Frequent Data Updating Using Random Checkpointing Arrangement in Decentralized Mobile Grid Computing

Mr. S. P. Santhoshkumar [#1], Mr. D. Prabakar [#2], Dr. S. Karthik[#3]

[#1]*Assistant Professor, Department of CSE, SNS Collefe of Technology, Coimbatore, India.*
E-Mail ID: spsanthoshkumar@gmail.com

[#2]*Assistant Professor, Department of CSE, SNS Collefe of Technology, Coimbatore, India*.
E-Mail ID: prabakaralam@gmail.com

[#3]*professor & Dean, Department of CSE, SNS Collefe of Technology, Coimbatore, India.*
E-Mail ID: profskarthick@gmail.com

***Abstract* – This Paper deals with an autonomic, decentralized, QoSaware, middleware, whose function is to establish checkpointing arrangements among MHs dynamically within the MoG, allowing its constituent MHs to support practical collaborative computation. As wireless links are less reliable and MHs move at will, a given job executed by multiple MHs collaboratively, relies on efficient checkpointing to enable execution recovery upon a MoG component failure by transferring recently saved intermediate data and machine states to a substitute MoG component, so that execution can resume from the last checkpoint, saved prior to the failure. Our checkpointing methodology requires no BS to achieve its function as checkpointing is handled within the MoG by keeping checkpointed data from a given MH at immediate neighboring MHs. Our methodology also facilitates encapsulation of the checkpointing function within the MoG, making it transparent to the wired-Grid or mobile client being served. Thus, in order to limit the use of relatively unreliable wireless links, and further to minimize the consumption of wireless host's memory resources and energy, each MH sends its checkpointed data to one selected neighboring MH, and also serves to take checkpointed data from one approved neighboring MH, realizing a decentralized form of checkpointing. It provides implications for resource scheduling, checkpoint interval control, and application QoS level negotiation. It fills a novel niche component of the ever developing field of MoG middleware, by proposing and demonstrating how QoS-aware functionality can be practically and efficiently added.**

***Keywords*—Checkpointing, computational Grids, Mobile Grid systems, Decentralized Checkpointing, Bayesian Estimation, Grid Computing, Mobile Computing**

I. INTRODUCTION

While most existing Grids refer to clusters of computing and storage resources which are wire-interconnected for offering utility services collaboratively, Mobile Grids (MoGs) are receiving growing attention and expected to become a critical part of a future computational Grid involving mobile hosts to facilitate user access to the Grid and to also offer computing resources. A MoG can involve a number of mobile hosts (MHs), i.e., laptop computers, cell phones, PDAs, or wearable computing gear, having wireless interconnections among one another, or to access points. Indeed, a recent push by HP to equip business notebooks with integrated global broadband wireless connectivity, has made it possible to form a truly mobile Grid (MoG) that consist of MHs providing computing utility services collaboratively, with or without connections to a wired Grid.

Due to mobility and intermittent wireless link loss, all such scenarios call for robust checkpointing and recovery to support execution, minimizing execution rewind, and recovery rollback delay penalties. Depending upon the application's or job's tolerance for such delay, its performance can be poor or it can be rendered totally inoperative and useless. Our Reliability Driven middleware, ReD, allows an MoG scheduler to make informed decisions, selectively submitting job portions to hosts having superior checkpointing arrangements in order to ensure successful completion by 1) providing highly reliable checkpointing, increasing the probability of successful recovery, minimizing rollback delay, and 2) providing performance prediction to the scheduler, enabling the client's specified maximum delay tolerance to be better negotiated and matched with MoG resource capabilities. Suitable for scientific applications, MoGs are particularly useful in remote areas where access to the wired Grid is infeasible, and autonomous, collaborative computing is needed.

