A Hierarchical Shared Memory Cluster Architecture with Load Balancing and Fault Tolerance

Minakshi Tripathy.
Department of Computer Science,
Sambalpur University, Burla,
Orissa, India.

C.R. Tripathy.
Department of Computer Science
and Engineering, V.S.S. University
of Technology, Burla, Sambalpur,
Orissa, India..

ABSTRACT
Recently a great deal of attention has been paid to the design of hierarchical shared memory cluster system. Cluster computing has made hierarchical computing systems increasingly common as target environment for large-scale scientific computations. This paper proposes hierarchical shared memory cluster architecture with load balancing and fault tolerance. Hierarchies of shared memory and caches structure the architecture. The hierarchical load balancing approach focuses on reducing the redistribution cost. The fault tolerant model is adopted to build highly available clusters in hierarchical shared memory clusters. Performance analysis and results reveal that hierarchical shared memory clusters performs much better creating a reliable hierarchical network cluster system with high scalability.

Keywords
Cluster Controller, Cluster Scheduler, Data Scheduler, Hierarchical Checkpointing, Levels of hierarchy, Recovery levels.

1. INTRODUCTION
The design of cluster system has received much attention in recent days. The cluster based architectures become quite appealing when a system is built with a large number of processors and memory modules [1-3]. A Shared memory cluster system needs less expensive interconnection network compared to that of a non clustered 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 [4-5]. The Hierarchical shared memory cluster system efficiently supports the hierarchy of memories sharing a common memory [6-8].

In a hierarchical shared memory cluster, there can be two sources of delays in satisfying memory requests: the access time of the main memory and communication delays imposed by the interconnection network [9-10]. The shared memory allows individual memory accesses for communication and synchronization. The access time delays of the interconnection network can be overcome by the use of private caches. The private caches significantly improve system performance due to the multiple copies of main memory. This ensures that the changes made to the shared memory by any one of the processors are visible to all other processors [11].

Hierarchical cluster architectures involve several issues such as sharing of global computation states, job scheduling and dynamic workload balancing among nodes. An effective load balancing scheme maximizes the efficiency by minimizing the processors idle time and interprocessor communication time. Job scheduling with an objective to minimize average job completion time consists of the allocation of tasks to the processors called as “space sharing” and the scheduling of the tasks over time called as “time sharing”. The scheduling involves effective and coordinated computation and data management mechanism. Data locality is also taken into account while mapping jobs to resources for load balancing among the clusters [12-15].

The decision making process in the schedulers improves overall resource utilization of the system. The shared networking resources are highly dynamic, which makes it difficult to make an accurate data transferring decision. Hence, the load balancing needs to focus on migrating excess workload from an overloaded processor to the underloaded processors. The load balancing is an essential part to reduce the inter processor communication during computation [16-18].

The cluster based hierarchical design is very appealing when a system is to be built with a very large number processors and memory modules. In a cluster, the services should be highly available at all times. Any single point of failure should be recoverable without affecting user’s application. So, for high availability the system should employ checkpointing and fault tolerant technologies to enable rollback recovery [19-22].

A fault tolerant cluster automatically demands the features of hot standby, failover and failback services after a node failure. In hot standby, a primary node provides services. While a backup node stands by without doing any work, the standby node takes over the work immediately when the primary node fails. The failover means that the surviving node takes over the services originally provided by the failed node. The failback allows the failed node to rejoin the workforce after it is repaired. Checkpointing is a software mechanism to periodically save the process state and intermediate computing results in memory or on disks. This allows the rollback recovery after a failure [23-26]. The DHCM [8] and HIN [10] follows a hierarchical network structure for clusters with main memory. However, the said work does not support the shared memory and cache memory. The load balancing issues proposed in AHS [13] and SST [14] does not calculate redistribution cost and there is no indication of the amount of workload to move from overloaded cluster to underloaded cluster. The work in HMA [24] has made fault tolerance analysis for l level hierarchical shared memory multiprocessors without hierarchical checkpointing and recovery.

This paper proposes a hierarchical shared memory cluster architecture (HSMC) that consists of a hierarchy of caches to maintain multicache coherency. At the first level of hierarchy, the shared memory structure is employed. At the second level of hierarchy, private cache copies are employed. A hierarchical load balancing model is proposed which estimates the load capacity of the available resources as a weighted sum of CPU load, available memory and link bandwidth. It emphasizes on reducing the cost associated with load redistribution. A hierarchical fault tolerant model is also proposed with hierarchical checkpointing and adaptive recovery levels. The proposed architecture features higher scalability with a good performance and reasonable cost by employing of the hierarchical structure and memory organizations with load balancing and fault tolerance.

The rest of the paper is organized as follows. The various notations used in the paper are presented in Section 2. In the Section 3, the overview of the proposed hierarchical shared memory cluster architecture with task scheduling is presented. In the Section 4, the proposed hierarchical load balancing model is presented. In the Section 5, the proposed hierarchical fault tolerant model with hierarchical checkpointing and recovery levels is provided. In the Section 6, the performance analysis and experimental results are presented followed by a brief discussion. Finally the concluding remarks are presented in Section 7.

2. NOTATION & ASSUMPTIONS

The following notations and assumptions are used throughout this paper to describe a number of different system parameters.

