Reliability-Aware Speedup Models for Parallel Applications with Coordinated Checkpointing/Restart

Ziming Zheng, *Member, IEEE Computer Society*, Li Yu, *Student Member, IEEE*, and Zhiling Lan, *Senior Member, IEEE Computer Society*

Abstract—Speedup models are powerful analytical tools for evaluating and predicting the performance of parallel applications. Unfortunately, the well-known speedup models like Amdahl's law and Gustafson's law do not take reliability into consideration and therefore cannot accurately account for application performance in the presence of failures. In this study, we enhance Amdahl's law and Gustafson's law by considering the impact of failures and the effect of coordinated checkpointing/restart. Unlike existing analytical studies relying on Exponential failure distribution alone, in this work we consider both Exponential and Weibull failure distributions in the construction of our reliability-aware speedup models. The derived reliability-aware models are validated through trace-based simulations under a variety of parameter settings. Our trace-based simulations demonstrate these models can effectively quantify failure impact on application speedup. Moreover, we present two case studies to illustrate the use of these reliability-aware speedup models.

Index Terms—Speedup, reliability, Amdahl's law, Gustafson's law, analytical modeling

1 Introduction

1.1 Motivations

Computing power has experienced tremendous growth over the past decades. Production systems today already contain hundreds of thousands of processors [6]. Exa-scale systems are expected to become available in less than a decade [37], which are projected to consist of millions of processing units. For parallel applications running on these extreme scale systems, speedup is a critical metric [38]. It not only measures the inherent parallelism of an application, but also provides an important guidance of application performance as system size increases.

Amdahl's law [3] and Gustafson's law [4] are two well-known speedup models. They are used to estimate the performance of parallel applications at scales: Amdahl's law focuses on parallel execution relative to the serial execution under the assumption of a fixed workload (i.e., fixed-sized speedup), while Gustafson's law emphasizes the amount of workload that can be finished in a fixed time (i.e., fixed-time speedup). *Both models implicitly assume that the application 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 ultra-reliable components [7]. As a result, parallel applications are very difficult to make any forward progress because of failures [5]. The occurrences of failures force the application to wait for system recovery, which may take upto nearly 100 hours [1], and then roll back to the beginning or the latest checkpoint. Due to the impact of failures, application speedup in the presence of failures is different from its speedup in an ideal failure-free environment [16].

Despite the importance of reliability-aware speedup modeling, only a few analytical studies have been conducted to understand application performance under failures [22], [12], [11], [16]. Reliability analysis is a hard problem, especially in parallel systems with unprecedented scale and complexity. To simply the problem, existing studies typically assume a constant failure rate and adopt Exponential failure distribution in their modeling process. Nevertheless, recent studies on field data from production systems clearly show that Weibull distribution with decreasing failure rate provides a better goodness of fit than Exponential distribution [2], [8], [46], [51]. The key challenge is that Weibull distribution is hard to study analytically due to its complicated and dynamic failure rate nature [40]. Furthermore, validating reliability-aware speedup models is also difficult due to the scarcity of failure data from production supercomputers.

To address the aforementioned problems, in this paper we present a set of *reliability-aware speedup models* to extend Amdahl's law and Gustafson's law by considering failure impact. The goal of this work is to provide more accurate measurement of application speedup in a practical failure-prone environment. More importantly, we consider both Exponential and Weibull failure distributions in the model construction. To tackle the challenge of dynamic failure rate inherent in Weibull distribution, we derive the lower and upper bounds of application speedup under Weibull failure distribution.

Checkpointing/restart is a well-known fault tolerance method to mitigate the impact of failures. In particular,

• *The authors are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616.*
E-mail: {zzheng11, lyu17, lan}@iit.edu.

Manuscript received 9 Mar. 2013; revised 24 Jan. 2014; accepted 16 Mar. 2014. Date of publication 14 Apr. 2014; date of current version 8 Apr. 2015.
recommended for acceptance by Z. Tari.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2317182

coordinated checkpointing is a widely used technique in the field of high performance computing due to its simplicity [26], [56], [58]. It periodically coordinates application processes and stores a consistent snapshot of the application. In case of failure from any process, all the processes roll back to the last checkpoint for recovery [30]. Although coordinated checkpointing can effectively reduce work loss, it also introduces some inevitable costs to the computation such as checkpointing overhead and recovery cost [47], [23]. As shown in the studies [16], [48], for an application running on a large-scale system, its overall checkpoint/restart overhead could take more than 50 percent of its execution due to high failure rate and high checkpoint frequency. In this study, we also quantify coordinated checkpointing effects on application speedup in the design of reliability-aware models. Furthermore, our models can be extended to study other fault tolerance technologies such as uncoordinated checkpoint with message logging [56] and hybrid checkpointing protocols [58], [60].

To comprehensively and realistically assess our models, we conduct trace-based simulations using the real failure traces from production supercomputers [35], [51]. We develop an event-driven simulator to simulate parallel execution. The simulator is designed to parse discrete events from the failure traces and also provides a flexible interface to allow various configurations of application parameters. Our experiments are structured to assess model accuracy under different application workloads, parallel fractions and computing scales. Together, these experiments provide a comprehensive evaluation of model accuracy under various configurations. Our results indicate that the newly derived speedup models can more accurately reveal application performance in a practical failure-present environment than the original Amdahl's and Gustafson's laws. Further, our results clearly show that the reliability-aware models based on Weibull distribution greatly outperform those based on Exponential distribution.

In this study we also present two case studies to illustrate the usefulness of the aforementioned speedup models. One is to identify the optimal computing scale and the maximal speedup for an application under a given system, and the other is to project node reliability and checkpoint overhead that are needed in future exascale systems in order to maintain good computing efficiency.

The rest of the paper is organized as follows. Section 2 briefly discusses related studies. Background and assumptions are presented in Section 3. In Section 4, we present reliability-aware application performance models. Based on the models listed in Section 4, the enhanced Amdahl's models and Gustafson's models are derived in Section 5 and 6 respectively. Model validation and model usage are listed in Section 7 and Section 8. We discuss the extension of our model for new fault tolerance techniques in Section 9. Finally, we conclude the paper in Section 10.

2 Related Work

Speedup has been studied for decades from various aspects. Amdahl's [3] and Gustafson's [4] are two well-known models. Amdahl's model is for fixed-size problems (strong scaling), while Gustafson's model is for fixed-time problems (weak scaling). Both models have made tremendous impact on parallel and distributed computing. A number of analytical studies have extended these basic models to examine application scalability under various system constraints. Kumar and Gupta develop scalability models for a specific parallel architecture [27]. Yero and Henriques analyze the speedup and scalability of Master-Slave applications on heterogeneous clusters [39]. Woo and Lee extend the Amdahl's law for many-core architectures [28]. Jogalekar and Woodside introduce an adaptation of scalability for the distributed system era [42]. Sun and Ni develop memory-constrained speedup in [9]. In [10], a power-aware speedup is proposed to predict the scaled execution time and power consumption. However, neither of them studies the impact of failures—which is an important aspect as systems and applications scale to very large sizes. This work is focused on extending Amdahl's model and Gustafson's model by considering system failures and resilience mechanisms. Further, our models can be easily integrated with existing studies to analyze more complicated scenarios.

Analytical modeling of failure and checkpointing on application performance has been presented in [22], [12], [11], [45], [17], and most studies assume failure arrivals follow a Poisson's process, i.e., the inter-arrival times of failures follow an identical Exponential distribution. In [22], Young derives the optimal checkpointing interval via the first order estimation. Daly improves the model by using higher order estimation and derives the expected completion time with checkpointing in [12]. Jones et. al. use Daly's model to study the impacts of failures and checkpointing on application efficiency in [43]. In [11] and [45],the authors 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. While Exponential distribution is commonly used for modeling, recent studies have shown that Weibull distribution provides a better goodness of fit [2], [8], [46]. There are very few studies considering Weibull distribution. In [41], Liu et. al. use a stochastic renewal reward process to study optimal checkpointing interval based on Weibull distribution. In [40], Gottumukkala et al. study system-wide time-to-failure distribution under Weibull failure arrivals. Distinguishing from existing analytical modeling studies, in this paper we derive reliability-aware speedup models under both Exponential and Weibull failure distributions, and examine these models under the scenarios where checkpointing may or may not be used.

