

Two phase algorithm to establish consistent checkpoints for recovery in multi process environment

Ashish Chauhan, Rohit Bishnoi, Shashank Aggarwal,
Shivam Srivastav, Shubham Tanwar

^{1 2 3 4 5} Department of Computer Science & Engineering (MEERUT INSTITUTE OF TECHNOLOGY, MEERUT)
¹ Assistant Professor ^{2 3 4 5} B.Tech Scholars

Abstract: - The problem of bringing a distributed multi-process system to a consistent state after transient failure is quite a difficult task. This paper addresses two components of this problem by describing a distributed algorithm to create consistent checkpoints, as well as rollback-recovery algorithm to recover the system to a consistent state. The motive of this algorithm is to make system more fault-tolerant. The likelihood of fault grows as systems are becoming more complex and applications are requiring more resources, including execution speed, storage capacity and many more. A checkpoint is a local state of a process saved on stable storage. In case of fault in distributed system, checkpoints enable the execution of the program to resumed from a previous consistent global state rather than resuming the execution from the beginning. When a process takes a checkpoint, minimal number of additional processes is forced to take checkpoints. Similarly when a process rollback and restarts from failure, a minimal number of additional processes are forced to rollback with it. This paper presents an efficient that algorithm works on Coordinator and Cohorts mechanism where the consistent set of checkpoints is established when it is directed by the coordinator. The checkpoints are established when all cohorts finishes the task assigned by the coordinator and there is no message in transit.

Keywords: - Distributed system, checkpoints, consistent state rollback recovery, orphan message and domino effect.

I. INTRODUCTION

As the technologies of processors and networks have rapidly been developed, message passing systems consisting of networked computers can provide supercomputer like performance parallel and distributed computing environments. The parallel processing capacity of a network of workstations is seldom exploited in practice. This is in part due to the difficulty in building applications program that can tolerate fully failure that is common in such environments. Check pointing and rollback are well known techniques that allow process to make progress instead of failure [1]. The failure under consideration are transient problems such as hardware errors and transient aborts, i.e., those that are unlikely to recur when a process restarts. The state of each process in a system is periodically saved on the stable storages, which are called checkpoints of the process. To recover from a failure the system restarts its execution from a previous error free, consistent state recorded by the checkpoints of all processes. More specifically, the failed processes are restarted on any available machine and their address spaces are restored from the latest checkpoints from stable storage. Other process may have to roll back to their latest checkpoints on the stable storage in order to restore the entire system to a consistent state. If there is a failure, one may restart computation from the last checkpoint, thereby avoiding repeating computation from the beginning. The process of resuming computation by rolling back to a saved state is called rollback recovery.

II. CONSISTANT GLOBAL STATES IN DISTRIBUTED SYSTEMS

The notion of a global consistent state is central to reasoning about distributed system. It was considered in [2], [3], [4] and formalized by Chandy and Lamport[5]. In a distributed system computation, an event can be spontaneous state transition by a process, or by sending or receipt of a message. Event A happens before event B[t] if and only if

1. A and B are events in the same process and A occur before B; or
2. A is the sending of a message m by a process B is the receiving of m by another process.

The transitive closure of directly happens before relation is that happens before relation. If A happen before B, B happens after A (abbreviate happens before, “before” and happens after, “after”). In a distributed system, since the process in a system do not share memory, a global state of the system is called or defined as a set of local states one from each process. The state of channels corresponding to a global state is the set of message sent but not yet received [6].

Two phase algorithm to establish consistent checkpoints for recovery in multi process environment

A system state is said to be consistent if it contains no orphan message; i.e., whose receive event is recorded in the state of the destination process, but its send event is lost [7]. In order to record a consistent global checkpoint on the stable storage, process must synchronize their checkpoint activities. In other words, when a process takes a checkpoint, it asks (by sending checkpoint requests to) all relevant process to take checkpoints. Therefore; consistent check pointing suffers from high overhead associated with the check pointing process.

