Thanmayee.T, P.K.Vaishali / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 1, Issue 4, pp. 2046-2050

A Review on Awareness of Decentralized QOS Using Wireless Check pointing In MoG

[1]Thanmayee.T, [2]P.K.Vaishali
Department of Computer Science & Information Technology,
Jyothishmathi Institute of Tech & Sciences, JNTU, Hyderabad, AP, INDIA.

Abstract- This paper deals with the overview of decentralized check pointing using QOS in MOG. Check pointing is more crucial in MOG systems than in their conventional wired counter parts due to host mobility. Dynamically, less reliable wireless links frequent disconnections and variation in mobile systems. The algorithms and protocols than we can take an adaptive protocol to reduce the overhead and leasing mechanism is for storage but have a problem that can overcome by synchronous snapshot algorithm is that we have rollback/recovery algorithm but have low communication, and storage overhead .The mutable checkpoints should be minimized. If we talk about a large scale super computing system with coordinated check pointing and rollback recovery. We have work fraction is relatively low correlated failures must be taken in to account. We can also see an NP-Complete problem, so these all problems can overcome by using an algorithm said to be as Reliability Driven (ReD). With ReD an MH simply sends its check pointed data to one selected neighboring MH, and also serves as a stable point of storage for check pointed data received from a single approved neighboring MH. Here ReD works to maximize the probability of check pointed data recovery during job execution. So we are going to implement one of them as NP-Complete problem can be solved by this we can see in future as Overview of Decentralized Check pointing Using QOS in MOG.

Index Terms- *Checkpoint, Mobile Computing System, Recovery, Mobile Grid System.*

I INTRODUCTION

As in earlier days we saw the wired networks and wireless networks, wired networks is going to have a wired network between two hosts and a base station for recovery purpose. If any failure occurs then it is going to take from the base station and but it is a centralized and storing the data at the base station so has disadvantage of wired network. Check pointing saves intermediate data and machine states periodically to reliable storage during the course and job execution. Check pointing in wired grid systems has been investigated earlier with various methodologies proposed [1], where hosts are connected by low latency, high-speed, wired links having low links and host failure rates [1]. Check pointing in wired large-scale grid systems generally assumes that the computation interval and check point over head are much smaller than the Mean Time between Failures (MTBF) [1] with the MOG however this assumption doesn't hold, since relatively high link and host failure rates make its MTBF correspondingly low. Hence maintaining the desired checkpoint interval and check pointing over head versus MTBF tradeoff is certainly more critical in MOG. Earlier wireless check pointing methods stored check pointed information at fixed, stationary hosts on the wired grid (i.e. base stations(BSs) via access points [4][2][3]). Mobile devices will be an integral part of distributed computing as their computational and storage abilities grow. Wireless communication advances leading to high band width and robustness will enables such devices to practically operate as a part of the computing systems has received growing attention with solution approaches treated [2][3][4]. Check pointing wireless MHs to BSs has its own drawbacks, however when not all MHs, are adjacent to BSs or when BSs don't exit. Mobility is a major impediment to moving check pointed data from MH to BS. A complication is that routes between MH and BS change frequently due to varying wireless links, complete and intermittent disconnection and mobility. The frequent need for multihop relays of checkpoint messages to access wired storage can lead to heavy traffic, significant latency, and needless power computation due to collisions and interference. Consequently a mechanism has been developed to reduce the number of checkpoint transmissions to only those actually requested and related to the needs of the distributed application [3].
In wireless communication is nothing but a host that

can communicate with the other host in the form of a base station the need of base station by using this if any failure or any disconnections occur or any crashes in between communication while communicatating between two hosts then that lost data can be taken from the base station but here it is a centralized one. So for getting the decentralized one, we should see many concepts if we are taking the data from the base station then it is a time consuming process because if all the host links may fail or disconnects then the data should be sent to all the hosts at a time and in that if some of them again disconnects then a last the host is going to be aborted. The location of mobile computers in the network may change with time. A mobile node communicates with the other nodes in the system through a fixed node to which it is connected. They communicate with each other through messages. Now in future we are not going to see the fixed nodes.

