AN EFFICIENT RECOVERY MECHANISM
WITH CHECKPOINTING APPROACH FOR
CLUSTER FEDERATION

Manoj Kumar
Department of Computer Science Engineering, Bhagwant University, Ajmer, Rajasthan

ABSTRACT

Checkpoint and recovery protocols are commonly used in distributed applications for providing fault
tolerance. A distributed system may require taking checkpoints from time to time to keep it free of arbitrary
failures. In case of failure, the system will rollback to checkpoints where global consistency is preserved.
Checkpointing is one of the fault-tolerant techniques to restore faults and to restart job fast. The algorithms
for checkpointing on distributed systems have been under study for years.

It is known that checkpointing and rollback recovery are widely used techniques that allow a distributed
computing to progress inspite of a failure. There are two fundamental approaches for checkpointing and
recovery. One is asynchronous approach, process take their checkpoints independently. So, taking checkpoints
is very simple but due to absence of a recent consistent global checkpoint which may cause a rollback of
computation. Synchronous checkpointing approach assumes that a single process other than the application
process invokes the checkpointing algorithm periodically to determine a consistent global checkpoint.

KEYWORDS

WAN, LAN, Checkpointing, Recovery, SAN’s, Distributed System, Cluster, VANET’s.

1. INTRODUCTION

Mobility management is one of the major functions of a GSM or a UMTS network that allows
mobile phones to work. The aim of mobility management is to track where the subscribers are,
allowing calls, SMS and other mobile phone services to be delivered to them. In a cellular
telephone network, handoff is the transition for any given user of signal transmission from one
base station to a geographically adjacent base station as the user moves around. In an
ideal cellular telephone network, each end user's telephone set or modem (the subscriber's
hardware) is always within range of a base station. The region covered by each base station is
known as its cell. The size and shape of each cell in a network depends on the nature of the
terrain in the region, the number of base stations, and the transmit/receive range of each base
station. In theory, the cells in a network overlap; for much of the time, a subscriber's hardware is
within range of more than one base station. The network must decide, from moment to moment,
which base station will handle the signals to and from each and every subscriber's hardware.
Vehicular ad hoc networks are gaining importance for inter-vehicle communication, because they
allow for the local communication between vehicles without any infrastructure, configuration
effort, and without the high costs of cellular networks. Besides local data exchange, vehicular
applications may be extended by accessing Internet services. The access is provided by Internet
gateways installed along the roadside. However, the Internet integration requires a respective
mobility support of the vehicular ad hoc network. In this paper we propose MMIP6, a communication protocol that integrates multihop IPv6-based vehicular ad hoc networks into the Internet. Whereas existing approaches are focused on small-scale ad hoc networking scenarios, MMIP6 is highly optimized for scalability and efficiency. The evaluation showed that MMIP6 is a suitable solution providing a scalable mobility support with an acceptable performance characteristic. Typical ITS applications can be categorized into safety, transport efficiency, and information/entertainment applications (i.e., infotainment) [1]. Vehicular ad hoc networks (VANETs) are emerging ITS technologies integrating wireless communications to vehicles. Different Consortia (e.g., Car-to-Car Communications Consortium (C2C-CC) [2]) and standardization organization (e.g., IETF) have been working on various issues in VANETs. C2C-CC aims to develop an open industrial standard for inter-vehicle communication using wireless LAN (WLAN) technology. For example, IEEE 802.11p or dedicated short range communications (DSRC) is an extension of 802.11 standards for inter-vehicle communication by IEEE working group. IETF has standardized Network Mobility Basic Support (NEMO BS) [3] for network mobility in VANETs. Originating from cellular networks, mobility management has been an important and challenging issue to support seamless communication. Mobility management includes location management and handoff management [4]. Location management has the functions of tracking and updating current location of mobile node (MN). Handoff management aims to maintain the active connections when MN changes its point of attachment. VANET is a special type of mobile ad hoc networks (MANETs) [5] with unique characteristics. Due to the high mobility of vehicles, topologies of VANETs are highly dynamic.