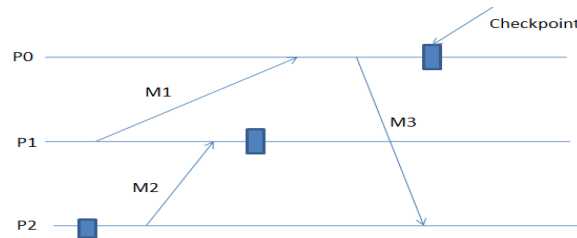


Figure 1: Shows an orphan message

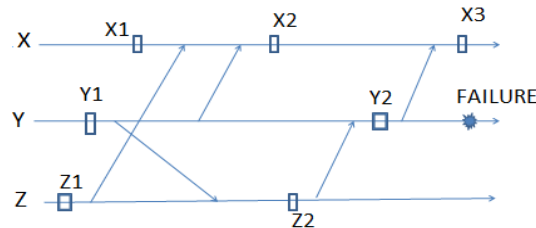


Figure 2: An example showing domino effect

From figure 2, it can be seen if Z rollbacks then all 3 processes must be rollback to their first recovery point i.e. X₁, Y₁, Z₁. So this effect of rolling back of one process causing one or more processes to rollback is known as Domino Effect [6].

III. SYSTEM MODEL

The distributed system computation consists of N separated sequential process denoted by P₁P₂P₃.....P_N. The processes do not share a common memory or a common clock. Message passing is the only way for process to communicate with each other. The computation is asynchronous: each process progresses its own speed and message are exchanged through reliable FIFO channels whose transmission delays are finite but arbitrary. The system is assumed to be consisting of single machine. The machine is connected to stable storage and secondary storage. A storage that doesn't lose information in the event of system is referred as stable storage.

IV. PROPOSED WORK

In the proposed system there are N processes P₁P₂.... P_N. One process will act **controlling agent** or **master** which remain active throughout the process and others are slaves which are idle. Master process assign task to slave processes. Only master process can initiate other process by sending a message to it. After receiving the message the slave process become active and start working on the task which is assigned by master.

Controlling agent has weight 'w' and slaves have zero weight. Whenever a task is assigned to the slave process by master process the weight of master process is divided between master process and slave process in exactly two halves. This activity of division of weight between master process and slave processes will continues until the complete works is divided by master to the slaves. When the task is completed by slave process it returns the weight back to the master process. When the weight of master process is again 'w' a checkpoint is assigned to the system.

The algorithm is given below:

Phase I:-

Step 1:- Initially all the processes are idle except the master process which is active. When a task starts, one process acts as master process also known as controlling agent and other as slave processes. Master process assigns work to slave processes.

Initially the weight of master process is w (usually 1) and slave processes have weight 0.

Two phase algorithm to establish consistent checkpoints for recovery in multi process environment

Step 2:-When a task is assigned to the system master process assigns the fragmented work to slave processes. Master process sends a message and half of its weight to slave processes with each message. This process is repeated every time until complete work is assigned to slave processes. Weight is assigned in such a way that at every moment the weight of the whole system is w or 1.

Step 3:-After completion of work that is assigned to slave processes, they return the weight back to master process and takes a tentative checkpoint. After taking checkpoints slave processes becomes idle and waits for acknowledgement from master. The returned weight is added to the master process's own weight.