Notation:

Z	Total number of clusters in the system
N	Total number of processors in a cluster
M	Total number of memory modules in the system
L	Total number of levels in the hierarchy
N_i	Number of processors in an ith level cluster
M_i	Number of memory modules in ith level cluster
n_i	Number of ith level processors of a memory module
m_i	Number of ith level memory modules of a processor
k_i	Number of (i-1)th level cluster used to make an ith level cluster.
C_i	Total number of ith level clusters in the system
T_s	Task execution time for serial part of application
T_p	Task execution time for parallel part of application
T_o	The overhead of communication
T_{comm}	Total data transfer time
T_{comp}	Total data computation time
P_i	Number of processors in the ith cluster
MRT	Mean Response Time of n number of jobs
MRT_i	Mean Response Time of processors in the ith cluster.
$R(J_i)$	Response Time of a job J_i
$T_{finish}(J_i)$	Completion Time of a job J_i
$T_{submit}(J_i)$	Submission Time of a job J_i
α	Network Latency
β	Bandwidth or transfer rate
L	Message Length
Nideal	Ideal number of jobs
Nwork	Number of works waiting to be scheduled
Tw	Amount of workload to transfer
Wi	Actual Workload
Wj	Desired Workload
RCost	Redistribution Cost
T1	Computation time overhead
T2	Communication time overhead
λp	Failure rate of a processor
λm	Failure rate of a memory module
λb	Failure rate of a bus
Rp(t)	Reliability of processor
Rm(t)	Reliability of memory module
Rb(t)	Reliability of bus
Ra(t)	Reliability of arbiter
RL(t)	Reliability of zeroth level cluster
Rl(t)	Reliability of subclusters in hierarchy
BWL	Bandwidth at hierarchy level l
BW	System Bandwidth

Assumptions:

i. All processors are heterogeneous in nature.

ii. All failures are statistically independent

iii.All failures are exponentially distributed.

3. HIERARCHICAL SHARED MEMORY CLUSTER SYSTEM

This section proposes a hierarchical shared memory cluster system with architecture and task scheduling (HSMC). The hierarchical shared memory cluster system, at the top level of the hierarchy passes the requests from the clusters to the desired memory locations. Since the private caches are also aware of this access, no special coherency is necessary. Each process on any local cluster's processors run with equal ease gaining the advantage of the automatic load balancing of tightly coupled clusters as well as executing on loosely coupled cluster. Thus, the hierarchical shared memory cluster system reduces the global bus traffic and average latencies by distributing the memory amongst a group of processors.The Fig.1 shows the proposed architecture, which consists of the following components.

i) A large number of clusters organized as Z clusters each containing N processors.

ii) The main memory (M) shared by all the clusters at the top level of hierarchy.

iii) Cluster shared memory (Ms^1) globally shared by processors in each cluster at the 1st level of hierarchy.

iv) Local private caches (Mc^2) associated with each processor at the 2nd level of hierarchy.

v) Local Interconnection Network (LIN) connects all the processors of clusters with the cluster shared memory. Intracluster communication is accomplished through the LIN.

vi) Global interconnection Network (GIN) connects the cluster shared memory with the main memory. Intercluster communication is accomplished through the GIN.

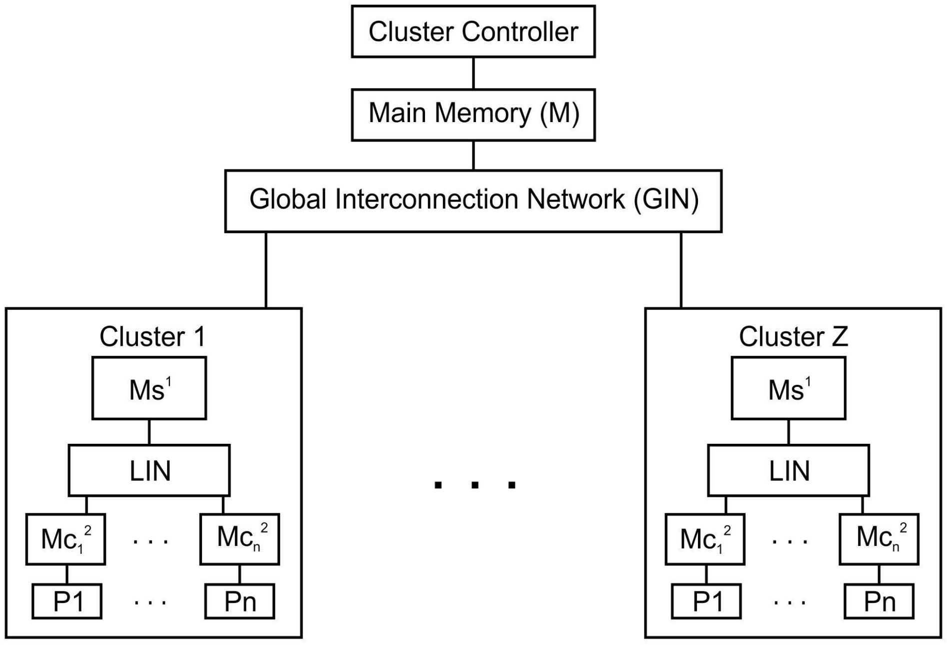

Fig.1 Hierarchical Shared Memory Cluster System Architecture

3.1 Overview of the Proposed Architecture

The proposed architecture consists of two levels of hierarchy for the management of processes inside each processor. The cluster controller provides a backup and co-ordinates the memory management services to the processes. The most important fact is that any memory location for which there are copies in the lower level caches will also have copies in the higher level shared memory. Since all the copies of memory locations contained in the lower level caches are also found in the higher level shared memory, the proposed cluster shared memory serve as multi cache coherency.

If any data is not found in the local cache, the cluster shared memory is searched making a LIN access. If it is not found in cluster shared memory then it performs a GIN access to the main memory. Once the data is found, a copy is forwarded to the requesting cluster shared memory and further reference to the same data by the processors of that cluster is serviced without accessing the main memory.

3.2 Task Scheduling

In the hierarchical shared memory cluster system, at the top level the tasks send messages to each other. In the next levels of hierarchies the data segment is transferred by passing a reference to the system segment number. When collaborating tasks with the shared memory, the user level communication is implemented.A task in the system is able to create new tasks in the form of subtask and determines the subsequent behaviour of the task. When the newly created task cooperates with its parent, a section of the parent segment is passed into the child, thereby enabling the shared access. It also allows control information to be passed across the child task.

