Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2014, Article ID 128438, 9 pages
http://dx.doi.org/10.1155/2014/128438

Research Article

A Comparative Analysis of Performance of Shared Memory Cluster Computing Interconnection Systems

Minakshi Tripathy[1] and C. R. Tripathy[2]

[1]*Department of Computer Science, Bidya Computer Education & Development, Chhend, Rourkela, Odisha 769015, India*
[2]*Department of CSE & A, VSS University of Technology, Burla, Sambalpur, Odisha 768018, India*

Correspondence should be addressed to Minakshi Tripathy; minakshiom@gmail.com
and C. R. Tripathy; minakshiom@yahoo.co.in

Received 10 June 2014; Revised 16 October 2014; Accepted 10 November 2014; Published 9 December 2014

Academic Editor: Eduardo da Silva

In recent past, many types of shared memory cluster computing interconnection systems have been proposed. Each of these systems has its own advantages and limitations. With the increase in system size of the cluster interconnection systems, the comparative analysis of their various performance measures becomes quite inevitable. The cluster architecture, load balancing, and fault tolerance are some of the important aspects, which need to be addressed. The comparison needs to be made in order to choose the best one for a particular application. In this paper, a detailed comparative study on four important and different classes of shared memory cluster architectures has been made. The systems taken up for the purpose of the study are shared memory clusters, hierarchical shared memory clusters, distributed shared memory clusters, and the virtual distributed shared memory clusters. These clusters are analyzed and compared on the basis of the architecture, load balancing, and fault tolerance aspects. The results of comparison are reported.

1. Introduction

The cluster computing is becoming increasingly popular. The latest technological developments and research innovations are pushing clusters into mainstream computing. This poses a number of new research challenges that need to be addressed [1].

Parallelism has been employed for many years, mainly in high performance computing. Parallel computing has become the paradigm in computer architecture, in the form of clusters that use multiple computers to work on the same task. A cluster is a type of parallel system that consists of a collection of interconnected computers used as a single unified computing resource. When failure occurs in a cluster, resources can be redirected and the workload can be redistributed through appropriate scheme of load balancing [2].

The collection of several servers in a single unified cluster makes it possible to share a computing load. If any resource in a cluster server fails, the clusters as a whole can continue to offer service by using resources on other cluster servers regardless of whether the failed component is hardware or software. In other words, when a resource fails, users connected to the server cluster may experience temporarily degraded performance but do not completely lose access to the service [2, 3]. The nodes of a cluster are connected each maintaining its own separate processor, memory, and operating system. Special communication protocols and system processes bind these nodes together and allow them to cooperate to provide outstanding levels of availability [4, 5].

The cluster management software provides services like failure detection, recovery, load balancing, and the ability to manage the servers as a single system. This system level software monitors local hardware and software subsystems, tracks the states of the nodes, and quickly responds to failures in a way that eliminates or minimizes application's downtime and provides a number of important other benefits including improved availability, easier manageability, and cost-effective scalability [3, 6].

The fault tolerant cluster is basically an open and distributed system, flexible and extensible, and its architecture

offers compliance to the principle of reusability and interoperability. The architecture of the cluster systems should be flexible enough to handle any component or subcomponent that is required to make the system. The fault tolerance in the clusters provides error information in real time and makes it possible to detect which part or component needs repair and accordingly initiates a service call to identify the parts needed for maintenance [5–11].

A shared memory system can be simultaneously accessed by multiple programs to provide communication among them and to avoid redundant copies. It refers to a large block of memory that is accessed by several processors. A shared memory cluster system is relatively easy to program since all processors share a single view of data and the communication between them can be as fast as memory accesses to a particular location. The distribution of tasks among clusters improves load balancing and provides better control of the use of communication resources [12, 13]. With the increasing performance needs, clusters provide additional mechanism to take advantage of aggregate performance of the participating nodes through different types of cluster architectures. The various important classes of shared memory clusters studied in the literature are shared memory clusters [4], hierarchical shared memory clusters [7], distributed shared memory clusters [9], and virtual distributed shared memory clusters [14].

Each of the shared memory cluster computing interconnection systems has its own advantages and limitations. Therefore, in order to make a fair choice for any particular application, it is very much essential to compare their performance on a common platform. The various desirable performance measures for comparison of clusters considered are availability, reliability, scalability, manageability, load balancing, fault tolerance, execution time, latency, bandwidth, communication, access, and time complexity. The main goal of cluster computing interconnection system is to reduce the execution time for better performance.

The principal objective of this paper is to compare some of the important performance parameters of the above said cluster computing interconnection systems. The rest of the paper is organized as follows. Section 2 describes the main characteristics of shared memory clusters. Section 3 describes the performance comparison of these clusters. Section 4 discusses the results of comparison and provides a brief summary of the important findings. Finally, the conclusions are drawn in Section 5.