2. PHASES OF CHECKPOINTING

Checkpointing has two phases:

• Saving a checkpoint
• Checkpoint recovery following the failure.

To save a checkpoint, the memory and system, necessary to recover from a failure is sent to storage. Checkpoint recovery involves restoring the system state and memory from the checkpoint and restarting the computation from the checkpoint stored [6].

3. DATA STRUCTURE

Notations used:
\(SN\) - 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
\([P_i, Y_i]\) - Process identity number of \(i^{th}\) process, flag \(Y\) for \(i^{th}\) process
\(v_j^i\) - keeps a record of SN for each process \(P_i\) in cluster \(j\)
\(C_i(x)^j\) - \(X^{th}\) checkpoint of process \(i\) in cluster \(j\)
\(Y[i][j]\) - is the flag used to identify active processes at \(x^{th}\) checkpoint
\(t\) - Time taken for a control or application message to reach from one CH to another CH
\(p_i^a(CH)\) - The checkpoint initiating cluster head process cluster \(a\)
The aim of this thesis is to present an efficient, better bandwidth utilization, maximum response time, decentralized and cost effective checkpointing algorithm suitable for cluster federation. Throughout this survey, we use \( N_p \) to denote the total number of processes and \( N_c \) is the clusters in the system where \( N_p \) is much larger than \( N_c \). Each process is assigned a unique id-number \( I \) (1\( \leq \) \( i \) \( \leq \) \( N_p \)).

In our checkpointing scheme, for each process in the cluster, the checkpointing dependency information is maintained by its cluster head process. Each Cluster Head \( CH \) sends the control messages to the cluster head of other clusters which further multicasts the message to all currently active processes in the cluster.

This scheme reduces the message passing and number of lost messages is also reduced drastically, thus making system more available, reliable and faster. When a checkpointing procedure begins, the sending and the receiving of control messages are mainly accomplished amongst cluster head processes.

To maintain such additional information for processes, each \( CH \) maintains a 2-tuple table \( [P_i, Y_i] \) where \( (1 \leq i \leq N_p \) \). A vector \( v_i \) for keeping a record of \( SN \) (Sequence Number) for each process \( p_i \) in cluster \( j \) where flag \( Y_{ji} j \) = 0 in case, process \( p_i \) neither receives or sends any message during current global interval \( \left( C_i(x)^{−} C(x−1)^{+} \right) \) at \( X^{th} \) check point. After the global check point is taken, both the fields in the table are set as empty and \( SN \) \( ^{+} \) are incremented.

4. RELATED WORK

S Kalaiselvi et.al [8] studied the algorithms for checkpointing parallel/distributed systems. It has been observed that most of the algorithms published for checkpointing in message passing systems are based on the seminal article by Chandy and Lamport. Number of reports have been published in this area by relaxing the assumptions made in this paper and by extending it to minimize the overheads of coordination and context saving.

Jiannong Cao et.al [9] proposed to address the need of applying different checkpointing schemes to different subsystems inside a single target system. The proposed algorithm has several advantages.

Ch. D. V. Subba Rao et.al [10] had proposed a new checkpointing protocol combined with selective sender based message logging . The protocol is free from the problem of lost messages Partha Sarathi et.al [11] several schemes for checkpointing and rollback recovery have been reported in the literature. We analyze some of these schemes under a stochastic model. We have derived expressions for average cost of checkpointing, rollback recovery, message logging and piggybacking with application messages in synchronous as well as asynchronous checkpointing. For quasi-synchronous checkpointing we show that in a system with \( n \) processes, the upper bound and lower bound of selective message logging are \( O(n^2) \) and \( O(n) \), respectively.

Y. Manable et.al [12] proposed a distributed coordinated checkpointing algorithm . A consistent global checkpoint is a set of states in which no message is recorded as received in one process
and as not yet sent in another process. This algorithm obtains a consistent global checkpoint for any checkpoint initiation by any process.