Experimental studies of failure impact on parallel computing have been discussed in [16], [21], [44]. 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 [16]. In [21], a reliability-aware resource allocation algorithm is presented to select an optimal number of nodes for the execution of an application. In [44], Taerat et. al analyze Blue Gene/L logs in a period of six months and study the job interruptions through simulation. Unlike these studies, this paper develops analytical models to explicitly express application speedup in the presence of failures, using both Exponential and Weibull failure distributions.

TABLE 1
Nomenclature

Symbol	Description
N	Number of computing nodes (i.e., the computing scale)
R_i	Reliability function of node i
R_{app}	Reliability function of an application running on N-node system
λ_i	Failure arrival rate of node i (per hour) in Exponential distribution
λ_{app}	Failure arrival rate of an application running on N-node system (per hour)
β, η	shape and scale parameters of Weibull distribution
μ	Mean-Time-To-Recover(MTTR) (hour)
W	Application workload (hour)
W_p	The parallel workload (hour)
T_W	Application execution time for a given workload of W, $T_W = W$ in the case of failure-free execution
α	The fraction of the application that can be parallelized
O_c, τ, δ	Checkpoint overhead, interval, segment, and $\delta = \tau + O_c$
SA	Amdahl's speedup model
SG	Gustafson's speedup model
S^f	Speedup model (SA or SG) in the presence of failures
S_{exp}	Speedup model (SA or SG)under Exponential failure distribution, without checkpointing
S_{wb}	Speedup model (SA or SG) under Weibull failure distribution, without checkpointing
$S_{exp,ckp}$	Speedup model (SA or SG) under Exponential failure distribution, with checkpointing
$S_{wb,ckp}$	Speedup model (SA or SG) under Weibull failure distribution, with checkpointing

3 Assumptions

For the development of reliability-aware speedup models, we make several assumptions based on existing studies.

First, the time interval between failures on each node i follows a certain distribution with a probability density function of $f_i(t)$. In this paper, we study two commonly used distributions, i.e., Exponential and Weibull. Their probability density functions are as follows:

$$f_i(t) = \begin{cases} \lambda_i e^{-\lambda_i t} & \textit{Exponential} \\ \frac{\beta_i}{\eta_i}\left(\frac{t}{\eta_i}\right)^{\beta_i - 1} e^{(-t/\eta_i)^{\beta_i}} & \textit{Weibull.} \end{cases} \quad (1)$$

In Exponential density function, the constant $\lambda_i > 0$ is the constant failure arrival rate of node i. In Weibull density function, $\eta_i > 0$ is the scale parameter and $\beta_i > 0$ is the shape parameter. We assume that all the nodes on a system have the same shape parameter, i.e., $\beta_1 = \cdots = \beta_N = \beta$, and $\beta < 1$. This assumption is based on the observations in [2], [46].

Second, we assume the system adopts a space sharing mechanism for job scheduling, where each compute node is dedicated to one application process once it is allocated to an application. Most supercomputing centers adopt space sharing mechanism for their job scheduling. Further, we assume a fail-stop mode [30], where the failure of any single node interrupts the entire application. Many tightly-coupled parallel applications like MPI applications fall into this category.

Third, failure repair time follows a general distribution with a mean of μ [11]. Further, μ is not sensitive to system size in homogeneous systems, which is based on the observation in [2].

Lastly, the overhead of a single coordinated checkpointing O_c is a linear function of application size N (i.e., number of nodes used for running the application), and is much less than application workload [47], [48], [12]. The overhead consists of two parts, i.e., I/O overhead and message passing overhead [18]. For augmented Amdahl's models, since the problem size does not change, we assume I/O overhead is fixed [18], [14], i.e., $O_c = a + bN$. For augmented Gustafson's models, 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. Hence the overhead is $O_c = aN + bN$ [25], [30].

Table 1 lists a set of nomenclatures that will be frequently used in the rest of the paper. Unless otherwise specified, in the rest of the paper, the term of checkpointing indicates coordinated checkpointing.

4 Expected Application Performance

Given a system with N nodes, we assume the inter-arrival times of failures for each node are independent and identically distributed (iid) and the failure of a single node interrupts the entire application (the fail-stop mode). Let W be the failure free execution time of an application running on N nodes, then application reliability function at time T is given as

$$R_{app}(T) = \prod_{i=1}^{N} \frac{1 - \int_0^{t_i+T} f_i(t)dt}{1 - \int_0^{t_i} f_i(t)dt}, \quad (2)$$

where t_i is the time of the last failure. The expected application execution time in the presence of failure $E^f(T_W)$, is different depending on whether checkpointing is adopted or not. For the systems without checkpointing support, it can be derived based on a model from [13] as below.

$$E^f(T_W) = W + \frac{\mu(1 - R_{app}(W))}{R_{app}(W)} - \frac{\int_0^W td(R_{app}(t))}{R_{app}(W)}. \quad (3)$$

In an environment with checkpointing support, we adopt a segment based model presented in [45], where the execution time of an application is divided into a set of checkpointing segments. Each segment is a period of time between two consecutive checkpoints and its length can be represented as $\delta = \tau + O_c$, where τ is the checkpoint interval and O_c is the checkpoint overhead. In each segment, once a failure occurs, the application needs to roll back to the last checkpoint (i.e., the beginning of the current segment). Assume there are k_f failures occurring during a segment and each failure results in a rework cost X and a system downtime Y, the expected completion time of a segment $E(T_\delta)$ can be defined as

$$E(T_\delta) = \delta + k_f * (E(X) + E(Y)), \tag{4}$$

where

$$E(X) = \int_{t=0}^{\delta} t \frac{f_i(t)}{\int_0^{\delta} f_i(t)dt} dt$$

and $E(Y) = \mu$. As we need at least $\lfloor \frac{W}{\tau} \rfloor$ segments to finish the whole application, the expected execution time of the application with checkpointing is estimated as

$$E^f_{ckp}(T_W) = \left\lfloor \frac{W}{\tau} \right\rfloor * E(T_\delta). \tag{5}$$

4.1 Exponential Distribution

In the case of Exponential distribution for failure arrivals, based on Equations (1) and (2), due to the memoryless property of this distribution, i.e., $P(t > t_i + T | t > t_i) = P(t > T)$, $R^{exp}_{app}(T)$ is independent of t_i and can be calculated as

$$R^{exp}_{app}(T) = e^{-T\lambda_{app}}, \tag{6}$$

where $\lambda_{app} = \sum_{i=1}^{N} \lambda_i$. If all the nodes have the same failure rate λ, it can be simplified as $\lambda_{app} = N\lambda$ [31], [36].

Based on Equation (3) and (6), we can obtain the expected application performance without checkpointing

$$E^f_{exp}(T_W) = \left(\mu + \lambda^{-1}_{app}\right)\left(e^{\lambda_{app}W} - 1\right). \tag{7}$$

In case of checkpointing support, based on Equation (4) and Equation (5), we can estimate the expected application execution time as follows:

$$E^f_{exp,ckp}(T_W) = \frac{e^{\mu\lambda_{app}}}{\lambda_{app}}\left(e^{\delta\lambda_{app}} - 1\right)\frac{W}{\tau}, \tag{8}$$

where τ is the checkpoint interval, $\delta = \tau + O_c$ is the checkpoint segment, and $O_c = a + bN$ is the checkpoint overhead. According to [12], [30], the optimal checkpoint interval can be approximated as follows:

$$\tau = \begin{cases} \sqrt{\frac{2O_c}{\lambda_{app}}}\left[1 + \frac{1}{3}\left(\frac{O_c\lambda_{app}}{2}\right)^{\frac{1}{2}} + \frac{1}{9}\left(\frac{O_c\lambda_{app}}{2}\right)\right] - O_c & O_c < \frac{2}{\lambda_{app}} \\ \frac{1}{\lambda_{app}} & O_c \geq \frac{2}{\lambda_{app}}. \end{cases} \tag{9}$$

4.2 Weibull Distribution

In the case of Weibull distribution for failure arrivals, based on Equation (1) and (2), $R^{wb}_{app}(T)$ can be derived as

$$R^{wb}_{app}(T) = e^{\sum_{i=1}^{N} \frac{t_i^{\beta} - (t_i + T)^{\beta}}{\eta_i^{\beta}}}. \tag{10}$$

Unlike the case of Exponential distribution, R^{wb}_{app} depends on the time of the last failure t_i, suggesting a change once failures occur in one or more nodes [40]. To tackle the problem, we derive a lower bound and an upper bound for R^{wb}_{app}.

