Does Partial Replication Pay Off?

Jon Stearley, Kurt Ferreira, David Robinson,
Jim Laros, Kevin Pedretti
Computer Science Research Institute
Sandia National Laboratories[1]
{jrstear,kbferre,drobin}@sandia.gov

Dorian Arnold, Patrick Bridges
Computer Science Department
University Of New Mexico
{darnold,bridges}@cs.unm.edu

Rolf Riesen
IBM Research, Ireland[2]
rolf.riesen@ie.ibm.com

***Abstract*—As part counts in high performance computing systems are projected to increase faster than part reliabilities, there is increasing interest in enabling jobs to continue to execute in the presence of failures. Process replication has been shown to be a viable method to accomplish this, but previous studies have focussed on full replication levels (dual, triple, etc). In this work, we present a model for studying job interrupt times on systems of arbitrary replication degree, and arbitrary node failure distribution. We show agreement of this model with a previously developed simulator and make three key observations for systems using process replication; 1) job interrupts are not exponentially distributed (even when underlying node failures are), 2) job mean time to interrupt increases exponentially between full replication degrees, and 3) while partial replication may pay off for interrupt-dominated jobs, full replication degrees offer the best overall value.**

I. Introduction

Supercomputer scale is growing in both component count and societal impact. Once tools for governments only, today they impact everything from the evening weather forecast to tire and drug design to fission and fusion research. Their unparalleled capabilities enable breathtaking science, and demand huge financial, power, and human resources. A dark cloud of concern is rising however, due to the unavoidable outcome on current systems that increasing component counts (up to 220K sockets by 2015 [1]) and stalled component failure rates [2] result in decreasing time to job interrupt. This stems directly from the fact that most current systems operate such that the failure of any non-redundant component interrupts the entire job. The most common failure mitigation strategy, checkpoint/restart, saves application state at fixed intervals, such that when interrupts do occur, jobs can restart from their last checkpoint. However, checkpoint sizes are increasing faster than checkpoint bandwidths [2]. It has been shown that the collision of these trends will render Exascale systems as "useless" due to checkpoint/restart overheads [1], and thus it is time for new reliability strategies to be explored [2], [3].

A variety of approaches is possible, including message logging, uncoordinated checkpointing, checkpoint compression, and checkpointing to rack-based flash with asynchronous push to disks [4], but this paper focusses on redundancy since it is unequalled as a proven-effective resilience strategy in other domains. It has been studied widely, and deployed in technical and non-technical settings ranging from circuits and space ships to communications and governance. In return for its benefits to completion and correctness in the midst of failures and faults, it unarguably involves significant costs. However, its benefits have been shown to outweigh its costs for many purposes. Redundancy is already deployed at various scales within supercomputers, including power, disks, and memory, but it has not been deployed at the macro-scale of computation itself. At that point, it is roughly assumed to completely double the cost of systems, for which the return on investment is unclear to most and silly to some. While full process replication coupled with traditional checkpoint/restart has been shown to be a viable resilience strategy for exascale systems [5]; in this work we explore partial replication.

We begin by showing in section II that job interrupts on systems with replication are not exponentially distributed, even when node failures are. We then describe a model for full or partial replication systems in section III, and explore mean time to job interrupt and total time to solution in sections IV and V respectively.

II. Job Interrupt Distribution

It is common to assume that node failures are independent and exponentially distributed [6]–[8]. On a non-replicated system, it follows that job interrupts are also exponentially distributed. However, prior works assume that this is also true on replicated systems [7], [8]. We now show this assumption to be incorrect and therefore motivate the importance of this model.

Quantile-quantile plots (Q-Q plots) are a standard means of assessing whether data observations match a theoretical distribution. Figure 1 plots simulated job interrupt times on non, dual, and triple replicated systems, versus exponentially distributed values. In each case, the mean of the exponential is set to match the observed mean for each system. While interrupts for the non-replicated system are clearly exponentially distributed, the times for the replicated systems are not. This motivates the need for more careful modeling of interrupt times on systems that use process replication.

[1]Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000. This document's Sandia identifier is SAND2012-2144C.