II RELATED WORK

A. MINIMAL SNAPSHOT COLLECTION ALGORITHM

A good snapshot collection algorithm should be nonintrusive and efficient. A nonintrusive algorithm doesn't force the nodes in the system to freeze their computation during snapshot collection. An efficient algorithm keeps the effort required for collecting consistent snapshot collection algorithms for static distributed systems have been proposed. The algorithm forces a minimal set of nodes to take their local snapshots. Thus the effort required for snapshot collection is reduced and nodes that have been dozing are unlikely to be disturbed. More over the algorithm is nonintrusive if we see the algorithm it is going to have snapshot initiation, reception of snapshot request, computation messages received during snapshot collection, and promotion and reclamation of check pointing.

NONBLOCKING SNAPSHOT COLLECTION ALGORITHM:

Hence this recovery algorithm is a compromise between two diverse recovery strategies. First recovery with high communication and storage over heads and slow recovery with very little communication over heads so this algorithm has some problems that can overcome by the following

B. ADAPTIVE PROTOCOL REDUCES CHECK POINTING OVER HEAD (OR) ADAPTIVE CHECK POINTING WITH LEASING STORAGE MANAGEMENT FOR BASE STATION

Check pointing creation, Hierarchical storage management, Hand off procedures, Failure detection and recovery. Provides flexibility when requested storage is less than the system capacity limit can control usage when the storage exceeds that limit. The leasing mechanism has 4 features:

Negotiation, Cancellation, Renewal, Expiration

NEGOTIATION PROTOCOLS:

Greedy, Greedy with delay, Reservation, Partial reservation

So the disadvantages of the previous one algorithm the over head can be reduced and in this algorithm we are going to face storage management base station problem because of negotiation schemes in that greedy, greedy with delay, reservation the storage space in the middle it is going to store information so if lease expires there may be a problem to get the information so the failure of storage problem can be overcome by using another following mobile computing system that is as follows

C MUTABLE CHECK POINTS ALGORITHM

Mutable check point which is neither tentative check point nor a permanent check point but it can be seen into a tentative check point. Mutable check points can be saved any where ex: the main memory or local disks of MHS. In this way taking a mutable check point avoids the over head of transferring a large amount of data to the stable storage at MHs over the wireless network.

III DECENTRALIZATION OF MOG USING CHECK POINTING

As the MH battery exhibits a much larger MTBF, than the wireless links, its failure impact will be considered later. Let MHk ! MHl define a check pointing relationship between MHk and MHl, functioning as the consumer and provider of check pointing services, respectively. The set of all such directed relationships, on a MoG instance, is called a check pointing arrangement. We define stability, in the context of a check pointing arrangement, as one where no consumer or provider prefers another

provider or consumer, respectively, to its current partner in the relationship, and all consumers and providers have found providers and consumers, respectively. Such an arrangement is called stable. Distinct arrangements result in differing MoG system reliability values and potential instabilities due to positional and wireless signal strength variations among MHs. Less reliable wireless links are more prone to frequent and intermittent disconnections. With MHs, relative locations, velocities, intervening obstacles, multipathing, interference, and other effects all partially determine link strength and reliability. Our Reliability Driven methodology is made QoS-aware, via wireless measurements, of their liabilities of links between MHs in the MoG.