S. Monnet et.al [13] suggested that a cluster takes two types of checkpoints, processes inside a cluster take checkpoint synchronously and a cluster takes a communication induced checkpoint whenever it receives an inter cluster application message.

J. Cao et.al [14] analyzed the need of integrating independent and coordinated checkpointing schemes for applications running in a hybrid distributed environment containing multiple heterogenous subsystems.

B. Gupta et.al [15] presented a simple non-blocking roll forward checkpointing/recovery mechanism for cluster federation. The effect of domino phenomenon is limited by the time interval between successive invocations of the algorithm and recovery is as simple as that in the synchronous approach.

Suriender Kumar et.al [16] focused on the hierarchical non blocking coordinated checkpointing algorithms suitable for distributed computing and eliminating the overhead of taking temporary checkpoints.

Guohui et.al [17] in distributed computing systems, processes in different hosts take checkpoints to survive failures. For mobile computing systems, due to certain new characteristics such as mobility, low bandwidth, disconnection, low power consumption and limited memory, conventional distributed checkpointing schemes need to be reconsidered. In this paper, a novel min-process coordinated checkpointing algorithm that

Qiangfeng Yang et.al [18] checkpointing and rollback recovery are widely used techniques for achieving fault-tolerance in distributed systems. In this paper, we present a novel checkpointing algorithm which has the following desirable features: A process can independently initiate consistent global checkpointing by saving its current state, called a tentative checkpoint. Other processes come to know about a consistent global checkpoint initiation through information piggy-backed with the application messages or limited control messages if necessary.

Bidyut Gupta et.al [19] had presented a non-blocking coordinated checkpointing algorithm suitable for mobile environments. The advantages make the proposed algorithm suitable for mobile distributed computing systems are following advantages: (a) the proposed algorithm does not take any temporary checkpoint and hence the overhead of converting temporary checkpoint to permanent checkpoint is eliminated. (b) the proposed algorithm does not use mutable checkpoints. Hence the overhead of converting them to permanent ones is eliminated. (c) their algorithm does not allow any process to take useless checkpoints. It uses very few control messages and participating processes are interrupted less number of times.

Lalit Kumar et.al [20][7] presented a non-blocking minimum process coordinated checkpointing protocol that not only minimizes useless checkpoints but also minimizes overall bandwidth required over wireless channels. In their proposed protocol the height of checkpointing tree proposed to reduce. This will reduce the uncertainty period and number of induced checkpoint.

J. L. Kim et.al [21] had presented a new efficient synchronized checkpointing protocol which exploits the dependency relation between processes in distributed systems. In their protocol, a process takes a checkpoint when it knows that all processes on which it computationally depends took their checkpoints, and hence the process need not always wait for the decision made by the checkpointing coordinator as in the conventional synchronized protocols.
5. WORKING MODEL

In 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, 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 each intercluster control message along with the first application message sent to any process in any cluster during \( X \) 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 prone to less network failure and for applications which use secure network media for message communication. (ii) \( CH \) communicates the intercluster received messages to all the active processes in the cluster within 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 \( CH \) must multicast the received messages to all processes in the cluster, participating during current global checkpoint interval \( (c_{x-1} - c_x) \).