[2]This research was partially supported by an Enterprise Partnership Scheme grant co-funded by IBM, the Irish Research Council for Science, Engineering & Technology (IRCSET), and the Industrial Development Agency (IDA) Ireland.

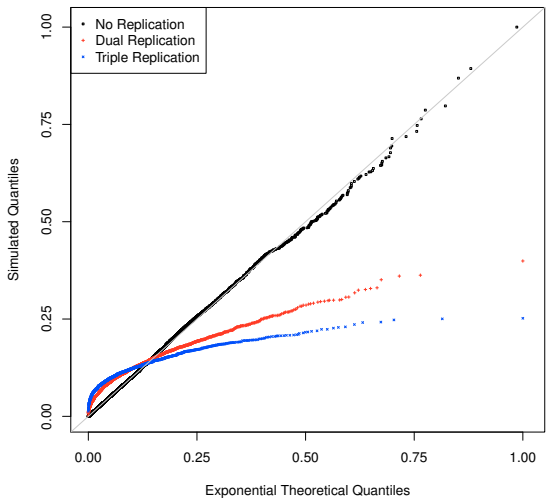


Fig. 1. Q-Q plot of job interupt times for a non-replicated, dual-replicated, and triple-replicated system. In each case, individual node failures times are taken from an exponentially distributed random variable. While interrupts for the non-replicated system are clearly exponentially distributed, the times for the replicated systems are not.

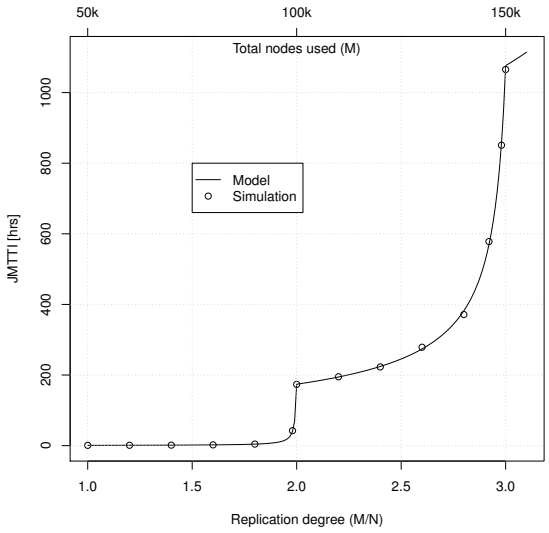


Fig. 2. Job mean time to interrupt (JMTTI) increases non-linearly, with sharp jumps at full replication degrees - suggesting that partial replication will not pay off. Model and simulation results agree tightly.

III. Model for Time To Job Interrupt

We now briefly derive a model for job interrupts on systems with arbitrary replication, using basic reliability theory and following previous work [7], [9]. The model makes no assumption on failure distribution and is useful for exploring a variety of reliability issues.

Let $F(t)$ be the probability that a device fails by time t, and $R(t)$ be the probability that the device fails after time t, $R(t) = 1 - F(t)$, and is commonly known as its *reliability*. Assume that device failures are independent and identically distributed (IID). A series system of N devices will fail upon the first device failure, $R_{series} = \prod_{i=1}^{N} R(t)$. Conversely, a parallel system of K devices fails upon the last failure of a device, or $F_{parallel} = \prod_{i=1}^{K} F(t)$.

Now consider a job running on a series of N replica groups, where the i'th group has K_i replicas. The total number nodes involved in the job will be $M = \sum_{i=1}^{N} K_i$. Each replica group is a parallel system, so the probability that all K_i nodes in group i have failed by time t is $R_i(t) = (1 - F(t)^{K_i})$. The job will be interrupted as soon as all the nodes in any replica group have failed, so the probability that the job will be interrupted by time t is

$$R_j(t) = \prod_{i=1}^{N} (1 - F(t)^{K_i}). \quad (1)$$

$F_j(t) = 1 - R_j(t)$
$F_i(t) = F(t)^{K_i}$
$JMTTI = \int_0^\infty \prod_{i=1}^{N} (1 - F(t)^{K_i}) dt$

We refer to the ratio of total nodes M to required nodes N used by a job as *replication degree*. An integer degree indicates a full replication level (1 is no replication, 2 is dual, etc), while a non-integer indicates *partial replication*. We use to N indicate number of nodes because it is an intuitive unit of job size and common unit of failure repair in the field. Without loss of generality however, N could track other failure granularities, such as sockets [2].