In addition to the memory management, the task scheduling is also another important aspect. It takes two forms as task distribution and scheduling within each processor. If two tasks are manipulated in a shared memory segment, the task is either located in the same processor where the segment resides or the segment is positioned in the memory close to the task.In case of global scheduling, the data to be processed by a processor are placed on the "ready for execution list" and the outputs are dispatched when the execution of a node is terminated. When a process specified in the above way is started, a task in each of the allocated processors, executes a procedure. The procedure monitors the "ready for execution list" and starts a "queued action". The queued actions are the codes described in the cluster shared memory to execute task and queues holding data are present in the writable shared segment.

3.3 Analytical Model

In this subsection we present an analytical model to determine system scalability and efficiency. The various levels in the proposed model are designed on the concept of the work described in [10]. The number of processors in an ith level cluster is:

Ni=n0 for i=0, kiNi-1 for 1<=i<=L-1 (1)

The number of memory modules in an ith level cluster is:

Mi=m0 for i=0, kiMi-1 for 1<=i<=L-1 (2)

The number of ith level processor of a memory module is:

ni=n0 for i=0, Ni-Ni-1 for 1<=i<=L-1 (3)

The number of ith level memory module of a processor is:

mi=m0 for i=0, Mi-Mi-1 for 1<=i<=L-1 (4)

The total number of ith level clusters in the system is:

$$Ci=\prod_{j=i+1}^{i-1} k_j \quad \text{for } 0<=i<=L-2,\ 1 \text{ for } i=L-1 \qquad (5)$$

Task execution time is the summation of communication and computation time for serial and parallel part of application given as follows.

T_s=Tcomm$_s$+comp$_s$ (6)

T_p=Tcomm$_p$+comp$_p$ (7)

Total communication time is:Tcomm=Tcomms+Tcommp (8)

Total data computation time is: Tcomp=Tcomp$_s$+Tcomp$_p$ (9)

Extending Amdahl's law [2], the system speedup is

$$S = \frac{Ts + Tp}{(Ts + \frac{Tp}{N}) + To} \qquad (10)$$

where, To~ $\frac{Tcomm}{Tcomp}.N$

3.4 Proposed Algorithm

This subsection presents the proposed algorithm (HSMC) for task scheduling in the hierarchical shared memory cluster.

```
For Clusters from 1 to Z
  For processors from 1 to N
    For levels from 1 to L
      Search the task queue
      Fetch data from "ready for execution" list
      Execute task in the shared memory of respective parent level
      Write the result into private memory cache of current level
      Calculate Ni, Mi, ni, mi.
    End For
    Calculate Ci.
  End For
  Calculate S.
Endfor
```

Theorem 1: Time Complexity (HSMC)

Proof: The algorithm consists of N number of processors in Z number of clusters with L number of levels. Hence the time complexity of the proposed algorithm is O (LNZ).

4. PROPOSED LOAD BALANCING

This section presents the proposed hierarchical load balancing model for the hierarchical shared memory cluster system. The model explores a 2-level hierarchical approach for the load balancing. The proposed load balancing consists of two processes: Local load balancing and Global load balancing.

i) Local load Balancing: - Local balancing process is performed on processors at finer levels for balancing of workload among the processors within a cluster. An overloaded processor can only transfer its excess workload to an under loaded processor in the same cluster.

ii) Global load Balancing: - Global balancing process is performed on clusters at level 0 for balancing of workload among clusters. When imbalances among the clusters are observed then load redistribution among clusters are performed.

The proposed hierarchical load balancing model for the hierarchical shared memory cluster system is shown in Fig.2. At the top level, cluster controller (CC) is present with the main memory. In the next level of hierarchy, there exists a cluster scheduler (CS). There are N shared processors that are grouped into clusters. The global balancing processes are much less during the runtime as compared to local balancing processes. The jobs are submitted to the cluster controller where they are placed in the job wait queue until task allocation decision is made. The required data to run a job is fetched before the task execution starts. All I/O requests from a job are placed in the data scheduler while the task is running.

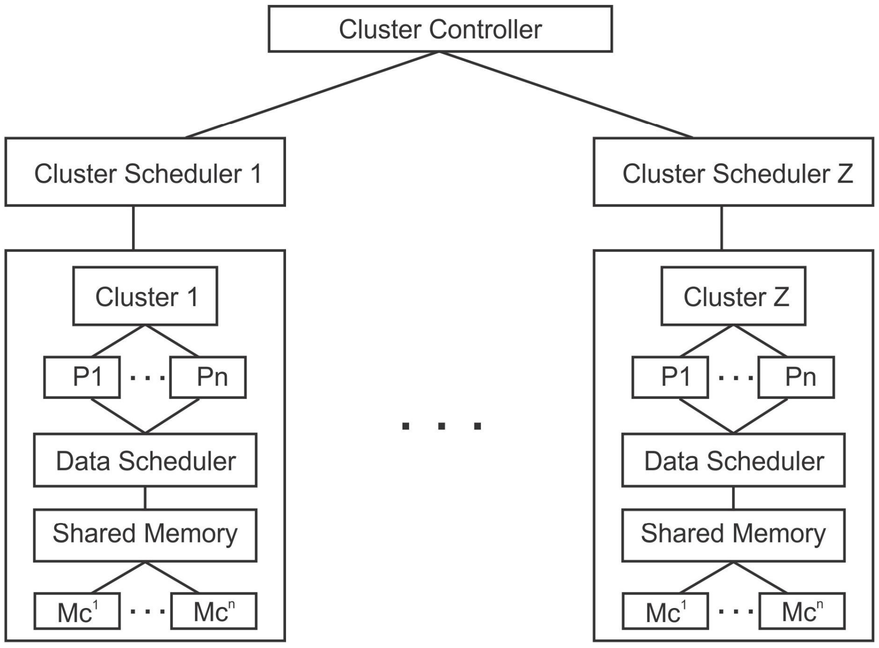

Fig.2: Herarchical Load Blancing Model

4.1 Theoretical Analysis

This subsection provides the theoretical analysis on global and local load balancing processes.