Theorem 1. *For Weibull failure distribution with $\beta_i < 1$ ($i = 1$ to N), the lower bound and the upper bound of R^{wb}_{app} are*

$$\underline{R}^{wb}_{app}(T) = e^{-N\left(\frac{T}{\underline{\eta}}\right)^{\beta}} \quad (lower\ bound) \tag{11}$$

$$\overline{R}^{wb}_{app}(T) = e^{N - N\left(\frac{\overline{\eta}+T}{\overline{\eta}}\right)^{\beta}}, \quad (upper\ bound) \tag{12}$$

where $\underline{\eta} = \min_i(\eta_i)$ and $\overline{\eta} = \max_i(\eta_i)$. If $\beta_i = 1$ and $\underline{\eta} = \overline{\eta}$, $\underline{R}^{wb}_{app}(T) = \overline{R}^{wb}_{app}(T)$

Proof. For the lower bound, we note the inequality $t_i^{\beta} - (t_i + T)^{\beta} \geq -T^{\beta}$ holds when $t_i \geq 0$ and $0 < \beta < 1$, therefore $R^{wb}_{app}(T) \geq e^{-\sum_{i=1}^{N} \frac{T^{\beta}}{\eta_i^{\beta}}} \geq e^{-N\left(\frac{T}{\underline{\eta}}\right)^{\beta}}$.

For the upper bound, we assume a situation in which all the nodes survive until a failure occurs at $t_i = \eta_i$, then system reliability becomes $R^{wb}_{app}(T) = e^{\sum_{i=1}^{N} \frac{\eta_i^{\beta} - (\eta_i + T)^{\beta}}{\eta_i^{\beta}}}$. To achieve this situation, all the nodes need to be failure-free during the time interval from $t_i = 0$ to η_i, with the probability $R_i(\eta_i)^N = e^{-N(\eta_i/\eta_i)^{\beta}} = e^{-N}$ decreasing to 0 as the growth of N. In other words, the probability that R^{wb}_{app} is as high as $e^{\sum_{i=1}^{N} \frac{\eta_i^{\beta} - (\eta_i + T)^{\beta}}{\eta_i^{\beta}}}$ is very low and decreases as the scale grows. As a result, an approximation for the upper bound is $e^{N - N\left(\frac{\overline{\eta}+T}{\overline{\eta}}\right)^{\beta}} \geq e^{\sum_{i=1}^{N} \frac{\eta_i^{\beta} - (\eta_i + T)^{\beta}}{\eta_i^{\beta}}}$. The reason of choosing $t_i = \eta_i$ is to simplify the mathematical expression.

If $\beta_i = 1$, $e^{\sum_{i=1}^{N} \frac{\overline{\eta}^{\beta_i} - (\overline{\eta} + T)^{\beta_i}}{\eta_i^{\beta_i}}} = e^{\sum_{i=1}^{N} \frac{\overline{\eta} - (\overline{\eta} + T)}{\eta_i}} = e^{\sum_{i=1}^{N} \frac{T}{\eta_i}} = e^{\sum_{i=1}^{N} \frac{T^{\beta_i}}{\eta_i^{\beta_i}}}$. Therefore if $\underline{\eta} = \overline{\eta}$, $\underline{R}^{wb}_{app} = \overline{R}^{wb}_{app}$. □

To estimate the expected execution time without checkpointing, we replace (3) with (11) and (12) as follows:

$$\underline{E}^f_{wb}(T_W) = W - \mu + \frac{\mu}{e^{-\underline{\varphi}}} + \frac{(\underline{\eta}/N^{\xi})\gamma(1+\xi, \underline{\varphi})}{e^{-\underline{\varphi}}} \tag{13}$$

$$\begin{aligned} \overline{E}^f_{wb}(T_W) = W - \mu + \frac{\mu}{e^{N-\overline{\varphi}}} \\ + \frac{\overline{\eta}/N^{\xi}(\gamma(1+\xi, \overline{\varphi}) - \gamma(1+\xi, N)) + \overline{\eta}e^{-\overline{\varphi}} - \overline{\eta}e^{-N}}{e^{-\overline{\varphi}}}, \end{aligned} \tag{14}$$

where $\xi = 1/\beta$, $\underline{\varphi} = N(W/\underline{\eta})^{\beta}$, $\overline{\varphi} = N(1 + W/\overline{\eta})^{\beta}$, and $\gamma(x, y) = \int_0^y t^{x-1}e^{-t}dt$ is the lower incomplete gamma function with parameter x and y [49].

To estimate the expected execution time with checkpointing, we integrate (11) and (12) into (5). We use $k_f = \frac{\delta}{MTBF}$ to represent the number of failures occurring in one checkpointing segment, where $MTBF$ is the mean time between failures. Under Weibull failure distribution, $MTBF =$

$\eta\Gamma(1+\xi)$ and $\Gamma(\cdot)$ is the gamma function. The results are shown below.

$$\underline{E}^{f}_{wb,ckp}(T_W) = \underline{\delta}\left(1 + \frac{\mu}{(\underline{\eta}/N^{\xi})\Gamma(1+\xi)} + \frac{\gamma(1+\xi,\underline{\psi})}{(1-e^{-\underline{\psi}})\Gamma(1+\xi)}\right)\frac{W}{\underline{\tau}} \tag{15}$$

$$\overline{E}^{f}_{wb,ckp}(T_W) = \overline{\delta}\left(1 + \frac{\mu}{\frac{\overline{\eta}e^{N}}{N^{\xi}}\Gamma(1+\xi,N) - \overline{\eta}} + \frac{\frac{\overline{\eta}e^{N}}{N^{\xi}}(\gamma(1+\xi,\overline{\psi}) - \gamma(1+\xi,N)) + \overline{\eta}e^{N-\overline{\psi}} - \overline{\eta}}{(1-e^{N-\overline{\psi}})\left(\frac{\overline{\eta}e^{N}}{N^{\xi}}\Gamma(1+\xi,N) - \overline{\eta}\right)}\right)\frac{W}{\overline{\tau}}, \tag{16}$$

where $\underline{\psi} = N(\delta/\underline{\eta})^{\beta}$, $\overline{\psi} = N(1+\delta/\overline{\eta})^{\beta}$, and $\Gamma(x,y) = \int_y^{\infty} t^{x-1}e^{-t}dt$ is the upper incomplete gamma function with parameter x and y [49]. The optimal checkpoint interval $\underline{\tau}$ and $\overline{\tau}$ can be identified via numeric method like Newton's method.

5 Reliability-Aware Amdahl's Models

According to Amdahl's law, application workload per process is $W_p = (1-\alpha)W + \alpha W/N$, where W is the entire application workload and α is the fraction of the application that can be parallelized. Hence the fixed-size speedup is defined as:

$$SA = \frac{W}{(1-\alpha)W + \alpha W/N} = \frac{N}{N(1-\alpha)+\alpha}. \tag{17}$$

5.1 Exponential Distribution

In the case of Exponential failure distribution, we can derive the following reliability-aware model based on Equation (7) and Equation (8)

$$SA^{f}_{exp} = \frac{W}{(\mu + \lambda^{-1}_{app})(e^{(1-\alpha+\frac{\alpha}{N})\lambda_{app}W} - 1)}, \tag{18}$$

$$SA^{f}_{exp,ckp} = \frac{\lambda_{app}\tau}{e^{\mu\lambda_{app}}(e^{\delta\lambda_{app}}-1)\left(1-\alpha+\frac{\alpha}{N}\right)}. \tag{19}$$

5.2 Weibull Distribution

In the case of Weibull failure distribution without checkpointing, we estimate the lower bound and upper bound based on (13) and (14) as follows:

$$\underline{SA}^{f}_{wb} = \frac{W}{\underline{E}^{f}_{wb}(T_{W_p})} = \frac{We^{-\underline{\vartheta}}}{e^{-\underline{\vartheta}}(W_p - u) + u + \left(\frac{\underline{\eta}}{N^{\xi}}\right)\gamma(1+\xi,\underline{\vartheta})} \tag{20}$$

and

$$\overline{SA}^{f}_{wb} = \frac{W}{\overline{E}^{f}_{wb}(T_{W_p})} = \frac{We^{N-\overline{\vartheta}}}{e^{N-\overline{\vartheta}}(W_p - u + \overline{\eta}) + u + \overline{\eta} + \frac{\overline{\eta}}{N^{\xi}}\left(\gamma\left(1+\xi, N\left(\frac{W_p}{\overline{\eta}}\right)^{\beta}\right) - \gamma(1+\xi,N)\right)}, \tag{21}$$

where $\underline{\vartheta} = N(W_p/\underline{\eta})^{\beta}$ and $\overline{\vartheta} = N(1+W_p/\overline{\eta})^{\beta}$.