2. Main Characteristics of Shared Memory Clusters

Shared memory clusters are tightly coupled clusters with attractive simple programming model. Tightly coupled clusters work together closely so that in many aspects they can be viewed as a single computer. The shared memory cache facilitates interprocess communication to exchange data between programs running at the same time [5]. The main characteristics of a shared memory cluster are

(i) efficient large-scale parallelism due to dynamic system decomposition of clusters,

(ii) efficient intercluster communication based on switching processors between clusters,

(iii) efficient intracluster communication based on multiple data reads on the fly,

(iv) efficient cache support to keep the overheads low.

A shared memory cluster system needs less expensive interconnection network compared to that of a non-cluster-based system. In this development the hierarchical shared memory cluster system prevails due to its hardware simplicity and cost-effectiveness. Several shared memory cluster systems with large number of processors have been designed using hierarchy of processors in clusters [7, 8]. The hierarchical shared memory cluster system efficiently supports the hierarchy of memories using a common memory. The advantages of hierarchical shared memory clusters are as follows.

(i) Hierarchical shared memory cluster significantly reduces the impact of inclusion of memory hierarchy that exploits program locality at all levels in the hierarchy.

(ii) A support for adaptive cache coherence to improve the performance of parallel applications and this approach is both manageable and advantageous.

(iii) The use of relaxed cache consistency models avoids the adverse performance impact of unnecessary invalidation and synchronization.

(iv) It supports the merging of access requests within the memory hierarchy.

(v) Hierarchical shared memory clusters are scalable up to very large processors that are an order of magnitude larger than the scalability of shared memory shared bus architecture.

The distributed shared memory clusters are loosely coupled distributed systems that have evolved using message passing as the main paradigm for sharing information. Distributed shared memory clusters are heterogeneous parallel distributed systems that enable sharing, selection, and aggregation of geographically distributed autonomous resources. The distributed shared memory cluster provides an abstraction to give the impression of a single monolithic memory, as in a traditional Von Neumann architecture [10, 11, 15]. The main characteristics of a distributed shared memory cluster are following.

(i) Communication across the interconnection network is achieved by read/write abstraction that simplifies the task of the programmer.

(ii) A global address space is provided for data movement across clusters using passing by reference.

(iii) While a data block is moved, the distributed shared memory cluster exploits locality of reference to reduce the communication overhead.

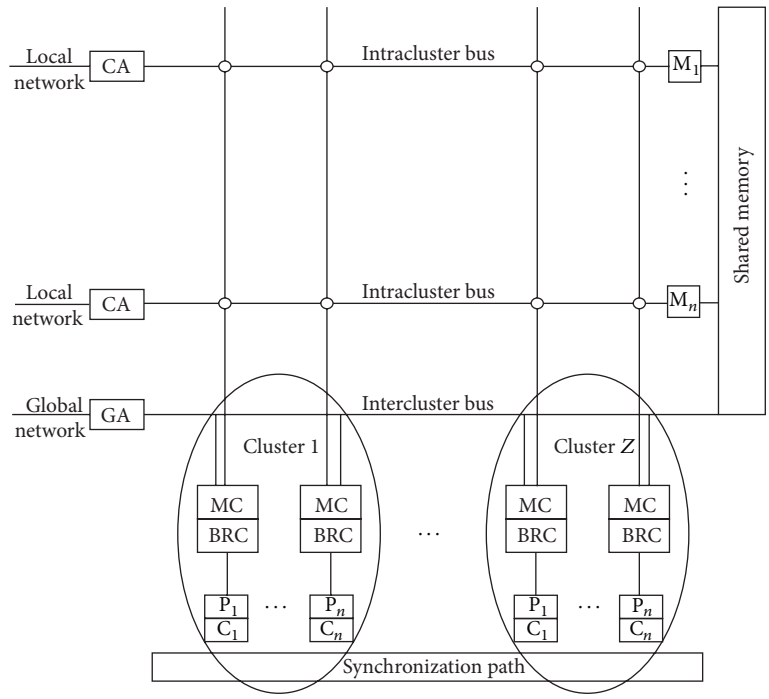

Figure 1: Shared memory cluster architecture (SMC).

(iv) Distributed shared memory cluster provides portability of programs independent of operating system and other low level system characteristics.

(v) Distributed shared memory cluster provides an abstraction of a virtual memory.

Modern cluster systems rely upon the virtual machines. Virtual shared memory cluster systems support the creation and management of virtual clusters. The virtual shared memory cluster is a shared memory abstraction implemented over distributed shared memory. Virtual shared memory clusters moved from tightly coupled relationship between logical and physical resources to more flexible, abstracted relationship where physical resources are allocated as needed [16, 17]. The advantages of virtual shared memory clusters are as follows.

(i) In virtual shared memory clusters, a guest operating system called virtual machine inside the main operating system is known as host sharing the computing resources from processor to memory.

(ii) It is good for testing and learning of software or virtual network inside a virtual machine. If the virtual machine ever crashes operating system, then the main operating system is not affected but only the virtual machine would be.