IV. Job Mean Time to Interrupt (JMTTI)

One of the first issues explored for HPC systems is the mean time between job interruptions. It is an intuitive metric of system reliability, and commonly used for calculating the optimal time between writing checkpoints [6]. We use the term JMTTI to refer to job mean time to interrupt to be consistent with a recent procurement contract [10], and we have found this distinctive acronym to promote clarity on exactly what is being discussed. Its use helps avoid getting sidetracked into what is meant by fault, failure, interrupt, etc. For any node failure distribution, JMTTI can be calculated as [9]:

$$JMTTI = \int_0^\infty R_j(t) dt. \quad (2)$$

Figure 2 shows JMTTI as a function of replication degree. The line depicts numerical evaluation of Equation 2 in Mathematica, and the dots depict results from a simple simulator we wrote which mimics the node failure and job interrupt behaviour on a system with process replication [11]. The tight agreement between these provides cross-verification. Here and throughout this paper, a job size of $N = 50,000$ is used, as this falls within the range where dual replication may be viable [5]. In addition, an exponential distribution for node failures is used, $F(t) = 1 - e^{-t/\theta}$ where $\theta = 5yr$ [2] is the mean time to failure for each node. Consideration of other node failure distributions is left for future work.

JMTTI does not increase linearly with replication degree. It increases very slowly just above integer replication degrees,

and very quickly just below them. JMTTI exhibits step-like increases as integer replication degrees are approached. This is a surprising trend, and has not been previously shown. By replicating only 50% of nodes, very little increase in JMTTI is realized. And why would one replicate only 95% of nodes when replicating an extra 5% brings the most substantial increases in JMTTI? The sharp increases at full replication degrees suggest that the benefits of partial replication will not outweigh its costs. More important than JMTTI however is replication's effect on total time to solution, which we examine in Section V.

A. Exponential Increases

We now explain the intuition for the sharp increase in JMTTI at integral replication levels. Consider a system with N nodes that fail independently. For a given time period, each individual node has a probability, P of failing within that interval ($F(t)$ in Section III). Therefore, the probability of the system (all its nodes) surviving the interval is $S = (1-P)^N$.

Now if each node is a part of a bundle with K replica nodes, the probability of a bundle failure in this period, the probability of all of its nodes failing, is P^K. So, with K-way replication, the probability of the system not failing during this period, the probability that no bundles fail, is: $S = (1-P^K)^N$ ($R_j(t)$ in section III).

For partial replication in which Q nodes have nodes have $K+1$ replicas and $N-Q$ nodes have K replicas, the probability of the system succeeding for the period is simply the probability of all of the bundles succeeding, based on their respective replication levels:

$$S(Q) = (1-P^K)^{N-Q} * (1-P^{K+1})^Q \quad (3)$$

$$= \frac{(1-P^K)^N}{(1-P^K)^Q} * (1-P^{K+1})^Q \quad (4)$$

$$= (1-P^K)^N * \left[\frac{1-P^{K+1}}{1-P^K}\right]^Q \quad (5)$$

As Q increases, a bundle that previously succeeded with probability $(1-P^K)$ is replaced with one that succeeds with greater probability $(1-P^{K+1})$, that is:

$$S(Q+1) = S(Q) * \frac{1-P^{K+1}}{1-P^K}. \quad (6)$$

This results in a growth function between levels of replication that is exponential.

Intuitively, as we increase the degree of partial replication, we exponentially increase the system's chance of success. However, since the base of the exponent is a number just very slightly above 1, replicating just a few of the nodes gives only a slight benefit. As we replicate more nodes, the replication benefit compounds, eventually getting us to the benefit of the next level of full replication. This behavior is independent of node failure distribution.

