Reliability-Aware Scalability Models for High Performance Computing

Ziming Zheng and Zhiling Lan

Department of Computer Science, Illinois Institute of Technology
Chicago, IL 60616, USA
{zzheng11, lan}@iit.edu

***Abstract*— Scalability models are powerful analytical tools for evaluating and predicting the performance of parallel applications. Unfortunately, existing scalability models do not quantify failure impact and therefore cannot accurately account for application performance in the presence of failures. In this study, we extend two well-known models, namely Amdahl's law and Gustafson's law, by considering the impact of failures and the effect of fault tolerance techniques on applications. The derived reliability-aware models can be used to predict application scalability in failure-present environments and evaluate fault tolerance techniques. Trace-based simulations via real failure logs demonstrate that the newly developed models provide a better understanding of application performance and scalability in the presence of failures.**

I. Introduction

Fueled by the ever-growing demand for more computational power in science and engineering, large-scale systems with hundreds of thousands of processors are being designed and deployed [7]. In these systems, scalability is a key factor for evaluating, predicting and optimizing application performance. It not only measures the inherent parallelism of parallel applications, but also provides an important guidance to build parallel computers [11].

Amdahl's law [3] and Gustafson's law [4] are two well-known scalability models. They are used to estimate the performance of parallel applications at scale; however, both implicitly assume that applications can complete without experiencing any failure. Nevertheless, with the increasing scale and complexity of computer systems, failure becomes a commonplace scenario rather than an exception. Recent studies have shown that MTBFs (mean-time-between-failures) of teraflop and petaflop-scale systems are only on the order of 10-100 hours, even for systems based on ultrareliable components [8]. As a result, long-running parallel applications may hardly make any forward progress because of failures [5], [6]. Failure forces the application to wait for system recovery, which may range from a couple of hours to nearly 100 hours [1], and roll back to the beginning or the latest checkpoint. Due to the impact of failures, application scalability is quite different from the ideal failure-free cases [19].

The impact of failures is commonly mitigated by checkpointing, which periodically stores a snapshot of the application and uses it for recovery in case of failure. Although the application does not need to be restarted from the beginning, recovery time and work loss are inevitable during the rollback process. Further, checkpoint overhead is not trivial in a large-scale system [25]. Hence, to comprehensively understand application scalability, it is essential to incorporate the checkpointing factor in the scalability models.

In this study, we derive a set of *reliability-aware scalability models* by extending Amdahl's law and Gustafson's law. Our models take into consideration the impact of failures on application performance. The extensions are examined in two cases: one where checkpointing is used, and the other where there is no checkpointing. They are intended to provide more accurate measurement of application performance in the presence of failures.

To evaluate the accuracy of these models, we conduct trace-based simulations based on real failures logs from production systems [37]. Experimental results demonstrate that these reliability-aware scalability models can better represent application performance in a practical failure-present environment, in both checkpoint and no-checkpoint cases. Moreover, they show the optimal speedup that could be achieved by the application in the presence of failures.

The rest of the paper is organized as follows. Section II provides a brief discussion of related work. The assumptions of our reliability-aware models are presented in Section III. In Section IV and V, the enhanced Amdahl's law and Gustafson's law are developed respectively. We present our evaluation methodology in Section VI, followed by a discussion of our experimental results in Section VII. In Section VIII we use our models to evaluate two fault tolerance technologies, fast recovery and proactive failure prevention using process migration. Finally, we conclude the paper in Section IX.

II. Related Work

Scalability has been studied in decades from various aspects. Among various scalability models, Amdahl's [3] and Gustafson's [4] are two well-known models. Amdahl's model is for fixed-size problems, while Gustafson's model is for fixed-time problems. Neither of them accounts for the performance impact induced by failures — which is an increasing concern as systems and applications scale to very large sizes.

The impact of failures on scalability has been discussed in [19], [23]. Based on simulation results, Elnozahy and Plank point out that the optimal speedup decreases rapidly as the number of nodes grows beyond a certain point [19]. In [23], a reliability-aware resource allocation algorithm is presented to select an optimal number of nodes for the execution of an application. Distinguished from these studies, this paper develops analytical models to explicitly express application scalability in the presence of failures.

Several studies have examined application scalability under various system constraints. Kumar et al. develop scalability models for a specific parallel architecture and problem size [29]. Sun and Ni develop memory-constrained scalability in [10]. In [11], [12], Chen et al. introduce iso-speedup scalability for heterogeneous computing systems. In [13], a power-aware speedup is proposed to predict the scaled execution time and power consumption. In [30], Woo et al. extend the Amdahl's law for many-core architectures. Unlike these works, this study is focused on extending Amdahl's law and Gustafson's law by considering system failures.

Checkpointing is a widely used fault tolerance mechanism for high performance computing. Job execution time with checkpointing has been studied in [24], [15], [14]. In [24], Young derives the optimal checkpointing interval via the first order estimation. Daly improves the model using higher order estimation and shows the expected completion time with checkpointing in [15]. Stochastic activity networks are used to study application performance by considering checkpointing overhead and failures in [20]. In [14], Wu et al. use an M/G/1 model to describe system failures and derive performance models to estimate the mean, variation and distribution function of application completion time. In this paper, we use Daly's model to derive reliability-aware scalability models in case of checkpointing.

III. Assumption

For the development of the reliability-aware scalability models, we make several assumptions. First, the time interval between failures on each node is exponentially distributed with an arrival rate of λ. This assumption is widely adopted in the literature [14], [19], [20]. In [18], Plank et al. point out that although exponential distribution may not fit well for some realistic scenarios, many theoretical results using exponential distribution are still applicable in practice.

Second, the failure of a node can lead to the crash of the entire application [32]. A large number of applications, e.g., MPI programs, fall into this category. Thus, if the failure arrival rate of node i is λ_i, then the failure arrival rate of P nodes is $\lambda_a = \sum_{i=1}^{P} \lambda_i$. In homogeneous systems, it can be simplified as $\lambda_a = P\lambda$ [33], [38].

Third, repair time follows a general distribution with a mean of μ [14]. Further, it is not sensitive to system size in homogeneous systems. This assumption is based on the observation in [2].

