RESULT ERROR DETECTION ON HETEROGENEOUS AND VOLATILE RESOURCES VIA INTERMEDIATE CHECKPOINTING *

Derrick Kondo
Laboratoire d'Informatique de Grenoble/INRIA Rhône-Alpes, France
dkondo@imag.fr

Filipe Araujo
CISUC, Department of Informatics Engineering, University of Coimbra, Portugal
filipius@dei.uc.pt

Patricio Domingues
School of Technology and Management, Polytechnic Institute of Leiria, Portugal
patricio@estg.ipleiria.pt

Luis Moura Silva
CISUC, Department of Informatics Engineering, University of Coimbra, Portugal
luis@dei.uc.pt

Abstract

Desktop grids use the free resources in Intranet and Internet environments for large-scale computation and storage. While desktop grids offer tremendous computational power and a high return on investment, one critical issue is the validation of results returned by participating hosts that are volatile, anonymous, and potentially malicious. We conduct a benefit analysis of a mechanism for result validation that we proposed recently for the detection of errors in long-running applications. This mechanism is based on using the digest of intermediate checkpoints, and we show in theory and simulation that the relative benefit of this method compared to the state-of-the-art is as high as 45%.

2

1.   Introduction

For over a decade, one of the largest distributed computing platforms in the world have been desktop grids, which use the idle computing power and free storage of a large set of networked (and often shared) hosts to support large-scale applications [14, 13–1]. Desktop grids are an extremely attractive platform because they offer huge computational power at relatively low cost. Currently, many desktop grid projects, such as SETI@home [14], FOLDING@home [13], and EINSTEIN@home [1], use TeraFLOPS of computing power of hundreds of thousands of desktop PC's to execute large, high-throughput applications from a variety of scientific domains, including computational biology, astronomy, and physics.

Despite the huge return-on-investment that desktop grids offer, one critical issue is the correctness of results returned from volatile, anonymous, and potentially malicious hosts. A number of different factors can influence the correctness of the results returned from the desktop grid worker to the server. These factors can be due to computational errors (for example, overclocking of the CPU or incorrectly modified application binaries [15]) or input/output errors (for example, a machine crash during an out-of-order flush of in-memory blocks [11]).

Given the risk of erroneous results, effective error detection mechanisms are essential. In this paper, we conduct a benefit analysis of a mechanism that we proposed recently for the detection of errors in long-running applications. This mechanisms uses the digest of intermediate checkpoints to accelerate the detection of result errors, especially for long running applications. A number of projects such as `climateprediction.net`, `climatechange`, and `seasonalattribution` have workunits whose execution span months [6], and we believe early error detection for these projects would be useful. We present theoretical upper and lower bounds on the benefits of our mechanism *for heterogeneous and volatile resources*, using error rates derived from a real desktop grid system. Finally, we present simulation results that loosen the assumptions of our theoretical analysis, but nevertheless confirm our theoretical results.

The paper is structured as follows. In Section 2, we describe how our work in this paper relates to previous research. Then, in Section 3, we detail our mechanism for error detection, give theoretical upper and lower bounds on its benefits, and confirm our analysis with simulation results. Finally, in Section 4, we summarize our conclusions and describe future research directions.

2.   Related Work

In [4], we presented the theoretical analysis and simulation results of the same error detection mechanism presented here, but there were two main limitations. First, the previous analysis was conducted using hypothetical error

rates instead of error rates obtained empirically from a real project. In fact, our previous work assumed error rates that were orders of magnitude higher than the rates we determined in this study. *Nevertheless, we show here that substantial benefits can still be achieved using this novel technique with real but relatively lower error rates.* Second, the theoretical analysis previously conducted made the assumption that checkpoints occur simultaneously across hosts at constant intervals. For reasons that we detail in the next section, this is an unrealistic assumption in volatile and heterogeneous desktop grids. *We loosen the assumption to consider variable checkpointing intervals, and give new theoretical upper and lower bounds on the benefits of this technique using a mathematical approach based on order statistics.*