Checkpointing is thus crucial for practical and feasible job completion, for without it, the MoG's potential is severely limited. ReD works to maximize the probability of checkpointed data recovery during job execution, increasing the likelihood that a distributed application, executed on the MoG, completes without sustaining an unrecoverable failure. It allows collaborative services to be offered practically and autonomously by the MoG. Simulations and actual testbed implementation show ReD's favorable recovery probabilities with respect to Random Checkpointing Arrangement (RCA) middleware, a QoS-blind comparison protocol producing random arbitrary checkpointing arrangements.

The rest of this paper is organized as follows: Section 2 outlines related checkpointing work in wire-connected Grid systems and wireless systems with BSs. Section 3 discusses our proposed Reliability Driven (ReD) middleware. Section 4 Bayesian Estimation Algorithm Description. Section 5 concludes the article.

II. RELATED WORK

At any time during job execution, a host or link failure may lead to severe performance degradation or even total job abortion, unless execution checkpointing is incorporated. Checkpointing forces hosts involved in job execution to periodically save intermediate states, registers, process control blocks, messages, logs, etc., to stable storage. This stored checkpoint information can then be used to resume job execution at a substitute host chosen to run the recovered application in place of the failed host. Upon host failure or inadvertent link disconnection, job execution at a substitute host can then be resumed from the last good checkpoint. This crucial function avoids having to start job execution all over again from the very beginning in the presence of every failure, thus substantially enhancing the performance realized by grid applications.

A. Checkpointing in Wired Grid Systems

Checkpointing in wired Grid systems has been investigated earlier with various methodologies proposed [7], [8], [9], [10], [11], [12], [13], [14] where hosts are connected by low latency, high-speed, wired links having low link and host failure rates [8], [11]. Diskless checkpointing sends checkpointed data to a cluster neighbor (instead of a local disk), in an attempt to reduce checkpointing time overhead on a LAN [10]. This works if message transmission time is less than disk-write time, a realistic possibility for a wired network. The method fails, however, to consider which neighbor or network path is best used to reach storage. Recent work on portable checkpointing for wired Grids assumes a centralized "middleware" support for applications, checkpointing, and recovery [10]. However, the MoG often requires decentralized ad-hoc support of Checkpointing and recovery, due to its highly unreliable wireless connections and mobile environment.

B. Checkpointing in Wireless Systems with BSs

Mobile devices will be an integral part of distributed computing as their computational and storage abilities grow. Wireless communications advances, leading to high bandwidth and robustness, will enable such devices to practically operate as part of the computational Grid. Hence, checkpointing in wireless computing systems has received growing attention, with solution approaches treated. Specifically, a checkpointing tool for the Palm Operating System has been developed [15] providing a set of APIs to enable checkpointing functionality on top of the Palm OS. It is useful because the Palm OS causes a reset of the handheld computer upon power loss. With this methodology, checkpointed data must be stored on stable safe storage (i.e., a computer server or PC, dubbed a base station, BS, on a wired network). The methodology is supported by recent routing mechanisms that interconnect inadvertently partitioned adhoc MH networks [19]. Checkpointing wireless MHs to BSs has its own drawbacks, however, when not all MHs are adjacent to BSs or when BSs do not exist (like the MoG at hand). Mobility is a major impediment to moving checkpointed data from MH to BS. A complication is that routes between MH and BS change frequently due to varying wireless links, complete and intermittent disconnections, 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 consumption due to collisions and interference.

III. DECENTRALIZED CHECKPOINTING IN THE MOG

This work focuses on the MH checkpointing arrangement mechanism, seeking superior checkpointing arrangements to maximize the probability of distributed application completion without sustaining an unrecoverable failure. It deals with MoG checkpointing among neighboring MHs without any access point or BS.

