A Dynamic Replica Placement Algorithm to Enhance Multiple Failures Capability in Distributed System

Mr. Sanjay Bansal[1] , Prof. Sanjeev Sharma[2] and Ishita Trivedi
[1] Chameli Devi School of Engineering
Rajiv Gandhi Prodhyogiki Vishwavidya
Indore, India
Sanju2.bansal@gmail.com
[2] Head, School of Information Technology
Rajiv Gandhi Prodhyogiki Vishwavidya
Bhopal, India
sanjeev@rgtu.net

Abstract:

Replicated check pointing is widely used to tolerate multiple node failure. Multiple node failure capability is enhanced by generating the replicas on demand. Replica on demand for mission critical data and their placement is a crucial task. Placement of such huge number of replicas at run time bottle necks the performance and dependability. A high overhead is due to recomputaion of placement from initial stage. The performance becomes more severe if large number of nods and their data need to replicate. In order to improve the dependability these replicas need to placed on totally distant set of nodes. This paper presents dynamic replication management architecture with improved performance and dependability. Improved performance is achieved by avoiding the recompuation for all replicas from initial stage and placement of replicas uniformly over all distant nodes. Improved dependability is achieved by placing the replicas over totally distant set of computing nodes. This paper proposed architecture as well as experimental result to show effectiveness of proposed approach.

Keywords:

Replica Placement, Distributed System, Replica on Demand; Dynamic Replication Management, Uniform Replica Placement,

1. Introduction:

Cluster computing is one way to perform distributed computing. Several computing nodes connected together form a cluster. Several Loosely coupled clusters of workstations are connected together by high speed networks for parallel and distributed applications. Cluster computing offers better price to performance ratio than mainframes. If one machine crashes, the system as a whole can still survive in distributed system. Computing power can be added in small increments in distributed systems. In this way incremental growth can be achieved. Cluster computing has increased in popularity due to greater cost-effectiveness and performance. Recent advancement in processors and interconnection technologies has made clusters more reliable, scalable, and affordable [RB, 99].

Fault-tolerance is an important and critical issue in cluster computing. Due to very large size and computation complexity, the chances of fault are more. As the size of clusters increases, mean time to failure decreases. Most of time, such failures are not due to one fault but due to more than one fault. M.J. Fischer raise the issue that any protocol can be overwhelmed by faults that

DOI: 10.5121/ijdps.2011.2607

are too frequent or too severe, so the best that one can hope for is a protocol that is tolerant to a prescribed number of "expected" faults[MJF,85]. In such a situation, inclusion of fault tolerance is very essential. There are certain areas like air traffic control, railways signaling control, online banking and distributed disaster system high dependability and availability is essential. In absence of sufficient multiple fault tolerance, huge human lives and money could be lost. Hence there is a strong need for improved algorithms for multiple fault tolerance with performance.

John Paul Walter proposed a checkpointing based replication [jpw, 09]. Replication is done on different computing nodes instead of dedicated checkpointing server in order to reduce the overheads. Number of replicas decides the number of faults it can tolerate. Replication placement is one of the crucial aspects for high dependability. Placing the different replicas on nearby nodes may prone to breakdown just due to switch failure. As the number of replicas increases fault tolerant capability increased but cost also increases drastically. Instead of uniform replicas for all data one may opt for more replicas for critical and important data or process and less for less critical and less important data. These all approaches and issues are addressed in this paper. In section two we have proposed architecture for replica on demand.