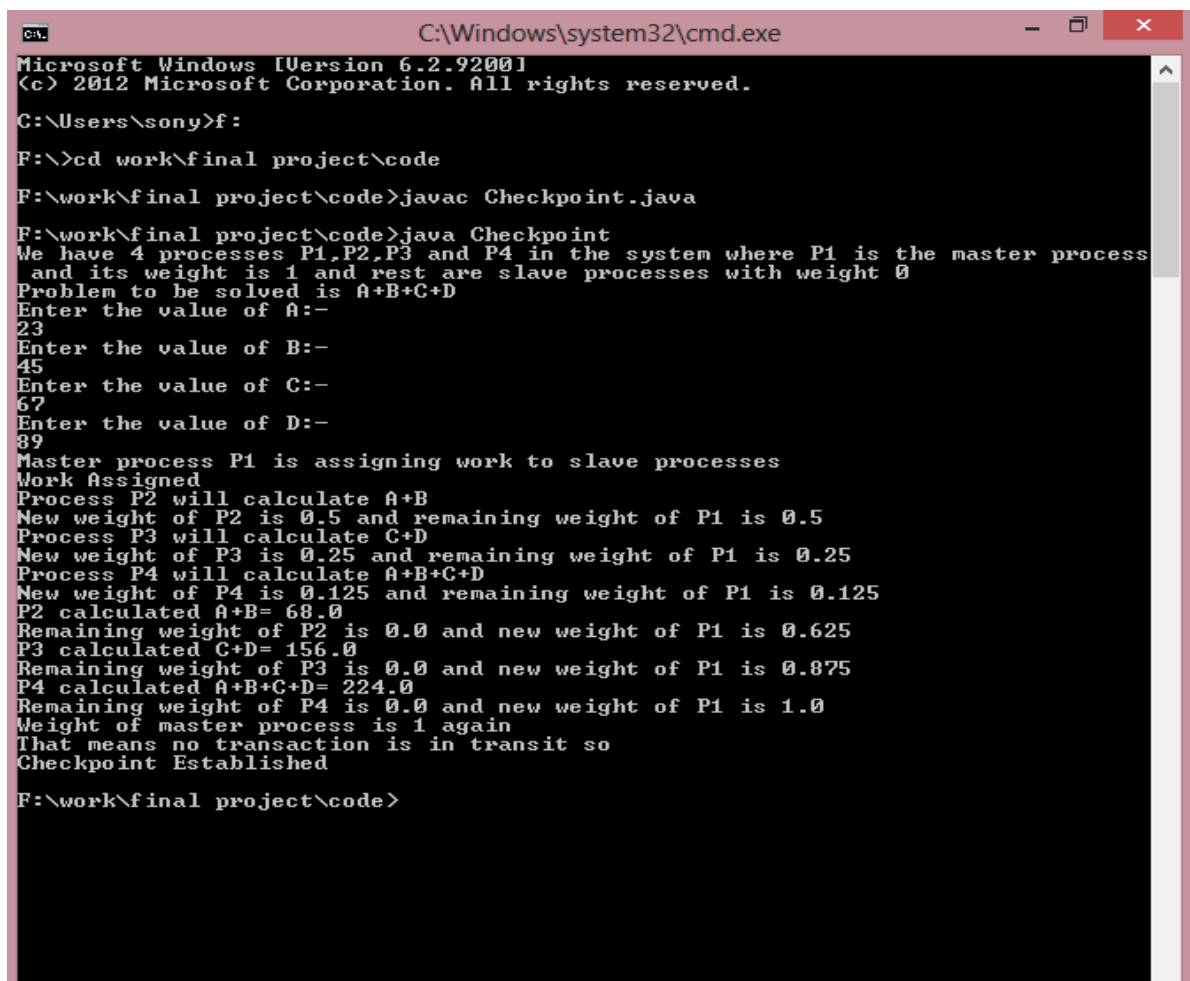
Step 4:- On receiving the responses from the slave processes the master process checks whether its weight is equal to ' w ' or not, if not then it waits for the response from the other processes. When master process's weight equals to ' w ' it is assumed that no transaction is going on, no message is in transit i.e. the tasks at slaves site has completed.

Phase II:

When the task gets completed, master process sends back acknowledgement all the slave processes to take a permanent checkpoint in their stable storage. If in the mean while if any problem occurs the master process sends message to slaves to discard their tentative checkpoint and will restart the task. At the completion of every work all the tentative checkpoints are made final assign a checkpoint to that system.

V. IMPLEMENTATION

The proposed algorithm is demonstrated for 4 processes here. Here the simple activity of addition is taken which is performed by four processes ie P_1, P_2, P_3, P_4 . Here P_1 will act as coordinator remaining processes will work as slaves. P_1 gives task to P_2, P_3, P_4 .