Finally, for Amdahl's law, the overhead of coordinate checkpointing is $O_{ckp} = a + bP$, where a is I/O overhead and bP denotes message passing overhead [21]. Since the problem size does not change, we assume I/O overhead is fixed [21], [17]. The message passing overhead is for achieving the global consistent state, which is linear with the number of processes. For Gustafson's law, since the problem size or the checkpoint image size is proportional to the number of nodes, the I/O overhead in this case linearly increases with the number of nodes. Thus the overhead is $O_{ckp} = aP + bP$ [27], [32]. In this study, we do not differentiate checkpoint overhead and checkpoint latency [26], and assume they are much less than application workload [15].

Before presenting our models, we first list a set of nomenclatures that will be frequently used in the rest of the paper (see Table I).

IV. Augmenting Amdahl's Model

A. Without Checkpointing

According to Amdahl's law, failure-free parallel execution time is $W_p = (1-\alpha)W + \alpha W/P$, where W is the workload and α is the fraction of the application that can be parallelized. So the fixed-size speedup is defined as:

$$S^A = \frac{W}{(1-\alpha)W + \alpha W/P} = \frac{P}{P(1-\alpha)+\alpha}. \tag{1}$$

In a failure-prone environment, the expected parallel execution time is $E_f(T(W_P))$. Thus, the reliability-aware Amdahl's model is defined as:

$$S_f^A = \frac{W}{E_f(T(W_p))}. \tag{2}$$

Without a checkpointing support, each time when a failure occurs, the application will roll back to the beginning. According to [16] the expected time under failure is

$$\begin{aligned} & E(T_f(W_p)) = W_p + \tfrac{\mu(1-e^{-\lambda_a W_p}) + \int_0^{W_p} t\lambda_a e^{-\lambda_a t} dt}{e^{-\lambda_a W_p}} \\ & = (\mu + \lambda_a^{-1})(e^{\lambda_a W_p} - 1) \\ & = (\mu + \lambda_a^{-1})(e^{(1-\alpha+\frac{\alpha}{P})\lambda_a W} - 1). \end{aligned} \tag{3}$$

Consequently, we can derive the following S_f^A model

$$S_f^A = \frac{W}{(\mu + (\lambda_a)^{-1})(e^{(1-\alpha+\frac{\alpha}{P})\lambda_a W} - 1)}. \tag{4}$$

S^A is a special case of S_f^A. When the recovery time can be ignored ($\mu = 0$), each node has the same failure rate ($\lambda_a = P\lambda$), and the λ_a is much larger than the parallel workload ($\frac{1}{P\lambda} >> W_p$), based on the first-order Taylor series, we obtain

$$\begin{aligned} S_f^A & = \tfrac{W}{(P\lambda)^{-1}(e^{(1-\alpha)P\lambda W} e^{\alpha\lambda W} - 1)} \\ & \approx \tfrac{W}{(P\lambda)^{-1}((1-\alpha)P\lambda W + \alpha\lambda W)} \\ & = S^A. \end{aligned} \tag{5}$$

Comparing Equation 1 and 4, we can see that S_f^A is quite different from S^A. S^A is independent of W, whereas S_f^A is dependent on W. When W increases, S_f^A exponentially decreases to 0. It means that if the application requires substantial amount of time to complete, the application becomes more vulnerable to failures.

TABLE I
NOMENCLATURE

Symbol	Description
P	Number of computing nodes
λ_i	Failure arrival rate of node i (per hour)
λ_a	Failure arrival rate of the nodes allocated to the application (per hour)
μ	Mean-Time-To-Recover(MTTR) (hour)
W	Application workload, the application failure-free execution time on a single node (hour)
W'	The scaled workload on a single node (hour)
W_p	The parallel workload (hour)
α	The fraction of the application that can be parallelized
O_{ckp}	Checkpoint overhead (hour)
a,b	Parameters used to denote checkpoint overhead
τ	Checkpoint interval (hour)
S^A	Amdahl's scalability model without checkpointing
S^A_f	Agumented Amdahl's scalability model without checkpointing
$S^A_{f\_c}$	Agumented Amdahl's scalability model with checkpointing
S^G	Gustafson's scalability model without checkpointing
S^G_f	Agumented Gustafson's scalability model without checkpointing
$S^G_{f\_c}$	Agumented Gustafson's scalability model with checkpointing

B. With Checkpointing

Checkpointing has been widely used for fault tolerance in high performance computing. Upon a failure, the application will be restarted from the most recent checkpoint. In the following, we extend Amdahl's law by considering the effect of checkpointing.

To estimate the expected parallel execution time with checkpointing, we adopt Daly's model, which is widely used in the field [15], [32]. Daly's model shares the same basic assumptions about failure and checkpointing as our models listed in the previous section. It derives the expected execution time with checkpointing as follows

$$E(T_{f\_c}(W_p)) = \frac{e^{\mu\lambda_a}}{\lambda_a}(e^{(\tau+O_{ckp})\lambda_a}-1)\frac{(1-\alpha+\frac{\alpha}{P})W}{\tau}, \quad (6)$$

where $O_{ckp} = a + bP$ is the checkpoint overhead and τ is the checkpoint interval. From [15], the optimal checkpoint interval can be approximated as follows

$$\tau = \begin{cases} \sqrt{\frac{2O_{ckp}}{\lambda_a}}[1+\frac{1}{3}(\frac{O_{ckp}\lambda_a}{2})^{\frac{1}{2}}+ \\ \frac{1}{9}(\frac{O_{ckp}\lambda_a}{2})]-O_{ckp} & O_{ckp} < \frac{2}{\lambda_a}, \\ \frac{1}{\lambda_a} & O_{ckp} \geq \frac{2}{\lambda_a}. \end{cases} \quad (7)$$

Based on Equation 6 and 7, we can derive the following $S^A_{f\_c}$ model:

$$\begin{aligned} S^A_{f\_c} &= \frac{W}{E(T_c(W_p))} \\ &= \frac{\lambda_a\tau}{e^{\mu\lambda_a}(e^{(\tau+O_{ckp})\lambda_a}-1)(1-\alpha+\frac{\alpha}{P})}. \end{aligned} \quad (8)$$

Distinguished from S^A_f presented in Equation 4, $S^A_{f\_c}$ is not dependent on W. Comparing Equation 3 and 6, we can see that the difference is due to the influence of checkpointing. With a checkpointing support, the expected execution time linearly increases with the workload, whereas it is exponentially increased with the workload without checkpointing. Thus checkpointing is helpful to keep scalability of the applications with high workload.

C. A Use Scenario

In Amdahl's model, S^A monotonically increases with the growth of P, and $\lim_{P\to\infty} S^A = \frac{1}{1-\alpha}$. Unlike S^A, S^A_f and $S^A_{f\_c}$ may decrease with the growth of P since application failure rate increases when P is getting larger. Our derived reliability-aware models can be used to identify the optimal number of nodes for running the application in the presence of failures.