(iii) A server with lots of computing resources can be divided and resold as virtual server. The virtual server gets a slice of computer resources and if the virtual server shuts down abnormally, the machine would not be affected as they only have access to their virtual machines.

(iv) For more than one monitor, each virtual machine can be assigned to a monitor so that each virtual machine has its own monitor.

(v) Virtual machines are portable.

(vi) Virtual machines provide greater resource allocation flexibility and improve the utilization efficiency.

3. Performance Comparison of Shared Memory Clusters

This subsection is concerned with the study and comparison of performance of different types of shared memory cluster computing interconnection systems. The basic motivation is to compare the performance of shared memory cluster [4], distributed shared memory cluster [9], hierarchical shared memory cluster [7], and the virtual distributed shared memory cluster computing interconnection systems [14]. In this subsection, the above said four types of shared memory clusters are compared on three common platforms: (i) cluster architecture, (ii) load balancing, and (iii) fault tolerance.

3.1. Comparison of Cluster Architectures. The dynamically reconfigurable shared memory cluster computing interconnection system architecture (SMC) proposed in [4] uses *communication on the fly* technique, which strongly speeds up the communication. The shared memory cluster architecture is shown in Figure 1. It is composed of a number of processors (Pi), a memory controller (MC), a set of data memory modules, a set of caches (Ci), and a set of buses. A memory controller arbitrates accesses to a memory module through the intercluster bus and intracluster buses. All the data

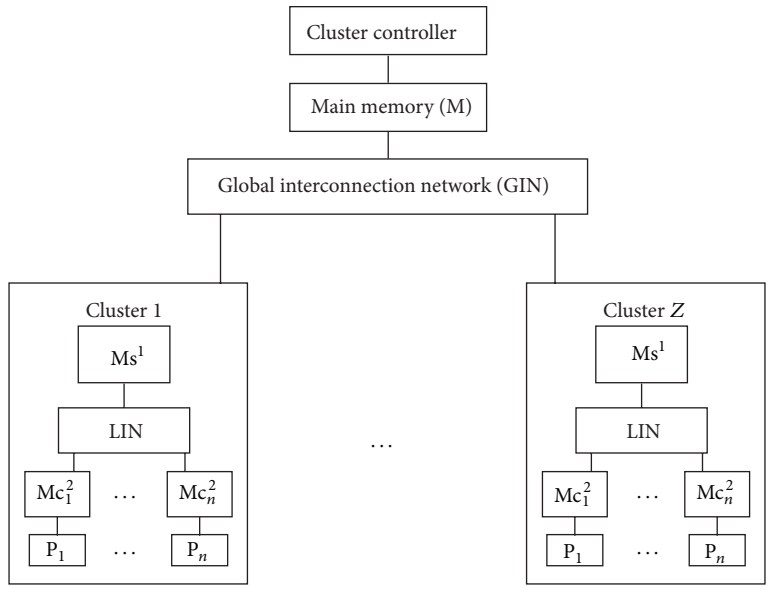

Figure 2: Hierarchical shared memory cluster architecture (HSMC).

memory modules are placed in the shared address space. All the processors attached to the intracluster bus of a data memory module constitute a processor cluster. Each processor in a cluster is equipped with a bus request controller (BRC). An arbiter selects the highest priority level request. At first the writes are examined and if there is no write then, it reads in the intercluster bus arbiters. Then, the arbiter allows a processor's bus request controller to perform the transmission. The proposed shared memory cluster interconnection architecture consists of dynamically reconfigurable shared memory clusters, which can dynamically adjust to computation and communication requirements. To control communication in the shared memory clusters, data prefetch, write, reads on the fly, and processor switching between clusters and communication on the fly operations are used. A read on the fly consists of capturing data written by a processor in a cluster. The processor switching between clusters involves disconnecting a processor from a cluster and connecting it to another cluster. A processor switched to a cluster brings new data in its cache for the target cluster. Processor switching with reads on the fly is called *communication on the fly* [4–6]. Thus *communication on the fly* eliminates many data transactions and decreases the overall interprocessors and memory traffic.

The hierarchical shared memory cluster computing interconnection system architecture (HSMC) uses *hierarchical memory* that reduces global and local intra- and intercluster communication [7]. The hierarchical shared memory cluster architecture is shown in Figure 2. It mainly consists of local interconnection network and global interconnection network [8]. The local interconnection network connects processors of each cluster to the cluster's shared memory. The global interconnection network connects the shared memory of each cluster to the global main memory at the top level of hierarchy. Cluster shared memory (Ms1) is globally shared by processors in each cluster at the 1st level of hierarchy. Local private caches (Mc2) are associated with each processor at the 2nd level of hierarchy. The bandwidth of the global interconnection network supports the intercluster traffic. Similarly, the bandwidth of each local interconnection network is sufficient enough for intracluster traffic. Thus the architecture effectively handles the available bandwidths at global and local levels of the cluster.