V. Time to Solution

Although Equation 1 is useful towards the development of a model for total wallclock time to solution, this is beyond the scope of this paper. However, in this section we take an exploratory first look via the aforementioned simulator, compared with the standard model for wallclock time without replication. Assuming that job interrupts are exponentially distributed (see Section II for contrast), Daly has shown that total wallclock time to solution T_w is given by [6]:

$$T_w = JMTTI * e^{R/JMTTI}\left(e^{(\tau+\delta)/JMTTI} - 1\right)\frac{T_s}{\tau}. \quad (7)$$

Here, the time to solve the problem is T_s if there are no interrupts, checkpoints, or restarts. However, time is spent writing and restarting from checkpoints (δ and R respectively), and τ is the time spent computing before initiating a checkpoint write. In this case, Daly shows that the optimal interval is well estimated by:

$$\tau_{opt} = A * \sqrt{2 * JMTTI * \delta} - \delta \quad (8)$$

where

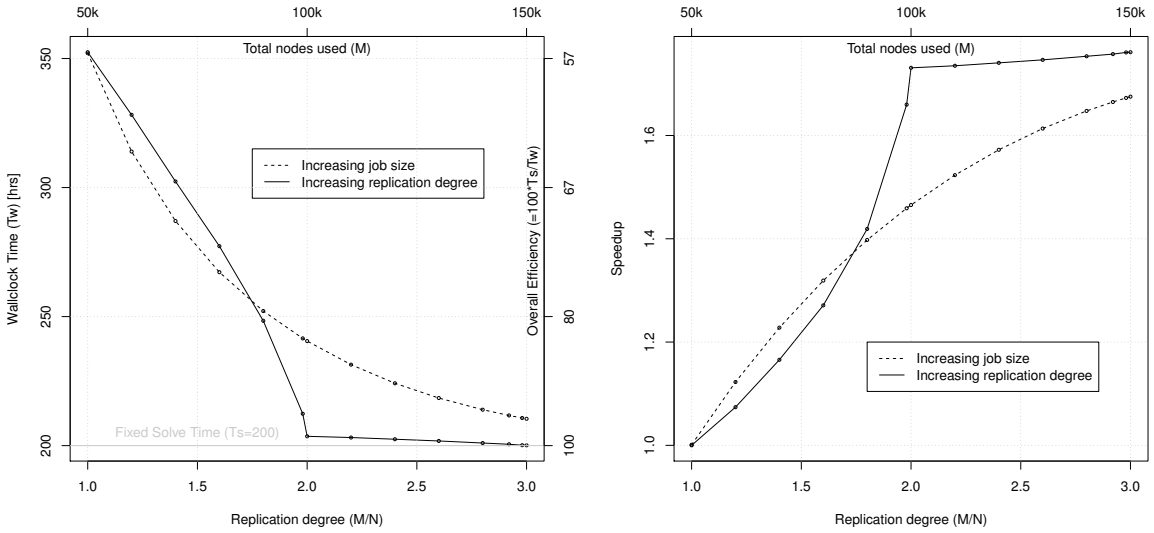
$$A = 1 + \frac{\delta}{18 * JMTTI} + \sqrt{\frac{\delta}{18 * JMTTI}}. \quad (9)$$

We examine the case where $R = \delta = 5$ minutes, which is an optimistic estimate for Exascale systems [5]. The dashed line in Figure 3(a) shows the evaluation of Equation 7, assuming perfect strong scaling and no replication. That is, we set $T_s = 200$ hours at $M = 50,000$, and linearly decrease it with additional nodes as job size M is increased, such that $T_s = 100$ hours at $M = 100,000$. JMTTI and τ_{opt} are set appropriately as M is increased. As the dashed runtime line flattens out, the effect of additional job interrupts from additional nodes begins to equal their computational benefits.

The solid line in 3(a) shows simulation results for total wallclock time to solution, if additional nodes are used for replication instead of increasing job size. In this case, job size $N = 50,000$ is held constant, and replication degree M/N is increased. Although replication does not decrease runtime as fast as perfect strong scaling initially, it starts beating it at about $M/N = 1.75$. Dual replication ($M/N = 2.0$) yields the most pronounced advantage.