Given the relation between λ_a and λ_i, the optimal number of nodes can be determined by solving the equations $\frac{\partial S^A_f}{\partial P} = 0$ and $\frac{\partial S^A_{f\_c}}{\partial P} = 0$. For example, if $\lambda_a = P\lambda$, as shown in Figure 1, S^A_f indicates that the maximal speedup of 4.55 is achieved when P is set to 20, and $S^A_{f\_c}$ points out that the maximal speedup of 7.9 can be obtained when P is set to 39. The three curves in the figure clearly show that in a failure-present environment, the achievable speedups by using S^A, S^A_f, and $S^A_{f\_c}$ are quite different. The regular Amdahl's law shows that by using a large amount of computing nodes like 512, it is possible to achieve a speedup of 9.8. Our derived reliability-aware models indicate that the maximal speedup is bounded by a certain number and using a large number of computing nodes may end up with a small speedup. Furthermore, by comparing S^A_f and $S^A_{f\_c}$, we can see that the use of checkpointing can considerably boost application scalability by increasing the maximal achievable speedup.

As shown in Figure 1, both S^A_f and $S^A_{f\_c}$ can quickly drop to 0 when P is set to a large number. Using a large amount of computing nodes, failure possibility will increase drastically. As a result, without checkpointing, failures will

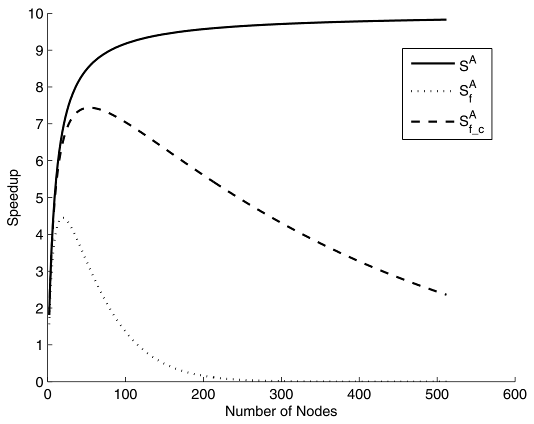

Fig. 1. Comparison of S^A, S^A_f, and $S^A_{f\_c}$ when $\lambda_a = P\lambda$. The parameters are $\alpha = 0.9$, $\lambda = \frac{1}{7500}$ hours, $\mu = 2$ hours, $W = 2000$ hours.

force the application to roll back to the beginning over and over, thereby resulting in an extreme long execution time. With a checkpointing support, application scalability can be improved. Nevertheless, as P is getting larger, checkpoint overhead increases as well, which will eventually overpower its benefits and lead to a speedup smaller than the ideal speedup given by Amdahl's law.

V. Augmenting Gustafson's Model

A. Without Checkpointing

Different from Amdahl's law, Gustafson's law emphasizes on the amount of workload that can be finished in a fixed time [4]. It assumes that the α fraction of the workload can be parallelized and scaled with the number of nodes, and the rest of the workload does not grow with the number of nodes. Hence it defines fixed-time speedup as follows

$$S^G = \frac{(1-\alpha)W + \alpha WP}{W} = 1 - \alpha + \alpha P. \tag{9}$$

In an ideal failure-free case, Gustafson's law shows that the fixed-time speedup is independent of W and linearly grows with P. In practice, as W increases, the application gets more vulnerable to failures. Considering the impact of failures, we define the achievable workload as $achievable\ workload = W - work\ loss - recovery\ time$. And the scaled workload W' is defined as $W' = achievable\ workload - (1-\alpha)W$. While the workload is scaled from W to $(1-\alpha)W + W'P$, we can derive the reliability-aware model S^G_f as follows,

$$\begin{aligned} S^G_f &= \tfrac{(1-\alpha)W + W'P}{W} \\ &= 1 - \alpha + \frac{P\ln(\frac{W}{u+(\lambda_a)^{-1}}+1)}{W\lambda_a} - P(1-\alpha). \end{aligned} \tag{10}$$

S^G is a special case of S^G_f. When the recovery time can be ignored ($\mu = 0$), each node has the same failure rate ($\lambda_a = P\lambda$), and λ_a is much larger than the execution time ($\frac{1}{P\lambda} >> W$), based on the first-order Taylor series we obtain

$$\begin{aligned} S^G_f &= 1 - \alpha + \tfrac{\ln(WP\lambda+1)}{W\lambda} - P(1-\alpha) \\ &\approx 1 - \alpha + \tfrac{WP\lambda}{W\lambda} - P(1-\alpha) \\ &= S^G. \end{aligned} \tag{11}$$

Comparing Equation 9 and 10, we can see that difference between S^G and S^G_f. S^G is not dependent on workload, while S^G_f decreases with the growth of W. It means that if the application requires substantial amount of execution time, the work loss and recovery time significantly increase due to the failures, thereby achievable workload is reduced.

B. With Checkpointing

In the following, we extend Gustafson's model by considering the impact of failures and checkpointing.

With checkpointing, the achievable workload is define as $W - work\ loss - recovery\ time - overhead$, and the scaled work load W' is defined as $W' = achievable\ workload - (1-\alpha)W$. As the problem size is scaled from W to $(1-\alpha)W + W'P$, we can derive $S^G_{f\_c}$ model as follows,

$$\begin{aligned} S^G_{f\_c} &= \tfrac{(1-\alpha)W + W'P}{W} \\ &= 1 - \alpha + \frac{\tau P\lambda_a}{e^{\mu\lambda_a}(e^{(\tau+O_{ckp})\lambda_a}-1)} - (1-\alpha)P, \end{aligned} \tag{12}$$

where $O_{ckp} = aP + bP$ and τ is decided by Equation 7. Distinguished from S^G_f presented in Equation 10, $S^G_{f\_c}$ is independent of W. Due to the checkpointing support, the achievable workload only linearly decreases with the workload, whereas it exponentially decreases with the workload without a checkpointing support.

C. A Use Scenario

In an ideal failure-free case, the Gustafson's law shows the workload linearly scaling up with the number of nodes without bound. Distinguished from S^G, S^G_f and $S^G_{f\_c}$ are limited by the failures and may decrease with the growth of P. Our reliability-aware models can be used to identify an optimal number of nodes for maximal achievable workload in the presence of failures.