The distributed shared memory cluster computing interconnection system architecture (DSMC) uses *data structure in the linked base type* that allows data and resource sharing in an effective manner [9, 18]. The distributed shared memory cluster architecture is shown in Figure 3. In this architecture, each cluster contains local distributed shared memory (LDSM), an intercluster controller (ICCL), an intercluster cache (ICC), processors with private caches, and a shared local bus. The private caches attached to the processors are meant for reducing the memory latency. The LDSM of each cluster is partially or entirely mapped to the globally distributed shared memory (GDSM). Regardless of the network topology, a specific ICCL is required to connect a cluster into the system. The LDSM reduces memory contention and improves data locality. The ICC facilitates data sharing among the clusters utilizing data locality. It contains data that are usually referenced by the intracluster processors. The local bus acts as an intraconnection network among intracluster processors, ICC, and LDSM, while the global bus acts as an interconnection network among intercluster nodes, intercluster interconnection network, and GDSM. Information about states or current locations of particular data blocks and the task scheduling queues are kept in the data structure (DS). On distributed shared memory cluster architecture, 2D cyclic distribution method is used to map data blocks in the form of matrix to different processes [9–11]. Block data layout is a technique used to improve memory

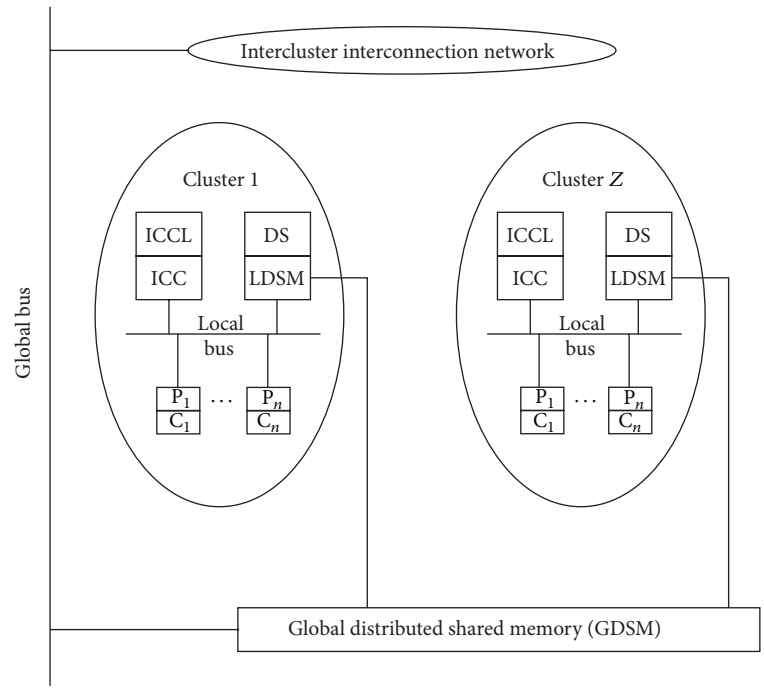

Figure 3: Distributed shared memory cluster architecture (DSMC).

hierarchy performance. A task is bind to its output block such that the computation is centered on data to minimize data movement and maximize data locality.

The virtual distributed shared memory cluster computing interconnection system architecture (VSMC) proposed in [14] uses virtual machines management techniques from virtual server that allows better performance of virtual clusters. The architecture provides an abstraction of logical clusters called virtual clusters consisting of virtual machines on available physical clusters consisting of physical processors. The virtual distributed shared memory cluster architecture uses *virtualisation* technology with virtual machine and virtual cluster to meet the virtual memory management requirement [14, 19]. The virtual distributed shared memory cluster architecture is shown in Figure 4. It is designed as a flexible platform for constructing virtual clusters (VC) with virtual machine management (VMM). The system supports dynamic creation and management of VCs on a physical cluster. It aggregates many virtual machines (VM) in a VC. Each node of a VC is connected to a private virtual network and to the underlying physical network. The running VC shares the physical resources according to a creation time mapping onto the physical cluster. The VCs may be reallocated by means of the run time migration of the VM between the physical nodes. One local node manager (LNM) runs on each physical node and interacts directly with the virtual machine monitor (VMM) to perform management operations such as creating, deleting, and migrating VMs on behalf of the controller. The LNM also collects resource usage data from the VMM and monitors local events. Next, the LNM reports resource usage updates and events back to the controller. The controller is the central component of the system. It communicates with the LNMs to collect usage data and manage VMs running on each physical node. The architecture provides an environment that creates, configures, monitors, and controls the placement, scheduling, and migration of virtual machines.