4.1.1 *Local Balancing Process*

The local balancing process is related with the issues like mean response time (MRT) of a job and intra cluster communication. The MRT of n jobs is calculated as follows.

$$MRT = \frac{1}{n}\sum_{i=1}^{n} R(Ji) \tag{11}$$

where, $R(J_i)$ is the response time of a given job J_i and is defined as: $R(J_i)=T_{finish}(J_i)-T_{submit}(J_i)$ (12)

The intra cluster communication (T_{comm}) is the time required for transferring data across the available processors within a cluster and is calculated as follows.

$T_{comm}= \alpha+ \beta.L$ (13)

4.1.2 *Global Balancing Process*

The global balancing process is related with issues like cluster imbalance, global redistribution and redistribution cost.

Theorem 2:

$$W_i \Leftrightarrow \sum_{j=1}^{Z} W_j * \frac{P_i * MRT_i}{\sum_{j=1}^{Z}(P_j * MRT_j)} \tag{14}$$

Proof: If the actual workload of a cluster is larger than the desired workload then the cluster is overloaded. If the actual workload is smaller than the desired workload then the cluster is under loaded. Otherwise, the cluster is balanced. Here left hand side represents the actual workload of ith cluster and right hand side represents the desired workload of ith cluster corresponding to its relative performance in terms of MRT. Hence the result for balanced cluster workload.

Theorem 3:

$$T_w = \frac{\sum_{j=1}^{Z} W_j \Big/ \sum_{j=1}^{Z} P_j * MRT_j}{W_i / P_i * MRT_i} \tag{15}$$

Proof: If the cluster imbalance exists, than the global redistribution is performed to move the workload from overloaded cluster to underloaded cluster. Here the amount of workload (Tw) is decreased and transferred from ith top level overloaded cluster to jth underloaded cluster. Hence the result for global redistribution.

Theorem 4:

$$R_{cost} = \max_{1\le i,j\le Z}\left[T_1 + (\alpha_{i,j} * Nmsg_{i,j}) + \beta_{i,j} * Lmsg_{i,j}) + T_2\right] \tag{16}$$

Proof: The redistribution cost is calculated as the maximum of all the redistribution costs between cluster i and cluster j with the estimated network latency(αi,j), data transfer rates(βi,j), number of messages(Nmsgi,j) and length of messages(Lmsgi,j) respectively. While redistributing the workload among clusters the global balancing process introduces some computation overhead (T1) and communication overhead (T2) which is defined as:

$$T1=\max (T1_i, T1_j),\ T2=\max (T2_i, T2_j) \quad (17)$$

Hence the result for the redistribution cost.

The imbalance ratio is the quality of load balancing and is defined as the ratio of maximum load to average load [17]. The imbalance ratio of the proposed hierarchical shared memory cluster is 1.8.

$$\text{Imbalance Ratio}= \max_{1\le i\le Z} [W_i] / \frac{1}{Z}\sum_{i=1}^{Z} W_i \quad (18)$$

4.2 Hierarchical Load Balancing

This subsection presents the hierarchical load balancing concept with an algorithm. The cluster controller contains a job queue i.e. QUEUE (job) to maintain unscheduled jobs or tasks and another queue to hold unsatisfied request for computation i.e. QUEUE (RFC). The processors can send request for computation (RFC) message to its upper level. If the upper level processor is in ideal state, it in turn generates its own RFC and sends it to the next upper level and so on upto the level of the cluster scheduler. When the RFC reaches the cluster scheduler, it backlogs the request if there are no jobs waiting to be scheduled. If the cluster scheduler or a processor along the hierarchy finds an unassigned work, the cluster scheduler allocates the task to the waiting jobs. The task allocation is done by ideal number of jobs that has to be moved in the hierarchy from upper level to the lower level as given below.

$$N_{ideal}=\beta * N_{work} \quad (19)$$

4.2.1 Proposed Algorithm: HLB

This subsection presents the proposed algorithm for the hierarchical load balancing model of the hierarchical shared memory cluster.

```
For Clusters from 1 to Z
  For processors from 1 to N
    For jobs from 1 to J
      Fetch jobs from coordinated checkpointing
      If level=0
        If QUEUE(job)=NULL
          Backlogs the request
        Else
          Perform task allocation
        End If
      End If
      If level>=1
        If QUEUE(job)=NULL
          If QUEUE(RFC)=NULL
            Send RFC message to upper level
          End If
          Backlogs the request
        Else
          Perform task allocation
        End If
      End If
      Calculate Response Time of a job
    End For
    Calculate Mean Response time of a processor
  End For
  Calculate intra cluster communication and workload of a cluster
  Perform redistribution of workload.
  Calculate redistribution Cost
End For
```

Theorem 5: Time Complexity (HLB)

Proof: The algorithm consists of N number of processors in Z number of clusters with J jobs in each of the L number of levels. Hence the time complexity of the proposed algorithm is O (LJNZ).

