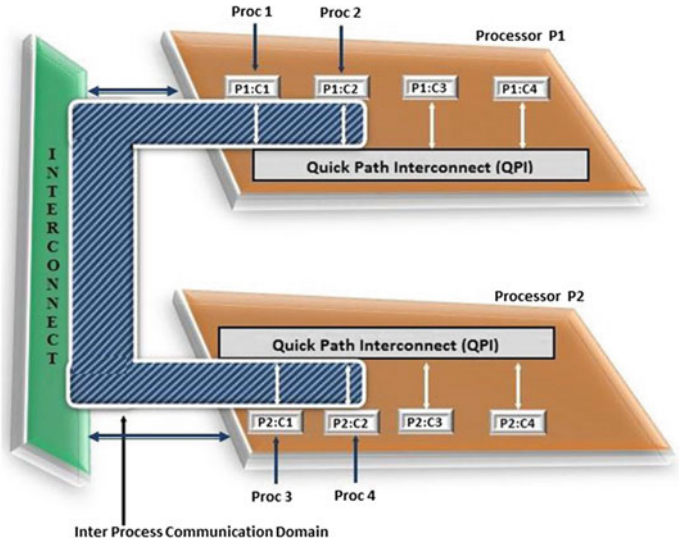

Fig. 4 Interprocess communication through interconnect

If four processes are executed on two cores of two different processors, then the interprocess communication takes place through the interconnect, as depicted in Fig. 4. In this case, Proc1 and Proc2 are spawned on C1 and C2 cores of processor P1, whereas Proc3 and Proc4 are allocated to C1 and C2 cores of processor P2. The interprocess communication domain is not confined to a single processor, instead it is through the interconnect. It is very interesting to carryout analytical studies using this model, and various experimental test runs of HPL provided set of results. Analysis of the result set revealed the behavior of cluster measured in terms of execution time for different process patterns expressed in terms of process per node (PPN). The two models discussed so far gave a clearer understanding of the cluster performance under varying processor core usage and process distribution.

Interconnect Impact Model: This model has been conceptualized to investigate impact of Infiniband interconnect. The model described in previous section is upgraded using 40 Gbps interconnect, and studies are carried out to compare the performance of clusters using 20 and 40 Gbps interconnects. Performance measured with increasing task complexity in case of 20 and 40 Gbps interconnect speed gives a clear picture of the positive impact of interconnect speed. Problem size is increased from 10,000 to 100,000 in steps of 10,000, and 24 cores have been used in three different distributive patterns. Execution time and peak computing power delivered are recorded [11, 12]. The lower latency offered by Infiniband has considerably nullified the bottleneck posed by conventional interconnects [12].

The three models described and discussed from operational requirement point of view have provided a clearer understanding of cluster performance for scientific applications. In the next section, an optimized and comprehensive model for executing multiple scientific applications concurrently on one cluster is proposed and described.

6 L3C Model of HPCC for Scientific Applications

After carrying out investigations on the three different models for scientific applications, it is logical to conceptualize an 'Optimized Model for Scientific Computing.' In a typical scientific computing scenario, the cluster is being used concurrently by multiple users for running their codes. All these scientific applications being resource hungry applications compete for available resources and at times result in resource contention, affecting the performance severely. The proposed model for scientific computing applications can be visualized as collection of multiple virtual clusters.

The current technological trends have enabled deployment of HPCCs having large number of PEs and cores. Each PE can typically have four, six, and even up to eight cores. Each node can have one, two, and even up to four PEs. This high density of PEs and cores in a single node results in availability of large number of cores for computational tasks. Application may need n number of cores, where $n < N$ (total no. of cores in the cluster), leaving some nodes unused if only one application is running. Limitation on usage of cores can come in from multiple fronts. First, there may be limitation in terms of license of the application for number of cores on which the task can be actually distributed. Second, the code may not be scalable beyond a limit and thus may leave certain cores/nodes unused. These unused cores may fruitfully be utilized by other applications.

In the proposed model, multiple scientific applications are made to utilize a subset of cores. For example, if application 1 needs 64 cores while application 2 and 3 require 32 cores each, then these three applications can coexist on the cluster without any degradation of performance. Concurrent execution of these three applications will enhance the overall throughput (along with optimum utilization of available resources) as compared to the situation where they were sequenced for execution one after another.