A. ReD's Heuristic Basis

ReD's algorithm takes into account desired behavioural controlling heuristics in the following ways. First, we require the MoG to be capable of autonomous operation without an access point or BS and further to reduce the use of relatively unreliable wireless links. ReD ensures this by storing checkpointed data only at neighboring MHs, within the MoG, and not requiring BS access or checkpoint transmission over multiple hops. Second, in a MoG, dynamicity ensures that a checkpointing arrangement must be converged rapidly and efficiently, even though it may only be close to optimal. While it is true that poor checkpointing arrangements play a role in reducing the Ri, we seek to maximize, so too do unconverged arrangements (i.e., arrangements where a significant percentage of consumers are still seeking to establish Checkpointing relationships with providers). To ensure convergence within a reasonable time, ReD employs four strategies:

1. ReD is supported by a clustering algorithm, which partitions the global MoG into clusters, allowing ReD to quickly and concurrently find superior arrangements within each cluster instead of having to labor toward a global MoG solution. While many clustering algorithms have been proposed for general ad-hoc networks, a simple clustering algorithm is devised and adopted both in our simulator and in our working testbed as a functional support layer for ReD,
2. ReD makes decisions about whether to request, accept, or break checkpointing relationships, locally (at the MH level) and in a fully distributed manner, instead of attempting a high-level centralized or global consensus,
3. ReD keeps checkpoint transmissions local, i.e., neighbor to neighbor, not requiring multiple hops and significant additional transmission overhead to achieve checkpointing relationships, and
4. ReD allows a given consumer or provider to breakits existing checkpointing relationship (when a provider breaks a checkpointing relationship, a *break message* is transmitted to the consumer) only when the arrangement reliability improvement is significant, thus promoting *stability*.

B. 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 checkpointing to the best, most reliably connected providers. In order to evaluate and compare the strength of progressive checkpointing arrangements, we calculate the reliability, Ri, of the whole arrangement on the MoG structure (Mi). 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, □i, assumed to be constant for the small time interval, □ti, typically a few milliseconds in the mobile environment. ReD's heuristic method ensures that checkpointing arrangement decisions aremade locally and individually at the host level, promoting rapid convergence, while a threshold mechanism is included in order to provide stability control.

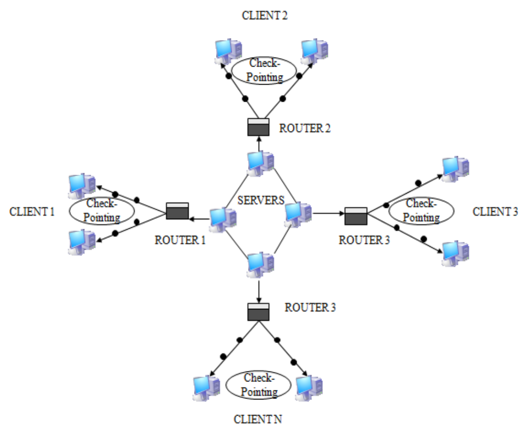

Fig: 1. Architecture Diagram

IV. Bayesian Estimation Algorithm Description

Given parametric model and data, estimate model paramters (→ same setting as MLE). Bayesian estimation ≠ Bayes' rule. Pick the class which is most probable given the data. For Bayes' rule, probability distribution is assumed to be given. Even if we use parametric inference to obtain it, we may e. g. use MLE rather than Bayesian estimation. Modeled by parametric likelihood. Considers only maximizer of the likelihood. Consider all possible values of θ.

1) To rank them against one another, we need their distribution.

2) To take the data into account, we need a conditional distribution of θ|x.

→ how can we obtain p(θ|x) from likelihood?. Posterior: Start with p(θ, x) and plug in definition of conditional distribution.

Result:

p(θ|x) = p(x|θ)p(θ)/p(x)

Consequence: To obtain data conditional distribution of θ ("posterior") from likelihood p(x|θ), we have to provide p(θ) ("prior"). In other words: For MLE, we need one model assumption (likelihood). To work with a full distribution of the parameter, we need a second model assumption (prior). Since distribution of θ|x is provided by Bayes' formula, estimation based on posterior distribution is called Bayesian estimation.