3. Comparing Intermediate Checkpoints for Long-Running Workunits

In this section, we present novel benefit analysis of a mechanism for error detection that we proposed recently in [4]. This mechanism is based on checkpointing and replication, and is well-suited for long-running workunits. The technique involves comparing intermediate checkpoint digests (provided for example by the MD5 [12]family of algorithms) of redundant instances of the same task. (Note that often computations occupy a large space in memory often near the 100MB range [6]and/or sending a small, intermediate result for comparison may not be possible nor efficient.) If differences are found, the conclusion is that at least one task's execution is wrong. In contrast to the simple redundancy mechanism, where diverging computations can only be detected after a majority of tasks have completed, intermediate checkpoint comparison allows for earlier and more precise detection of errors, since execution divergence can be spotted at the next checkpoint following any error. This allows one to take proactive and corrective measures without having to wait for the completion of the tasks, and it allows for faster task completion, since faulty tasks can immediately be rescheduled.

We assume the following. First, if the digests differ from the correct digest, then the divergent digest differs from all other digests (including other divergent ones). Second, the errors occur independently of one another. Finally, each task is checkpointed locally and periodically (as is done in several existing desktop grid systems [3, 9]). (Note that later we relax these assumptions in our simulations.) With respect to CPU time, the application could conduct local checkpointing periodically (for example, every 10 minutes). However, with respect to wall-clock time, the time between checkpoints is random because of non-deterministic events that could delay checkpointing such as a host being powered off, or the worker being suspended or killed because of user activity [8].

Parameter	Definition
W	Benefit in time of intermediate checkpointing relative to the state-of-the-art method
$T_{k,j}$	Time from start of workunit to the time of checkpointing segment j on worker k
R	Number of workers on which a checkpointed task is replicated
c	Number of segments or equivalently checkpoints per task
$S_{k,g}$	Time from start of segment g to the time of checkpointing segment g on worker k
p, v	p is the probability of getting an error within a segment on any host. $v = 1-p$
X	Random variable distributed geometrically with parameters p and v representing the number of task segments before an error occurs

Table 1.   Parameter Definitions.

Thus, we model the time between checkpoints as a random variable. In particular, each checkpoint delineates the end of a task segment to create a total of c segments. Let R be the number of workers on which a checkpointed task is replicated (see Table 1). Let $S_{k,g}$ be a random variable that represents the time to checkpoint the current segment g, beginning from the last checkpoint (or start of the task, in the case of the first checkpoint), on worker k where $1 \leq g \leq c$, and $1 \leq k \leq R$.

Let $T_{k,j}$ be a random variable that represents the amount of time elapsed since the start of the task up to the checkpoint time of segment j, on worker k. Specifically, $T_{k,j} = \sum_{g=1}^{j} S_{k,g}$ (see Figure 1 for an example).

We assume that $S_{k,g}$ is distributed exponentially with parameter λ across all workers. While a number of previous studies have characterized the distribution of availability intervals on *enterprise* desktop resources (for example, [8]), it is unclear how these periods of availability relate to the time of checkpointing a segment on *Internet* environments. Thus, for future work, we will verify our assumption using resource traces, for example, those currently being collected on Internet desktop environments [10].

Given that $S_{k,g}$ is distributed exponentially, $T_{k,j}$ has a gamma distribution with parameters $\alpha = j$ and $\beta = 1/\lambda$.
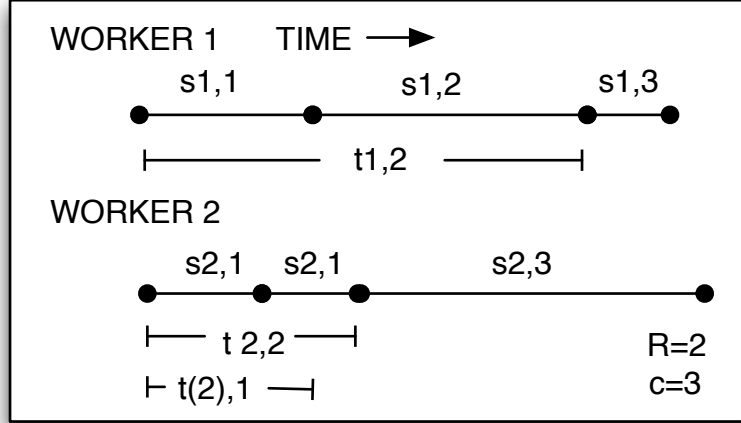
Figure 1.   Example of Intermediate Checkpointing

The time to validate the ith segment is given by $T_{(R),i}$, which is the Rth order statistic of the set $T_{1,i}, ..., T_{R,i}$. That is, $T_{(R),i}$ represents the maximum time to complete segment i among all R workers.

