Improved Bandwidth Utilization based Check pointing Algorithm in Distributed Database

Neera Batra and Manpreet Singh
M. M. Engineering College, M. M. University
Mullana, Ambala
Haryana, India

ABSTRACT

In this paper, we present an optimal-bandwidth, min-process coordinated check pointing algorithm suitable for network failure prone applications in distributed systems. In the developed algorithm, during normal computation message transmission, dependency information among clusters is recorded in the corresponding cluster head processes. When a check pointing procedure begins, the initiator from a cluster concurrently sends composite message to all the cluster head processes which after extracting individual messages from it, further multicasts individual messages to the corresponding currently active receiving processes in their corresponding clusters thus resulting in reduced transmission delay and communication cost, better bandwidth utilization and faster speed of execution. Quantitative analysis shows that proposed algorithm works efficiently in terms of better response time and maximum bandwidth utilization for applications running under critical conditions such as low bandwidth availability and thereby resulting in frequent disconnections.

Keywords

Optimal bandwidth cluster fed check point, non-blocking.

1. INTRODUCTION

Cluster federations contain a large number of nodes and are heterogeneous. Nodes in a cluster are often linked by a SAN (System Area Network) while clusters are linked by LANs (Local Area Network) or WANs (World Area Network) [5]. Clusters communicate with each other by message passing. To survive failures, clusters take checkpoints periodically or non-periodically. Checkpoint is defined as a designated place in a program at which normal processing is interrupted specifically to preserve the status information necessary to allow resumption of processing at a later time. Check pointing is the process of saving the status information. A checkpoint is a snapshot of the state of a process saved on the stable storage which can be reloaded into memory to reduce the amount of lost work in recovery. [6] Checkpoints correspond just to processes states and have no information about the states of the communication channels. Therefore, in order to record a consistent global state of a process, whenever the state of any process indicates that it has sent a message to another process, the state of the receptor must indicate that it has received this message. This requirement is the major condition to the implementation of techniques to record global states of distributed system.

A global checkpoint [8][9][10][14] consists of certain number of checkpoints such that each of these checkpoints corresponds to one of the clusters uniquely in a cluster federation.

In this paper, we specifically address a reduced bandwidth usage and low cost scheme for distributed database based on processes check pointing . Processes take checkpoints periodically managed by the local cluster head and log their output/input in a common table maintained by cluster head. The developed scheme reduces the cluster-to-cluster communication to a single composite message and the cluster head of each cluster is responsible for extracting the individual messages from the composite message and multicast them to the corresponding receiving processes.

The remainder of this paper is organized as follows. In section 2, we state the data structures used. Section 3 describes the working model of the algorithm. Section 4 presents the proposed check pointing algorithm. In section 5, some relevant observations along with comparison to other algorithms are presented. Section 6 comprises of performance analysis. Section 7 concludes the paper.

2. DESIGN PRINCIPLES

2.1 Models and assumptions

Application model: Distributed database applications using the composite message model is designed. Processes of this kind of applications are divided into clusters (modules). Processes inside the same cluster communicate a lot while communications between processes belonging to different clusters are limited due to global bandwidth usage resulting in more communication cost incurred and frequent transaction restarts due to unavailability of optimal bandwidth required for Inter-cluster communications.

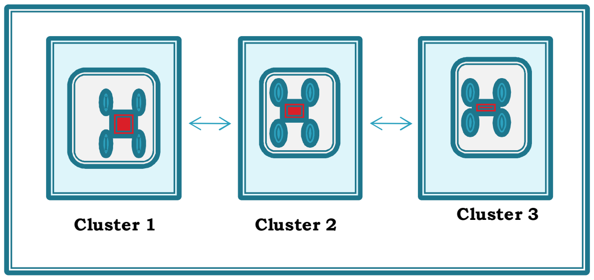

Figure 1: Cluster Federation

2.2 Architecture model and network assumptions

We assume a cluster federation as a set of clusters interconnected by a WAN, inter-cluster links being either dedicated or even Internet, or a LAN. Each group of processes may run in a cluster where network links have small latencies and large bandwidths (SAN).