Terms of Bayes' formula:

p(θ|x) = p(x|θ)p(θ) / p(x)

posterior = likelihood × prior / evidence

Evidence p(x): Since data is assumed to be given (→ x fixed), p(x) is a normalization constant. Always think of the likelihood as a function of θ:

1) Likelihood p(x|θ) is a density w.r.t. x, but x is fixed to one particular value.

2) p(x|θ) is no density w.r.t. θ (i. e. not normalized). Some people emphasize this by writing e. g. l(θ) instead of p(x|θ).

Generative model of the data. Prior - User input! This is the key point of criticism often voiced concerning Bayesian methods

Approach 1: Maximize it. This is called maximum a posteriori estimation (MAP) and is the direct counterpart to MLE (Considers only maximizer of the likelihood). We can apply the logarithm trick and obtain:

$$\hat{\theta}_{\mathsf{MAP}} := \arg\max_{\theta}\left\{\sum_{i=1}^{n}\log(p(x_i|\theta) + \log(p(\theta))\right\}$$

Evidence p(x) not required. Not really Bayesian estimation: Estimate once again restricted to single value. We have penalized MLE by prior knowledge.

Approach 2: Compute expectations.

1. If interested in parameter estimate: Compute expectation w.r.t. posterior,

$$\hat{\theta} := \mathsf{E}_{\theta|x}[\theta] = \int \theta p(\theta|x_1, ..., x_n)d\theta$$

2. If interested in some statistic f(θ): Compute Eθ|x [f(θ)]. This is a "full" Bayesian approach.

Problem 1: Normalization. How do we compute the normalization constant p(x) (the evidence)?

1) Evidence is value of joined distribution of sample x1, ..., xn at the single point (x1, ..., xn). We cannot hope to estimate p(x) from a single point!

2) Evidence is also normalization constant of posterior, so

p(x) = ∫p(x|□)p(□)d□

p(x|□), p(□) known□□p(x) computable, but have to integrate.
Problem 2: Expectations require integration against p(x|□)p(□). Analytic integration- Perfect if it works, but even for many simple standard models (e. g. Gaussian likelihood + Cauchy prior), integral has no analytic solution. Quadrature-Next step if analysis does not work. Problem: Curse of dimensionality. (Example: Estimate parameters of 3D Gaussian□□quadrature on 9D grid). Monte Carlo integration-E. g. MCMC sampling. Very powerful, but requires some expertise.

C. Conjugate pairs

Let the likelihood be in a family P1 of distributions (e. g. P1 = Gaussian densities), and the prior in P2. Likelihood and prior form a *conjugate pair*, if the resulting posterior is again in P2. This **Meaning is** Many standard models have a known conjugate prior, which is also a standard model. Most standard models can be handled analytically. Therefore: *If* our model has a conjugate prior, and *if* it is a meaningful prior for our problem, we will be able to deal with the posterior. **Some examples** (likelihood/prior): Gaussian (unknown μ, fixed)/Gaussian, Gaussian (fixed μ, unknown)/gamma, Gaussian (both par. unknown)/Wishart, multinomial/Dirichlet. The data "updates" the parameter values.

V. Conclusion

As earlier proposed checkpointing approaches cannot be applied directly to MoGs and are not QoS-aware, we have dealt with QoS-aware checkpointing and recovery specifically for MoGs, with this paper focusing solely on Checkpointing arrangement. It has been demonstrated via simulation and actual testbed studies, that ReD achieves significant reliability gains by quickly and efficiently determining checkpointing arrangements for most MHs in a MoG. ReD is 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 a lightweight, distributed manner, while maintaining both low memory and transmission energy footprints.This work has marked implications for resource scheduling, checkpoint interval control, and application QoS level negotiation. It fills a novel niche component of the ever developing field of MoG middleware, by proposing and demonstrating how QoS-aware functionality can be practically and efficiently added and how Bayesian estimation algorithm used for updating the data's.

