www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 9 September 2014 Page No. 7966-7967

Enhancement in the neighbor-based diskless checkpointing approach

Parase Dipali B[1], Dr Mrs Apte S S[2], Shegadar A R[3]

[1] Solapur University, Walchand Institute Of Technology, Solapur,Computer Science Department
Akkalkot Road, Solapur, Maharashtra,India
deepali.parase@rediffmail.com
[2] Solapur University, Walchand Institute Of Technology, Solapur,Computer Science Department
Akkalkot Road, Solapur, Maharashtra,India
headcse@gmail.com

[3] Solapur University, BIGCE college of engineering, Solapur,Computer Science Department
Kegaon, Solapur, Maharashtra,India
amrutashegadar8887@gmail.com

Abstract: ***In this paper we are providing a solution for the stable storage requirement in the disk-based system. Fault tolerant is very essential for distributed and parallel systems. To handle multiple processor failure here, we are using a diskless checkpointing approach. In that we are enhancing the neighbor-based approach. Instead of storing checkpoint in the stable storage we are storing it in peer processor's main memory. To overcome the memory overhead we use a parity technique.***

Keywords: diskless checkpointing, parity technique, neighbor based, XOR

1. Introduction

In a disk based system when we are using checkpointing approach the checkpoint is to be stored onto the stable storage. As number of checkpoint increases the disk overhead also increases, which degrade the system's performance. To overcome this we use diskless checkpointing approach. There are various methods available for implementation of diskless checkpointing approach. Among that in this paper we are discussing neighbor-based diskless checkpointing approach [2].

1.1 Neighbor based diskless checkpointing

In neighbor based diskless checkpointing approach each processor stores its checkpoint on its peer processor's memory. Dedicated checkpoint processor is not needed here. Here, we assign each processor a set of peer processor's for storing their checkpoint it is called as checkpoint storage node (CS). The processor itself responsible for storing checkpoint from other processors. Set of such processors are called checkpoint coverage node (CC)[1]. Each processor stores checkpoint into memory so problem of memory overhead occurs. To overcome this problem we apply parity technique. Whenever processor stores checkpoint into its memory it performs the XORing of own local copy of checkpoint with the received copy of the checkpoint i.e. parity-based diskless checkpointing technique [3]. There is no need for dedicated checkpoint processor.

When processor failure occurs, the failed processor is recovered from one of its checkpoint storage node. Condition here is that all nodes in checkpoint coverage nodes assigned for particular node in checkpoint storage node should take participate for recovering the failed processor's checkpoint. In this way a processor failure recovery can be performed.

2. Implementation

2.1 Design of CS

First design checkpoint coverage node (CC) and checkpoint storage nodes (CS). All CSi's and CCi's have the same size implies that load balance can be achieved for all the processors in the system. The number of processors should participate into our paper are 5. This can be calculated by using partial sum restricted sequence (PSR sequence) as follows-

A sequence of r positive integers $d_0, d_1, \ldots, d_{r-1}$ is defined as a partial sum restricted sequence (or PSR sequence)if there exists no l,m, p, and q, $0 \le l \le m < p \le q \le r-1$, for which $\sum_{i=l}^{m} di = \sum_{i=p}^{q} di$. It is given in Table 1 below.

Table 1: Minimum n and One Possible PSR Sequence for k =2 to 5

k	*Possible* $d_0, \ldots, d_{k-2}$	*Minimum d*	*Minimum n*
2	1	1	5
3	1,2	3	11
4	1,3,2	6	20
5	1,3,5,2	11	35

In the above Table 1 n is number of processors, k is size of CS and CC, d is sum of PSR sequence. The total number of n

processors in the system is n=3d + 2. In our project we are allowing k=2 i.e. minimum 2 simultaneous failures can be allowed. So, the according to Table 5.1 value of d is 1. Hence, n= (3*1) +2=5. Therefore, we are considering 5 processors system. Here, we have to form a cyclic chain [4]. It is having one element common in respective CS's. Suppose we allow processor failure up to 2 i.e. k=2 then the assignment of CS and CC's for processors P0,P1,P2,P3,P4 is shown in Table2 below.

Table 2: List of CS and CC's for k=2 and n=5

Processor (Slave id)	*CS*	*CC*
P0	{P1,P2}	{P3,P4}
P1	{P2,P3}	{P4,P0}
P2	{P3,P4}	{P0,P1}
P3	{P4,P0}	{P1,P2}
P4	{P0,P1}	{P2,P3}

2.1 Working

We use a distributed or parallel system. Here, we take an application called MAT. This application performs a matrix multiplication. We are having two matrices of size 4000*4000. Consider, we are allowing simultaneous k processor failures in our system. For example k=2. So we require number of processors equal to 5. To work in parallel we divide the task of multiplication among processors. After dividing each processor performs its computations. A row is taken as a checkpoint. While performing computation each processor stores its checkpoint into own memory as local copy. Also sends its checkpoint to nodes in CS. While storing checkpoint each processor performs XORing of own and received checkpoint hence calculate parity. Therefore only parity is stored into memory. So memory consumption problem is removed here.

If one of the processor failures occurs, then at least one of the node in CS should remains alive. And all CC's of that CS node including failed node should take participate in calculating checkpoint for failed processor. Here we are storing previous checkpoint as well as current checkpoint for the system. While recovering first previous checkpoint is made nullify using XORing with calculated parity. After this take checkpoints from all of its CC's including failed processor. Perform XORing with current copy of checkpoint and calculate checkpoint for the failed processor.

3. Conclusion

This study addresses diskless checkpointing issues in a distributed or parallel computing environment and presents a new approach to enhancing neighbor-based schemes to tolerate multiple failures. The proposed scheme is unique in that it only uses simple XOR operations for checkpointing and failure recovery and does not require dedicated checkpoint processors. This method allows checkpoint related operations to be evenly distributed among all processors, achieving good load balance. We have proposed that our approach works for k simultaneous failures. But right now it works for only one processor at a time. In future we will try for k simultaneous failures at a time.

4. References

[1] Ge-Ming Chiu, Member, IEEE Computer Society, and-Ferng Chiu, "A New Diskless Checkpointing Approach for Multiple Processor Failures",IEEE transactions on dependable and secure computing, vol. 8, no. 4, july/august 2011

[2] Z. Chen, G.E. Fagg, E. Gabriel, J. Langou, T. Angskun, G. Bosilca,and J. Dongarra, "Fault Tolerant High Performance Computing by a Coding Approach," Proc. ACM Symp. Principles and Practice of Parallel Programming (PPoPP '05), pp. 213-223, June 2005.

[3] J.S. Plank, Y. Kim, and J. Dongarra, "Fault-Tolerant Matrix Operations for Networks of Workstations Using Diskless Checkpointing," J. Parallel Distributed Computing, vol. 43, no. 2, pp. 125-138, 1997.

[4] Z. Chen, G.E. Fagg, E. Gabriel, J. Langou, T. Angskun, G. Bosilca,and J. Dongarra, "Fault Tolerant High Performance Computing by a Coding Approach," Proc. ACM Symp. Principles and Practice of Parallel Programming (PPoPP '05), pp. 213-223, June 2005.