Impact of Sub-optimal Checkpoint Intervals on Application Efficiency in Computational Clusters

William M. Jones[*]
Department of Computer Science
Coastal Carolina University
Conway, SC, USA
wjones@coastal.edu

John T. Daly[†]
Center for Exceptional Computing
Department of Defense, ACS
Fort Meade, MD, USA
john.t.daly@ugov.gov

Nathan DeBardeleben[‡]
High Performance Computing
Los Alamos National Laboratory
Los Alamos, NM, USA
ndebard@lanl.gov

ABSTRACT

As computational clusters rapidly grow in both size and complexity, system reliability and, in particular, application resilience have become increasingly important factors to consider in maintaining efficiency and providing improved computational performance over predecessor systems. One commonly used mechanism for providing application fault tolerance in parallel systems is the use of checkpointing.

By making use of a multi-cluster simulator, we study the impact of sub-optimal checkpoint intervals on overall application efficiency. By using a model of a 1926 node cluster and workload statistics from Los Alamos National Laboratory to parameterize the simulator, we find that dramatically overestimating the AMTTI has a fairly minor impact on application efficiency while potentially having a much more severe impact on user-centric performance metrics such a queueing delay. We compare and contrast these results with the trends predicted by an analytical model.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance Attributes

General Terms

Performance, Reliability

Keywords

checkpointing, resilience, simulation, prediction

[*]Corresponding author.

[†]Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the Los Alamos National Laboratory, the US Department of Defense, or Coastal Carolina University.

[‡]This paper has been approved for public release under LA-UR-09-05697.

1. INTRODUCTION

As the size of a cluster increases, the system mean time between failures (SMTBF) tends to decrease inversely proportional to the number of components and in some cases, even more rapidly when failures exist that simultaneously impact multiple jobs. Applications can experience an interruption in service due to such failures, and as system sizes grow, application failures will become a much more critical issue in addressing overall system performance. In addition to hardware failures, novel system software stacks coupled with legacy parallel scientific applications deployed on modern cluster platforms push the envelope of reliability.

One commonly used technique to recover from application failure is the use of checkpointing. During checkpointing, an application writes its entire state to non-volatile secondary storage so that in the event that it is interrupted, it can resume its work from the last checkpoint rather than from the beginning. While writing and reading the checkpoint data is a type of overhead that consumes valuable system resources, the savings in rework times due to failure can often outweigh the cost of performing checkpointing in the first place.

One aspect of employing checkpointing is properly assigning a checkpoint interval, i.e., the time from the beginning of one checkpoint to the beginning of the next. Given a set of job and failure parameters, it is possible to assign this interval in such a way that it maximizes application efficiency, i.e., the ratio of time the job spends making forward progress compared to the entire wall-clock time that includes checkpoint, restart, and rework overhead. A principal question here is the extent to which sub-optimal interval assignment impacts this efficiency. This is of particular importance due to the fact that the underlying parameters necessary to optimize the interval width may be difficult to determine or potentially contain a large amount of estimation error depending on what type of application and system monitoring is used and the period over which the requisite data is gathered. In this paper we compare and contrast both an analytical and simulation-based study using a real cluster model and workload characteristics to demonstrate that dramatically overestimating the checkpoint interval has little impact on application efficiency; however, this impact may result in other undesirable performance characteristics such as excessive queueing delays.

2. ANALYTICAL MOTIVATION

For applications that use checkpoint restart as their primary means of fault tolerance, only a fraction of the application's execution time, or *run time* (t_r), is spent performing actual computational work that represents forward progress towards a solution. That time is referred to as *solve time* (t_s). The difference between the solve time and the execution time consists of time spent writing out checkpoint data, restarting after an interrupt, and performing rework to move the calculation from the latest checkpoint back to the point where the interrupt occurred. Daly [1, 2] demonstrated that the relationship between the application's solve time and execution time when checkpointing at regular intervals on a system where interrupts arrive according to a Poisson process can be expressed in terms of the *checkpoint interval* (t_c), *dump time* (δ), *application MTTI* (M), and *restart overhead* (R) as

$$\frac{t_s}{t_r} = e^{-\frac{R}{M}} \left(\frac{\frac{t_c}{M} - \frac{\delta}{M}}{e^{\frac{t_c}{M}} - 1} \right) \quad \text{for} \quad \delta \ll t_s \ . \tag{1}$$

The dump time is the wall-clock time required to create a checkpoint, M is the application mean time between unscheduled system events that result in an application interrupt (a.k.a AMTTI), and the restart overhead is wall-clock time elapsed from the point at which the application ceased making progress until the point when it resumes computational work. Daly [1] demonstrated that the ratio of solve time to execution time is maximized when the checkpoint interval is chosen according to this first order approximation

$$t_c \approx \sqrt{2\delta M} \quad \text{for} \quad \delta < \frac{1}{2}M \ , \tag{2}$$

which agrees with Young's [8] original work in this area.