2. Related Work:

Replica placement is one of the important issues for high dependable distributed system. Replica placement deal with how many different should be deployed and how to locate them. Replica placements become more crucial for dynamic distributed system. D.L McCue et. suggested a need of dynamic and adaptable replica placement[DLM,92]. A dynamic replica placement architecture is proposed by Byoung-Dai Lee for dynamic number of replicas [BL,01].These dynamic and adaptive replica placement policies must ensure performance over long period of system operation. Jaun Calos Leonardo et.al. propose an adaptable replication scheme for reliable distributed object oriented computing. He proposed an adaptable replication scheme that permits replacement of down replicas or change the number of replicas when partial failures occur and chooses the most adequate consistency protocol for the current configuration [JCL, 03]. However he has not address the issue of none varying the replicas at run time for some important processes to enhance the fault tolerant capability at run time. Xueyan Tang proposed a polynomial-time algorithm is then proposed to compute the optimal replication strategy which designates where each object should be replicated and how to keep the replicas up-to-date [XT, 04].However the minimal cost replication problem under dynamic replication creation is not addressed here. Qiao Lian el.at proposed a analytical framework to reason and quantify the impact of replica placement policy to system reliability. In this framework he addressed impact of replication placement on handling multiple failures [QL,05]. Bassam A. Alqaralleh also addresses the need of adaptive replica placement strategy [BAA, 07]. Wei FU et. proposed a QOS aware replica placement[WF,08]. Vinodh Venkatesan investigated impact of various replica placements on reliability of system [vv, 2010]. The Random Node Selection algorithm originally designed by Sankaran et. al., revised by John Paul Walters and Vipin Chaudhary, aimed to place replica at some far random node rather than placing it near the critical place. The goal is to randomly generate 'r' replicas per node, subject to certain constraints [jpw, 09].S.bansal el.at. proposed stair case based replication algorithm[sb,2011].We have proposed a algorithm in this paper in order to place the replicas on distant nodes in case new nodes join to the system or number of replicas are increased for important processes or nodes.We have also distribute the number of replicas uniformly among all computing nodes by stair case mechanism . In this, replicas of placed on such nodes first which having lower number of replicas, Due to uniform distribution of replicas among all computing nodes , we observe a significant improvement in performance.

3. Formal Description of architecture:

Architecture of proposed adaptive distant replication placement is shown in fig1. It consist of following modules

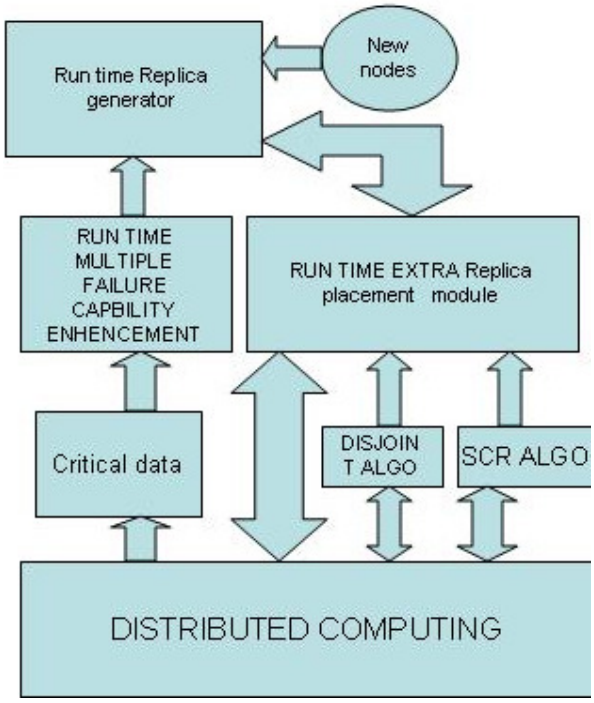

Fig. 1. Architecture of Proposed Replica Placement for Dynamic Distributed System

(3.1)Distributed system: -It consists of geographically distant computers or nodes with Remote Method Invocation.

(3.2)Run Time multiple capability enhancement module: It decides the fault tolerance capability of different nodes and process. In order to enhance the multiple failure capability of some mission critical data, number of replicas need to increase for those mission critical data. Alternatively if any new node joins, in that case extra replicas are generated. Replica numbers corresponding to some nodes or processes are varied at run time. Placement of these run time replicas or replica on demand is done by the run time extra replica placement module.