The expected gain $E[W]$ for using intermediate checkpoints compared to state-of-the-art methods where the comparison is done at the end of the workunit is then given by:

$$E[W] = E[T_{(R),c} - T_{(R),i}] \tag{1}$$

where $1 \leq i \leq c$.

Let X be the number of trials, i.e., the segment in which an error occurs on any of the hosts, and let X have a geometric distribution with parameters p and v, where p is the probability of getting an error within a segment in any of the hosts, and $v = 1 - p$.

From [5], a lower bound on the expectation of the maximum of a set of random variables is the maximum of the expected value of each random variable in the set. Moreover, Hartley and David [7]report that an upper bound for the expectation of the maximum is $\mu + \sigma \times (n - 1)/\sqrt{2n - 1}$, given a set of n independent random variables with identical means and variances (μ, σ^2).

In Figure 2, we show the upper and lower bounds on the benefit $E[W]$ relative to the upper and lower bounds of the expected maximum time $E[T_{(R),c}]$ for checkpointing at the end of the task. In particular, in Figure 2(a), the number of checkpoints c is fixed to 1000, and p varies between [0.0005, 0.0015]. In
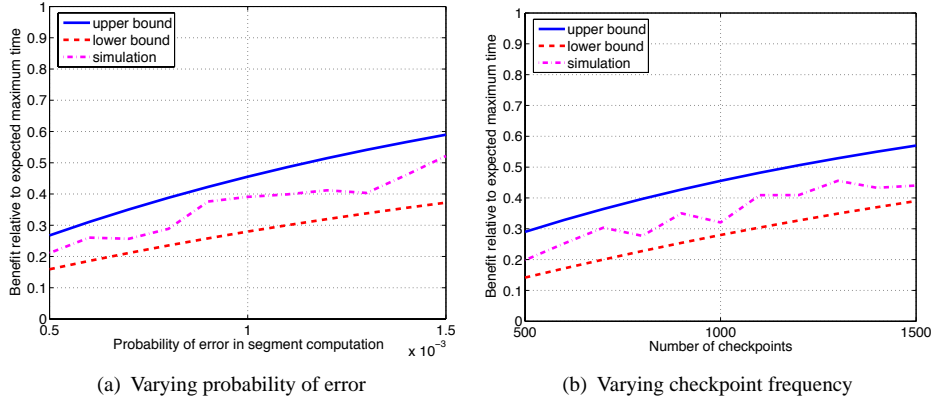
(a) Varying probability of error   (b) Varying checkpoint frequency

Figure 2.   Benefits of intermediate checkpointing

Figure 2(b), the probability of error within each segment p is fixed at 0.001, and c varies between [500, 1000]. (The range of error rates are based upon those observed in a real desktop grid system [16]. In that study, the authors checked syntactically and semantically the results returned from about 600 hosts in an Internet-wide desktop grids to determine error rates of hosts.)

We observe potentially significant gains even for small error rates. For example, in Figure 2(a), we find that if the probability of error p is 0.001 and the number of checkpoints per task c is 1000, then the potential benefit of intermediate checkpointing is between $\sim 30 - 45\%$. While 1000 checkpoints may seem abnormally large, if we assume a task checkpoints every 10 minutes a thousand times, this equates to a 7-day workunit. (This is a reasonable checkpoint frequency and workunit length as the frequency in real projects EINSTEIN@home, PREDICTOR@Home, and SIMAP is on the order of minutes [2]and execution is on the order of days or months [6].) In Figure 2(b), we find that if the number of checkpoints is 1050 (and the probability of error is 0.001), then the potential benefit of intermediate checkpointing is between $\sim 30 - 45\%$.