The issue in this case is that variability in accurately measuring the requisite parameters will lead to calculations of checkpoint intervals that are sub-optimal. The resulting question is the extent to which this variability leads to significant degradation in application efficiency. In order to model this impact, we assign an error factor, *err* to the calculation of the optimal checkpoint interval, as follows in Equation 2

$$\overline{t_c} = err\sqrt{2\delta M} \quad \text{for} \quad .1 \leq err \leq 4.0 \ . \tag{3}$$

Note that the range provided here for *err* does not imply that it is undefined elsewhere, but rather the area of interest over which we chose to explore the parameter space.

In order to parameterize the model, we need to specify values of dump time (δ), restart time (R), and mean time to failure for the application (M). As an example to simply illustrate trends, we set δ and R to 10 minutes and vary M across a reasonable range of values that are similar in magnitude to the SMTBF's that we will later use in the simulation section of this paper. This analytical result applies to the impact to a single job whereas the later simulation applies to the aggregated impact across the entire workload presented. As such, it is not appropriate to directly compare the absolute results here with those that are obtained later. Note that the results presented here are intended only to illustrate the general trends of insensitivity to *err* in application efficiency.

By substituting Equation 3 into Equation 1, we plot the application efficiency as a function of the specified values of *err* and application mean time between failures, M, as seen in Figure 1.

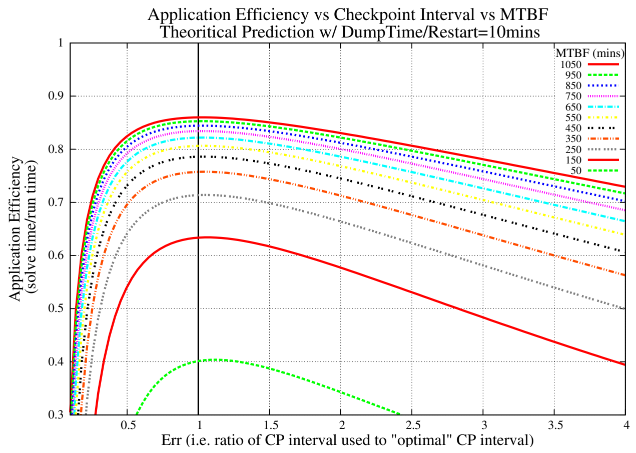

Figure 1: Theoretical application efficiency as a function of failure times and error in checkpointing interval *for a single job* with the given MTBF. *Note that the impact of fairly significant error in checkpointing interval does not dramatically impact efficiency, even at fairly severe failure times.*

The derivation of the above equations assume we are looking at a single job with a single failure rate. This will be compared and contrasted to the simulation-based results that utilize a real workload where the widths of the jobs play a major role in how often a job will fail due to the multiplicity of nodes across which they are mapped.

3. SIMULATION

In this section, we describe the simulation-based study conducted to demonstrate the impact of sub-optimal checkpoint interval assignment on overall application efficiency.

Our cluster simulator [7, 4] is parametrized to model the Pink cluster at Los Alamos National Laboratory (LANL). Pink was a 1024 node Myrinet connected cluster at LANL of which 963 nodes were available for user applications. Each node consisted of two Intel 2.4-GHz Xeon processors. Pink was available for use by external LANL collaborators and was decommissioned in 2008. During its lifetime, Pink was used to develop and test the Beowulf Distributed Process Space (BProc) [3] software for system administration.

For the initial experiments, our simulator's workload generator has been tuned to both the sizes and run times of jobs seen on Pink during an 11-month period in 2007 [6]. We assume that jobs arrive according to a Poisson process, and have adjusted the inter-arrival times to load the system. It should be noted that the ratio of arrival to departure rate is less than one so that we can guarantee that the queueing system is operating in a stable mode after simulation warm-up and prior to cool down. Any measures of application efficiency as a function of estimate error would not be affected by this ratio; however, the queueing delay would be affected. Both are addressed in the results section.

4. RESULTS AND OBSERVATIONS

In this section, we present the simulation-based results

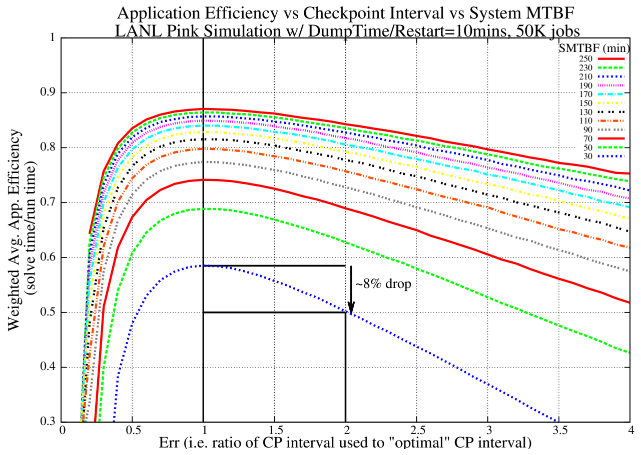

Figure 2: Simulation-based results that show the impact of error in optimal checkpoint interval assignment as a function system failure load. ***Note than a factor of 2 error in interval calculation results in less than a 10% drop in average application efficiency.***