For the models based on checkpointing support, we can obtain the following lower and upper bounds based on Equations (15) and (16):

$$\underline{SA}^{f}_{wb,ckp} = \frac{W}{\underline{E}^{f}_{wb,ckp}(T_{W_p})} = \frac{\underline{\tau}}{\underline{\delta}\left(1 + \frac{\mu}{(\underline{\eta}/N^{\xi})\Gamma(1+\xi)} + \frac{\gamma(1+\xi,\underline{\phi})}{(1-e^{-\underline{\phi}})\Gamma(1+\xi)}\right)\left(1-\alpha+\frac{\alpha}{N}\right)} \tag{22}$$

and

$$\overline{SA}^{f}_{wb,ckp} = \frac{W}{\overline{E}^{f}_{wb,ckp}(T_{W_p})} = \frac{\overline{\tau}}{\overline{\delta}\left(1-\alpha+\frac{\alpha}{N}\right)\left(1 + \frac{\mu}{\frac{\overline{\eta}e^{N}}{N^{\xi}}\Gamma(1+\xi,N)-\overline{\eta}} + \frac{\frac{\overline{\eta}e^{N}}{N^{\xi}}(\gamma(1+\xi,\overline{\phi}) - \gamma(1+\xi,N)) + \overline{\eta}e^{N-\overline{\phi}} - \overline{\eta}}{(1-e^{N-\overline{\phi}})(\frac{\overline{\eta}e^{N}}{N^{\xi}}\Gamma(1+\xi,N)-\overline{\eta})}\right)} \tag{23}$$

where $\underline{\phi} = N(\underline{\delta}/\underline{\eta})^{\beta}$ and $\overline{\phi} = N(1+\overline{\delta}/\overline{\eta})^{\beta}$.

Theorem 2. *SA is a special case of SA^{f}_{exp}, and SA^{f}_{exp} is a special case of SA^{f}_{wb}.*

Proof. For SA^{f}_{exp}, 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 ($\frac{1}{P\lambda} \gg W_p$), based on the first-order Taylor series, we obtain

$$\begin{aligned} SA^{f}_{exp} &= \frac{W}{(P\lambda)^{-1}(e^{(1-\alpha)P\lambda W}e^{\alpha\lambda W} - 1)} \\ &\approx \frac{W}{(P\lambda)^{-1}((1-\alpha)P\lambda W + \alpha\lambda W)} \\ &= SA. \end{aligned}$$

Based on Theorem 1, it is obvious that $\overline{SA}^{f}_{wb} = \underline{SA}^{f}_{wb} = SA^{f}_{exp}$ when $\beta = 1$. □

5.3 Model Analysis

The above models provide two interesting properties about reliability-aware fixed-size speedup.

Property 1. Reliability-aware fixed-size speedups, in case of Exponential or Weibull failure distribution *without checkpointing*, decrease with the growth of application workload.

Property 2. Reliability-aware fixed-size speedups, in case of Exponential or Weibull failure distribution *with checkpointing*, are independent of application workload.

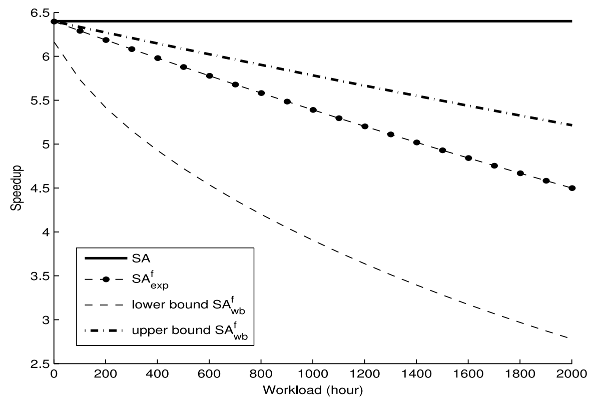


Fig. 1. Comparison of SA, SA^f_{exp}, and SA^f_{wl} under different workloads where computing scale N is 16. The parameters are $\alpha = 0.9$, $\lambda = \eta = \frac{1}{7500}$ hours, $\mu = 0.2$ hours, $\beta = 0.8$.

To illustrate Property 1, Fig. 1 presents reliability-aware fixed-size speedups under different workloads. By comparing Property 1 and 2, we can clearly observe that the use of checkpointing can promote application speedup.

6 Reliability-Aware Gustafson's Models

Different from Amdahl's law, Gustafson's law emphasizes 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 computing nodes, and the rest of the workload does not grow with the number of nodes [4]. Hence it defines fixed-time speedup as follows:

$$SG = \frac{(1-\alpha)W + \alpha WN}{W} = 1 - \alpha + \alpha N. \tag{24}$$

In an ideal failure-free case, Gustafson's law shows that the fixed-time speedup is independent of W (i.e., application workload) and linearly grows with N (i.e., computing scale). In practice, however, as W increases, the application becomes more vulnerable to failures. Considering the impact of failures, we define the *achievable workload* W^* as the workload for user application during W execution.

6.1 Exponential Distribution

In the case of Exponential failure distribution, SG^f_{exp} can be derived from Equation (7) as

$$SG^f_{exp} = 1 - \alpha + \frac{N\ln(\frac{W}{u+\lambda^{-1}_{app}}+1)}{W\lambda_{app}} - (1-\alpha)N. \tag{25}$$

Similarly, we can derive $SG^f_{exp,ckp}$ based on Equation (8) as follows:

$$SG^f_{exp,ckp} = 1 - \alpha + \frac{\tau N \lambda_{app}}{e^{\mu\lambda_{app}}(e^{\delta\lambda_{app}}-1)} - (1-\alpha)N. \tag{26}$$

Theorem 3. *SG is a special case of* SG^f_{exp}.

Proof. When the recovery time can be ignored ($\mu = 0$), each node has the same failure rate ($\lambda_{app} = P\lambda$), and λ_{app} is much larger than the execution time ($\frac{1}{P\lambda} \gg W$), based on the first-order Taylor series we obtain

$$\begin{aligned} SG^f &= 1 - \alpha + \frac{\ln(WP\lambda + 1)}{W\lambda} - P(1-\alpha) \\ &\approx 1 - \alpha + \frac{WP\lambda}{W\lambda} - P(1-\alpha) \\ &= SG. \end{aligned} \tag{27}$$

□

6.2 Weibull Distribution

In the case of Weibull failure distribution without checkpointing, the achievable workloads are derived from the transcendental equations $\underline{E}^f_{wb}(T_W) = W$ and $\overline{E}^f_{wb}(T_W) = W$, which have no analytical solutions. We adopt a numeric method to solve the equations, as shown in Algorithm 1. Here the value of SG^f_{exp} is used as the initial point and the Newton's method is used to search the numeric solutions.