Let us assume that the time taken on an average by a cluster head to send a control 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 [35], If a control message is to be sent to processes in a cluster, time taken by a sending process \( p_i \) in cluster \( a \) for any processes \( 1 \leq p_j \leq n \) in cluster \( b \) is \( t \). If the process \( p_i \) is supposed to send the control message to all the processes in cluster \( b \) directly, it will take \( n \times t \). In the proposed algorithm, the \( CH \) of cluster \( b \) checks for the value of \( Y_i \) where \( 1 \leq i \leq n \) and multicasts control messages to all the processes with value of \( Y_i \). Suppose time taken by \( CH \) to multicast the control message \( m^c \) among active processes is \( \tau \) which is a small fraction of time \( t \) as cluster \( b \) uses SAN, 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 \( (t + \tau) \). This value \( (t + \tau) \) is considered as a threshold value to keep a check on transmission delay caused by \( CH \). Although this threshold value varies during each global checkpoint 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 figure 1.
Now process \( p_1 \) of cluster ‘\( a \)’, which is initiating cluster head process \( p^{a}_{1}(CH) \) sends a control message \( m^c \) to \( CH \) of cluster ‘\( b \)’ in time interval \( t \) say 2 ms (micro seconds). \( CH \) of cluster \( b \) on receiving this control message \( m^c \) further multicasts it to all the processes of cluster \( b \) who are active in current global checkpoint interval ‘\( I \)’ say within \( \tau \) ms (say 2ms). So total time taken for control message \( m^c \) sent by cluster \( a \) to reach all the active processes in cluster \( b = (t + \tau) = 4ms \). Now say after 2ms of sending the control message by \( p^{a}_{1}(CH) \) of cluster ‘\( a \)’, a process \( p_4 \) 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 \( p^{a}_{1}(CH) \). \( CH \) of cluster \( b \) after extracting the information from the received message , sends the message to \( p_6 \) for processing taking total time of 4ms(2+2) i.e. \( (t + \tau) \). Total time taken for processing first application message \( m^a = (2+2+2) \) i.e. \( (t + 2\tau) = 6 \) 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 \( \equiv \tau \). Accordingly within 6 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 control message yet.

On basis of above observations, maximum global checkpoint interval \( I = (C_{s-1} - C_s) \) is such that \( T = (2 + 2 + 2 + 2) \) i.e. \( (t + 3\tau) = 8 \) ms and 2 sec. is for time taken to composite message. 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 \( p^{a}_{1}(CH) \) of cluster ‘\( a \)’ sends a piggybacked computation message to cluster ‘\( b \)’, it takes again \( l \) 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, the process \( p_6 \) of cluster ‘\( b \)’ takes a forced checkpoint and updates its \( SN \) value with piggybacked \( SN^a \) value, if \( Y[6][b] = 1 \). The first application message sent by a \( CH \) to any other cluster only contains piggybacked \( SN^a \) value. 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.
6. CHECKPOINTING ALGORITHM

\*p[j][i] is the \(i\)th process in \(j\)th cluster & we assume \(p[j][1]\) as cluster head of each cluster \(j, 1 \leq j \leq N_c\) *\n
**Step 1:** \(N_p \geq N_c \& N_p \in N_c\)

where \(N_p\) - Number of processes
\(N_c\) - Number of clusters

**Step 2:** \*Assigning process id*\n
\(k = 1\);

For \(j = 1\) to \(N_c\)

{For \(j = 1\) to \(N_p\)
  \(p[j][i] = K;\)
  \(k = k + 1;\)
  \(i = i + 1;\)
}
\(j = j + 1;\)

**Step 3:** \*Identifying and Assigning cluster head-id*\n
For \(j = 1\) to \(N_c\)

{\(CH[j] = p[j][1];\) \* for \(j\)th cluster*\n  \(j++ ;\)
}

**Step 4:** \(Y[i][j] = 0;\) \(\forall 1 \leq j \leq N_c\)

1 \(\leq i \leq N_p\)

**At Sender:**

\* Assume \(p_{mi}\) is the initiator in cluster \(c\)*\n
If \(p_{mi} == CH[c]\)

Step 1: takes a checkpoint

Step 2: checks \(Y[k][c] == 1\) for each process \(k\) Cluster \(c\)

Step 3: sends \(\langle m^{c}, SN^{mi} \rangle\) to processes with \(Y[k][c] == 1\) and to each element of \(CH[j], \forall 1 \leq j \leq N_c\).