This concept of deploying multiple applications concurrently is in no way limited in terms of number of such applications. The limitation actually comes from the total number of cores available. All the available cores can be divided in n number of (n_1, n_2, n_3 …) virtual clusters. Number of cores used by each virtual cluster could be different (c_1 cores by n_1 cluster, c_2 cores by n_2 cluster, and so on) as long as $c_1 + c_2 + c_2$ … is not greater than c * n where c * n is total number of cores in the cluster.

This model can thus be termed as a loosely coupled cluster of clusters (L3C model). The model allows on-the-fly creation of clusters of varying size as per the requirement of the applications running concurrently on the cluster. We can visualize the cluster as a loosely coupled cluster of clusters.

Number of 'virtual clusters' created in this model depends on number of applications being deployed and the resources available. Thus, number of virtual clusters can vary from 1 to many, depending on size and complexity of applications being run concurrently. One can deploy multiple applications concurrently on the cluster by dedicatedly assigning/allocating x no. of cores to an application, y no. of cores to another application, and so on. It is already verified in the model (interconnect impact model) presented earlier that the 40 Gbps Infiniband interconnect ensures very high throughput for inter processor communication. This feature provided by Infiniband interconnect is exploited in L3C model to allocate cores to applications irrespective of which processor those cores belong to.

The above diagram depicts the L3C model of HPCC for a typical requirement of four applications running concurrently on HPCC. Each application requires different number of cores as shown in the Fig. 5. CE1 and CE2 are control elements, which are responsible for job distribution (scheduling among cores), access to storage (data and program files), and job submission. In the typical example depicted above, Scientific Application 1 is deployed on core C1–C64, Scientific Application 2 uses core C65 to C96, Scientific Application 3 uses another set of 16 cores from C97–C112, and Scientific Application 4 is being run on core C113–C128.

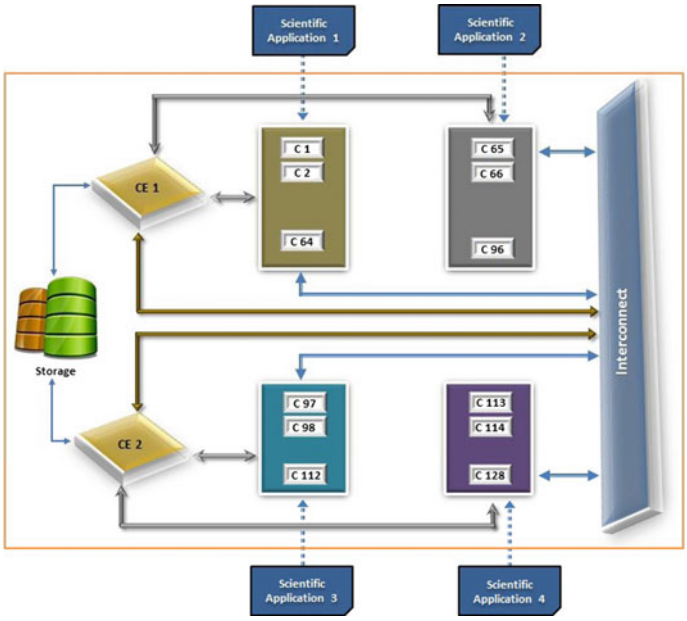

Fig. 5 Loosely coupled cluster of clusters (L3C) model

7 Implications of L3C Model

Concurrent Execution of Scientific Codes: The L3C model proposes to concurrently execute multiple scientific applications or scientific codes on computing cluster, as long as total number of cores required by all the applications put together does not exceed the total number of cores available in the computing cluster. If these multiple tasks are queued up for sequential execution one after another, all available cores may not be utilized all the time, thus resulting in wastage of computational power.

Resource Allocation Approach: A set of scientific applications requiring varied number of cores is listed in Table 1. Estimated execution time is also included in the table, which is only indicative in nature, since actual execution time depends on many other factors like size of data set, result convergence methodology, processing power of the PE, and available memory per core/per PE.

Just for an example, VORPAL, ADF, and WIEN2k_08 can coexist on a computing cluster having 128 cores. Similarly, other combination of applications is also possible, which solely depends on number of available cores, presuming other requirements like memory per core, scratch area, etc., are not constrained.