Given the relation between λ_a and λ_i, the optimal number of nodes can be determined by solving the equations $\frac{\partial S^G_f}{\partial P} = 0$ and $\frac{\partial S^G_{f\_c}}{\partial P} = 0$. For example, considering $\lambda_a = P\lambda$, as shown in figure 2, S^G_f points out that the maximal fixed-time speedup of 42.32 is obtained when P is set to 243, and $S^G_{f\_c}$ suggests that the maximal speedup of 103.5 can be achieved when P is set to 288. Obviously, as shown in the three curves, the achievable fixed-time speedups by using S^G, S^G_f, and $S^G_{f\_c}$ are quite different. The regular Gustafson's law shows that the fixed-time speedup can be infinite as the computing nodes increases. Our reliability-aware models indicate that the maximal fixed-time speedup is bounded by a certain number and overzealously using computing nodes may reduce the speedup. Furthermore, the difference between S^G_f and $S^G_{f\_c}$ indicates that the use of checkpointing can increase the maximal achievable fixed-time speedup and boost application scalability.

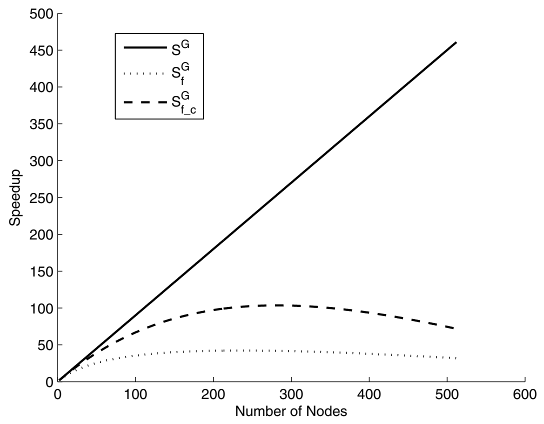

Fig. 2. Comparison of S^G, S^G_f, and $S^G_{f\_c}$ when $\lambda_a = P\lambda$. The parameters are $\alpha = 0.9$, $\lambda = \frac{1}{7500}$ hours, $\mu = 2$ hours, $W = 200$ hours.

VI. Evaluation Methodology

In order to choose a suitable scalability model for performance prediction and optimization, it is necessary to compare the accuracy of the models. In this section, we conduct trace-based simulations to compare *percentage prediction errors* between the scalability models and the real measurements. Percentage prediction error is a widely used metric [13], [14], which is defined as

$$Percentage\ Prediction\ Error = \left| \frac{Pediction - Measurement}{Measurement} \right| . \quad (13)$$

To use the simulator, the user provides the application level parameters like the workload W and the fraction α. The system level parameters such as failure rate and repair time are obtained from the trace fed into the simulator. Based on these parameters, the simulator traces the discrete events of failures and returns the measurement results like application execution time, achievable workload, etc. Meanwhile, we can also apply the scalability models discussed above to get the prediction results. By comparing the measurement results returned by the simulator and the prediction results obtained from the models, we can calculate percentage prediction errors of the models.

To truly assess the models in a realistic environment, we use a failure trace from a production system at Los Alamos National Lab . The trace is from the system #8, which is publicly available at [37]. For simplicity, we choose 128 nodes that have similar failure rates. The average MTBF of these nodes is 3872.7 hours and the mean time to recovery μ is 2.26 hours.

To study checkpoint overhead, six parallel benchmarks are tested in the open-source checkpointing package MPICH-VCL 0.76 in our previous work[21]. Based on the experiments, the IO overhead a is set as 0.335 minutes and the message passing overhead b is set as 0.0364 minutes.

For Amdahl's law, given the workload W and the fraction α, the simulator calculates the parallelized workload W_p on different number of nodes. Then it scans the failure trace in the time order and simulates a failure when a real failure entry is encountered. Without checkpointing, the job waits for the recovery and rolls back to the beginning. The job execution time includes W_p, the accumulated work loss and the accumulated recovery time. With checkpointing, the simulator obtains the optimal checkpoint interval τ based on Equation 7. The checkpoint overhead is added during every checkpoint interval. When a failure occurs, the work performed between the failure time and the most recent checkpoint is lost. The job waits for the recovery and rolls back to the most recent checkpoint. The job execution time includes W_p, the accumulated work loss, the accumulated recovery time and the accumulated checkpoint overhead.

For Gustafson's law, the execution time is fixed as W and the simulator calculates the achievable workload. Without checkpointing, only the work between the last failure and the completion time is achievable. With checkpointing, the simulator performs the checkpoints and traces each failure. All the execution time is for the achievable workload except for the checkpoint overhead, the recovery time and work loss. By recording the achievable workload and the fraction α, the simulator can calculate the fixed-time speedup as described in the previous section.

VII. Experimental Results

Our experiments are designed to study application level parameters like W and α at two aspects. In the first scenario, we fix the parameter α and vary W to analyze the relationship between application workload and its scalability. In the second scenario, we fix application workload W and vary α to study how α impacts prediction errors. In both scenarios, we change the number of nodes from 2 to 128 and calculate the average of percentage prediction errors.

A. Comparison of Enhanced and Original Amdahl's Models

In the first set of experiments, we set $\alpha = 0.9$ and test various workloads from 2000 hours to 10000 hours. As shown from Table II, without checkpointing, the accuracy of Amadal's model dramatically decreases with the growth of workload. When an application requires substantial amount of execution time, it becomes more vulnerable to failures. As a result, application completion time in a failure-prone environment becomes much longer than application completion time in a failure-free environment. The results here indicate that without considering failure impact, Amdahl's model cannot accurately represent application scalability, especially long-running applications.

With checkpointing, the accuracy is stable with different workload. The accuracy of S^A is significantly improved since checkpointing greatly reduces the work loss. However, Amdahl's model does not account for the impact of failures and fault tolerance techniques, whereas $S^A_{f\_c}$ comprehensively considers the recovery time, work loss and checkpoint overhead. As compared to S^A, $S^A_{f\_c}$ can better represent application performance. For example, the average percentage prediction

TABLE II

PERCENTAGE PREDICTION ERRORS OF S^A, S_f^A AND $S_{f\_c}^A$ WITH DIFFERENT WORKLOADS. THE $MTBF$ IS 3539.5 HOURS,$MTTR$ IS 2.26 HOURS AND $\alpha = 0.9$. THE CHECKPOINT PARAMETER a IS 0.335 MINUTES AND b IS 0.0364 MINUTES.