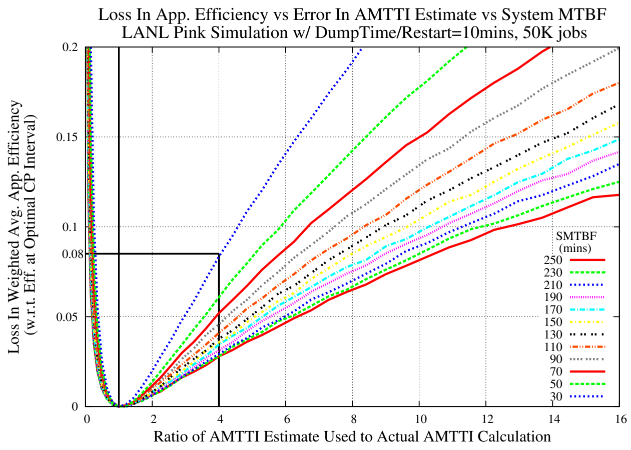

Figure 3: Simulation-based results that show the total loss in average application efficiency as a function of error in the estimation of M and of failure load of system. ***Note that even when there is an order of magnitude error in M, for $SMTBF \geq 170$, the results are less than 10% losses in application efficiencies.***

using the metrics defined above in BeoSim, a custom event-driven simulator that has been previously used in the study of network aware multi-cluster parallel job scheduling, [5, 6, 7].

By making use of LANL's Pink cluster workload characterization of 50,000 jobs, we were able to conduct a parameter study of average application efficiency as a function of various failure loads and error in optimal checkpoint interval assignment. The results of this study are summarized in Figures 2 and 3.

The results presented in Figure 2 corroborate the general trends found in the analytical model presented in Figure 1. In fact, at system mean time between failures (SMTBFs) of 30 minutes, an error factor of two results in a drop of only 8% in average application efficiency (as highlighted in Figures 2 and 3). This is especially significant considering that a 30-minute SMTBF is extremely severe for a cluster of only 1926 processors. In a cluster this size, where node failures are independent, the individual nodes would need to have a MTBF of slightly over one month to result in such a short SMTBF. This suggests that error in optimal interval assignment would likely result in a much smaller impact on application efficiencies than what is suggested by the more severe SMTBFs used in the simulation-based study.

From the data we can also see another important trend regarding the choice of application interval. It appears to be far more important to err on the side of overestimation, rather than underestimation. As the checkpoint interval becomes shorter, the impact becomes far more severe. However, the non-symmetry is exacerbated by the fact that small changes in the "err" axis here actually represent large error in underestimation. For example an "err" value of .1 actually means that the chosen checkpoint interval is an order of magnitude smaller than optimal.

Thus far we have studied the impact to application efficiency with respect to the specified error factor in checkpoint interval assignment. While this is a useful starting point, perhaps the more important question is how the efficiency varies with respect to error in the underlying parametrization of the interval assignment. As Equation 3 implies, there can be two sources of error in the assignment of a first-order optimal checkpoint interval, specifically, the dump time δ and M application mean time to interrupt (AMTTI). If we assume for a moment that the specified dump time is fairly accurate, any error in interval assignment must be the result of errors in the estimation of M for the given job. By moving the err term in Equation 3 under the square root, we see that an error of err in the checkpoint interval calculation results in an error of err^2 in the estimation of M (Equation 4).

$$\overline{t_c} = \sqrt{2\delta(err^2)M} \quad \text{for} \quad .1 \leq err \leq 4.0 \ . \tag{4}$$

In order to more clearly illustrate the results obtained from the parameter study, we cast the data in terms of loss of application efficiency (with respect to optimal) as a function of error in the estimation of M, rather than t_c. This is perhaps a more useful organization of the data because it correlates with what one may expect in assigning a "poor" checkpoint interval using a coarse estimate of his application's expected AMTTI (M). The results of this interpretation are provided in Figure 3.

4.1 Impact On User Experience

Both the analytical and simulation-based results imply that the AMTTI can be vastly overestimated without having a large impact on application efficiency. The natural question that remains is whether these relatively small changes in application efficiencies can have grave impacts on such performance metrics as throughput and application turnaround time. To answer these questions, we must simultaneously consider system load and application efficiency.