Table 1 Set of scientific applications and certain related details

Name of the scientific application	No. of cores	Estimated execution time (h)
Amsterdam density functional (ADF)	32	75
WIEN2k_08	32	200
Versatile object-oriented code for relativistic plasma analysis with laser (VORPAL)	64	400
Virtual laser plasma lab (VLPL) code	64	300
Objective ring beam injection and tracking (ORBIT_MPI)	16	24
Crystal	32	350
LS-Dyna	16	20

Optimizing Allocation of Resource: When multiple scientific applications are deployed on a computing cluster, they may end at different time, meaning they may need different execution time. This will surely render some cores free earlier as compared to cores engaged for applications requiring longer execution time. Optimization of resources as proposed in L3C model suggests scheduling those applications from the waiting queue, which can be accommodated without any overlapping of applications on cores. Thus, as soon as a set of cores is free, another application can be scheduled to occupy the free cores. The virtual cluster of clusters as visualized in L3C model thus allows an application requiring very large execution time to coexist with other applications which require comparatively smaller execution times. The state of this virtual cluster of clusters will vary dynamically depending on the scientific applications being executed concurrently at any instance of time.

8 Conclusions

The L3C model conceptualizes the division of cluster into virtual multiple clusters. Since the Infiniband interconnect ensures that there is no constrain on the inter-processor communication, thus allocation of cores to a set of concurrent applications can be done on the fly as per the demand of application, irrespective of whether the cores belong to same processor or not. Concurrent execution of the scientific applications increases HPCC utilization considerably and leads to much enhanced throughput delivered by the computing cluster.

Scientific applications require varied number of cores. Number of cores required for an application depends on the problem size being attempted and also on some other factors like software license, simulation model, etc. This concept of allocating nodes/cores to a set of applications for concurrent execution is a novel approach for enhanced utilization of computing resources.

The comprehensive L3C model proposed for scientific applications is a novel idea ensuring maximum throughput and optimum utilization of computing resources. The concept of deploying multiple applications concurrently has resulted in enhanced throughput of HPCC leading to better return on investment (RoI).

References

1. Alam, S.R., Barrett, R.F., Kuehn, J.A., Roth, P.C., Vetter, J.S.: Characterization of scientific workloads on systems with multi-core processors. In: IEEE International Symposium on Workload Characterization, pp. 225–236 (2006)
2. Dongarra, J., Luszczek, P., Petitet, A.: The LINPACK Benchmark: past, present, and future. Concurrency: Pract. Exp. **15**, 803–820 (2003)
3. Langou, J., Dongarra, J.: The problem with the Linpack benchmark matrix generator. Int. J. High Perform. Comput. Appl. **23**(1), 5–14 (2009)
4. Petitet, R.C., Whaley, J. Dongarra, A.: Cleary, HPL—a Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. Innovative Computing Laboratory, Computer Science Department, University of Tennessee, September 2008
5. Buyya, R. (ed.): High Performance Cluster Computing: Architectures and Systems, vol. 1. Prentice Hall PTR, NJ (1999) ISBN: 0-13-013784-7
6. Buyya, R. (ed.): High Performance Cluster Computing: Programming and Applications, vol. 2. Prentice Hall PTR, NJ, USA (1999) ISBN: 0-13-013785-5
7. Hwang, K., Dongarra, J., Fox, G.: Distributed and Cloud Computing, 1st edn. Morgan Kaufmann (2011)
8. Rajan, A., Joshi, B.K., Rawat, A.: Critical analysis of HPL performance under different process distribution patterns. In: CSI 6th International Conference on Software Engineering (CONSEG 2012), Devi Ahilya Vishwavidyalaya (DAVV), Indore, MP, India, 5–7 Sept 2012
9. Vaidya, M.: Parallel processing of cluster by map reduce. Int. J. Distrib. Parallel Syst. (IJDPS) **3**(1), 167 (2012)
10. Rajan, A., Joshi, B.K., Rawat, A.: Analysis of process distribution in HPC cluster using HPL. In: The Second IEEE International Conference on Parallel, Distributed and Grid Computing 2012 (PDGC 2012), Jaypee University of Information Technology, Solan, HP, India, 6–8 Dec 2012
11. Rajan, A., Joshi, B.K., Rawat, A.: Analytical studies of peak computing power deliverable by small and mid size HPCC. In: INDIACom 2013—7th International Conference on 'Computing for Nation Development', BVICAM, New Delhi, 7–8 Mar 2013
12. Rajan, A., Joshi, B.K.: Performance comparison of 20 Gbps and 40 Gbps Infiniband Interconnect. In: IEEE International Conference on Global Sustainable Development (IndiaCom 2014), BVICAM, pp. 5–6, New Delhi, Mar 2014