W	S^A (W/O CKP)	S^A(W/ CKP)	S_f^A	$S_{f\_c}^A$
2000	1.99	0.14	0.27	0.09
5000	18.89	0.24	0.73	0.11
8000	20.55	0.16	0.87	0.04
10000	21.98	0.15	0.82	0.04

TABLE III

PERCENTAGE PREDICTION ERRORS OF S^A, S_f^A AND $S_{f\_c}^A$ WITH DIFFERENT FRACTIONS α.

α	S^A (W/O CKP)	S^A (W/ CKP)	S_f^A	$S_{f\_c}^A$
0.7	15.66	0.52	0.95	0.34
0.75	8.73	0.34	0.95	0.19
0.8	10.67	0.28	0.27	0.14
0.9	21.98	0.15	0.82	0.04
0.95	23.83	0.24	0.79	0.10
0.999	3.32	0.07	0.93	0.06

error introduced by $S_{f\_c}^A$ is 4 times smaller than that introduced by S^A when $W = 8000$ hours.

In the second set of experiments, we test different values of α. The workload is set as 10000 hours and α ranges from 0.7 to 0.999. The average percentage errors are listed in table III. Without checkpointing, the results of S_f^A are always less than 1 and distinctly outperform the Amdahl's model. The accuracy of S^A is improved when α is extreme large, e.g., $\alpha > 0.95$. The main reason is that, the parallel execution time is significantly reduced by extreme large α. And the short term applications are significantly less vulnerable to the failures.

With the support of checkpointing, the average percentage error of S^A can be reduced to 0.07 when the application can be perfect parallelized, e.g., $\alpha = 0.999$. However, $S_{f\_c}^A$ still outperforms S^A no matter the value of α. It means that even the application with checkpointing can be perfect parallelized, the performance is still distinct from the ideal failure-free environments.

In this set of experiments, we compare enhanced Amdahl's models and Amdahl's model as the number of nodes increases from 2 to 128 (see Figure 3). The figure presents two interesting patterns. First, compared to Amdahl's model, our reliability-aware models, including both S_f^A and $S_{f\_c}^A$, can better model the actual measurements. Second, the gap between Amdahl's model and the actual measurement become larger with the growth of number of nodes. This is due to the fact that Amdahl's model does not consider failure impact. As the number of nodes increases, application failure probability grows, thereby resulting in a growing discrepancy between the Amdahl's model result and the actual measurement. In summary, the above experiments demonstrate that our derived reliability-aware models can better represent application scalability by considering failure impacts and checkpointing effects.

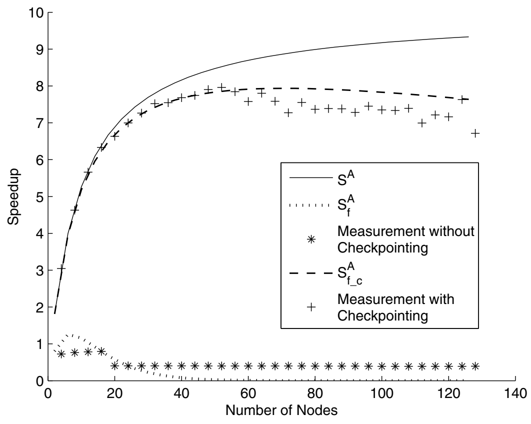

Fig. 3. Comparison of enhanced Amdahl's models and original Amdahl's model with different numbers of nodes. $\alpha = 0.9$, workload=10000 hours.

TABLE IV

PERCENTAGE PREDICTION ERRORS OF S^G, S_f^G AND $S_{f\_c}^G$ WITH DIFFERENT WORKLOADS.

W	S^G (W/O CKP)	S^G (W/ CKP)	S_f^G	$S_{f\_c}^G$
1000	1.32	0.07	0.58	0.05
1200	15.7	0.13	2.3	0.01
1800	29.86	0.14	6.91	0.01
2000	12.71	0.14	2.33	0.01

B. *Comparison of Enhanced and Original Gustafson's Models*

Similar to the first set of experiments, α is set as 0.9 and workload is changed from 1000 hours to 2000 hours. As shown in Table IV, Gustafson's model can lead to substantial errors in the failure-prone environments, where the achievable workload is much less than the ideal failure-free case. S_f^G adequately quantify the impact of failures, so it outperforms S^G with smaller errors. One observation is that the errors of S^G without checkpointing and S_f^G fluctuate with different workloads. The reason is that the time between last failure and the completion time is a random variable with large variance. In future we plan to extend the models to predict the variance of fixed-time speedup under failures.

With checkpointing, Gustafson's model achieves better results than the case without checkpointing no matter the value of workload. It shows that checkpointing has the great effect on saving achievable workload. However, $S_{f\_c}^G$ is even better than S^G with smaller error. It means that, even with the support of checkpointing, the workload is not achievable as Gustafson's model due to the inevitable recovery time, work loss and checkpoint overhead.

In the forth set of experiment, we test how the fixed-time speedup is impacted by different values of α. In this experiments, the workload is set as 1000 hours. As shown in table V, our derived reliability-aware models still outperform the Gustafson's model, no matter the value of α. One interesting observation is that, the error decreases with the growth of α even when the achievable workload is fixed. The main reason

TABLE V

PERCENTAGE PREDICTION ERRORS OF S^G, S^G_f AND $S^G_{f\_c}$ WITH DIFFERENT FRACTIONS α.

α	S^G(W/O CKP)	S^G (W/ CKP)	S^G_f	$S^G_{f\_c}$
0.7	3.70	0.12	1.43	0.05
0.8	1.9	0.08	0.80	0.06
0.9	1.32	0.07	0.58	0.05
0.95	1.15	0.06	0.51	0.05
0.999	1.01	0.06	0.46	0.05

is if α is small, the work loss for sequential part e.g., $(1-\alpha)W$ could be high, and the scaled part could be much less than the failure-free case.

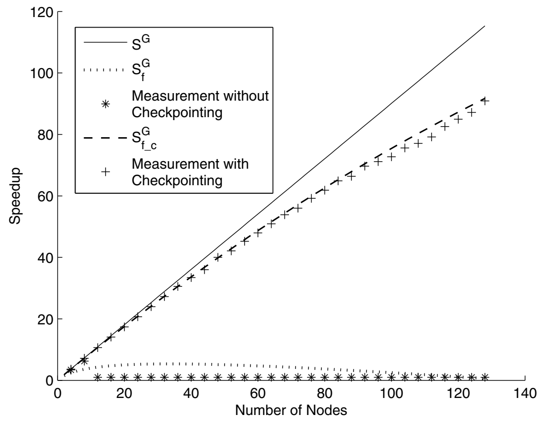

Fig. 4. Comparison of enhanced Gustafson's models and original Gustafon's model with different numbers of nodes. $\alpha = 0.9$, workload=2000 hours.