From a queueing theory point of view, as application efficiency degrades, the service time per job increases. This, in turn, impacts the departure rate of jobs, and by definition, throughput. The issue occurs when the system load is high. In this case, the departure rate starts coming closer to the arrival rate, and as this occurs, the system becomes much

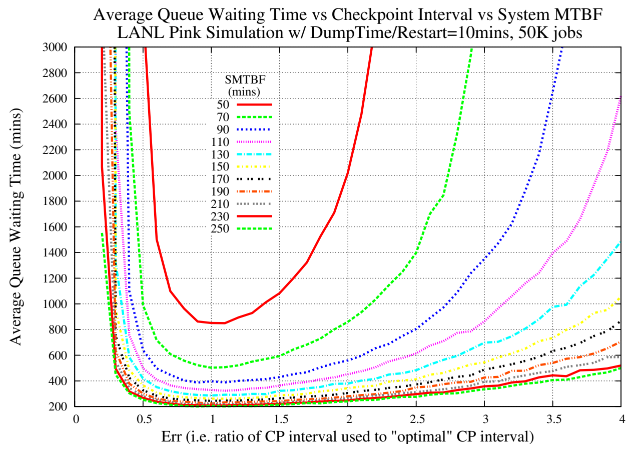

Figure 4: Simulation-based results that show the impact to queue waiting time as a function of error in checkpoint interval assignment. ***Note that while the previous results suggest minimal impact on application efficiency, small error in checkpoint interval assignment can have a dramatic impact on queueing delay, and by extension, turnaround time and job throughput.***

more sensitive to changes in service time. This, in turn, dramatically impacts queueing time, and therefore overall application turnaround time. The results are provided in Figure 4.

For example, at a SMTBF of 50 minutes, the average queue waiting time increases from 850 minutes at optimal to 2000 minutes when $err = 2$, an over two-fold increase in waiting time. This result paints quite a different picture of the impact of error when compared to the data presented in Figure 3. There we see that the loss in application efficiency at $SMTBF = 50$ and $err = 2$, is a mere 5%. While 5% does not initially seem like such a bad compromise, a two-fold increase in queueing delay certainly does. Note that the queueing delay is calculated as the difference between the job dispatch event time and the job submit event time in our discrete event-driven simulator. The presented results represent the system average across all submitted jobs.

These results imply that while there is a relatively minor impact on application efficiency due to error in optimal checkpointing interval assignment, it can nonetheless have a relatively major impact on such system metrics a queueing time. The extent to which this is the case depends on system load. We consider higher system loads to be the more important scenario since it represents the situation where both execution time and waiting time come into play.

5. CONCLUSIONS

In this paper, we present the results of a simulation-based study that addresses the extent to which error in optimal checkpoint interval assignment impacts both the overall application efficiency as well as the queue waiting time. We demonstrate that these simulation-based results corroborate the trends predicted by the analytical model. Our results indicate that underestimating an application's optimal checkpoint interval is generally more severe than overestimation. Additionally, we support, through simulation, that the underlying estimation of AMTTI can actually sustain a much larger range of errors due to the relationship between interval width and AMTTI. Furthermore, we demonstrate that while the absolute loss in application efficiency may not appear to be significant, that other system performance metrics, especially one that the end-user finds important, i.e., queue waiting time, is in fact much more significantly degraded compared to the system-centric measure of application efficiency.

6. REFERENCES

[1] J. T. Daly. A Higher Order Estimate of the Optimum Checkpoint Interval for Restart Dumps. *Future Generation Computer Systems*, 22:300–312, 2006.

[2] J. T. Daly. Methodology and metrics for quantifying application throughput. In *Proceedings of the Nuclear Explosives Code Developers Conference*, 2006.

[3] E. Hendriks. Bproc: the beowulf distributed process space. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 129–136, New York, NY, USA, 2002. ACM.

[4] BeoSim Website. http://www.parl.clemson.edu/beosim.

[5] W. M. Jones. Network-aware selective job checkpoint and migration to enhance co-allocation in multi-cluster systems. In *Journal of Concurrency and Computation: Practice and Experience*, volume 21, pages 1672–1691. John Wiley and Sons, Ltd., September 2009.

[6] W. M. Jones, J. T. Daly, and N. A. DeBardeleben. Application resilience: Making progress in spite of failure. In *The Workshop on Resilience held in conjunction with the IEEE International Conference on Cluster Computing and the Grid (CCGRID 2008)*, pages 789–794, May 2008.

[7] W. M. Jones, L. W. Pang, D. Stanzione, and W. B. Ligon III. Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. In *Journal of Supercomputing, Special Issue on the Evaluation of Grid and Cluster Computing Systems*, volume 34, pages 135–163. Springer Science and Business Media B.V, November 2005.

[8] J. W. Young. A first-order approximation to the optimum checkpoint interval. In *Communications of the ACM*, pages 530–531, September 1974.