5. PROPOSED FAULT TOLERANCE

This section presents the proposed hierarchical fault tolerant model, which consists of two levels of hierarchy. The processors and memory modules of the hierarchical shared memory cluster system are grouped into a number of processor-memory clusters at the zeroth level. The Fig.3 shows the proposed hierarchical fault tolerant model for hierarchical shared memory cluster system. Every zeroth level cluster has n0 processors and m0 memory modules. The L level includes n_1* n_2*----* n_{L-1} base clusters, which contains n_L processors and nL memory modules. The Intracluster communication is established through the shared bus interconnection network and intercluster communication is established through the shared memory modules of the corresponding level. The total number of processors is N which is equal to N= n_1n_2*----* $n_{L-1}n_L$. The total number of memory modules is M which is equal to M= $n_1+n_1n_2++n_1n_2n_3+----+ n_1n_2.....n_{L-1}n_L$.

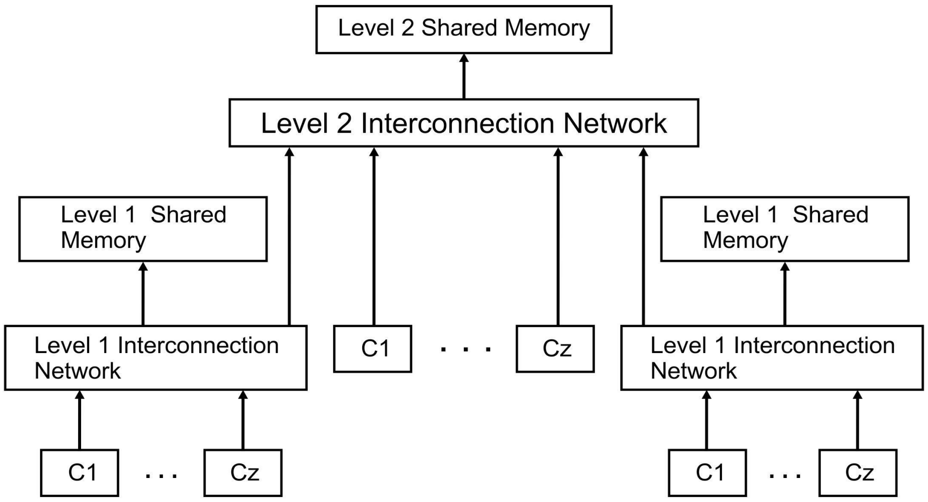
Level 2 Shared Memory
Level 2 Interconnection Network
Level 1 Shared Memory
Level 1 Shared Memory
Level 1 Interconnection Network
C1
. . .
Cz
Level 1 Interconnection Network
C1
. . .
Cz
C1
. . .
Cz

Fig. 3: Hierarchical Fault Tolerant Model

5.1 Hierarchical Checkpointing

This subsection proposes hierarchical checkpointing scheme. The proposed scheme is based on the 3-level adaptive recovery. The three types of checkpointings, which are stored by host processor periodically, are given as follows.

i) Local disk Checkpointing: The processes store coordinated or independent checkpoints in their own local storages periodically. This type of checkpointing can tolerate only a transient processor failure.

ii) Mirrored Checkpointing: The processes save consistent checkpoints in their local disks periodically and copy the mirrored images to their neighbor's disk. This type of checkpointing can tolerate single failures and multiple, isolated permanent failures.

iii) Stable Storage Checkpointing: The process saves consistent checkpoints periodically in the stable storage. It can tolerate any number of failures as stable storage is assumed to be failure free.

5.1.1 Recovery Schemes

Here, we describe the various recovery schemes. The hierarchical checkpointing schemes offer three levels of recovery. These three types of recovery are given as follows.

Level1: In case of a processor's transient failure, it immediately rollsback to local disk checkpoint as it can not follow a mirrored or stable storage checkpoint.

Level2: In case of permanent node failure, it rolls back to mirrored checkpoint and do not follow stable storage checkpoint.

Level3: It rollbacks to stable storage checkpointing in the following cases of failures.

a) Failure in processor

b) Failure in local disk

c) Loss of mirrored checkpoint

d) Failures occur after stable storage checkpoints.

5.2 Theoretical Analysis

This subsection is devoted towards theoretical analysis required for hierarchical checkpointing and recovery used in hierarchical fault tolerant model. The failure rates of a processor, memory modules and bus are exponential distributed [23]. The corresponding reliabilities of processor, memory and bus are given as:

$$R_p(t)=e^{-\lambda}p^t \quad (20)$$

$$R_m(t)=e^{-\lambda}m^t \quad (21)$$

$$R_b(t)=e^{-\lambda}b^t \quad (22)$$

The base cluster is divided into three sub modules: i processors, j cluster memory modules and c cluster buses. Then Pijc(t) is the probability that the base cluster is in (i,j,c) state and is given by [24]:

$$P_{ijc}(t)=R_a(t)*(n_L)\underset{(i)}{C_p}n_L^{-i}\,(R_p(t)^i(1-R_p(t))n_L^{-i} * (B_L)\underset{(c)}{C_b}B_L^{-c}\,(R_b(t)^c(1-R_b(t))B_L^{-c} * (n_L)\underset{(j)}{C_n}n_L^{-j}\,(R_m(t)^j(1-R_m(t))n_L^{-j} \quad (23)$$

The reliability of the zeroth level cluster in the shared bus interconnection network is given by:

$$R_L(t) = \sum_{i=I}^{N_L} \sum_{j=j}^{N_l} \sum_{c=C}^{B_L} P_{ijc}(t) \tag{24}$$

Each cluster in the level l(l<L) consists of nl subclusters connected to nl memory modules through the cluster interconnection network. The probability Pdks(t) gives the probability for the cluster to be in state(d,k,s) for d subclusters, k memory modules and s buses is:

$P_{dks}(t)=(n_l)Cc_l n_i^{-d} (R_l(t)*R_b(t)* R_i(t))^d * (1-R_l(t)*R_b(t)* R_l(t)n_l^{-d}$
(d)
$*(n_l)C_m n_L^{-k}(R_m(t))^k*(1-R_m(t))n_l^{-k} *(B_l)C_b B_l^{-s}(R_b(t))^s(1- R_b(t))B_l^{-s}$
(k) (s) (25)