3.2. Comparison of Load Balancing in Shared Memory Clusters. For the load balancing, the shared memory cluster computing interconnection system uses *centralized dynamic load balancing* technique through worker manager model with *master slave* paradigm [20]. A central manager collects load information of each node and performs workload distribution among the processors at run time. It minimizes the response time of job and average load of the system giving high speedup. The various load balancing activities in master slave paradigm are determination of slave node, selection of task, positioning of task, and collection of information. In the first phase, the master and slave node for task migration are determined. Then in the selection phase, the master selects the most suitable task of a slave for efficient and effective load balancing. In the positioning phase, task's receivers are determined. Finally, the last phase decides where and how to collect information.

The load balancing in hierarchical shared memory cluster computing interconnection system uses *hierarchical load balancing* concept and follows local and global load balancing in different hierarchical levels [21]. It focuses on reducing execution time and redistribution cost while distributing workload with quality load balancing. The load balancing processes in hierarchical shared memory cluster are carried out in two phases: global and local. In local load balancing an overloaded processor transfers its excess workloads to

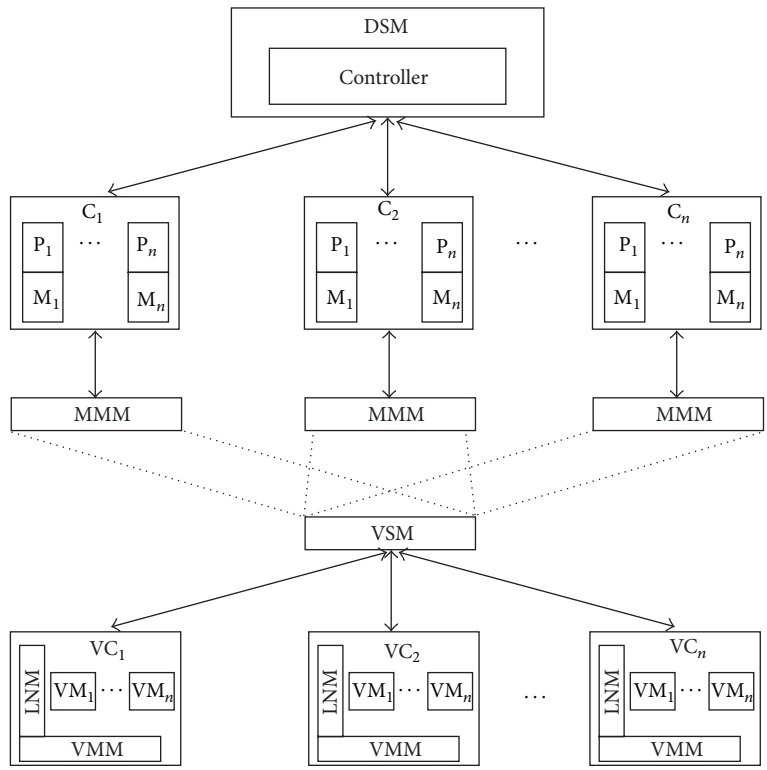

Figure 4: Virtual distributed shared memory cluster architecture (VSMC).

an underloaded processor to distribute the workload evenly and equally among the processors of a cluster. In global load balancing if imbalance is observed among clusters, load redistribution is performed. For the global balancing, the major challenge is to effectively reduce the redistribution cost.

The proposed distributed shared memory cluster computing interconnection system uses *distributed dynamic load balancing* technique with *work stealing* [22]. In *work stealing* when the system has a large number of processors with many jobs running, the idle processors steal tasks from busy processors for balancing of workload. *Work stealing* improves execution time and efficiency of the system. Under work stealing, whenever a processor runs out of work, it becomes a *thief* and attempts to *steal* task from another processor called *victim*. Idle processors select victim processors at random to steal work from them maximizing the availability of work. The load balancing activities in distributed shared memory cluster are collecting information, task distribution, and redistribution of task. In the first phase, the master collects the slave node status for task assignment. In the distribution phase, the master distributes task based on work stealing. In the last phase, the redistribution is performed by transferring tasks from heavily loaded processors to lightly loaded processors.

The load balancing technique in virtual distributed shared memory cluster computing interconnection system uses *centralized dynamic load balancing* principles with virtual cluster server [23]. It focuses on the concept of *on the fly* that can add virtual machines and resources any time with the ongoing activities of the system. The virtual load balancing provides support for multiple concurrently executing virtual machines to utilize cluster wide memory. It also supports seamless migration of virtual machines with large memory workloads.

3.3. Comparison of Fault Tolerance in Shared Memory Clusters. For the fault tolerance, the shared memory cluster computing interconnection system uses *shared memory checkpoints* that reduce task migration overhead and avoid establishing communication with failed processes [24]. It performs data and cluster availability to achieve high performance and reliability. In the first phase, an analytical model is used to describe the system's response to fault along with the mean time to failure and mean time to repair. It uses high availability techniques to regain the lost availability. In the second phase, fault tolerance is frequently implemented through periodic checkpointing. Whenever a node fails, the jobs running on it are stopped and restarted on a different node from the most recent checkpoint.

