A SURVEY OF THE LITERATE ON CHECK POINTING ALGORITHMS

***T. KISHAN RAO, #DR. V.K. GAUR**
Ph.D Scholar (COMPUTER SCIENCE)
#Associate Professor, Govt. Dungar College, Bikaner

ABSTRACT

A distributed system is a collection of computers that are spatially separated and do not share a common memory. The processes executing on these computers communicate with one another by exchanging messages over communication channels. The messages are delivered after an arbitrary delay. We give theoretical foundations of distributed systems.
Keywords. *Checkpointing algorithms; parallel & distributed computing; shared memory systems; rollback recovery; fault-tolerant systems*

INTRODUCTION

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. Checkpointing is the process of saving the status information. This paper surveys the algorithms which have been reported in the literature 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. A large number of articles have been published in this area by relaxing the assumptions made in this paper and by extending it to minimise the overheads of coordination and context saving. Checkpointing for shared memory systems primarily extend cache coherence protocols to maintain a consistent memory. All of them assume that the main memory is safe for storing the context. Recently algorithms have been published for distributed shared memory systems, which extend the cache coherence protocols used in shared memory systems. They however also include methods for storing the status of distributed memory in stable storage. Most of the algorithms assume that there is no knowledge about the programs being executed. It is however felt that in development of parallel programs the user has to do a fair amount of work in distributing tasks and this information can be effectively used to simplify checkpointing and rollback recovery.

PROPOSED RESEARCH WORK

A survey of the literate on check pointing algorithms

REVIEW OF STUDIES

The concept of designing distributed algorithms that explicitly cope with host mobility, is still in its infancy. An overview of the impact of host mobility on distributed computations is presented in [7], while [6] contains preliminary ideas that have been fully developed in this paper. The implications of mobile for distributed data management are considered in [20, 19, 4, 8]. Other related work includes addressing schemes and protocols at the network-layer for routing messages to and from mobile hosts [9, 22, 28, 30], while [11] quantifies the effects of host mobility on transport-layer connections. Application –layer protocols to deliver multicast messages to mobile recipients from exactly-one location are presented in [2, 1]. The design of distributed filesystems has also been influenced by mobility of users, as exemplified by [17, 18, 23, 27]. Various operating systems issues related to mobile hosts can be found in [26, 5, 13, 31].

CONCLUSION

A survey of the literate on checkpointing algorithms show that a large number of papers have been published on checkpointing message-passing distributed computers. A majority of these algorithms are based on the seminal article by Chandy & Lamport (1985) and have Checkpointing algorithms for parallel and distributed computing been obtained by relaxing many of the assumptions made by them; the main aim of improving the earlier extensions of the Chandy & Lamport (1985) algorithms was to minimize the overhead of coordination between processes in a multiprocessor system. More recent published work attempts to minimise the context-saving overhead. A smaller number of algorithms have been proposed to checkpoint shared-memory multiprocessors. These algorithms primarily extend cache coherence protocols to maintain a consistent memory. These algorithms assume the main memory to be safe and do not save context in disk. More recently, algorithms have been proposed for distributed shared-memory systems. In these systems also maintenance of cache coherence of the logical global memory is important for checkpoints. As the physical memory is distributed it is necessary to save main memory contents in the disk. Thus context saving overhead is higher when compared to shared-memory systems.

We also see that most of the algorithms assume no prior knowledge on the structure of programs meant for execution on multiprocessors. In practice, considerable information about such programs is available. We suggest that use of this knowledge by checkpointing algorithms can considerably reduce the coordination and context-saving overheads of such algorithms.

The design of algorithms for distributed systems and their communication costs have been based on the assumptions that the location of hosts in the network do not change and the connectivity amongst the hosts is static in the absence of failures. However, with the emergence of mobile computing, these assumptions are no longer valid. Additionally, mobile hosts have tight constraints on power consumption and bandwidth of the wireless links connecting MHs to their local MSSs is limited. This paper first presents a new system model for the mobile computing

environment and then describes a general principle for structuring distributed algorithms in this model. The two tier principle defined our approach to designing efficient distributed algorithms in this model, viz. localize the communication and data structures necessary for an algorithm within the static portion of the network to the extent possible; the core objective of the algorithm is achieved through a distributed execution amongst the fixed hosts while performing only those operations at the mobile hosts that are necessary for the desired overall functionality. Power consumption at the mobile hosts is thus kept to a minimum, and since updates to the data structures are performed at the fixed hosts, the overall search cost is reduced. A fundamental algorithm in distributed systems viz. a token circulating in a logical ring, served as an illustrative example. Algorithm R-MH established the logical ring amongst MHs. This was shown to be inefficient in terms of search cost, power consumption at the MHs, usage of wireless links and handling disconnection of MHs. The two tier principle was applied to remedy these drawbacks by establishing the logical ring amongst the MSSs. The two tier principle by itself is not sufficient to handle the effects of varying location of MHs. This required location management of migrant MHs and we presented three strategies: (1) search (2) inform (3) search within a local area with location updates after wide-area 11 moves. The relative merits of the three strategies were quantitatively compared. It was then shown that mobility of a host could determine how often it was able to access the token per traversal of the ring. Since algorithm R-MH allowed each MH to access the token atmost once per traversal, we needed an equivalent functionality when the logical ring was shifted to the static segment. A scheme was presented to this end which was equally applicable to all three location management strategies. Finally, we considered an alternative approach to handling the effects of varying location of MHs namely, replicating the queue of pending requests at all MSSs. This eliminated the need for location management strategies for migrant MHs, but increased the number of messages exchanged within the fixed network to globally order pending requests among all MSSs.