Step 4: \(SN^{c} = SN^{c} + 1;\)

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

Else

Step 6: takes a checkpoint & informs \(CH[c]\).

Step 7: \(CH[c]\) repeats the Step 2 to Step 5.

**At Receiver:**

On receiving \(\langle m^{c}, SN^{c} \rangle\) from cluster \(c\), each \(CH[j], \forall 1 \leq j \leq N_c\) checks for process \(p_i\) satisfying condition \(Y[i][j] == 1, \forall 1 \leq i \leq N_p\)

Step 1: \(CH[j]\) sends \(\langle m^{c}, SN^{c} \rangle\) to processes with \(Y[i][j] == 1\).

Step 2: \(SN^{j} = SN^{j} + 1;\) \(1 \leq j \leq N_c\) & \(CH[c] \notin N_c\)
Step 3: Set $Y[i][j]=0$ for $1 \leq j \leq N_C, \ 1 \leq i \leq N_P$
End of algorithm.

**RECOVERY ALGORITHM**

For each Process $P_k$ and $1 \leq i \leq n, i \neq k$
if $S_{ik} > R_{ki}$

$P^*$ records these sequence numbers $(R_{ki}^* + 1)$ to $S_{ik}$ in lost-form-$P_i^k$;

//message with sequence numbers $(R_{ki}^* + 1)$ to $S_{ik}$ are the lost messages from

$P_i$ to $P_k$

$P^*$ forms the total order of all lost messages sent by every $P_i, i \neq k$ to $P_k$ using

lost-form-$P_i^k$ and the message log $MESG_k$ for $P_k$

7. **SYSTEM MODEL**

In the existing scheme, when a sender sends a message it is received by all the processes whether they are participating in current checkpoint interval or not, resulting in bandwidth wastage, increased communication cost and traffic congestion. In proposed checkpointing algorithm, message moves in composite form and it’s the cluster head who is responsible for sending message to other cluster heads and further each cluster head multicasts the message to all active processes. It results in efficient bandwidth utilization and making the system more cost effective and less traffic congestion prone.

![Figure 5: Without Clustering System Model](image)

![Figure 2: With Clustering System Model](image)
8. IMPLEMENTATION OF SYSTEM MODEL

This experiment uses sets of PC memory distributed databases with java platform. To evaluate the implementation of algorithm, following parameters have been taken into consideration: Bandwidth utilization, Number of clusters, Number of messages to be sent individually, Number of messages sent as a composite message, number of checkpoints taken, number of messages to be recovered since this thesis is an attempt to develop a recovery system which may succeed in reducing the number of messages required to be recovered.

- Bandwidth Utilization Versus Number of clusters

In the proposed algorithm, effort has been focused to find the fact that whether the number of composite messages depend on the number of clusters? Now consider the given Figure 6.1:

![Figure 6.1: Bandwidth Utilization Versus Number of Clusters](image)

From Figure 6.1, it is obvious that with increase in number of clusters there is increase in number of composite messages but in a graceful way. Now let us see the advantage of this fact:

**Less Number of Clusters:** If there are less number of clusters, than number of messages to be sent are almost equal to number of clusters. In case, the number of clusters sending the messages is less, the number of composite messages sent is also low and hence the bandwidth is used efficiently.

**Average Number of clusters:** If there are average number of clusters, than number of messages sent are almost two third of the number of clusters. So, with increase in number of clusters, there is a little increase in number of composite messages and hence usage of bandwidth is still efficient.

**Increased Number of Clusters:** If there is large number of sending clusters, the number of messages sent is almost half of the number of sending clusters. Hence usage of bandwidth is still efficient.

- Bandwidth Usage: As shown in the Figure 6.2, the bandwidth usage by the proposed technique is the least as compared to other techniques.
Initially, proposed technique has higher bandwidth usage, this is due to the overheads incurred in the sending of composite message. But this overhead is neutralized as soon as the number of clusters increases. Further, increase in number of clusters exponentially increases the bandwidth usage in traditional method. But in proposed technique, there is only linear increase in the bandwidth usage. So, proposed technique proves to be of great usage in the scenarios where large number of processes interacts with each other which is not so rare in real life systems.