For the fault tolerance, the hierarchical shared memory cluster computing interconnection system uses hierarchical fault tolerant model with *hierarchical checkpointing* and *recovery* scheme [25]. The *hierarchical checkpointing* and *recovery* is performed through local disk, mirrored and stable storage checkpointing in different levels of hierarchy. This scheme handles several types of failures improving the overall system availability. The hierarchical fault tolerant model is based on the hierarchical interconnection network. A fault tolerant analysis has been made under fault free and faulty condition to give an overall analysis of the hierarchical shared

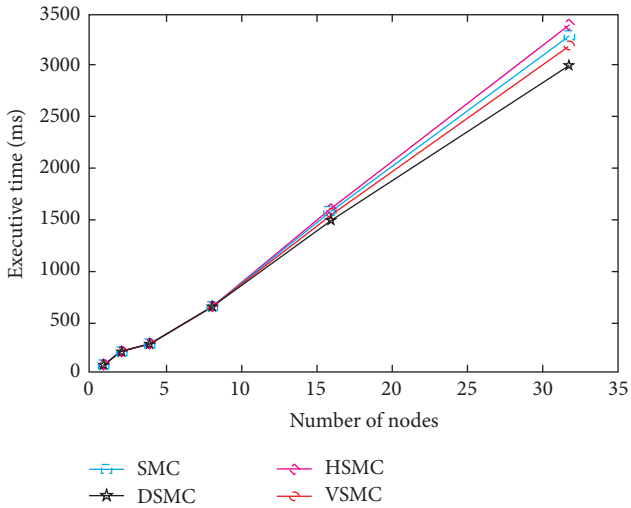

Figure 5: Comparison of cluster architecture.

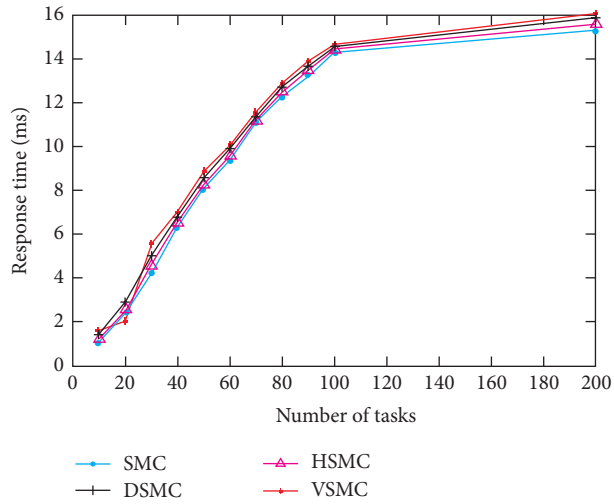

Figure 6: Comparison of load balancing in clusters.

memory cluster system based on availability and bandwidth. The system availability produces useful information about the available request rates at the inputs of clusters at a given level of hierarchy. The bandwidth availability is the expected value of available bandwidth of the system at a particular time.

For the fault tolerance, the distributed shared memory cluster computing interconnection system uses *time based coordinated checkpointing* and *rollback recovery* that satisfies consistency and recoverability properties [18]. It leads to better performance, requires less space in stable storage, is easier to implement, and obtains a predictable rollback distance. With coordinated checkpointing and rollback recovery, the fault tolerance brings a distributed shared memory cluster system to a consistent state after failures. The *time based coordinated checkpointing* and *rollback recovery* relies upon the approximately synchronized clocks. It creates checkpoints periodically when the local clock arrives at checkpoint time. To recover from failure, the system rolls back to the last stored global state and starts reexecution from there. This scheme achieves failure free execution and avoids message coordination overheads.

For the fault tolerance, the virtual distributed shared memory cluster computing interconnection system uses *replicated* and *distributed checkpointing* with virtual cluster server and virtual machines [26, 27]. The checkpoints stored in distributed file system guarantee system recovery when a failure occurs with robustness and ease of use. The virtual shared memory cluster system handles the virtual fault tolerant model with virtual server. This model achieves data replication in the virtual machine and transparently recovers from faults. The checkpointing and recovery scheme is integrated with distributed file system to store checkpoints. The virtual fault tolerant model allows the system to restart computation by restoring the replicated checkpoints from other nodes without a global restart of the system.

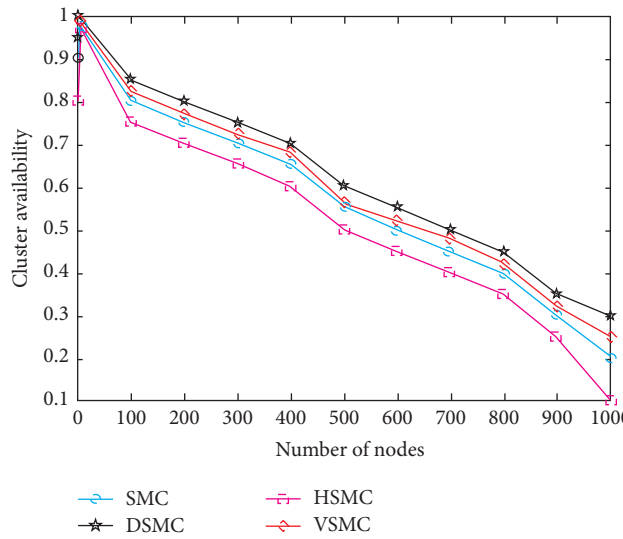

Figure 7: Comparison of fault tolerance in clusters.