In this set of experiments, we compare enhanced Gustafson's models and Gustafson's model as the number of nodes increases from 2 to 128 (see Figure 4). Similar to Figure 3, the results show that our derived reliability-aware models can better represent application scalability by considering failure impact and checkpointing effect, especially with a large number of nodes.

C. Heterogeneous Failure Rates

In the previous experiments, 128 nodes are selected with similar failure rate to study the case $\lambda_a = P\lambda$. In practice, different nodes may have different failure rates [2]. As an example, for the system #8, some of its nodes have very high failure rates such as the node #33 with MTBF of 1378.07 hours, whereas other nodes have very low failure rates like the node #160 with MTBF of 7208.7 hours.

In this set of experiments, we assess our reliability-aware models by randomly selecting 128 nodes for the system #8. The node-level MTBF ranges from 13 to 5950.5 hours, with an average of 144.625 hours. Table VI shows the percentage prediction errors of different models. Here, we set W to 2000 hours and α is set to 0.9. The results clearly show that our derived reliability-aware models outperform Amdahl's model and Gustafson's model in case of heterogeneous failure rates.

TABLE VI

PERCENTAGE PREDICTION ERRORS OF ENHANCED AMDAHL'S/GUSTAFSON'S MODELS AND AMDAHL'S/GUSTAFSON'S MODELS.

S^A (W/O CKP)	S^A (W/ CKP)	S^A_f	$S^A_{f\_c}$
2.368	0.20	0.45	0.06
S^G (W/O CKP)	S^G (W/ CKP)	S^G_f	$S^G_{f\_c}$
8.18	0.12	0.77	0.02

VIII. DISCUSSION

In this section, we further discuss the use of our reliability-aware models to demonstrate the benefits of fast recovery and proactive failure prevention via process migration.

A. Fast Recovery

The occurrences of failures force the application to wait for recovery, which ranges from a couple of hours to nearly 100 hours [1]. Long recovery time can significantly impact application performance. Fast recovery can reduce MTTR (Mean-Time-To-Recovery), thus mitigates the impact of failures on scalability.

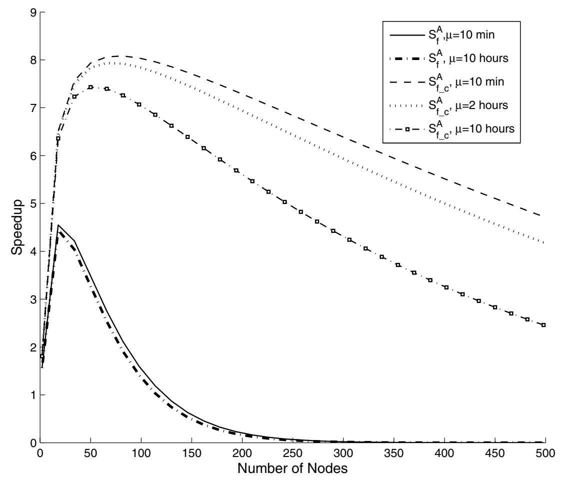

Fig. 5. Fast recovery mitigates the impact of failures on S^A_f and $S^A_{f\_c}$ with varying MTTRs. $\alpha = 0.9$, workload=2000 hours, $\lambda = \frac{1}{7500}$ hours.

To examine the benefits of fast recovery, we vary the MTTR value from 10 minutes to 10 hours (see Figure 5 and 6). As shown in these Figures, without checkpointing fast recovery can not significantly improve application scalability. Due to the lack of fault tolerance support, the work loss caused by failures can significantly impact application scalability, and fast recovery cannot help much in this case. With checkpointing, the maximal $S^A_{f\_c}$ is improved from 7.4 to 8.1, and the maximal $S^G_{f\_c}$ is improved from 103.5 to 189.9 as μ is decreased from 20 hours to 10 minutes. Based on Equation 8 and 12, both $S^A_{f\_c}$ and $S^G_{f\_c}$ exponentially decrease with the growth of μ. The results demonstrate that fast recovery with checkpointing can significantly improve application performance and scalability.

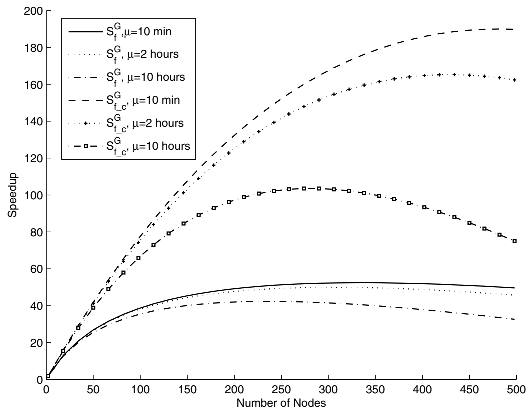

Fig. 6. Fast recovery mitigates the impact of failure on S_f^G and $S_{f\_c}^G$ with varying MTTRs. $\alpha = 0.9$, workload=200 hours, $\lambda = \frac{1}{7500}$ hours.

B. Failure Prediction

Failure prediction is a process to tell whether a failure event will occur in the near future [22]. With failure prediction in hand, proactive actions such as process migration [36] can prevent failure experiencing and avoid rollbacks. Nevertheless, prediction misses and false alarms are commonly associated with failure predictors. The prediction misses can lead to performance damage, whereas false alarms can invoke unnecessary proactive action overhead. In this set of experiments, we assume that the process migration overhead is two times of checkpoint overhead, e.g., $O_{pm} = 2O_{ckp}$ [35]. Similar method can be used to analyze live migration where migration overhead is small.

Prediction accuracy is generally measured by two metrics: *precision* and *recall*. *Precision* is proportion of correct predictions to all the predictions (i.e., $\frac{TP}{TP+FP}$), and *recall* is the proportion of correct predictions to the number of failures (i.e., $\frac{TP}{TP+FN}$). Here TP is true positive, FP is false positive, and FN is false negative. A good predictor engine should obtain a high value for both metrics [35].