A.ReD'S METHODOLOGY

An executing host is considered to be in "failure," if wireless connections to all of its neighbors are disrupted temporarily or permanently, resulting in its isolation and inability to achieve timely delivery of intermediate or final application results to other hosts. Executing MHs with poor connectivity, have greater likelihood of experiencing failure than do those with greater connectivity and are thus in greater need of check pointing to the best, most reliably connected providers.Inorder to evaluate and compare the strength of progressive check pointing arrangements, we calculate the reliability, Ri, of the whole arrangement on the MoG structure (Mi), as depicted in Fig. 1, where each of thesymbol labels on the model's reliability diagram blocks areas previously defined (see Table 1). Link signal strength decreases inversely with the square of the distance between linked hosts. Reliability mapping for the link is thus based on this assumed signal strength profile with failure rate assumed to be constant for the small time interval, typically a few milliseconds in the mobile environment. We conclude from Fig. 1 that no unrecoverable failure occurs under the following two cases: Case 1: No failure at participating hosts (i.e., no isolation of any host when its results are required). A given application will run to completion without restarting or rewinding. This is true whether or not a check pointing is done. Case 2: Failure of a participating host (isolation). The application will run to completion, with recovery assistance, provided that upon a host failure, its latest saved checkpoint data exists on the provider host, which is no itself isolated and can then be accessed by the MoG during the recovery process. This is possible only if 1) the link for sending checkpoints to the provider host did not fail when being used to checkpoint, and 2) the provider host did not fail (i.e., did not itself become isolated from the rest of the MoG) during the attempt by the MoG to access saved check pointed data upon recovery.

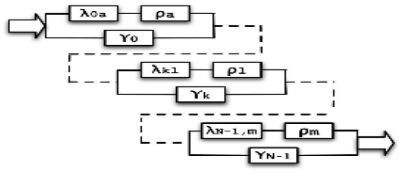

Fig. 1. ReD reliability model.

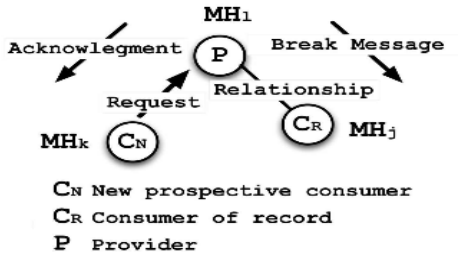

Fig. 2. Protocol function.

B. ReD's Algorithm Description

Upon initiation or refresh, if some consumer, MHk, does not have a designated provider, it begins to look for one. In doing so, it examines and compares a products of each of $\lambda_{ka} \times \rho_a$(where $\rho_a = \psi_a$), $(a = \exists\ neighbor\ ID)$,MHa products in decreasing order (max to min). Next it transmits a checkpoint request, first to the list top hosts the $(\text{having the greatest } \lambda_{ka} \times \rho_a \text{ product})$e.g., MHl. In essence ,a checkpoint request asks the provider ,MHl's permission to send check pointed data to it. If prospective provider, MHlhas no consumer of record, it readily grants permission and sends a positive acknowledgment back to MHk, establishing a MHk !->MHl relationship. On the other hand, if a relationship, say, MHj !->MHl already exists, MHl checks to see if the requesting consumer's pairing reliability gain isgreater than that for its existing paired consumer,i.e.,is

$(\psi_k||\lambda_{kl} \times \rho_l) \,/\, \psi_k alone > (\psi_j||\lambda_{jl} \times \rho_l) \,/\, \psi_j alone$a

true statement ?If so, it breaks its relationship with MHj by sending ita break message, and then grants

permission to MHk by sending it an acknowledgment. If, on the other hand, the statement proves false ,MHk is sent negative acknowledgment, and MHl maintains the relationship ,MHj!->MHl, unless otherwise severed due to mobility, or weak signal .Fig. 2 depicts ReD's protocol messages while the pseudo code used is listed in the Appendix. We call this apparent gain inreliability, achieved by the prospective pairing of some consumer, MHk, with some provider, MHa, the pairing gainofkona, ΔG_{ka}, or $G(MH_k \rightarrow MH_a)$, where $\Delta G_{ka} =$ i.e., $(\psi_k||\lambda_{ka} \times \rho_a) \ / \ \psi_k alone.$ ReD has been enhanced since our initial preliminary work [49] to include pairing gain considerations when comparing prospective relationships as opposed to juststrictly comparing alternative relationship reliabilities directly. In summary, ReD, essentially greedily, attempts to globally maximize Ri, through local, decentralized ,MHk !->MHa pairing decisions. To allow for mobility, tables of connectivity and link reliabilities are updated and aged via the soft state process. A consumer periodically $\lambda_{ka} \times \rho_a$ refreshes, checks product list, and determines if it might do better to find another provider, whereupon it takes action. A provider, upon loss of relationship with a consumer, for any reason, deletes its consumer pointer and admits requests from other consumers. Finally, upon receiving a break message, a paired consumer initiates the process of finding a checkpoint provider all over again. Note that ReD is designed to be IID (meaning torunoneach host independently and identically). Because messages can be lost in transmission, especially over poor wireless links, tables of host connectivity ,link reliabilities, and consumer and provider pointers are maintained at hosts by the soft state registration process, e.g., due to mobility or a weak signal, a host may declare that it has lost its provider (or consumer), attempting to find a new one (or ready to admit a checkpoint request).