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.
C:\Users\sony>f :
F:\>cd work\final project\code
F:\work\final project\code>javac Checkpoint.java
F:\work\final project\code>java Checkpoint
We have 4 processes P1,P2,P3 and P4 in the system where P1 is the master process
and its weight is 1 and rest are slave processes with weight 0
Problem to be solved is A+B+C+D
Enter the value of A:-
23
Enter the value of B:-
45
Enter the value of C:-
67
Enter the value of D:-
89
Master process P1 is assigning work to slave processes
Work Assigned
Process P2 will calculate A+B
New weight of P2 is 0.5 and remaining weight of P1 is 0.5
Process P3 will calculate C+D
New weight of P3 is 0.25 and remaining weight of P1 is 0.25
Process P4 will calculate A+B+C+D
New weight of P4 is 0.125 and remaining weight of P1 is 0.125
P2 calculated A+B= 68.0
Remaining weight of P2 is 0.0 and new weight of P1 is 0.625
P3 calculated C+D= 156.0
Remaining weight of P3 is 0.0 and new weight of P1 is 0.875
P4 calculated A+B+C+D= 224.0
Remaining weight of P4 is 0.0 and new weight of P1 is 1.0
Weight of master process is 1 again
That means no transaction is in transit so
Checkpoint Established
F:\work\final project\code>
```

Figure 3: Working of Algorithm

VI. CONCLUSION

Fault tolerance in Distributed systems pose new challenging problems of maintaining the consistency of the overall system as a result the check pointing is evolved to overcome the effect of intermediate failure or faults that occurs in the system. Although the check pointing scheme saves the status of system at some intermediate points (Checkpoints) and a rollback to the latest saved state is done at the occurrence of a failure. Therefore, it reduces the rollback portion through at the cost of additional overheads for checkpoints The algorithm proposed here is quite effective as either all or none of the process of the system will take the checkpoint to which the system will rollback in case of failure.

REFERENCE

- [1] B. Randell, P. A. Lee, and P. C. Treleaven, "Reliability issues in computing system design," *ACM Comput. Surveys*, vol. 10, no. 2, pp. 123-166, June 1978.
- [2] D. L. Presotto, "Publishing: A reliable broadcast communication mechanism," *Comput. Sci. Division, Univ. California, Berkeley, Tech. Rep. UCB/CSD 83-165*, Dec. 1983.
- [3] B. Randell, "System structure for software fault tolerance," *IEEETrans. Software Eng.*, vol. SE-1, pp. 226-232, June 1975.
- [4] D. L. Russel, "Process backup in producer-consumer systems," in *Proc. ACM Symp. Operat.Syst. Principles*, Nov.1977.
- [5] K. M. Chandy and L. Lamport, "Distributed snapshots: Deterining global states of distributed systems," *ACM Trans. Comput. Syst.*, vol. 3, no. 1, pp. 63-75, Feb. 1985.
- [6] Adnan Agbaria, Wiilliam H Sanders, " Distributed Snapshots for Mobile Computing Systems", *IEEE Intl. Conf. PERCOM'04*, pp. 1-10, 2004.
- [7] Alvisi, Lorenzo and Marzullo, Keith, " Message Logging: Pessimistic, Optimistic, Causal, and Optimal", *IEEE Transactions on SoftwareEngineering*, Vol. 24, No. 2, February 1998, pp. 149-159.
- [8] Silva, L.M. and J.G. Silva, "Global checkpointing for distributed programs", *Proc. 11th symp. Reliable Distributed Systems*, pp. 155-62, Oct.1992
- [9] Koo R. and Toueg S., "Checkpointing and Roll-Back Recovery for Distributed Systems," *IEEE Trans. on Software Engineering*, vol. 13, no.1, pp. 23-31, January 1987.
- [10] Cao G. and Singhal M., "Mutable Checkpoints: A New CheckpointingApproach for Mobile Computing systems," *IEEE Transaction OnParallel and Distributed Systems*, vol. 12, no. 2, pp. 157-172, February2001
- [11] Huang, S-T., " Detecting termination of distributed computation by external agents " proceedings of the 9th international conference on distributed computing systems, 1989, pp. 79-84