4. Results and Discussions

This section presents the results of comparison of the above said four types of cluster computing interconnection system based on architecture, load balancing, and fault tolerance and the results are depicted in Figures 5–7, respectively. For the purpose of experiments on the cluster computing interconnection systems, 4–64 number of clusters are assumed with 1–1000 number of nodes, 1–100 number of processes, and 1–10 number of faults. The numbers of checkpoints are generated randomly where the checkpoint size and checkpoint interval is set in between 100–1000 ms, respectively. The results were obtained using the average of 5–10 experiments using MATLAB for final results and comparison. MATLAB is a numerical computation and simulation tool developed into commercial tool with a user friendly interface. It is

Table 1: Summary of performance comparison of cluster computing interconnection systems.

Parameters	SMC	DSMC	HSMC	VSMC
Network architecture	Intra- and intercluster bus	Interconnection network scheme	Intraconnection network hierarchical bus	Virtual network scheme
Memory access	All nodes access global shared memory with local cache for each	Each node has local memory with large global distributed shared memory	All nodes access global shared memory with hierarchical cache for each	Each node has local memory with large distributed and virtual memory
Communication	Explicit	Implicit	Explicit	Implicit
Data sharing	Fine grain between processes	Coarse grain process migration	Fine grain between processes	Coarse grain process migration
Abstraction	No abstraction	Simple abstraction	No abstraction	Full abstraction
Availability	High	Higher	Less	Same as that of physical cluster
Architecture	Communication on the fly	Distributed data structure in linked base type	Hierarchical memory	Virtual machine monitor with virtual cluster
Load balancing	Centralized dynamic load balancing	Distributed dynamic load balancing	Hierarchical load balancing	Virtual cluster server centralized dynamic load balancing
Fault tolerance	Shared memory checkpointing	Time based coordinated checkpointing	Hierarchical checkpointing	Replicated and distributed checkpointing
Scalability	Limited scalability	Highly scalable	Hard to scale	Good scalability
Latency	Lower	Higher	Low	High
Bandwidth	Higher	Lower	High	Low
Programming	Easier to program	Easy to program	Critical to program	Hard to program
End user	Difficult to use	Easier to use	Hard to design and use	Easy to use
Execution	Fastest	Fast	Faster	Moderate fast
Cost	Cheapest	Moderate	Cheap	High

an independent programming language with a library of mathematical functions capable of treating complex problems. Figure 5 compares the four types of cluster architectures based on their execution times. The results infer that the distributed shared memory cluster has the highest scalability with fast execution. Figure 6 shows the comparison of load balancing in different types of shared memory clusters. We compare the number of tasks with response time in Figure 6. The shared memory cluster is found to be the best as compared to three others in terms of load balancing. Figure 7 shows the comparison of fault tolerance in different types of clusters. There, we compare the number of nodes with cluster availability. As shown in Figure 7, the distributed shared memory cluster provides the high availability. In virtual shared memory cluster, virtual machines may not handle the load when all physical nodes stop working at the same time. Hence, virtual shared memory cluster provides almost the same availability of the underlying physical cluster connected through the distributed shared memory cluster environment. Table 1 shows the summary of comparison of performance of the four types of shared memory clusters.

5. Conclusions

In this paper, a comparative analysis of four different types of shared memory cluster computing systems is made. Our comparison mainly concentrates on the cluster architecture, load balancing, and fault tolerance aspects of the above said clusters. The important findings are presented.

The hierarchical shared memory cluster is observed to support flexible form of communication through shared memory. It also performs much better and enjoys high availability. The distributed shared memory clusters intelligently balances the load among nodes with the work stealing approach. The virtual distributed shared memory clusters hide dynamic changes of physical hardware configuration of underlying distributed shared memory cluster. This architecture also allows better fault tolerance of the system.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] I. Ahmad, "Cluster computing: a glance at recent events," *IEEE Concurrency*, vol. 8, no. 1, pp. 67–69, 2000.

[2] C. Kobhio, S. Pierre, and A. Quintero, "Redundancy schemes for high availability computer clusters," *Journal of Computer Science*, vol. 2, no. 1, pp. 33–47, 2006.

[3] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing systems," in *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 249–258, June 2006.

[4] M. Tripathy and C. R. Tripathy, "Design and analysis of a dynamically reconfigurable shared memory cluster," *International Journal of Computer Science and Network Security*, vol. 10, no. 9, pp. 145–158, 2010.