Given the failure rate λ_a, and prediction accuracy (i.e., precision and recall), the failure rate is reduced to $\hat{\lambda}_a = (1 - recall)\lambda_a$. The expected number of failures is $\lambda_a \times executiontime$. Then the number of true positives is $TP = recall \times \lambda_a \times executiontime$ and the number of all the positive predictions is $TP + FN = \frac{recall \times \lambda_a \times executiontime}{precision}$. Suppose every positive prediction will trigger a process migration, the total migration overhead is $2O_{ckp} \times \frac{recall \times \lambda_a \times executiontime}{precision}$. Thus, the expected execution time is as follows:

$$E(T_{f\_cm}(W)) = \frac{e^{\mu\hat{\lambda}_a}}{\hat{\lambda}_a}(e^{(\hat{\tau}+O_{ckp})\hat{\lambda}_a} - 1)\frac{W}{\hat{\tau}}(1 + \frac{recall \times 2\lambda_a O_{ckp}}{precision}), \tag{14}$$

where $\hat{\tau}$ is the optimal checkpoint interval based on failure rate $\hat{\lambda}_a$.

Based on Equation 14, we can derive the following models from $S_{f\_c}^A$ and $S_{f\_c}^G$:

$$S_{f\_cm}^A = \frac{1}{\frac{e^{\mu\hat{\lambda}_a}}{\hat{\lambda}_a}(e^{(\hat{\tau}+O_{ckp})\hat{\lambda}_a} - 1)\frac{P(1-\alpha)+\alpha}{\hat{\tau}P}(1+\frac{recall \times 2\lambda_a O_{ckp}}{precision})}, \tag{15}$$

where $O_{ckp} = a + bP$.

$$S_{f\_cm}^G = 1 - \alpha + \frac{P}{\frac{e^{\mu\hat{\lambda}_a}}{\hat{\lambda}_a}(e^{(\hat{\tau}+O_{ckp})\hat{\lambda}_a} - 1)\frac{1}{\hat{\tau}} + \frac{recall \times 2\lambda_a O_{ckp}}{precision}} -(1 - \alpha)P, \tag{16}$$

where $O_{ckp} = aP + bP$.

To study how prediction accuracy impacts application scalability, we keep one metrics as 0.75 and shift the other one from 0.5 to 0.9. As shown in Figure 7, failure prediction with process migration can boost the maximal achievable fixed-size and fixed-time speedup. Another observation is that recall has more impact on application performance than precision. For example, the maximal $S_{f\_c}^A$ is improved from 8.26 to 8.81 as recall increases from 0.5 to 0.9, whereas the maximal $S_{f\_c}^A$ only increases 0.06 as precision grows up from 0.5 to 0.9. The main reason is that, high recall can not only prevent work loss but also reduce the frequency of checkpointing; whereas high precision only reduces unnecessary process migration overhead.

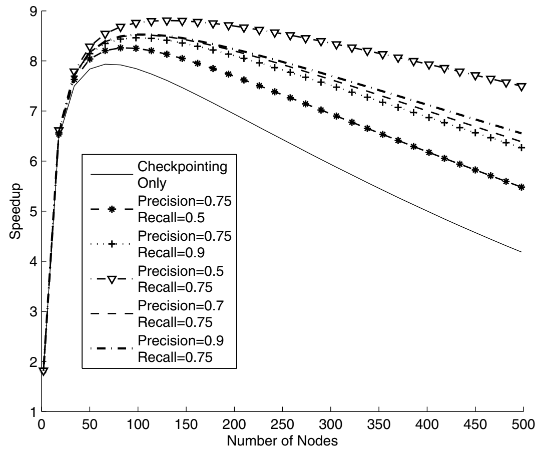

Fig. 7. Failure prediction with process migration mitigates the impact of failures on $S_{f\_c}^A$ and $S_{f\_m}^A$. $\alpha = 0.9$, workload=2000 hours,$\lambda = \frac{1}{7500}$ hours, $\mu = 2$ hours.

IX. Conclusions

In this paper, we have presented several reliability-aware models to extend Amdahl's law and Gustafson's law by considering the impact of failures and fault tolerance techniques. Trace-based simulations with failure logs collected from production systems have demonstrated that our models can better represent application performance and scalability in the presence of failures. Further, we have also used the models to demonstrated the benefits of fast recovery and proactive failure prevention via process migration.

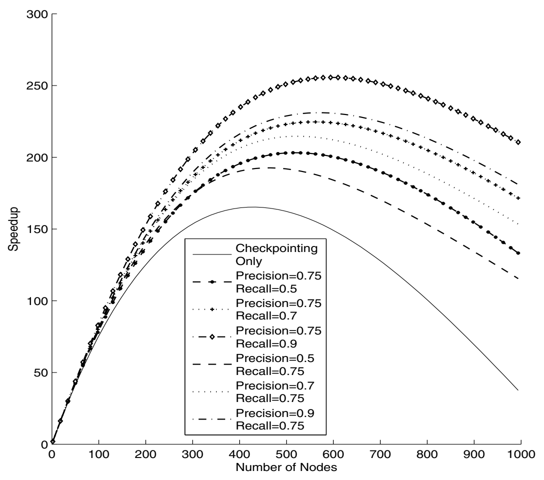

Fig. 8. Failure prediction with process migration mitigates the impact of failures on $S^G_{f\_c}$ and $S^G_{f\_m}$. $\alpha = 0.9$, workload=200 hours,$\lambda = \frac{1}{7500}$ hours, $\mu = 2$ hours.

Our study has some limitations that remain as our future work. First, in addition to checkpointing, we plan to incorporate other fault tolerance technologies, such as replication [14], live migration [36], and cooperative checkpointing [31] into our models. Next, extensive experiments will be conducted to evaluate our reliability-aware models with the failure logs collected from various large-scale systems.

ACKNOWLEDGMENT

Zhiling Lan is supported by US NSF grants CNS-0834514, CNS-0720549, and CCF-0702737. The authors like to thank Yong Chen and Hui Jin from the Scalable Computing Software Laboratory at IIT for their valuable discussions. We would like to thank Prof. Bianca Schroeder for helping us on the LANL failure logs.