(3.3) Run Time Replica Placement Module: We have used following algorithm 1 proposed by John Paul Walter with following assumptions.

1. A node should not replicate to itself.
2. A node should replicate to exactly r nodes, each of which may only store r rep licas.
3. Each of a node's r replicas should be unique.

The algorithm proposed by John Paul Walter for replica placement is as follows

Algorithm 1: Compute random replica placements.
Input: Integer r, the number of replicas
Input: Integer n, the number of nodes
Output: Replica array, Replicas [0...n - 1][0...r - 1]
1: for all i such that 0 = i < n do
2: Preload node i's replicas with i
3: end for

```
4: Circular-shift each column (j index) of Replicas
   by column index - 1
5: for all i such that 0 = i < n do
6: for all j such that 0 = j < r do
7: repeat
8: z = random node, s.t. 0 = z < n
9: v = Replicas[z][j]
10: until z  = i
11: if v  = i and Replicas[i][j]  = z then
12: valid replica = 1
13: for all k such that 0 = k < r do
14: if Replicas[i][k] == v or Replicas[i][j] == Replicas[z][k] then
15: valid replica = 0
16: end if
17: end for
18: if valid replica then
19: Replicas[z][j] = Replicas[i][j]
20: Replicas[i][j] = v
21: end if
22: end if
23: end for
24: end for
```

It has no provision for the replicas generated at run time for existing node as well as new nodes arrival. Replicas placement for new node arrival or additional replicas generated at run time is done by disjoint module.

(3.4)Disjointing Module: This module only take care about placement of replica of following two cases

Case 1: When New Node needed to be inserted in an already developed distributed system.

Case 2: When number of replicas required to be increased for some important and crucial data. This module does not disturb too many already placed replicas. This only computes placement of additional replicas for either cases. The algorithm used by this module is as follows

Algorithm 2 Disjoint Algo: Compute runs time and new replica placements

```
1:-while new nodes are joined
2: for all such that 0 = i < n do
3: Preload node nn's replicas with r replica of itself(number of replicas)
4: end for
5 While (no of new nodes * replica)
6. for all k= 1 to r/2-1( number of replicas at every node)
7   Take a random node j;
8. Remove node j from list
9 If replica (j+k) exist
10 Exchange replica j+k with new node replica
11 end if
12 end while
13 if number of replicas are increase for some or all nodes
14 preload addition replicas to itself
15.  for m=1 to dr(additional replicas of r nodes)
16 choose a random node k
17  (if k==m)
```

18 continue
19 end if
20 if (m replica already exists) than
21 continue;
20 else
21 transfer extra replica to m node
23 remove k from node list;
23 end for

(3.5) Stair case algorithm module: In order to distribute all replicas uniformly among all distant set of computing nodes. Indexing is based on giving the lowest rank to a processor having the least number of replicas node and the highest to the node having the highest least number of replicas. In this way, a rank is allocated to each node based on their load value. Placement of replicas is first done on distant low index node and so on.

Algorithm 3 Stair-Case (Replica Result Array)Input: Input the Array
Output: enhanced replica model for fault tolerance

1. For all I such that 0< I <n do
2. Apply asynchronous replication
3. Eliminate central storage
4. Calculate the index by replica method
5. for all I such that 0 <=i < n
6. for all J such that 0 <=j <n do
7. apply replication on the lower node to the higher node
8. Finish

(4) Experimental set up: We have performed a experiment on 64 nodes. Distributed environment is set up by Remote Method Invocation. The simulation creates an interactive distributed environment that shows the dynamic changes as follows:

Case 1: When New Node needed to be inserted in an already developed distributed system.

Case 2: When number of replicas required to be increased in an already developed distributed system to improve its fault tolerance capability.

(5) Result: We obtain following results as shown in Table 1. Result is obtained by simulating the algorithm in RMI based distributed computing. Since proposed algorithm eliminates the need of computation from initial stage, performance is improved. Once all placement of replica done and if extra replicas are generated either due to new nodes arrival or due to increase of replicas for existing node than computation is required for only extra replicas instead of all.