Thus the reliability of the cluster with D subclusters, K global memory modules and S buse is given as:

$$R_l(t) = \sum_{d=D}^{n_l} \sum_{k=K}^{n_l} \sum_{s=S}^{B_l} P_{dks}(t) \tag{26}$$

At the hierarchy level l, with Bl buses and nl memory modules, the bandwidth BWl for shared bus interconnection network is given by

$$BW_l = \sum_{i=1}^{B_l} i * f(i) + \sum_{i=B_l+1}^{n_l} B_l * f(i) \tag{27}$$

Here f (i)=(n)Pi(1-P)n-i (28)
(i)

where, P is the ratio of bandwidth with number of memory modules for crossbar interconnection network that is given as :

$$P = \frac{BW_l}{n_l} \tag{29}$$

Now the total bandwidth of the system is the sum of bandwidth of all the subclusters of the system is given as follows:

$$BW = \sum_{i=0}^{L-1} \left(\prod_{l=0}^{i} n_l \right) * BW_{i+1} \tag{30}$$

5.3 Proposed Algorithm: HCR

This section proposes a hierarchical checkpointing and recovery algorithm for the hierarchical shared memory cluster system. For 2-level hierarchical shared memory cluster, the host processor saves mth checkpoints in the mirrored disk and mnth checkpoints in stable storage. This scheme rolls back to a local memory for transient failures. For a cluster of N workstations, the mirrored checkpoint can tolerate at most N/2 permanent failures. The level-1 checkpoint provides higher recovery latency while stable storage checkpoint provides lower latency and mirrored checkpoint provides latency in between the other two checkpoints.

```
Let T be the checkpointing period
If level=1
Host processor saves level-1 checkpoint in local disk
 If transient failure in processor
  Rollback to local disk checkpoint
 End If
 Loss of computation=T
 Else If level=2
  Host processor saves every mth checkpoint as mirrored checkpoint
  If permanent failure in processor
    Rollback to mirrored checkpoint
  End If
  Loss of computation=mT
  Else If level=3
  Host processor saves every mnth checkpoint as stable storage checkpoint
   If failure in processor or disk or level1 or level2 checkpoint
     Rollback to stable storage checkpoint
   End If
   Loss of computation=mnT
  End If
Calculate the reliability and bandwidth of the system.
```

Theorem 6: Time Complexity (HCR)

Proof: The algorithm consists of L levels with T number of checkpointing periods. Hence the time complexity of the proposed algorithm is O(LT).

6. PERFORMANCE ANALYSIS

This section analyses and evaluates the performance of the hierarchical shared memory cluster system. The program is developed in MATLAB. Based on the analytical model, the performance of the proposed hierarchical shared memory cluster system architecture is evaluated and compared with the previous works [8][13-14][24]. The performance is based on the task execution time including serial and parallel execution in all the levels of hierarchy with local cache memory and global cluster shared memory. A procedure containing codes which describes analytical model and the proposed algorithm is run with different configurations of nodes for different hierarchical level as shown in Table1. Then the average execution time is calculated. The Fig. 4 shows how the execution time affects the overall system performance in hierarchical shared memory cluster system as compared to that of DHCM and ISP [8]. As it is shown in the Fig.4, the execution time of Invalidation Set Protocol (ISP) worsens with the increase of nodes. While both the DHCM and HSMC improve with increase of nodes, the proposed cluster (HSMC) shows superior performance until the number of nodes approaches 32 as compared to existing models.

The performance of hierarchical load balancing model is evaluated against AHS [13] and SST [14]. The program uses the parameters for a set of J jobs with arrival time, the number of processors requested and execution completion time. The performance evaluation includes execution time, communication time, overheads, workload and redistribution cost. At runtime each process collects its own performance data and gives the summary result at the end of execution. Here in all the experiments, the number of clusters is taken as 8 with processors in the range of 4-64, jobs in the range of 1-50, latency is set to 50µs with bandwidth i.e. transfer rate to 100 Mbytes/s. The main goal of high performance clusters is to reduce the execution time. So, we evaluate the proposed system with the performance of workload and imbalance ratio as system utilization. The imbalance ratio if 1.0 or higher the better load balancing quality it achieves. The Fig.5 shows workload vs MRT for the proposed HLB compared with the existing methods AHS [13] and SST [14]. As shown in the figure the proposed HLB performs substantially better than SST and AHS at all system workload or utilization levels. Also after evaluation for imbalance ratio, it is found to be 1.8, which achieves better quality of load balancing.

The proposed hierarchical fault tolerant model is evaluated in terms of reliability and bandwidth. A program is developed describing the proposed hierarchical checkpointing and recovery algorithm. The hierarchical level is taken from L=1 to 4. For all the values of I, J, K, D sets to 8 and the program is run for several times and the average values are plotted. The number of clusters, processors and memory modules are varied from 1 to 10. As it can be seen in the Fig.6 and Fig.7 that HCR scheme uses less network bandwidth and provides high reliability as compared to that of the existing system HMA [24].