REFERENCES

[1] A. Oliner and J. Stearly, "What Supercomputers Say: A Study of Five System Logs," *Proc. of DSN*, 2007.
[2] B. Schroeder and G. Gibson, "A Large-scale Study of Failures in High-performance-computing Systems," *Proc. of DSN*, 2006.
[3] G. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *Proc. of AFIPS Spring Joint Computer Conference*, 1967.
[4] J. Gustafson, "Reevaluating Amdahl's law," *Communications of the ACM*, 31(5):532-533,1988.
[5] D. Mogilevsky, G. Koenig, and W. Yurcik, "Cluster Survivability with ByzwATCH: A Byzantine Hardware Fault Detector for Parallel Machines with Charm++", *Proc. of the 2nd Workshop on High Performance Computing Reliability Issues*, 2006.
[6] Y. Tanaka, H. Takemiya, S. Sekiguchi, S. Ogata, A. Nakano, R. Kalia, and P. Vashishta, "Adaptive Grid-enabled SIMOX Simulation on Japan-US Grid Testbed", *Proc. of TeraGrid*, 2006.
[7] Top500 supercomputing sites. http://top500.org/.
[8] D. Reed, C. Lu, and C. Mendes, "Big systems and big reliability challenges," *Proc. of Parallel Computing*, 2003.
[9] T. Lin and D. Siewiorek, "Error log analysis: statistical modeling and heuristic trend analysis," *IEEE Trans. on Reliability*, 39(4):419-432, 1990.
[10] X. Sun and L. Ni, "Another View on Parallel Speedup," *Proc. of Supercomputing*, 1990.
[11] Y. Chen, X. Sun, and M. Wu, "Algorithm-System Scalability of Heterogeneous Computing," *Journal of Parallel and Distributed Computing*, 68(11):1403-1412, 2008.
[12] X. Sun, Y. Chen, and M. Wu, "Scalability of Heterogeneous Computing," *Proc. of ICPP*, 2005.
[13] R. Ge and K. Cameron , "Power-Aware Speedup," *Proc. of IPDPS*, 2007.
[14] M. Wu, X. Sun, and H. Jin, "Performance under Failure of High-End Computing," *Proc. of SuperComputing*, 2007.
[15] J. Daly, "A Higher Order Estimate of the Optimum Checkpoint Interval for Restart Dumps," *Future Generation Computer Systems*, 22(3): 303-312, 2006.
[16] S. Garg, Y. Huang, C. Kintala, and K. Trivedi, "Minimizing Completion Time of a Program by Checkpointing and Rejuvenation," *Proc. of SIGMETRICS*, 1996.
[17] J. Plank and M. Thomason, "Processor allocation and checkpoint interval selection in cluster computing systems," *Journal of Parallel and Distributed Computing*, 61(11): 1570-1590, 2001.
[18] J. Plank and W. Elwasif, "Experimental Assessment of Workstation Failures and Their Impact on Checkpointing Systems," *Proc. of FTCS*, 1998.
[19] E. Elnozahy and J. Plank, "Checkpointing for Peta-Scale Systems: A Look into the Future of Practical Rollback-Recovery," *IEEE Trans. on Dependable and Secure Computing*, 1(2):97-108, 2004.
[20] L. Wang, K. Pattabiraman, Z. Kalbarczyk, and R. Iyer, "Modeling Coordinated Checkpointing for Large-Scale Supercomputers," *Proc. of DSN*, 2005.
[21] Z. Lan and Y. Li, "Adaptive Fault Management of Parallel Applications for High Performance Computing," *IEEE Trans. Computers*, 57(12): 1647-1660, 2008.
[22] J. Gu, Z. Zheng, Z. Lan, J. White, E. Hocks, and B-H. Park, "Dynamic Meta-Learning for Failure Prediction in Large-scale Systems: A Case Study", *Proc. of ICPP*, 2008.
[23] N. Gottumukkala, C. Leangsuksun, R. Nassar, M. Paun, D. Sule, and S. Scott, "Reliability Aware Optimal K Node of Parallel applications in Large Scale HPC Systems," *Proc. of High Availability and Performance Computing Workshop*, 2008.
[24] J. Young, "A First Order Approximation to the Optimal Checkpoint Interval," *Comm. ACM*, 17(9): 530-531, 1974.
[25] Y. Zhang, M. Squillante, A. Sivasubramaniam, and R. Sahoo, " Performance implications of failures in large-scale cluster scheduling," *Proc. of Workshop on JSSPP, SIGMETRICS*, 2004.
[26] A. Oliner, R. Sahoo, J. Moreira, and M. Gupta, "Performance Implications of Periodic Checkpointing on Large-scale Cluster Systems," *Proc. of IPDPS*, 2005.
[27] S. Arunagiri, J. Daly, P. Teller, S. Seelam, R. Oldfield, M. Varela, and R. Riesen, "Opportunistic Checkpoint Intervals to Improve System Performance," *Technical Report UTEP-CS-08-24*, 2008.
[28] A. Bouteiller, P. Lemarinier, G. Krawezik, and F. Cappello, "Improved message logging versus improved coordinated checkpointing for fault tolerant MPI," *Proc. of Cluster*, 2003.
[29] V. Kumar and A. Gupta, "Analysis of scalability of parallel algorithms and architectures: a survey," *Proc of ICS*, 1991.
[30] D. Woo and H. Lee, "Extending Amdahl's law for energy-efficient computing in the many-core era," *IEEE Computer*, 41(12):24-31, 2008.
[31] A. Oliner, L. Rudolph, and R. Sahoo, "Cooperative checkpointing: A robust approach to large-scale systems reliability," *Proc. of ICS*, 2006.
[32] R. Oldfield, "Investigating lightweight storage and overlay networks for fault tolerance," *Proc. of High Availability and Performance Computing Workshop*, 2006.
[33] F. Petrini, K. Davis, and J. Sancho, "System-level faulttolerance in large-scale parallel machines with buffered coscheduling," *Proc. of IPDPS*, 2004.
[34] V. Nicola, "Checkpointing and modelling of program execution time. Software Fault Tolerance," John Wiley and Sons, 1995.
[35] Y. Li and Z. Lan, "A Fast Recovery Mechanism for Checkpointing in Networked Environments," *Proc. of DSN*, 2008.
[36] C. Wang, F. Mueller, C. Engelmann, and S. Scott, "Proactive process-level live migration in HPC environments," *Proc. of Supercomputing*, 2008.
[37] Los Alamos National Laboratory, Operational Data to Support and Enable Computer Science Research, http://institute.lanl.gov/data/lanldata.shtml.
[38] J. Daly, L. Pritchett-Sheats, and S. Michala, "Application MTFE vs Platform MTBF: A Fresh Perspective on System Reliabilty and Application Throughput for Computations at Scale," *Proc. of CCGRID*, 2008.