ReD's Pseudocode:

Consumer Code:
ReD basic pseudocode listing:
Node: while not refresh, as consumer, $\exists C_k$
Sort neighbor host providers, $\forall P_a$ on list by $\lambda_{ka} \times \rho_a$ (max to min); send checkpointing request to list top, e.g., MH_l;
if(no reply after five tries $||MH_l$ sends NACK $||(MH_k \rightarrow \exists P_a$ && P_a sends Break to this $C_k))$, then
if((list of $\lambda_a \times \rho_a)! = \emptyset$), then
send checkpointing request to next P_a on list; (repeat)
else -- sleep until next refresh period; // empty
else if ACK from selected P_a
then set toPointer $\rightarrow$ to selected P_a;
upon refresh: start consumer process anew

Provider Code:
Node: as a Provider, $\exists P_l$
if(receive checkpointing request from $\exists C_k$) Then
if $(C_i \rightarrow P_l == \emptyset)$ // i.e., if a relationship does not already exist
then send ACK to requesting C_k; set fromPointer toC_k;
else if$(C_j \rightarrow P_l != \emptyset)$, then if($\Delta G_{kl}$ requesting > ΔG_{jl} existing)
send ACK to requesting C_k; // ack requesting consumer
send Break to existing C_j; // break with existing consumer
set fromPointer to requesting C_k;
else send NACK to requesting C_k;
Repeat for each checkpointing request received

IVCONCLUDING REMARKS

Nodal mobility in a large MoG may render a MH participating in one job execution, unreachable from the remaining MHs occasionally ,calling for efficient check pointing in support of long job execution. As earlier proposed check pointing approaches cannot be applied directly to MoGs and are notQoS-aware, we have dealt with QoS-aware check pointing and recovery specifically for MoGs, with this paper focusing solely on check pointing arrangement. It should be demonstrated via simulation and actual testbed studies, that ReD achieves significant reliability gains by quickly and efficiently determining check pointing arrangements for most MHs in a MoG. ReDis shown to outperform its RCA counterpart in terms of the average reliability metric and does so with fewer required messages and superior stability (which is crucial to the checkpoint arrangement, minimization of latency, and wireless bandwidth utilization). Because ReD was tailored for a relatively unreliable wireless mobile environment,

its design achieves its checkpoint arrangement functions in lightweight, distributed manner, while maintaining both low memory and transmission energy foot prints. This work has marked implications for resource scheduling, check point interval control, and application QoS level negotiation. It fills a no Venice component of the ever developing field of MoG middleware, by proposing and demonstrating how QoS-aware functionality can be practically and efficiently added.

REFERENCES

[1] L. Wang et al., "Modeling Coordinated Checkpointing for Large- Scale Supercomputers, Proc. Int'l Conf. Dependable Systems andNetworks, pp. 812-821, July 2005.

[2] K. Ssu et al., "Adaptive Checkpointing with Storage Management for Mobile Environments," IEEE Trans. Reliability, vol. 48, no. 4, pp. 315-324, Dec. 1999.

[3] G. Cao and M. Singhal, "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems," IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 2, pp. 157-172, Feb. 2001.

[4] R. Prakash and M. Singhal, "Low-Cost Checkpointing and FailureRecovery in Mobile Computing Systems," IEEE Trans. Parallel andDistributed Systems, vol. 7, no. 10, pp. 1035-1048, Oct. 1996.