Algorithm 1 Numeric algorithm to obtain augmented Gustafson's model in the case of Weibull failure distribution, without checkpointing

```
Definition: $\underline{g}(T^*_W) \equiv \underline{E}^f_{wb}(T^*_W) - W$, $\overline{g}(T^*_W) \equiv \overline{E}^f_{wb}(T^*_W) - W$, $\varepsilon$ is a positive real number sufficiently close to zero.
Objective: $\underline{SG}^f_{wb}$ and $\overline{SG}^f_{wb}$
  $T^*_{W_s} \leftarrow \frac{N}{\eta}\ln(\frac{W}{u+N/\eta}+1)$
  $T^*_{W_{iter}} \leftarrow T^*_{W_s}$
  while $|\underline{g}(T^*_{W_{iter}})| > \varepsilon$ do
    $T^*_{W_{iter}} \leftarrow T^*_{W_{iter}} - \frac{\underline{g}(T^*_{W_{iter}})}{\underline{g}'(T^*_{W_{iter}})}$
  end while
  $\underline{T_W}^*_{opt} \leftarrow T^*_{W_{iter}}$
  $\underline{SG}^f_{wb} \leftarrow \frac{(1-\alpha)W+(\underline{T_W}^*_{opt}-(1-\alpha)W)N}{W}$
  $T^*_{W_{iter}} \leftarrow T^*_{W_s}$
  while $|\overline{g}(T^*_{W_{iter}})| > \varepsilon$ do
    $T^*_{W_{iter}} \leftarrow T^*_{W_{iter}} - \frac{\overline{g}(T^*_{W_{iter}})}{\overline{g}'(T^*_{W_{iter}})}$
  end while
  $\overline{T_W}^*_{opt} \leftarrow T^*_{W_{iter}}$
  $\overline{SG}^f_{wb} \leftarrow \frac{(1-\alpha)W+(\overline{T_W}^*_{opt}-(1-\alpha)W)N}{W}$
```

In the case of Weibull failure distribution with checkpointing, based on Equations (15) and (16), the lower bound and upper bound speedups are estimated as follows:

$$\begin{aligned} \underline{SG}^f_{wb,ckp} &= 1 - \alpha - (1-\alpha)N \\ &\quad + \frac{\underline{\tau}N}{\underline{\delta}\left(1 + \frac{\mu}{(\underline{\eta}/N^{\xi})\Gamma(1+\xi)} + \frac{\gamma(1+\xi,\underline{\psi})}{(1-e^{-\underline{\psi}})\Gamma(1+\xi)}\right)} \end{aligned} \tag{28}$$

$$\begin{aligned} \overline{SG}^f_{wb,ckp} &= 1 - \alpha - (1-\alpha)N \\ &\quad + \frac{\overline{\tau}N}{\overline{\delta}\left(1 + \frac{\mu}{\frac{\overline{\eta}e^N}{N^{\xi}}\Gamma(1+\xi,N)-\overline{\eta}} + \frac{\frac{\overline{\eta}e^N}{N^{\xi}}(\gamma(1+\xi,\overline{\phi})-\gamma(1+\xi,N))+\overline{\eta}e^{N-\overline{\phi}}-\overline{\eta}}{(1-e^{N-\overline{\phi}})\left(\frac{\overline{\eta}e^N}{N^{\xi}}\Gamma(1+\xi,N)-\overline{\eta}\right)}\right)}. \end{aligned} \tag{29}$$

6.3 Model Analysis

The above models provide two interesting properties about reliability-aware fixed-time speedup.

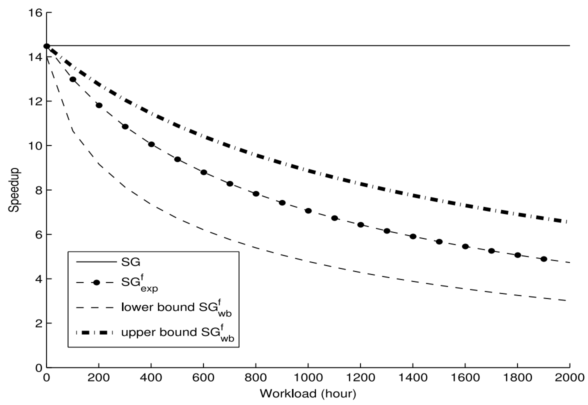


Fig. 2. Comparison of SG, SG^f_{exp}, and SG^f_{wb} with different application workloads where the computing scale is set to 16. Here, the parameters are $\alpha = 0.9$, $\lambda = \eta = \frac{1}{7500}$ hours, $\beta = 0.8$, $\mu = 0.2$ hours.

Property 3. Reliability-aware fixed-time speedups, in case of Exponential or Weibull failure distribution *without checkpointing*, decrease with the growth of application workload.

Property 4. Reliability-aware fixed-time speedups, in case of Exponential or Weibull failure distribution *with checkpointing*, are independent of application workload.

To illustrate Property 3, Fig. 2 presents reliability-aware fixed-time speedups under different workloads. By comparing Property 3 and 4, we can clearly observe that the use of checkpointing can promote application speedup with high workload.

7 Model Validation

We evaluate our models by means of real failure traces from production supercomputers. Specifically, we select two failure traces from the public failure archive [35], denoted as LANL #8 and LANL #9 in the rest of the paper. We also use a failure log from the 40-rack Blue Gene/P system named *Intrepid* at Argonne [51]. The use of multiple traces from different machines is to ensure that our models are not biased to any specific systems. For the LANL systems, we select 128 nodes that have sufficient failure records for testing; for the BlueGene/P system, we test our models at the rack-level. The summary of failure traces are listed in Table 2.

Fig. 3 depicts our evaluation design. For the purpose of model verification, we have developed an event-driven simulator. It takes two inputs: failure events and parallel applications. The failure event is extracted from the failure trace log. Each failure event is associated with a time stamp, location, and recovery time. The parameters of parallel application is specified by users, or randomly created by job generator. Each application is described in terms of its failure-free execution time W, its computing scale N, its parallel fraction α, the checkpointing configuration (i.e., with or without checkpointing) and the checkpointing overhead O_c.

TABLE 2
Summary of Failure Traces

	LANL #8	LANL #9	Blue Gene/P
MTBF (hour)	30.26	37.41	30.47
MTTR (hour)	2.26	3.33	0.21
Number of failures	3,292	3,007	549

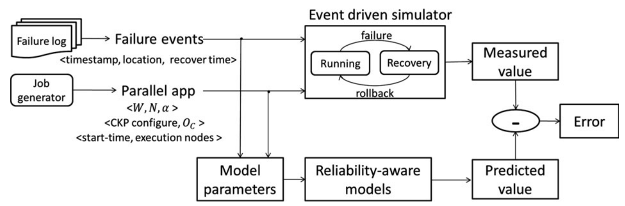


Fig. 3. Model validation.

Upon initiating the execution of an application, the simulator randomly assigns a set of machine nodes to the application and selects a time stamp within the failure trace to represent application start time. For each application, the results presented in the following sections are the average of 10,000 simulation tests with randomly selected start times and execution nodes. When the execution is completed, the simulator returns a value (denoted as *measured value*).

With respect to Amdahl's models, the simulator scans through the failure trace in the order of event occurrence time and emulates a failure when any of the assigned nodes to the application encounters a fatal event according to the failure trace. Without checkpointing, the application is stopped, waits for the recovery, and then rolls back to the beginning. The measured application time includes the failure-free execution time, the work loss, and the recovery time. With checkpointing, the simulation process is similar except that the application performs periodic checkpointing. Upon a failure, the application rolls back to the most recent checkpoint, and the checkpointing overhead is added into the measured application time.

With respect to Gustafson's models, the simulator measures the achievable workload of the application during the T_W time. Similar to the cases of calculating Amdahl's speedup, the simulator scans through the failure trace in the order of failure time stamp. Upon a failure on any of the assigned nodes, without checkpointing, the achievable workload is calculated as the application workload between the last failure and the completion time; with checkpointing, it is calculated as the sum of workload except for the checkpoint overhead, the recovery time and work loss due to rolling back.

Meanwhile, we extracted reliability related parameters from the failure log, and these include failure rate and repair time. These parameters, along with application information, are fed into our reliability-aware models to calculate various speedup values (denoted as *predicted values*). By comparing measured values and predicted values, we assess model accuracy by calculating their relative difference (i.e., $error = \frac{|predicted\ value - measured\ value|}{|measured\ value|}$).