On the right hand axis of 3(a), we show overall efficiency, calculated as T_s/T_w. Note that this job's overall efficiency is never less than 50%, which has widely been assumed to be the point at which replication might be worthwhile. Yet, using additional nodes to reduce the number of job interrupts via replication brings greater reduction in total runtime than perfect strong scaling. This motivates the need for more investigation on where nodes should be used for replication instead of increasing job size. A full investigation will incorporate the overhead costs of replication (e.g. we have projected communication overhead to be no greater than 5% at 200k sockets [5]), and realistic application scaling instead of our first order assumptions of no overhead and perfect strong scaling. These are left for future work.

Figure 3(b) shows the same simulation results, but in terms of speedup instead of raw wallclock time. The dashed line indicates the ratio of wallclock time with increasing job size



(a) Wallclock time and overall efficiency

(b) Speedup

Fig. 3. 3(a) compares the wallclock benefit of using additional nodes to increase job size (assuming perfect strong scaling) versus using additional nodes to increase replication degree (where job size is held constant at $N = 50,000$). Figure 3(b) shows the same data, but in terms of speedup, calculated as the ratio of wallclock time at $M = N = 50,000$ to wallclock time with additional nodes. Partial replication begins to "pay off" at $M/N = 1.75$, but full dual replication offers the best overall value (the greatest speedup compared to using nodes for increased job size). In this case, dual replication yields an overall efficiency of 98%, leaving little to be gained by further replication.

M versus wallclock time at $M = 50,000$ (assuming perfect strong scaling as before). The solid line indicates the ratio of wallclock time with increasing replication degree M/N versus wallclock time at $M = N = 50,000$. Their intersection is consistent with Figure 3(a), at about $M/N = 1.75$. Note that the speedup curve with replication mimics the exponentially increasing behaviour described in Section IV. For this job, dual replication does so well that there is very little gain left for triple replication. Our experiments show that this is the case for all system size and failure rate estimates for Exascale systems - replication beyond dual is unlikely to be worthwhile for these systems.

Note that the vertical scale of Figure 2 accomidates the full range of increase from no replication to triple replication. As such, JMTTI appears to be nearly flat at low replication degrees. In contrast, Table I provides JMTTI in minutes, along with the percentage of jobs which are interrupted before they can complete their first checkpoint write. We call such jobs "wasted" as they make no forward progress - the next job will be starting from the same place as the "wasted" one. For this simulation, JMTTI increases fast enough at low replication degrees to significantly reduce the number of wasted jobs. This explains the smooth decrease in total wallclock time in Figure 3(a).

From this analysis we conclude that while partial replication may "pay off" (yield a shorter runtime than perfect strong scaling), the best overall value is offered by full replication.

M/N	JMTTI [minutes]	Wasted jobs
1.0	52	43%
1.2	65	39%
1.4	88	34%
1.6	131	28%
1.8	266	20%
2.0	10,416	.05%

TABLE I
AS JMTTI INCREASES WITH REPLICATION DEGREE M/N, FEWER JOBS ARE INTERRUPTED BEFORE THEY CAN WRITE THEIR FIRST CHECKPOINT. WE CALL SUCH JOBS "WASTED" BECAUSE THEY MAKE NO FORWARD PROGRESS.

VI. RELATED WORK

Fault-tolerance for extreme-scale systems is an active research area with a number of alternative approaches. Most of these alternative approaches decrease overheads by improving the performance of the checkpoint and restart mechanisms. Replication, on the other hand, dramatically increases the application MTTI therefore dramatically decreasing the frequency at which we need to take checkpoints. In the remainder of this section, we describe these approaches and briefly discuss their potential benefits and costs.

A. Replication-Based Work

Ferreira et al. [5] used modeling, simulation, and empirical analysis to outline the benefits of full dual replication for extreme scale systems. This study looked at a prototype replication library implementation, called *r*MPI, to outline the runtime overheads of this approach. Using these overheads we found the break-even socket count where full dual replication will be more efficient that traditional checkpoint/restart. Similarly, Engelmann et al. [7] investigate the benefits of full dual and triple redundancy but in terms of the metric

system availability. In this work the authors show how redundancy can significantly increase system availability and correspondingly lower the required component reliability or quality. Most recently, Elliot et al. [8] investigate the benefits of partial replication using an analytical model. This model has different results from those in this paper, due in part to the author's assumption that failures of both the physical hardware and the replica groups is exponentially distributed. In fact, this assumption can lead to results that are an order of magnitude higher than those from the model presented in this paper and the simulator results presented in [12]. As we show in Section II, if we assume physical node failures are exponentially distributed, the failures of the replica groups are not exponentially distributed. In addition, this work does not fully evaluate the costs of using replication. In the work all replicated nodes are assumed to be free and in scenarios where replication is used the solves times used are for problems containing different amounts of work.