2.3 Data Structures

Notations used:

N	- Global Bandwidth
η	- Local Bandwidth
SN	- Sequence Number of a Process
SN^a	-Sequence number of cluster A
N_P	- Total number of processes

N_c - Total number of clusters

CH -Cluster Head

$\lfloor P_{i,i} Y_i \rfloor$ -Process identity number of i^{th} process, flag Y for i^{th} process

v_i^a - keeps a record of SN for each process p_i in cluster A

$C_i(x)^a$ - X^{th} checkpoint of process i in cluster A

$Y[i][a]$ - is the flag used to identify active processes at x^{th} checkpoint

t - Time taken for a composite message to reach from one CH to another CH

$p_1^a(CH)$ - The checkpoint initiating cluster head process cluster A

m^c -control message

m^a - Application message

m^{Ca} - Composite message

Suppose there are N_P processes and N_c clusters in the system where N_P is much larger than N_c. Each process is assigned a unique id, i where $(1 \le i \le N_P)$.

In our check pointing scheme, for each process in the cluster, the checkpointing dependency information is maintained by its cluster head process. Each Cluster Head CH sends the composite messages consisting of control and application messages to the cluster head of other clusters which further multicast the message to all the corresponding currently active processes in the cluster after extracting the individual single or multiple messages from the composite message meant for each receiving process. This scheme reduces the message passing and number of lost messages drastically, thus making system more available, reliable and faster and resulting in optimal bandwidth utilization for network failure prone applications.

When a check pointing procedure begins, the sending and the receiving of composite messages is mainly accomplished amongst cluster head processes. To maintain such additional information for each process p_i^a in cluster A, each sending cluster head CH maintains a log file $\langle p_i^a, p_j^b, B \rangle$ with p_i^a as a sending process in cluster A and p_j^b as a receiving process in cluster B where $(1 \le i, j \le N_P)$ and $(1 \le a, b \le N_c)$. A vector v_i^a for keeping a record of SN (Sequence Number) for each process p_i in cluster A where flag $Y[i][a] = 0$ in case, process p_i neither receives or sends any message during current global interval $\left(C_i(x)^a - C(x-1)^a\right)$ at X^{th} check point. After the global check point is taken, both the fields in the table are set as empty and SN^a is incremented.

3. WORKING MODEL

In the proposed algorithm, when communication occurs between two processes in different clusters, then dependencies are generated between checkpoints taken in different clusters. Dependencies must be tracked in order to allow the application to be restarted from a consistent state. In our work based on idea adopted from [1], it is the sending process that ensures that none of its sent messages can remain an orphan (received-not-sent).

When the CH of any cluster initiates the checkpointing procedure by sending the control message to other clusters, then the current cluster's sequence number SN is piggybacked on inter-cluster control message embedded in the composite message which is sent to any active process in any cluster during X^{th} global checkpoint interval. CH of each other cluster is responsible for storing these SN values for synchronization among clusters.

The communication scheme based on message passing from one CH to other is beneficial only if (i) there are very few chances of message loss due to network failure. So the proposed algorithm works best for the applications which are network failure prone and for applications which do not use secure network media for message communication as it introduces the concept of composite message passing to overcome these two significant shortcomings of communication induced check pointing (ii) CH communicates the inter-cluster received messages to all the active processes in the cluster within a finite period of time so that there is no synchronization delay. To deal with synchronization delay, the algorithm assumes a threshold value of time interval within which sending CH creates a composite message comprising of control messages and application messages both and receiving CH multicasts the received messages to all corresponding processes in the cluster which are participating during current global checkpoint interval $(c_{x-1} - c_x)$,after extracting them from the composite message .