Table 1. Hierarchical Structure of Nodes

N	Configurations				
	Lev0	**Level1**	**Level2**	**Level3**	**Level4**
1	1x1				
2	1x2				
4	1x4, 2x2	1x2x2, 1x4x1			
8	1x8, 4x2	1x2x4, 1x4x2	1x2x2x2		
16	1x168 x2, 4x4	1x8x2, 1x4x4	1x2x2x41x 2x4x2	1x2x2 x2x2	
32	1x321 6x28x 4	1x16x2, 1x8x4	1x2x4x41x 4x4x2	1x2x2 x2x4, 1x2x2 x4x2	1x2x2 x2x2x2

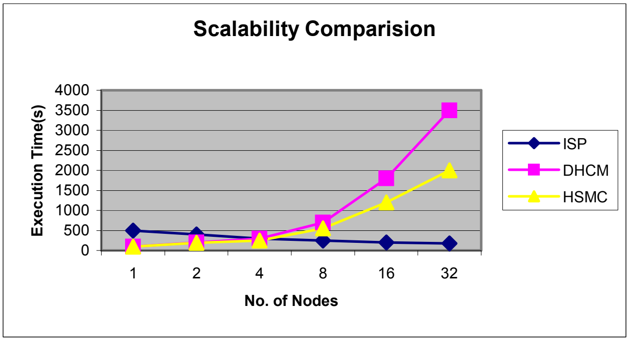

Fig.4. No. of nodes vs Execution Time

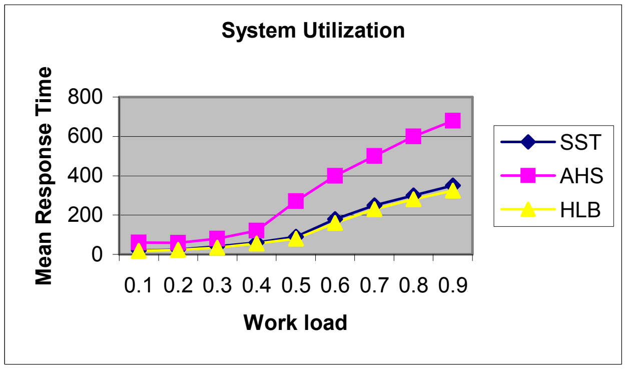

Fig.5. Workload vs Mean Response Time

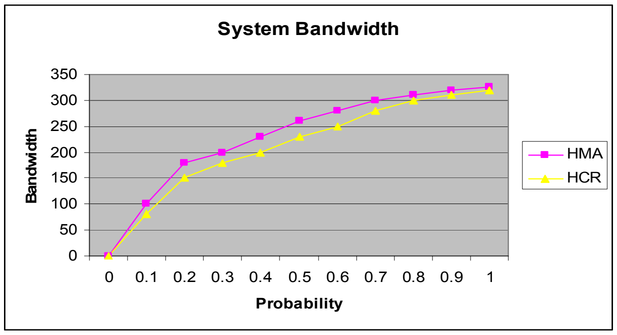

Fig.6. Probability vs Bandwidth

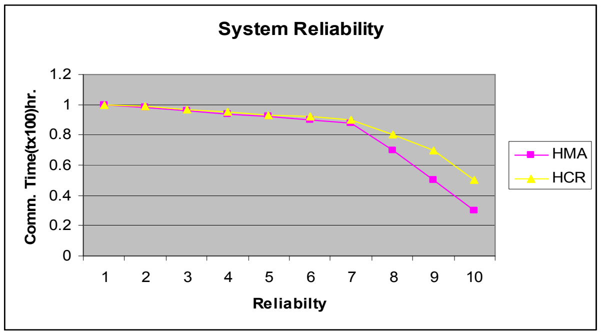

Fig.7. Reliability vs Communication Time

7. CONCLUSION

In this paper, we proposed a new hierarchical shared memory cluster system architecture which is scalable up to very large number of processors The proposed architecture demonstrated the effectiveness of hierarchical memory. The hierarchy used allows the expansion of the cache coherency technique beyond single cluster. It also reduces the global and local communication through intra and inter cluster communication. A hierarchical load balancing model is proposed which focuses on reducing execution time and redistribution cost with quality load balancing. It adjusts load balancing based on the observations of workload for the current system and desired system. We compared the results of system utilization with existing methods while minimizing the response time. The results establishes the hierarchical load balancing to be the appropriate load balancing scheme for efficient system utilization with shorter execution time of hierarchical shared memory cluster system. This paper also presented an overall analysis of hierarchical fault tolerant model with the hierarchical checkpointing and recovery schemes. This scheme handles several types of failures improving the system availability. In comparision with HMA, it uses less bandwidth occupying less network traffic and provides high reliability with efficient utilization of hierarchical shared memory cluster system

8. REFERENCES

[1] Guzman A., Krall E.J., McGehearty P.F. and Bagherzadeh N., "The effect of application characteristics on performance in a parallel architecture", Proceedings of the Twenty-First Annual Hawaii International Conference on Architecture Track, 5-8 Jan 1988, pp 167-172.

[2] ShenggangChen, Shuming Chen, and YamingYin, "Performance Impact of SMP-Cluster on the On-chip Large-scale Parallel Computing Architecture", IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 19-23 April 2010, pp 1 – 7.

[3] Minakshi 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, Sept 2010, pp 145-158.

[4] Yen-Jen Oyang, David Jinsung Sheu, Chih-Yuan Cheng and Cheng-Zen Yang, "The M2 hierarchical multiprocessor", International journal of Future Generation Computer Systems, Vol. 9, No. 3, Sept 1993, pp 235-240.

[5] Morris, D. and Theaker, C.J., "Hierarchical multiprocessor architecture", IEEE Proceedings on Computers and Digital Techniques, July 1987, Vol. 134, No. 4, pp 161 – 167.

[6] A. W. Wilson Jr., "Hierarchical cache/bus architecture for shared memory Multiprocessors", In Proceedings of the 14th Annual International Symposium on Computer Architecture, June 1987, pp 244-252.

[7] Khokhar A. and Dubois M., "Matching algorithms and architecture in hierarchical shared-memory multiprocessor (HSM) systems", Proceedings of Sixth International Symposium on Parallel Processing, 23-26 Mar 1992, pp 558 – 561.