- **Number of individual messages to be sent versus number of composite messages sent**

In the proposed algorithm, if one or more processes in the sending cluster have to send messages to one or more processes at the receiving end, may be a cluster or a site, then the sending cluster first makes a composite message comprising of all the individual messages received from processes under it. This composite message is then sent by the sending cluster to the receiving cluster and after receiving this message, the receiving cluster multicasts the appropriate extracted messages to the receiving active processes.

Figure 6.3 shows the comparison between numbers of actual messages to be sent versus number of composite messages sent. From the above figure, it is clear that during various checkpoints, the number of composite messages sent remain almost constant. And also, the number of composite messages sent are largely less than the actual individual messages to be sent, thus saving the actual bandwidth. Hence this graph clearly shows that the proposed algorithm has a caliber of improving the bandwidth usage.
• **Number of messages to be recovered with increased number of clusters**

As shown in the figure 6.4, it is clear that in the proposed technique, less number of messages need to be recovered than in the B. Gupta et.al method.

![Figure 6.4 Messages recovered versus number of clusters](image)

This is due to the fact that in proposed technique, initially a control message is sent to the receiving clusters from the sending cluster. In case, if the receiving cluster does not receive the control message in time, still it comes to know about the latest checkpoint taken when it receives the first application message embedded with latest SN sent to it by sending clusters, thus minimizing the chances of lost or orphan messages and hence, resulting in minimized recovery of messages. Moreover, no acknowledgement is sent back by the receiving cluster since even if it does not receive the control message, first application message sent to any one of its node, informs about the latest checkpoint taken and hence all the active processes in the cluster updates its synchronization number with the latest received SN.

### 3. CONCLUSIONS

Checkpointing protocols require the processes to take periodic checkpoints with varying degrees of coordination. At one end of the spectrum, coordinated checkpointing requires the processes to coordinate their checkpoints to form global consistent system states. Coordinated checkpointing generally simplifies recovery and garbage collection, and yields good performance in practice. At the other end of the spectrum, uncoordinated checkpointing does not require the processes to coordinate their checkpoints, but it suffers from potential domino effect, complicates recovery, and still requires coordination to perform output commit or garbage collection. Between these two ends are communication-induced checkpointing schemes that depend on the communication patterns of the applications to trigger checkpoints. These schemes do not suffer from the domino effect and do not require coordination. Recent studies, however, have shown that the nondeterministic nature of these protocols complicates garbage collection and degrades performance.

In this thesis, we have presented a simple non-blocking efficient and low cost checkpointing algorithm for cluster federation. The time interval considered between successive invocations of algorithm ensures minimum number of lost or delayed messages. The main features of the algorithm are: 1) Minimum number of processes takes check points in this approach. 2) Cluster to
cluster communication is minimum. 3) Each cluster maintains its own data structures for keeping the check pointing dependency information resulting in decentralized approach and faster speed of execution. 4) Wastage of bandwidth is minimum

**Future Scope**

Message is not secure. Here message is travel in plain text form so work on security. On peer to peer model it is implemented. It is used in share data base.