Let us assume that the time taken on an average by a cluster head to send a composite message to other cluster head is a constant t with the assumption that the bandwidth available during message passing remains constant. As seen in most of the previous works[1][2][3], If a control message is to be sent to processes in a cluster, time taken by a sending process p_i^a in cluster A for any processes p_j^b where $(1 \le p_j \le n)$ in cluster B is t. If the process p_i is supposed to send the control message to each process in cluster B directly, it will take $n * t$ if there are n processes. In the proposed algorithm, the CH of cluster B checks for the value of Y_i where $1 \le i \le n$ and multicasts extracted control messages and application messages to all the processes with value of $Y_i = 1$. Suppose time taken by CH to multicast the control message m^c and application messages m^a among active processes is τ which is a small fraction of time t as cluster B uses SAN(System Area Network), a very fast and reliable media in comparison to LAN or WAN used for communication amongst clusters. So total time taken by CH to inform all the active processes for the next checkpoint is $(\tau + t + \tau)$ i.e. $t + 2\tau$ where τ is time taken by sending process in cluster A to composite the message, t is the time taken by cluster A to send the message to CH of cluster B and τ is the time taken by CH of cluster B to extract the individual messages and multicast them to corresponding active processes. This value $(t+2\tau)$ is considered as a threshold value to keep a check on transmission delay caused by CH. Although this threshold value varies during each global check point interval depending upon number of active processes in current global checkpoint interval but this variation is very small, since number of participating processes during each checkpoint interval remain almost constant. Now this threshold value will be a common constant for all the clusters in cluster federation. Hence, each sending and receiving cluster will know a priori about message transmission delay caused by any other CH. So no acknowledgement is required to ensure that cluster head CH has sent the message to all other processes or not, which belong to same cluster.

Suppose there are two clusters A and B with 4 processes each uniquely identified as $p_{1,}p_2, p_3, p_4$ and $p_5, p_{6,}p_7, p_8$ respectively as shown in Fig. 2.

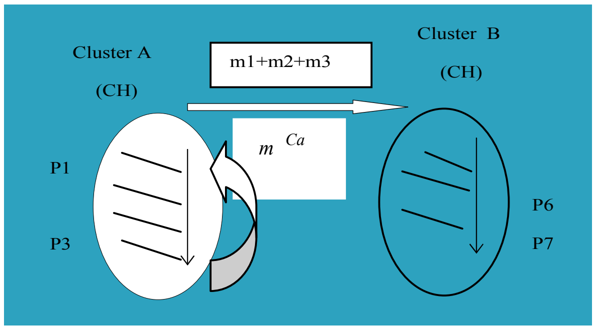

Figure 2. Cluster Communication Through Meassage Passing

Now process p_1 of cluster A which is initiating cluster head process $p_1^a(CH)$, when takes a check point, creates a composite message m^{Ca} comprising of a single control message with latest SN and application messages received during a fixed time τ say 2 ms and then sends this composite message to CH of cluster B in time interval t say 2 ms(micro seconds). CH of cluster B on receiving this composite message m^{Ca} extracts each individual control message m^c and application message m^a from it and further multicasts them o all the corresponding receiving processes respectively in cluster B who are active in current global checkpoint interval ' I ' say within τ ms(say 2ms) .

So total time taken for control message m^c sent by cluster A to reach all the active processes in cluster $B = (\tau + t + \tau)$= 6 ms. Now say after 2ms of sending the first composite message by $p_1^a(CH)$ of cluster A, a process p_3^a belonging to same cluster sends an application message $\langle m^a, SN^a, p_6\rangle$ piggybacked with SN^a along with process identity number of receiving process to cluster B through $p_1^a(CH)$ embedded in second composite message. CH of cluster B after extracting the application message $\langle m^a, SN^a, p_6\rangle$ from the received composite message , sends the message to p_6 for processing taking total time of 6ms(2+2+2) i.e. $(\tau + t + +\tau)$ i.e.. $(t+2\tau)$ .Total time taken for processing second composite message = (2+2+2+2) i.e. $(t+3\tau)$ = 8 ms where first 2 ms taken are considered on the basis that this message is sent after 2ms of recent global checkpoint interval starts which is $\cong \tau$. Accordingly within 8 ms, all the processes in the cluster come to know about the next global checkpoint to be taken even if they haven't received the first composite message containing the control message in it yet.