Table 1. Performance comparison after new nodes are added

No of nodes initially	Number of new nodes joining	Replica Placement Time in (ms)	
		Purposed Method	Simple
8	4	95	145
12	8	123	247
20	16	212	349
36	18	324	567

Table 2. Performance comparison when number of replicas are increased for some critical data nodes

No of replicas initially	Number of new replicas added	Replica Placement Time in (ms)	
		Pur-posed Method	Simple
8	2	98	112
8	4	192	252
8	6	278	362
8	8	489	589

(6) Graphical Representation: In this section a graph is plotted for results obtained in section 5. Graphical presentation is shown below in fig 2.

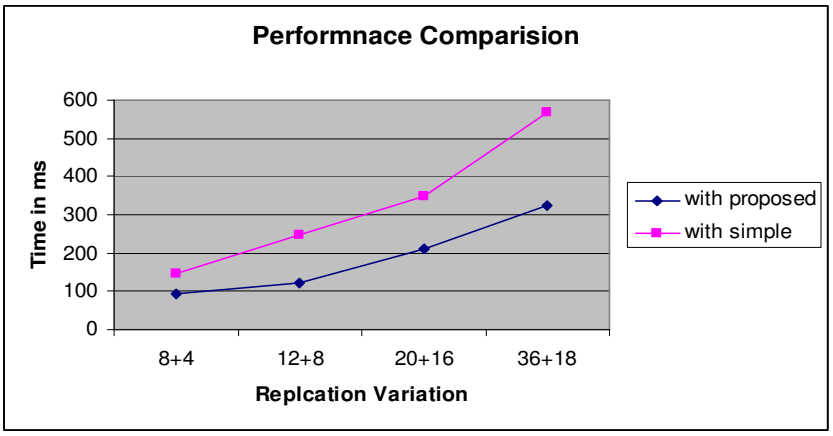

Fig. 2. Graphical Representation for Result Obtain

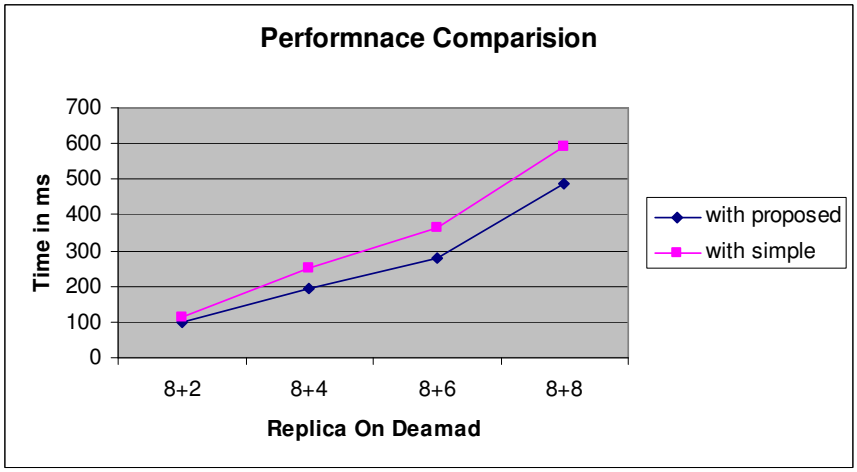

Fig. 3. Graphical Representation for Result Obtain

Conclusion:

From the result table and graphical representation it is clear that proposed approach takes less time as compare to simple non adaptive placement approach that computes the placement from initial stage if number of replica changes rapidly or new nodes joins the distributed system. Our approach is adaptive to change in number of replicas of each process or some process. In case of new nodes arrival the disjoint module only compute placement for new nodes replicas without doing much computation for all existing replicas. It takes less bandwidth and latency since only fever number of replica or moved in distributed system in case of significant number of new nodes joins to the distributed system. Fault tolerance capability is also improved since placements of replicas are done on disjoined nodes. In case of several nodes failure of same link, disjoint replica placement increases dependability of distributed system as well. Using stair case mechanism, we placed extra replicas uniformly among all computing nodes which further improve the performance.