In our experiments, checkpoint overhead O_c on LANL systems is a linear function of computing scale N (i.e., number of nodes used for running the application), and is much less than application workload [47], [48], [12]. It consists of two parts, namely I/O overhead and message passing overhead [18]. For augmented Amdahl's models, as the problem size does not change, the I/O overhead is assumed fixed [18], [14], i.e., $O_c = a + bN$. For augmented Gustafson's models, as the problem size or the checkpoint image size is proportional to the number of nodes, the overhead is defined as $O_c = aN + bN$ [25], [30]. Based on our experience as well as existing literatures [18], [50], [47], [48], we set a to

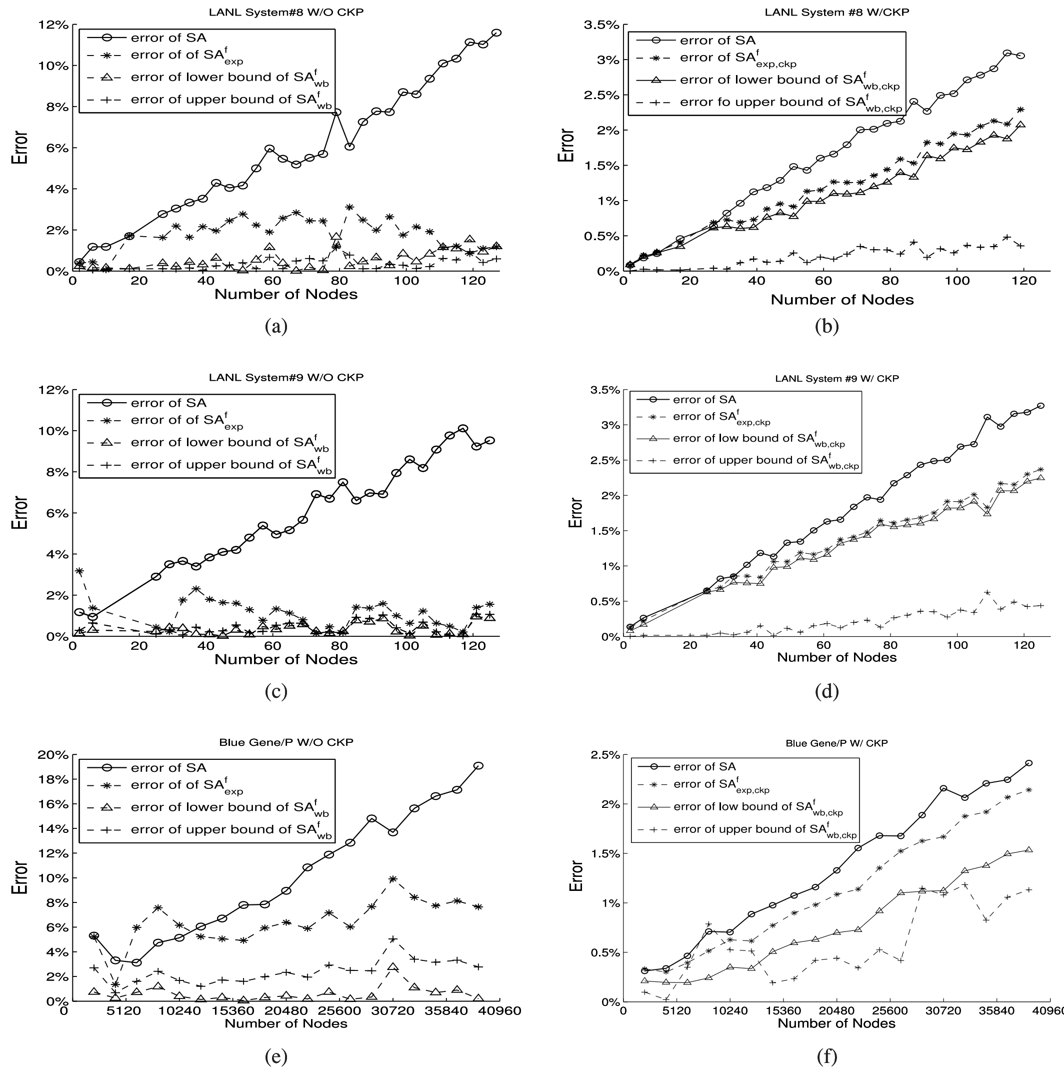


Fig. 4. Error comparison of various Amdahl's models under different computing scales. Here application workload is 100 hours and its parallel fraction is set to 0.9.

0.335 and b to 0.0364. For the Blue Gene/P system, I/O bandwidth scales linearly with the number of nodes at the beginning and then becomes level off at about 25 GB/s when the application scales to 16 racks (16,384 nodes) [50], hence we set 120 seconds of O_c for augmented Amdahl's models and 600 seconds of O_c for augmented Gustafson's models if the application uses less than 16 racks; otherwise we set 240 seconds of O_c for augmented Amdahl's models and 1,200 seconds of O_c for augmented Gustafson's models.

7.1 Under Different Computing Scales

In the first set of experiments, we study the accuracy of our reliability-aware speedup models under different computing scales. We set the workload W to 100 hours and the parallel fraction α to 0.9. The results are shown in Figs. 4 and 5.

From the figures, we make two important observations. First, the original Amdahl's model and Gustafson's model tend to deviate far from the actual application performance as the computing scale increases, no matter whether checkpointing is adopted or not. Our reliability-aware models can better represent application speedup, especially under large scales. For example, as shown in Fig. 4 a, the error of Weibull based models is always less than 2 percent, which is only about one-sixth of the original Amdahl's model. Second, our Weibull based models generally outperform the Exponential based models. Without checkpointing, either $\underline{SA}^f_{wb}$ or $\overline{SA}^f_{wb}$ shows the best accuracy. With checkpointing, $\overline{SA}^f_{wb}$ always outperforms other models with an error of less than 1 percent under different computing scales. Furthermore, its error typically does not increase with the growth of computing scale.

7.2 Under Different Workloads

In the second set of experiments, we study the accuracy of our reliability-aware speedup models under different workloads. We use the maximal computing scales, i.e., 128 nodes for the LANL systems and 40,960 nodes for the BlueGene/P system, and set α to 0.9. In terms of Amdahl's models, we test application workloads from 200 hours to 1,000 hours. In terms of Gustafson's models, we test application execution time from 40 hours to 200 hours. Due to space limitation, we

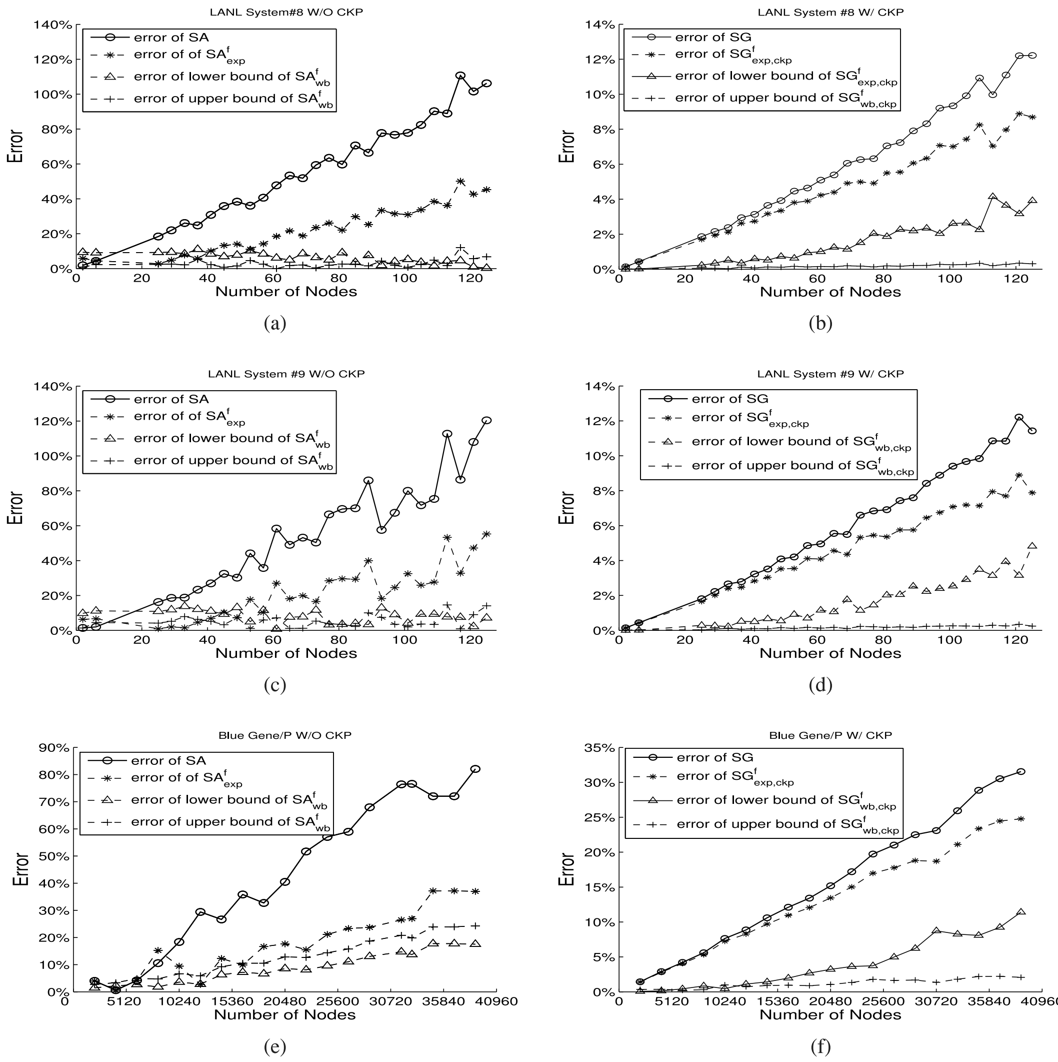


Fig. 5. Error comparison of various Gustafson's models under different computing scales. Here application workload is 100 hours and its parallel fraction is set to 0.9.