On basis of above observations, maximum global checkpoint interval $I = (C_{x-1} - C_x)$ is such that $T = (2+2+2+2)$i.e. $(t+3\tau)$ = 8 ms.

The proposed algorithm makes system resilient against any message delay or message loss. Since this threshold value considered is a constant and already known to each cluster, so if any process $p_1^a(CH)$ of cluster A sends a piggybacked message to cluster B, it takes again t time to reach the cluster head CH of cluster B and now the cluster extracts the SN^a piggybacked with application message . If $SN^b < SN^a$, then CH of cluster B informs all the active processes in cluster B about the next checkpoint to be taken and sends the received application message for processing to the concerned process. Therefore instead of waiting for the control message m^c to arrive which was embedded in the first composite message, the process p_6 of cluster B takes a forced checkpoint and updates its SN value with piggybacked SN^a value received in second composite message, if $Y[6][b]=1$. The first application message embedded in the second composite message , sent by a CH to any other cluster, only contains piggybacked information. However, any other process in source cluster doesn't need to piggyback SN value if it sends any other message to the same cluster before the next invocation of the proposed algorithm.

4. CHECKPOINTING ALGORITHM

*p[j][i] is the ith process in jth cluster & we assume p[j][1] as cluster head of eack cluster j, $1 \le j \le N_c$ *\

$$N_p \ge N_c \;\&\; N_p \in N_c$$

where N$_p$-Number of processes
N$_c$-Number of clusters

At Sender :

```
\* Assume $p_{ini}$ is the initiator in cluster c*\
If $p_{ini} == CH[c]$
Step 1: $CH[c]$ takes a checkpoint
Step 2: checks Y[k][c]==1 for each process $k$ in
        Cluster $c$
Step3: $CH[c]$ sends $m^{Ca}$ to $CH[j]$ where
$CH[j], \forall 1 \le j \le N_c$.
Step 4: $SN^c = SN^c + 1$;
```

	Hierarchical [3]	Roll-forward [1]	Hybrid [2]	Proposed Algorithm
Cost	$2*n_{broad} + n*c_{air}$	$n_{min}*c_{air}$	$3n*c_{air}$	$N-1*c_{air}$
Temporary Checkpoints	Yes	No	Yes	No
Non-blocking	Yes	No	Yes	No
No. of check points	n_{min}	n_{min}	n+1	n_{min}
Message complexity	O(n)	O(n)	O(n)	O(1)

Step 5: Set Y[k][c]=0 for each process k in cluster c

At Receiver:

On receiving m^{Ca} from cluster c, each $CH[j], \forall 1 \le j \le N_c$ checks for process p_i satisfying condition $Y[i][j] == 1, \forall\, 1 \le i \le N_p$

Step 1 : $CH[j]$ after extracting control message $\langle m^c, SN^c \rangle$ from m^{Ca} sends it to processes with Y[i][j]==1.

Step 2 : $SN^j = SN^j + 1$; $1 \le j \le N_c$ & $CH[c] \notin N_c$

Step 3: Each extracted m^a from m^{Ca} is sent to corresponding receiving active process in cluster j where $\forall 1 \le j \le N_c$

Step 4: Set Y[i][j]==0 for $1 \le j \le N_C$, $1 \le i \le N_p$

End of algorithm.

5. IMPLEMENTATION

It is reasonable to say that the major source of overhead in checkpointing schemes is the response time and network communication latency. Communication overhead becomes a minor source of overhead as the latency of network communication decreases. In this scenario, the coordinated checkpoint becomes worthy since it requires less accesses to stable storage then uncoordinated checkpoints. Moreover, Composite message composition further helps in reducing the network latency significantly thereby resulting in reduced transmission delay, communication cost, better bandwidth utilization and faster speed of executioN.