We then confirmed and extended the theoretical results through simulation. We assign a number of tasks to a set of workers. Whenever a worker computes a checkpoint, it randomly determines whether that computation is wrong or correct. Once a checkpoint is wrong, all the remaining checkpoints from that worker are also considered as wrong. In our experiments, the time that a worker needed to compute a checkpoint was given by an exponential distribution. We chose an arbitrary average checkpoint time (as it does not impact the *relative* benefit of our technique). We varied the number of checkpoints of each task and the probability of error in each checkpoint. (We used a constant value for

the probability of error. We also tried random variables (truncated Gaussian, exponential and others), with little if any impact on the outcome of the trials.)

In Figures 2(a) and 2(b), we show the results of our experiments for the same range of parameters as used for the theoretical analysis. The curve of the observed benefit is the average of 300 trials.

Our results show that the there is a considerable benefit in comparing intermediate checkpoints, especially for long-running workunits. Even for very small probabilities of error, which correspond to real values observed in real systems, the time savings can amount to 20%-45% of the time corresponding to state-of-the-art solutions. (One potential limitation of this method is scalability of receiving the high-frequency digest messages if digests are sent centrally to a "supervisor" for comparison. We are currently working on secure load-balancing techniques via distributed hash tables (DHT) to remove this limitation, and we will report on this in future work.)

4.   Conclusion

We showed the benefits of a recently proposed method for accelerating error detection on large-scale and volatile resources. In particular, we gave novel theoretical analysis for our proposed method based on the digest of intermediate checkpoints, *where each task segment can take a variable amount of time* due to host volatility or heterogeneity, for example. For error rates often found in real systems, we found that the time savings bounded by theoretical analysis can often range from 20% to 45%. We then verified our theoretical bounds on the potential benefit through simulation experiments, while loosening assumptions of the analysis. We find that our simulation experiments validate our theoretical analysis, even for various distributions of the probability of error. For future work, we will develop scalable ways to collect and compare checkpoint digests, for example using mechanisms based on DHT's.

References

[1] EINSTEN@home. http://einstein.phys.uwm.edu.

[2] B. Allen, C. An, and T. Rattel. Personal communications, April 2006.

[3] D. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, 2004.

[4] F. Araujo, P. Domingues, D. Kondo, and L. M. Silva. Validating Desktop Grid Results By Comparing Intermediate Checkpoints. *Submitted to 2nd Coregrid Integration Workshop*, 2006.

[5] T. Aven. Upper (lower) bounds on the mean of the maximum (minimum) of a number of random variables. *Journal of Applied Probability*, 22:723–728, 1985.

[6] Catalog of boinc projects. http://boinc-wiki.ath.cx/index.php?title=Catalog_of_BOINC_Powered_Projects.

[7] H. Hartely and H. David. Universal bounds for mean range and extreme observations. *The Annals of Mathematical Statistics*, 25:85–89, 1954.

[8] D. Kondo, M. Taufer, C. Brooks, H. Casanova, and A. Chien. Characterizing and Evaluating Desktop Grids: An Empirical Study. In *Proceedings of the IPDPS'04*, April 2004.

[9] M. Litzkow, M. Livny, and M. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems (ICDCS)*, 1988.

[10] P. Malecot, D. Kondo, and G. Fedak. Xtremlab: A system for characterizing internet desktop grids (abstract). In *in Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*, 2006.

[11] A. Oltean. How to do atomic writes in a file. http://blogs.msdn.com/adioltean/archive/2005/12/28/507866.aspx, December 2005.

[12] R. Rivest. RFC-1321 The MD5 Message-Digest Algorithm. *Network Working Group, IETF*, April 1992.

[13] M. Shirts and V. Pande. Screen Savers of the World, Unite! *Science*, 290:1903–1904, 2000.

[14] W. T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, G. Gedye, and D. Anderson. A new major SETI project based on Project Serendip data and 100,000 personal computers. In *Proc. of the Fifth Intl. Conf. on Bioastronomy*, 1997.

[15] M. Taufer, D. Anderson, P. Cicotti, and C. L. B. III. Homogeneous redundancy: a technique to ensure integrity of molecular simulation results using public computing. In *Proceedings of the International Heterogeneity in Computing Workshop*, 2005.

[16] D. Kondo, F. Araujo, P. Domingues, P. Malecot, G. Fedak, F. Cappello Characterization Error Rates on Internet Desktop Grids. To appear in *Proceedings of Euro-Par*, 2007.