VI. References

[1] Paul J. Darby and Nian-Feng Tzeng, "Decentralized QoS-Aware Checkpointing Arrangement in Mobile Grid Computing" IEEE transactions on Mobile Computing, vol. 9, no. 8, August 2010.

[2] SUN Microsystems, "Sun Grid Compute Utility," http://www.sun.com/service/sungrid, 2006.

[3] Hewlett-Packard Development Company, L.P., "Grid-Computing—Extending the Boundaries of Distributed IT," http://h71028.www7.hp.com/ERC/downloads/4AA03675ENW.pdf?jumpid=reg_R1002_USEN, Jan. 2007.

[4] "IBM Grid Computing," http://www-1.ibm.com/grid/about_grid/what_is.shtml, Jan. 2007.

[5] S. Wesner et al., "Mobile Collaborative Business Grids—A Short Overview of the Akogrimo Project," white paper, Akogrimo Consortium, 2006.

[6] Computerworld, "HP Promises Global Wireless for Notebook PCs," http://www.computerworld.com/mobiletopics/mobile/story/0,10801,110218,00.html?source=NLT_AM&nid=110218, Apr. 2006.

[7] J. Long, W. Fuchs, and J. Abraham, "Compiler-Assisted Static Checkpoint Insertion," Proc. Symp. Fault-Tolerant Computing, pp. 58-65, July 1992.

[8] K. Ssu, B. Yao, and W. Fuchs, "An Adaptive Checkpointing Protocol to Bound Recovery Time with Message Logging," Proc. 18th Symp. Reliable Distributed Systems, pp. 244-252, Oct. 1999.

[9] N. Neves and W. Fuchs, "Coordinated Checkpointing without Direct Coordination," Proc. Int'l Computer Performance and Dependability Symp., pp. 23-31, Sept. 1998.

[10] W. Gao, M. Chen, and T. Nanya, "A Faster Checkpointing and Recovery Algorithm with a Hierarchical Storage Approach," Proc. Eighth Int'l Conf. High-Performance Computing in Asia-Pacific Region, pp. 398-402, Nov. 2005.

[11] R. de Camargo, F. Kon, and A. Goldman, "Portable Checkpointing and Communications for BSP Applications on Dynamic Heterogenous Grid Environments," Proc. Int'l Symp. Computer Architecture and High Performance Computing, pp. 226-234, Oct. 2005.

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

[13] A. Agbaria and W. Sanders, "Application-Driven Coordination-Free Distributed Checkpointing," Proc. 25th IEEE Conf. Distributed Computing Systems, pp. 177-186, June 2005.

[14] A. Oliner, R. Sahoo, J. Moreira, and M. Gupta, "Performance Implications of Periodic Checkpointing on Large-Scale Cluster Systems," Proc. 19th IEEE Int'l Conf. Parallel and Distributed Processing Symp., Apr. 2005.

[15] C. Lin, S. Kuo, and Y. Huang, "A Checkpointing Tool for Palm Operating System," Proc. Int'l Conf. Dependable Systems and Networks, pp. 71-76, July 2001.

[16] D. Pradhan, P. Krishna, and N. Vaidya, "Recoverable Mobile Environment: Design and Trade-Off Analysis," Proc. Symp. Fault- Tolerant Computing, pp. 16-25, June 1996.

[17] H. Higaki and M. Takizawa, "Checkpoint-Recovery Protocol for Reliable Mobile Systems," Proc. 17th IEEE Symp. Reliable Distributed Systems, pp. 93-99, Oct. 1998.

[18] C. Ou, K. Ssu, and H. Jiau, "Connecting Network Partitions with Location-Assisted Forwarding Nodes in Mobile Ad Hoc Environments," Proc. 10th IEEE Pacific Rim Int'l Symp. Dependable Computing, pp. 239-247, Mar. 2004.

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

[20] 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.