Reference:

1. [GG, 08]G. Georgina., "D5.1 Summary of parallelization and control approaches and their exemplary application for selected algorithms or applications," LarKC/2008/D5.1 /v0.3, pp 1-30.
2. [RB, 99] R.Buyya., "High Performance Cluster Computing: Architectures and Systems," Vol. 1, Prentice Hall, Upper Saddle River, N.J., USA, 1999
3. MJF, 85] MICHAEL J. FISCHER, NANCY A. LYNCH AND MICHAEL S. PATERSON, "Impossibility of Distributed Consensus with One Faulty Process, " Journal of the Asscctktion for Computing Machinery, Vol. 32, No. 2, April 1985, pp. 374-382.
4. [JPW, 09]John Paul Walters and Vipin Chaudhary, "Replication-Based Fault Tolerance for MPI Applications," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 20, NO. 7, JULY 2009
5. [DLM,92] D. L. McCue AND M. C. Liule, "Computing Replica Placement in Distributed Systems," 0-8186-3170492 $3.00 Q 1992 IEEE. [BL,01] Byoung-Dai Lee and Jon B. Weissman, "Dynamic Replica Management in the Service Grid," 0-7695-1296-8/01 $10.00 0 2001 IEEE.
6. [JCL,03] Juan Carlos Leonardo and Takaichi Yoshida, "An Adaptable Replication Scheme for Reliable Distributed Object-Oriented Computing," Proceedings of the17 th International Conference on Advanced Information Networking and Applications (AINA'03) 0-7695-1906-7/03 $17.00 © 2003 IEEE.
7. [Xt,04] Xueyan Tang and Samuel T. Chanson, "Minimal Cost Replication of Dynamic Web Contents under Flat Update Delivery," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 15, NO. 5, MAY 2004
8. [QL, 05] Qiao Lian, Wei Chen, Zheng Zhang, " On the Impact of Replica Placement to the Reliability of Distributed Brick Storage Systems, " Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICSCS'05) 1063-6927/05 $20.00 © 2005 IEEE.
9. [BAA, 07] Bassam A. Alqaralleh, Chen Wang, Bing Bing Zhou, and Albert Y. Zomaya, "Effects of Replica Placement Algorithms on Performance of structured Overlay Networks," 1-4244-0910-1/07/$20.00 ©2007 IEEE.
10. [WF, 08] Wei FU, Nong XIAO and Xicheng LU, "A Quantitative Survey on QoS-aware Replica Placement," 2008 Seventh International Conference on Grid and Cooperative Computing, 978-0-7695-3449-7/08 $25.00 © 2008 IEEE DOI 10.1109/GCC.2008.23
11. [vv,10] Vinodh Venkatesan, Ilias Iliadis, Xiao-Yu Hu, Robert Haa, "Effect of Replica Placement on the Reliability of Large-Scale Data Storage Systems, " 2010 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1526-7539/10 $26.00 © 2010 IEEE,
12. [SB, 11] S. Bansal ,S Sharma and Ishita Trivedi "A Novel StairCase Replication (SCR)Based Fault Tolerance for MPI Applications" International Conference on Advances in Information Technology and Mobile Communication – AIM 2011 April 21-22, 2011, Nagpur, Maharashtra, India,. Information Technology and Mobile Communication Communications in Computer and Information Science Springer, 2011, Volume 147, Part 3, 445-448, DOI: 10.1007/978-3-642-20573-6_80

S. Bansal S. Sharma and Ishita Trivedi , "A Fast Adaptive Replication Placement for Multiple Failures in Distributed System," accepted in First International Conference on Parallel, Distributed Computing Technologies and Applications (PDCTA-2011) , Communications in Computer and Information Science (CCIS) Series(Springer).