6. PERFORMANCE

The main advantage of our algorithm over the algorithms [1][2][3] is that the inter-cluster transmission delay, communication cost and bandwidth usage remains minimum with increasing number of processes also ,thereby making the system more efficient, less prone to network failure and worth using for application running with low strength bandwidth available. We have presented the comparison of performance of the above three algorithms with our algorithm in Table 1.

Consider a cluster federation consisting of n clusters where

N-	number of clusters
n_{min} -	minimum number of processes that need to take a check point
C_{air} -	cost of sending a message from one process to another
n_{multi} -	time taken to multicast a message to all processes in the system
n_{broad} -	time taken to broadcast a message to all processes in the system

Table 1: Performance Comparison of the Check pointing Algorithms

The cost to complete the checkpointing process using [1] is given as $n_{min} * c_{air}$. It is a single phase algorithm and it multicasts only one type of control message. In [2], the initiator sends control messages to all the processes costing $n * c_{air}$. Then all the receiving processes acknowledge the initiator process about the receiving of control message costing $n * c_{air}$, then finally the initiator sends all the processes a control message requesting for commit. Hence total cost to complete the checkpointing process using algorithm [2] is given as $(3 * n * c_{air})$.

In [3] the initiator in each cluster broadcasts a check pointing request to all the processes costing n_{broad}. The initiator receives replies from n processes, the cost of which is $n * c_{air}$. Finally the initiator broadcasts a commit message to all processes to convert their temporary check points to permanent ones, the cost of which is n_{broad}. Hence the total cost of [3] is $n * c_{air} + 2 * n_{broad}$.

In proposed algorithm, communication takes place directly between clusters. So when one cluster sends a control message to other, it is received by cluster head CH and multicast to the participating processes in current global checkpoint interval. Since only one control message per cluster is sent and if there are N clusters, one cluster sends $N-1$ messages to all the remaining clusters. Hence the cost is $N - 1 * c_{air}$.

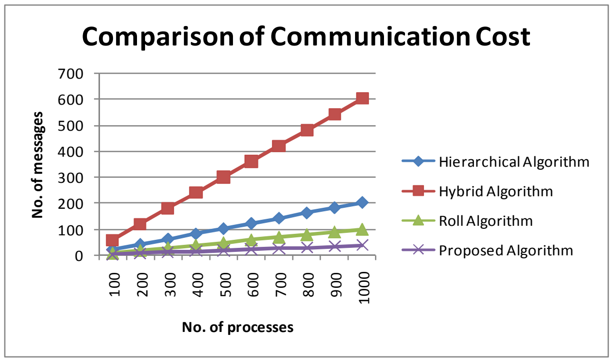

Figure 2. Performance Comparison of Proposed Algorithm with Existing Algorithms

In Fig. 2, the ordinate represents the cost of sending the control messages to complete the check pointing algorithm in the best case for the four algorithms. Fig. 2 clearly demonstrates the better performance of our approach than the ones in [1],[2] & [3] in terms of message passing. In our approach, with the increase in number of processes the number of control messages transferred is minimum. Hence the least chances of traffic congestion and bandwidth utilization