**REFERENCES**

[1] Jalote P. “Fault Tolerance in Distributed Systems”. 1st. edition of Englewood Cliffs, USA: Prentice Hall, 1994
[2] Randell, B. “Fault tolerance in decentralized systems”, In proceedings of the 14th international symposium on Autonomous Decentralized systems (ISA’99), pp. 174-179, March 1999
[3] Russell, D.L. “State Restoration in systems of communicating processes”, IEEE transactions on software Engineering, 6(2), pp. 183-194, March 1980
[4] Strom, R. and Yemini, S.”Optimistic recovery in distributed systems”, ACM transactions on Computer Systems, 3(3), pp. 204-226, August 1985
[5] Elnozahy, E.N., Alvisi, L., Wang, Y.-M. and Johnson, D.B. “A Survey of Rollback-recovery protocols in message passing systems”, ACM computing surveys, 34(3), pp. 375-408, September 2002
[6] Bhargava, B. and Shu-Renn, L., “Independent Checkpointing and Concurrent rollback for recovery in distributed Systems-an optimistic approach”, Proceedings of The 17th Symposium on Reliable Distributed Systems, pp. 3-12, Columbus, USA, October 1988.
[7] Wang, Y.-M. “Consistent global checkpoints that contain a given set of local checkpoints”, IEEE transactions on Computers, 46(4), pp. 456-468, April 1997
[8] S Kalaiselvi and V Rajaraman “A survey of checkpointing algorithms for parallel and distributed computers”, 25(5), pp. 489-510, October 2000
[9] Jiannong Cao, Yifeng Chen, Kang Zhang, Yanxiang He, “Checkpointing in Hybrid Distributed Systems”, In proceedings of the 7th international Symposium on Parallel Architectures, Algorithms and Networks (ISPAN’04), 2004
[10] Ch. D.V. Subba Rao and M.M. Naidu. “A New, Efficient Coordinated Checkpointing Protocol Combined with Selective Sender-Based Message Logging”, AICCSA, IEEE/ACS International Conference on Computer Systems and Applications, pp. 444-447, 2008
[11] Partha Sarathi Mandel, Krishnendu Mukhopadhyaya, “Performance analysis of different checkpointing and recovery schemes using stochastic model” Journal of Parallel and Distributed Computing, 66(1), pp. 99-107, January 2006
[12] Y. Manable. “A Distributed Consistent Global Checkpoint Algorithm with minimum number of Checkpoints”, Technical Report of IEICE, COMP97-6, April, 1997
[13] S. Monnet, C. Morin, R. Badrinath, “Hybrid checkpointing for Parallel Applications in Cluster Federations”, In 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, Chicago, IL, USA, pp. 773-782, April 2004
[14] J. Cao, Y. Chen, K. Zhang and Y. He, “Checkpointing in Hybrid Distributed Systems”, In Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN’04), pp. 136-141, Hong Kong, China, May 2004
[15] B. Gupta and S. Rahimi, and R. Ahmad “A new Roll-Forward checkpointing/Recovery Mechanism for Cluster Federation”, International journal of computer science and Network security, 6(11), pp. 292-297, November 2006
[16] Surender Kumar, Parveen Kumar, R.K. Chauhan “Design and performance analysis of coordinated checkpointing algorithms for distributed mobile systems”, In the proceedings of International Journal of Distributed and Parallel systems (IJDPS), 1(1), September 2010.
[17] Guo-Hui Li, Hong-Ya Wang, “A Novel min-process checkpointing scheme for mobile computing systems” Journal of system Architecture, 51(1), January 2005
[18] Qiangfeng Jiang, Yi Luo, D. Manivannan, “An Optimistic Checkpointing and message Logging approach for consistent global checkpoint Collection in distributed Systems” Journal of Parallel and Distributed Computing, 68(12), pp. 1575-1589, December 2008

[19] Bidyut Gupta, S. Rahimi and Z. Lui. “A New High Performance Checkpointing Approach for Mobile Computing Systems”. IJCNS International Journal of Computer Science and Network Security, 6(5B), May 2006

[20] Lalit Kumar Awasthi, Kumar “A Sychronous Checkpointing Protocol For Mobile Distributed Systems.” Probabilistic Approach. Int J. Information and Computer Security, 1(3), pp. 298-314, 2007

[21] J. L. Kim and T. Park. “An efficient protocol for checkpointing recovery in Distributed Systems” IEEE Transaction On Parallel and Distributed Systems, 4(8), pp. 955-960, August 1993