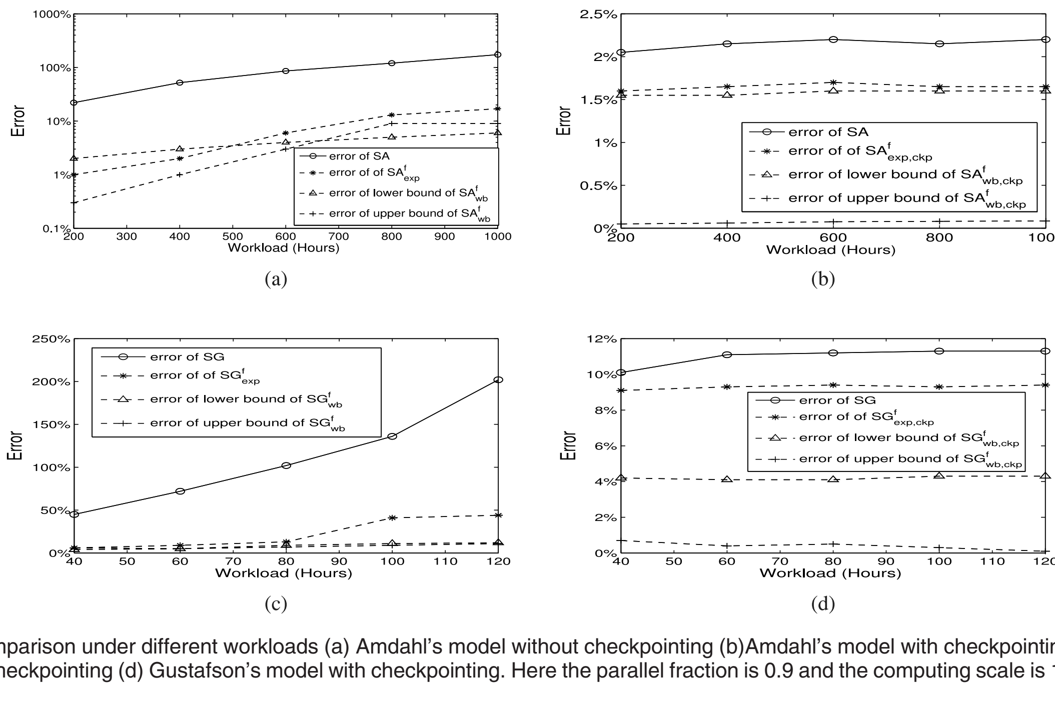


Fig. 6. Error comparison under different workloads (a) Amdahl's model without checkpointing (b)Amdahl's model with checkpointing (c) Gustafson's model without checkpointing (d) Gustafson's model with checkpointing. Here the parallel fraction is 0.9 and the computing scale is 128.

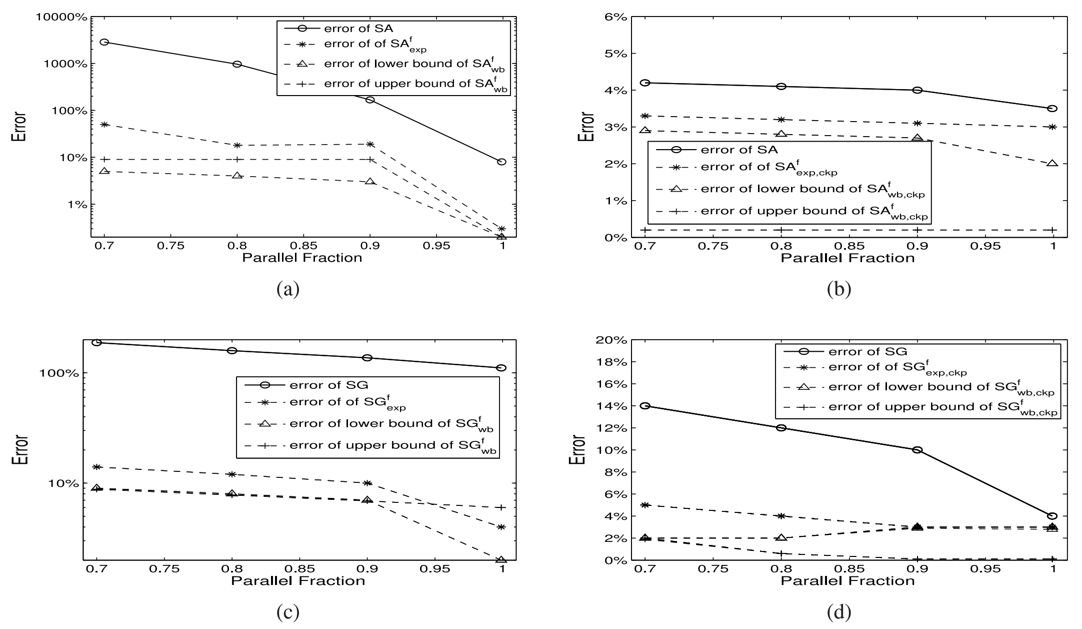


Fig. 7. Error comparison under different parallel fractions (a) Amdahl's models without checkpointing (b) Amdahl's models with checkpointing (c) Gustafson's models without checkpointing (d) Gustafson's models with checkpointing. Here the computing scale is 128, workload for Amdahl's models is 1,000 hours, and workload for Gustafson's models is 200 hours.

only present the results from the LANL #8 system in Fig. 6. The results from other systems are very similar.

Without checkpointing, the accuracy of the original Amdahl's model SA dramatically decreases with the growth of application workload. For example, as shown in Fig. 6, the error of SA is more than 100 percent when application workload is increased beyond 800 hours. Our reliability-aware models significantly outperform SA and SA^f_{exp} with the error less than 9 percent. Similarly, the original Gustafson's model SG can lead to substantial errors as the growth of workload. Instead, SG^f_{wb} quantifies the impact of failures, thereby greatly outperforming SG. These results clearly indicate that without considering failure impact, the original Amdahl's model and Gustafson's model cannot accurately represent application speedup, especially for long-running applications. Furthermore, the models based on Weibull distribution (e.g., SA^f_{wb}, $SA^f_{wb,ckp}$, SG^f_{wb}, and $SG^f_{wb,ckp}$) outperform the models based on Exponential distribution (e.g., SA^f_{exp}, $SA^f_{exp,ckp}$, SG^f_{exp}, and $SG^f_{exp,ckp}$), especially under high application workload. For example, for the application with a workload of 1,000 hours with checkpointing, the Exponential based model $SA^f_{exp,ckp}$ introduces an error of 13 percent, whereas the upper bound Weibull based model $SA^f_{wb,ckp}$ only has an error of 0.1 percent. This is due to the fact the Exponential based models do not consider the dynamic feature of failure arrivals, which can greatly influence model accuracy under high application workload.

7.3 Under Different Parallel Fractions

In the third set of experiments, we examine our reliability-aware models under different parallel fractions. We set the application workload W to 1,000 hours for Amdahl's models and 200 hours for Gustafson's models, with α varying from 0.7 to 0.999. Again, we only present the results from the LANL system #8 in Fig. 7, and omit the results from other systems as they are similar.