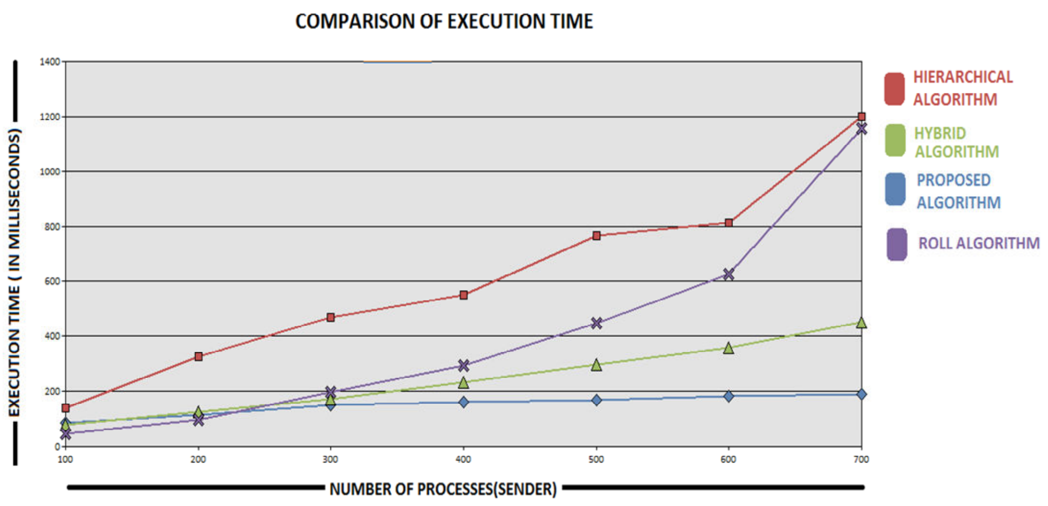

Fig.3 Cpu Utilization Comparison

7. REFERENCES

[1] B. Gupta, S. Rahimi, and R. Ahmad, " A New Roll-Forward Checkpointing / Recovery Mechanism for Cluster Federation", International Journal of Computer Science and Network Security, vol. 6, no.11, pp. 292-297, Nov. 2006.

[2] J. Cao, Y. Chen, K. Zhang and Y. He, "Checkpointing in Hybrid Distributed Systems", IEEE ISPAN'04, 2004.

[3] S. Monnet, C. Morin and R. Badrinath,"A Hierarchical Checkpointing Protocl for Parallel Applcations in Cluster Federations", IEEE IPDPS 2004, 2004.

[4] B. Gupta, S. Rahimi, and Z. Liu, "A New High Performance Checkpointing Approach for Mobile Computing Systems", International Journal of Computer Science and Network Security, vol. 6, no. 5, May 2006.

[5] G. Cao, and M. Singhal, "On coordinated checkpointing in distributed systems", IEEE Transactions on Parallel and Distributed Systems, vol. 9, no. 12, pp. 1213 – 1225, Dec. 1998.

[6] R. Prakash, and M.Singhal, "Low-Cost Check pointing and Failure Recovery in Mobile Computing Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 7, no. 10, pp. 1035-1048, October 1996.

[7] G.Cao, and M. Singhal, "Mutable checkpoints: a new checkpointing approach for mobile computing systems," IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 2, pp. 157-172, Feb. 2001.

[8] K.M. Chandy, and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," ACM Transactions on Computer Systems, vol. 3, no. 1, pp. 63-75, Feb. 1985.

[9] P. Kumar, L. Kumar, R.K. Chauhan, and V.K. Gupta, "A non-intrusive minimum process synchronous checkpointing protocol for mobile distributed systems," ICPWC 2005, IEEE International Conference on Personal Wireless, vol. 3, no. 1, pp. 63-75, Feb. 1985.

[10] L. M. Silva, and J.G. Silva, "Global checkpointing for Distributed Programs," Proceedings of 11th symposium on Reliable Distributed Systems, pp. 155 –162, Oct. 1992.

[11] B. Gupta, S. Rahimi, and Z. Liu, "A New Non-Blocking Synchronous Checkpointing Scheme for Distributed Systems," Proceeding of 20th International Conference on Computers and Their Applications, pp. 26–31, Mar. 2005.

[12] R. Prakash, and M.Singhal, "Low-Cost Check pointing and Failure Recovery in Mobile Computing Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 7, no. 10, pp. 1035-1048, Oct. 1996.

[13] D. Manivannan, and M. Singhal, "Asynchronous Recovery Without Using Vector Timestamps," Journal of Parallel and Distributed Computing, vol. 62, no. 12, pp. 1695-1728, Dec. 2002.

[14] B. Gupta, S. Rahimi and Yixin Yang, "A Novel Roll-Back Mechanism for Performance Enhancement of Asynchronous Checkpointing and Recovery", Informatica, pp. 1–13, 2007.