B. High-speed Storage for Checkpoint/Restart

High speed local storage, for example local disk and flash memory systems has been proposed to speed up checkpoint/restart. This method works by placing large amounts of high-speed storage near the data that must be checkpointed. Actually deploying large amounts of local non-volatile storage in an exascale system is potentially very challenging, however. Local disk-based storage has traditionally been avoided because of the increased failures it causes, for example. Upcoming non-volatile phase change PCRAM [13] and resistive RRAM devices provide high bandwidth and reliability, but are potentially very expensive. Modern NAND and NOR flash technologies are potentially the most promising for buffering and storing local checkpoints because of their comparatively low cost, high density, and high reliability. However, their write durability may require periodically replacing all flash memory in the system, significantly altering the total cost of the machine.

C. Asynchronous Checkpointing and Message Logging

Another approach that has been suggested to improve the performance of checkpointing systems is uncoordinated or asynchronous checkpointing [14]–[16]. Nodes generally checkpoint and restore from local storage without the synchronization used by coordinated checkpointing. To ensure consistent checkpoints, nodes in this approach keep a log of recent messages that they have sent. When a node restores from a previous checkpoint, it can then replay reception of messages using remote nodes' logs. While this approach can increase checkpointing performance by decreasing overheads, it also assumes the availability of local storage. In addition, logging increases the latency of messaging operations and can take a significant amounts of space on the node.

D. Memory-based and Multi-level Checkpointing

Memory-based checkpointing [17], [18] uses the memory of a remote machine to checkpoint node state. Since memory is regarded as a key budget and power constraint in exascale systems, it is unclear if the benefits of replicating only memory are superior to the qualitative advantages of replication. Similarly, multi-level checkpointing [19] is a library-based approach for checkpointing to multiple storage targets, including memory-based checkpoints, local checkpoint storage, and remote checkpoints, all in a single system. Because of its similarity, it shares many of the advantages and disadvantages of memory-based checkpointing and local storage techniques. Unlike these techniques, however, multi-level checkpointing has the flexibility to choose between multiple levels of storage based on system design parameters and application behavior, making it a promising technique for exascale systems.

VII. Conclusions and Future Work

In this work, we evaluated the suitability of partial replication combined with traditional checkpoint/restart as the primary fault tolerance method for extreme-scale systems. Using a combination of modeling and simulation, we outline the impact of arbitrary levels of replication on an applications interrupt time. Our results show that, when the underlying node failures are exponentially distributed, the job interrupt times are not exponential. In addition, we show that the mean interrupt time for an application increases exponentially between full replication degrees. Also, we show that while partial replication can dramatically decrease the the time to solution for interrupt-dominated applications, the greatest benefits are seen with integral levels of replication. Most importantly, the model and simulation described in this paper more accurately define the benefits of replication than previous work as it does not assume that interrupt times with replication are exponentially distributed.

While the research described in this work outlines many of the potential costs and benefits of partial process replication, there is a great deal of additional work that remains. For example, in this work we only considered exponentially distributed hardware failures. Considering additional and more realistic failure distributions is important as results can vary dramatically dependent on the distribution. Also, while we show that the failure times of replica groups are not exponential, we have not yet determined what distribution best fits these times. Having that distribution is important in determining the optimal number of replicas to use for an application on a given platform. A closed-form solution for total time to solution and easy-to-evaluate estimates for JMTTI and optimal checkpoint interval are also needed. Lastly, the analytical model for evaluating the benefit and cost of partial replication is too simple. A more realistic performance model which consider the overheads of the replication mechanism and application scaling characteristics is needed to better quantify the viability of partial replication.