DISCUSSIONS

Algorithms, coordination are achieved through markers, headers or both. The main distinction between approaches used in independent schemes is the method used to log messages. They follow pessimistic, optimistic or a combination of pessimistic and Figure 2.2 Classification of checkpointing algorithms. Checkpointing algorithms for parallel and distributed computing 503 optimistic logging methods. Optimistic methods can further be classified based on where e messages are logged: sender-based logging or receiver-based logging.

Desirable features of a checkpointing algorithm:

(1) The time taken by the checkpointing algorithm should be minimum during a failure free run. In other words, increase in total execution time due to checkpointing should not be significant.

(2) Recovery should be fast in the event of a failure. Availability of a consistent global state in stable storage expedites recovery.

(3) Domino effect or rollback propagation should be eliminated completely. Cascading rollbacks of processes due to dependencies among them is termed the domino effect.

(4) Selective rollback should be possible.

(5) Dependency on the cache/memory coherence protocols in shared-memory systems should be minimum.

(6) Resource requirements (memory and processor) for checkpointing should be minimum.

(7) Modifications introduced on the network transmission protocols in case of message passing systems and cache/memory coherence protocols in case of shared-memory systems should be minimum.

FUTURE DIRECTION OF RESEARCH IN CHECKPOINTING

In the first subsection (2.11 a)) we summaries shared and distributed shared-memory checkpointing algorithms and their current status. Section 6.2 summarizes message-passing algorithms and their current status, while 6.3 considers the potential for improvement if one concentrates on minimizing context-saving overhead. Section 6.4 points out the advantage of adopting static approaches in multiprocessor systems which minimize the runtime overhead of a checkpointing algorithm.

Shared and distributed shared-memory systems:

From the discussion about shared-memory checkpointing algorithms, it is clear that cache coherence protocols already maintain consistent memory and those checkpointing algorithms are part of them. Only the number of checkpoints needs to be controlled and this depends on the node interactions, which in turn depend on the program behavior. Distributed shared memory (DSM) systems are considered to be better than the conventional bus-based shared memory systems as they are scalable. DSM systems should allow sufficient time between checkpoints to complete the context-saving process. We saw that DSM systems should save the context in stable storage, if they have to withstand node failures. Periodic checkpointing algorithms would be suitable for DSM systems since they have control over the number of checkpoints unlike algorithms which depend on the interactions among nodes. One such attempt was made recently for DSM systems based on Scalable Coherent Interface (SCI) (IEEE 1992) standards. The algorithm is initiated periodically to maintain consistency in memory and to maintain the

consistent state in stable storage. One may proceed to develop improved checkpointing algorithms for DSM systems by considering the following: (a) Control over number of checkpoints, and (b) scalability of the checkpointing algorithm along with system scalability.

REFERENCES

[1] Kalaiselvi S, Rajaraman V 2000 Task graph based checkpointing in parallel/distributed systems. J. Parallel Distributed Comput. (submitted)

[2] Leong H V, Agrawal D 1994 Using message semantics to reduce rollback in optimistic message logging recovery schemes. Proc. IEEE 14th Conf. on Distributed Computing Syst. Pp 227-234

[3] James Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Trans. On Computer Systems*, 10(1), Feb. 1992. 12

[4] G. Le Lann. Distributed systems, towards a formal approach. *IFIP Congress*, *Toronto*, pages 155–160, 1977.

[5] M. Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3(2), May 1985.

[6] Brian Marsh, Fred Douglis, and Ramon Caceres. Systems issues in mobile computing. Technical Report MITL-TR-50–93, MITL, 1993.

[7] Carl D. Tait and Dan Duchamp. Service interface and replica management algorithm for mobile file system clients. In *Proc. First Intl. Conf. on Parallel and Distributed Information Systems*, 1991. [28] Fumio

[8] eraoka, Yasuhiko Yokote, and Mario Tokoro. A network architecture providing host migration transparency. *Proc. of ACM SIGCOMM'91*, September, 1991.

[9] David Vaskevitch. Database in crisis and transition: A technical agenda for the year 2001. In *Proc. Of the 1994 ACM SIGMOD Intl. Conf. on Management of Data*.