[8] X. Liu, H. Jiang, and L. K. Soh, "A Distributed Shared Object Model Based on Hierarchical Consistency Protocol for Heterogeneous Clusters", Proceedings of 4th IEEE/ACM Int. Symp. On Cluster Computing and the Grid, April 2004, pp. 515-522.

[9] Feipei Lai, Lea-Ming Tzeng, Thom-Ling Chang and Tai-Ming Parng, "MARS performance evaluation with different interconnection networks", Proceedings of the First International Conference on Systems Integration, 23-26 April 1990, pp 248-257.

[10] Syed Masud Mahmud , L. Tissa Samaratunga and Shilpa Kommidi,"Fault-Tolerant Hierarchical Networks for Shared Memory Multiprocessors and their Bandwidth Analysis", The Computer Journal, Vol. 45, No. 2, Feb. 2002, pp. 147-161.

[11] Sokolinsky L. B., "Survey of Architectures of Parallel Database systems", Programming and Computer Software, Vol.30, No.6, 2004, pp 337-346.

[12] Rocco Aversa, Beniamino Di Martino, Nicola Mazzocca and Salvatore Venticinque, "A hierarchical distributed-shared memory parallel Branch & Bound application with PVM and OpenMP for multiprocessor clusters", Parallel

Computing, Vol. 31, No. 10-12, Oct-Dec. 2005, pp 1034-1047.

[13] J. H. Abawajy, "Adaptive hierarchical scheduling policy for enterprise grid computing systems", Journal of Network and Computer Applications, Vol. 32 No. 3, May 2009, pp 770-779.

[14] Emilia Rosti, Giuseppe Serazzi, Evgenia Smirni and Mark S. Squillante, "The impact of I/O on program behavior and parallel scheduling", Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, Vol. 26, No. 1, June 1998, pp 56-65.

[15] James D. Teresco, Jamal Faik and Joseph E. Flaherty, "Hierarchical Partitioning and Dynamic Load Balancing for Scientific Computation", 7th International Conference on Applied Parallel Computing, June 20-23, 2004, pp 911-920.

[16] Zhiling Lana, Valerie E. Taylorb, and Yawei Lia, "DistDLB: Improving cosmology SAMR simulations on distributed computing systems through hierarchical load balancing",---- Volume 66, Issue 5, May 2006, pp 716-731.

[17] Zhiling Lan, Valerie E. Taylor and Greg Bryan, "A novel dynamic load balancing scheme for parallel systems", Journal of Parallel and Distributed Computing, Vol.62, No. 12, Dec. 2002, pp 1763-1781.

[18] Cristiana Amza, Alan L. Cox., Sandhya Dwarkadas and Pete Keleher, "Treadmarks: Shared Memory computing on networks of workstations", Computer Journal, Vol 29, No.2, Feb 1996, pp 18-28.

[19] Tzu Fang Sheu, Nen Fu Huang, Hsiao Ping Lee, "Hierarchical multi pattern matching algorithm for network content inspection", International journal of Information Sciences, Vol. 178, No. 14, July 2008, pp 2880-2898.

[20] B.Panja, S.K. Madria, B.Bhargava, "A role-based access in a hierarchical sensor network architecture to provide multilevel security", Computer Communications Vol.31, No.4, April 2008, pp 793– 806.

[21] Minakshi Tripathy, C.R.Tripathy, "A new Coordinated Checkpointing and Rollback Recovery scheme for Distributed Shared Memory Clusters", International Journal of Distributed and Parallel systems, Vol.2, No.1, Jan 2011, pp 49-58.

[22] Arun K. Somani and Allen M. Sansano, "Achieving Robustness and Minimizing Overhead in Parallel Algorithms through Overlapped Communication/Computation", The Journal of Supercomputing - Special issue on embedded fault-tolerance systems, Vol. 16, No. 1-2, May 2000, pp 27-52.

[23] Chita R. Das, and Laxmi N. Bhuyan, "Bandwidth availability of multiple-bus multiprocessors", IEEE Transactions on Computers, Vol. 34, No. 10, Oct. 1985, pp 918-926.

[24] Veglis A.A. and Pombortsis A.S., "Performance related analysis of L-level hierarchical shared-memory multiprocessors", 8th Mediterranean Electro technical Conference, 13-16 May 1996, pp 1055-1060.

[25] Kai Hwang, Hai Jin, Chow, E., Cho-Li Wang and Zhiwei Xu, "Designing SSI clusters with hierarchical checkpointing and single I/O space", IEEE Concurrency, Vol. 7, No. 1, Jan-Mar 1999, pp 60 – 69.

[26] K. Li, J. Naughton and J. Plank, "Low latency concurrent checkpointing for parallel programs", IEEE transactions on Parallel and distributed systems, Vol. 5, No. 8, 1994, pp 874-879.

Authors Profile

Ms. Minakshi Tripathy received the degree of B.Sc. (PCM), M.Sc. (Statistics) and MCA from Sambalpur University. She has done 'A' level course from DOEACC, New Delhi. She is currently a Ph.D. (Computer Science) student at Sambalpur University, Burla, Orissa. She has publications in five different international conferences and three different international journals. Her research interest includes shared memory, cluster computing, load balancing and fault tolerance.

Prof. (Dr.) C.R. Tripathy received the B.Sc. (Engg.) in Electrical Engineering from Sambalpur University and M. Tech. degree in Instrumentation Engineering from I.I.T., Kharagpur respectively. He got his Ph.D. in the field of Computer Science and Engineering from I.I.T., Kharagpur. He has more than 50 publications in different national and international Journals and Conferences. His research interest includes Dependability, Reliability and Fault–tolerance of Parallel and Distributed system. He was recipient of "Sir Thomas Ward Gold Medal" for research in Parallel Processing. He is a fellow of Institution of Engineers, India. He has been listed as leading scientist of World 2010 by International Biographical Centre, Cambridge, England, Great Britain.