References

[1] P. M. K. (editor), "Exascale computing study: Technology challenges in achieving exascale systems," Univ. of Notre Dame, CSE Dept., Tech. Report TR-2008-13, Sept. 28 2008.

[2] B. Schroeder and G. A. Gibson, "Understanding failures in petascale computers," *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012022, 2007.

[3] M. Wu, X.-H. Sun, and H. Jin, "Performance under failures of high-end computing," in *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 2007, pp. 1–11.

[4] G. Grider, "Speed matching and what economics will allow," HEC FSIO Research and Development Workshop, Tech. Rep., 2010.

[5] K. Ferreira, R. Riesen, J. Stearley, J. H. L. III, R. Oldfield, K. Pedretti, P. Bridges, D. Arnold, and R. Brightwell, "Evaluating the viability of process replication reliability for exascale systems," in *Proceedings of the ACM/IEEE International Conference on High Performance Computing, Networking, Storage, and Analysis, (SC'11)*, Nov 2011.

[6] J. T. Daly, "A higher order estimate of the optimum checkpoint interval for restart dumps," *Future Gener. Comput. Syst.*, vol. 22, no. 3, pp. 303–312, 2006.

[7] C. Engelmann, H. H. Ong, and S. L. Scott, "The case for modular redundancy in large-scale high performance computing systems," in *Proceedings of the* 8^{th} *IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN) 2009*. Innsbruck, Austria: ACTA Press, Calgary, AB, Canada, Feb. 16-18, 2009, pp. 189–194. [Online]. Available: http://www.csm.ornl.gov/~engelman/publications/engelmann09case.pdf

[8] J. Elliot, K. Kharbas, D. Fiala, F. Mueller, K. Ferreira, and C. Engelmann, "Combining partial redundancy and checkpointing for HPC," in *International Conference on Distributed Computing Systems*. Los Alamitos, CA, USA: IEEE Computer Society Press, June 2012, pp. 1–11, [to appear].

[9] J. C. Turner, "Reliability and operability of systems of components in series and in parallel," *Electronics Reliability and Microminiaturization*, vol. 1, pp. 21–26, 1962.

[10] "Scope of work and technical specification (for the cielo supercomputer)," Los Alamos National Laboratory, Tech. Report Subcontract 65483-001-10 Exhibit D, February 2010.

[11] R. Riesen. Process replication simulator. [Online]. Available: http://www.cs.sandia.gov/~rolf/app_model.html

[12] R. Riesen, K. Ferreira, J. Stearley, R. Oldfield, J. H. L. III, K. Pedretti, and R. Brightwell, "Redundant computing for exascale systems," Sandia National Laboratories, Technical report SAND2010-8709, Dec. 2010.

[13] X. Dong, N. Muralimanohar, N. Jouppi, R. Kaufmann, and Y. Xie, "Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09. New York, NY, USA: ACM, 2009, pp. 57:1–57:12. [Online]. Available: http://doi.acm.org/10.1145/1654059.1654117

[14] J. Ahn, "2-step algorithm for enhancing effectiveness of sender-based message logging," in *SpringSim '07: Proceedings of the 2007 spring simulation multiconference*, 2007, pp. 429–434. [Online]. Available: http://portal.acm.org/citation.cfm?id=1404748

[15] Q. Jiang and D. Manivannan, "An optimistic checkpointing and selective approach for consistent global checkpoint collection in distributed systems," in *Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium*, Mar. 2007.

[16] D. B. Johnson and W. Zwaenepoel, "Recovery in distributed systems using asynchronous and checkpointing," in *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, 1988, pp. 171–181.

[17] J. S. Plank, Y. B. Kim, and J. J. Dongarra, "Algorithm-based diskless checkpointing for fault tolerant matrix operations." in *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*. Pasadena, CA, USA: Los Alamitos, CA, USA : IEEE Comput. Soc. Press, 1995, June 1995, pp. 351–360.

[18] L. M. Silva and J. G. Silva, "An experimental study about diskless checkpointing." in *24th EUROMICRO Conference*. Vasteras, Sweden: IEEE Computer Society Press, August 1998, pp. 395 – 402.

[19] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–11. [Online]. Available: http://dx.doi.org/10.1109/SC.2010.18