[5] J. Protic, M. Tomasevice, and V. Milutinoviec, "A survey of shared memory," in *Proceedings of the 28th annual Hawaii International Conference of System Sciences*, pp. 74–84, January 1995.

[6] M. Tudruj and L. Masko, "Communication on the fly and program execution control in a system of dynamically configurable SMP clusters," in *Proceedings of the 11th Euro Micro Conference on Parallel and Network Based Processing*, pp. 67–74, February 2003.

[7] M. Tripathy and C. R. Tripathy, "A hierarchical shared memory cluster architecture with load balancing and fault tolerance," *International Journal of Computer Application*, vol. 25, no. 6, pp. 8–18, 2011.

[8] Y.-J. Oyang, D. Jinsung Sheu, C.-Y. Cheng, and C.-Z. Yang, "The M^2 hierarchical multiprocessor," *Future Generation Computer Systems*, vol. 9, no. 3, pp. 235–240, 1993.

[9] M. Tripathy and C. R. Tripathy, "A distributed shared memory cluster architecture with dynamic load balancing," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 2, pp. 231–240, 2012.

[10] S. Zhou, M. Stumm, K. Li, and D. Wortman, "Heterogeneous distributed shared memory," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, pp. 540–554, 1992.

[11] J.-C. Ueng, C.-K. Shieh, T.-Y. Liang, and J.-B. Chang, "Proteus: an efficient runtime reconfigurable distributed shared memory system," *The Journal of Systems and Software*, vol. 56, no. 3, pp. 247–260, 2001.

[12] K. J. Barker, K. Davis, A. Hoisie et al., "Using performance modeling to design large-scale systems," *Computer*, vol. 42, no. 11, Article ID 5331904, pp. 42–49, 2009.

[13] B. Gopal and P. Kumar, "Framework for improving parallelism by write-update coherence protocol in distributed shared memory system," *International Journal of Recent Trends in Engineering*, vol. 1, no. 2, pp. 201–204, 2009.

[14] M. Tripathy and C. R. Tripathy, "On a virtual shared memory cluster system with virtual machines," *International Journal of Computer and Electrical Engineering*, vol. 3, no. 3, pp. 754–761, 2011.

[15] M. Protic and V. Tomasevic, "An overview of distributed shared memory," *IEEE Computer Journal*, vol. 24, no. 8, pp. 12–50, 1991.

[16] C. Clark, K. Fraser, S. Hand et al., "Live migration of virtual machines," in *Proceedings of the 2nd Symposium on Networked Systems Design & Implementation (NSDI '05)*, pp. 273–286, May 2005.

[17] K. Li and P. Hudak, "Memory coherence in shared virtual memory systems," *ACM Transactions on Computer Systems*, vol. 7, no. 4, pp. 321–359, 1989.

[18] M. Tripathy and C. R. Tripathy, "A new co-ordinated checkpointing and rollback recovery scheme for distributed shared memory clusters," *International Journal of Distributed and Parallel Systems*, vol. 2, no. 1, pp. 49–58, 2011.

[19] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker, "Usher: an extensible framework for managing clusters of virtual machines," in *Proceedings of the 21st Large Installation System Administration Conference*, pp. 167–181, November 2007.

[20] H. Zhang, "On load balancing model for cluster computers," *International Journal of Computer Science and Network Security*, vol. 8, no. 10, pp. 1590–1602, 2008.

[21] Z. Lan, V. E. Taylor, and Y. Li, "DistDLB: improving cosmology SAMR simulations on distributed computing systems through hierarchical load balancing," *Journal of Parallel and Distributed Computing*, vol. 66, no. 5, pp. 716–731, 2006.

[22] J. Dinan, D. B. Larkins, P. Sadayappan, S. Krishnamoorthy, and J. Nieplocha, "Scalable work stealing," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09)*, pp. 45–54, November 2009.

[23] M. R. Hines and K. Gopalan, "MemX: Supporting large memory workloads in Xen virtual machines," in *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing (VTDC '07)*, pp. 1–8, Reno, Nev, USA, November 2007.

[24] H. Naik, R. Gupta, and P. Beckman, "Analyzing checkpointing trends for applications on the IBM Blue Gene/P system," in *Proceedings of the 38th International Conference Parallel Processing Workshops (ICPPW '09)*, pp. 81–88, September 2009.

[25] A. A. Veglis and A. S. Pombortsis, "Performance related analysis of L-level hierarchical shared - memory multiprocessors," in *Proceedings of the 8th Mediterranean Electrotechnical Conference (MELECON '06)*, pp. 1055–1059, Bari, Italy, May 1996.

[26] K.-L. Wu and W. K. Fuchs, "Recoverable distributed shared virtual memory," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 460–469, 1990.

[27] Í. Goiri, F. Julià, J. Guitart, and J. Torres, "Checkpoint-based fault-tolerant infrastructure for virtualized service providers," in *Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS '10)*, pp. 455–462, Osaka, Japan, April 2010.