Without checkpointing, the original SA and SG provide extremely low accuracy, especially when parallel fraction is low. When α is 0.7 (i.e., meaning that 70 percent of the application can be parallelized), the original Amdahl's model produces an error of $2,851$ percent and the original Gustafson's model gives an error of 188 percent, whereas our models achieve much better accuracy with error being less than 10 percent. With checkpointing, our models still significantly outperform the original models. For example, the best accuracy achieved by the original Amdahl's model is 3.5 percent when α is set to 0.999, whereas $\overline{SA}^f_{wb,ckp}$ can provide an almost perfect prediction with an error of only 0.2 percent; the best accuracy achieved by the original Gustafson's model is 9 percent, whereas $\overline{SG}^f_{wb,ckp}$ has an error of only 0.1 percent. Moreover, by comparing the errors produced by Exponential based models and Weibull based models, we can observe that Weibull based models are more accurate than Exponential based models. For example, without checkpointing, the error of SA^f_{exp} is 50 percent when the α is 0.7, whereas the Weibull based models always present less than 10 percent error. With checkpointing, $\overline{SA}^f_{wb,ckp}$ and $\overline{SG}^f_{wb,ckp}$ produce extremely lower errors under different parallel fractions. This demonstrates that Weibull based models can better describe application speedup in the failure-present environment as compared to Exponential based models.

7.4 Validation Summary

Our trace-based simulations demonstrate the significant impact of failures on application speedup and the high accuracy of our newly developed speedup models. Below we summarize the key observations.

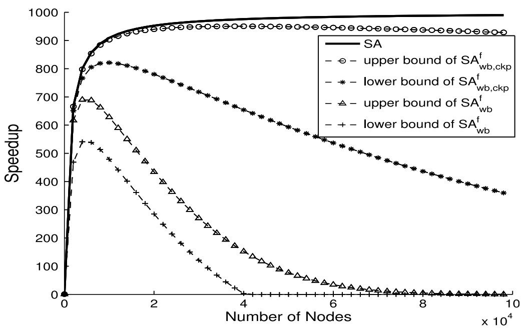


Fig. 8. Identification of the optimal computing scale and the maximum fixed-size speedup in a homogenous system. Here MTBF of 10^5 nodes is 109,718 seconds (i.e., 30.5 hours), $\alpha = 0.999$, $\beta = 0.8$, $\mu = 0.2$ hours, and the application workload is 1000 hours.

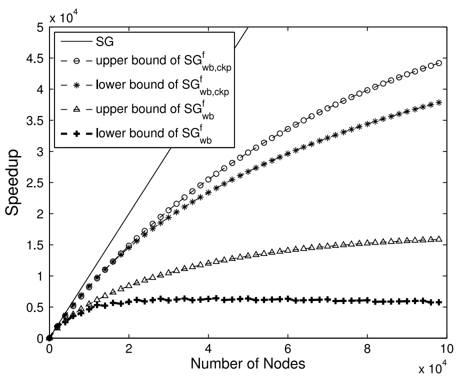


Fig. 9. Identification of the optimal computing scale and the maximum fixed-time speedup in a homogenous system. Here MTBF of 10^5 nodes is 109,718 seconds (i.e., 30.5 hours), $\alpha = 0.999$, $\beta = 0.8$, $\mu = 0.2$ hours, and the application workload is 10 hours.

- The original SA and SG cannot accurately represent application speedup in a failure-present environment, especially under large computing scale, high workload, and small parallel fraction.
- The Exponential based speedup models quantify the impact of failures and the effect of checkpointing and their accuracy decrease under high workload and small parallel fraction. Without checkpointing support, the Exponential based speedup models may lead to an error of up to 50 percent.
- The Weibull based speedup models are highly accurate with the error typically ranging between $0.4-5.1$ percent for Amdahl's models and $0.3-17.5$ percent for Gustafson's models, which significantly outperform the Exponential based models.

8 Model Usage

As shown in Section 7, our Weibull based models can represent application speedup with high accuracy. In this section, we present two case studies to demonstrate the use of Weibull based models. Given an application, the first case study shows the use of our models to identify an optimal computing scale. The second one is to estimate node MTBFs and checkpoint overhead that are needed in future exascale systems in order to maintain the computing efficiency achieved on current systems.

8.1 Identification of Optimal Computing Scale

According to the original Amdahl's law and Gustafson's law, the fixed-size speedup SA monotonically increases as the computing scale N grows with the bound $\frac{1}{1-\alpha}$, and the fixed-time speedup SG can grow infinitely as the computing scale increases in a failure-free system. Nevertheless, in a realistic failure-present environment, our derived models show that both fixed-size speedup and fixed-time speedup drop down when the computing scale increases beyond a certain point due to the increasing failure rate. As a result, for a parallel application, it is necessary to identify the optimal computing scale at which the application can achieve the maximal speedup.

Suppose a homogenous system where all the nodes have the same Weibull failure distribution, the optimal computing scale can be determined by solving the equations $\partial S^f/\partial N = 0$ and $\partial S^f_{ckp}/\partial N = 0$. Fig. 8 presents the fixed-size speedup results on a homogeneous system where the computing scale ranges from 1 to 10^5. There are five curves in the plot, representing the original Amdahl's model and our reliability-aware models with and without checkpointing respectively. As shown in the figure, unlike the original SA, reliability-aware fixed-size speedup decreases as the computing scale increases beyond a certain point. Furthermore, by comparing the curves with and without checkpointing, we observe that the use of checkpointing can considerably boost application speedup by increasing the maximal achievable speedup. For the specific application listed in the figure, without checkpointing, the optimal scale is about 4,500-5,000 which achieves a speedup of about 550-680; with checkpointing, the optimal scale is about $16{,}000 - 21{,}000$ which achieves a speedup of about 820-920.

Similarly, Fig. 9 presents the fixed-time speedup as the computing scale grows from 1 to 10^5. Obviously, these curves indicate that the achievable fixed-time speedups of SG^f and SG^f_{ckp} are much smaller than the speedup given by the original Gustafson's law. The curves of SG^f show that the maximal speedup can only achieve about 1.5×10^4 although α is as high as 0.999. With checkpointing, the growth of SG^f_{ckp} is still much slower than the original Gustafon's law, but SG^f_{ckp} can achieve 4.5×10^4 when computing scale is 10^5. The difference between SG^f and SG^f_{ckp} indicates that the use of checkpointing can increase the optimal computing scale and boost application speedup.

8.2 Projection on Exascale Systems

The *efficiency* of parallel application is defined as the proportion of speedup to the computing scale. It measures the extent to which time is well used for the intended computation. Assume a future system composing of nodes with the same reliability as those on the current Blue Gene/P machine "Intrepid" at Argonne. We further assume checkpoint overhead on this future system is maintained at the same level as on Intrepid. Our field data show that on Intrepid, system-wide MTBF is 30.5 hours, and checkpointing overhead typically takes 30 minutes [51]. Fig. 10 presents *efficiency* trend on a large scale system (composing of $10^5 - 10^6$ nodes). Here, we use the reliability-aware Gustafson's model under checkpointing. The curves show that efficiency exhibits decreasing trend. Especially, the

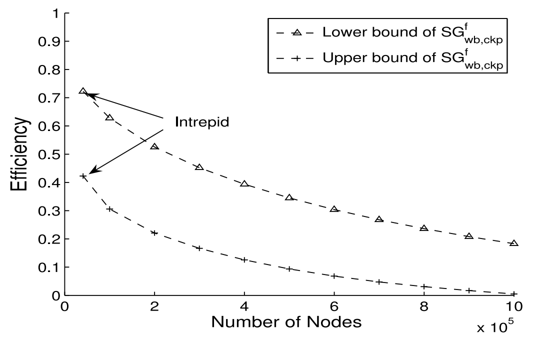


Fig. 10. The efficiency with different computing scales. Here the parameters are $\alpha = 0.9$, $\beta = 0.6$, $W = 168$ hours. The MTBFs of Intrepid system is 109,718 seconds, and checkpointing overhead on existing system $O_c = 30$ minutes.

lower bound of efficiency becomes less than 0.1 when computing scale reaches 10^6. In other words, the results here indicate that in order to effectively harness the potential of extreme scale systems, we need to improve fault tolerance.

A number of approaches have been studied to reduce the impact of failures. These approaches can be broadly classified into two main directions: improving the node reliability [19], [52] or reducing the overhead of fault tolerance methods such as checkpointing [16], [26], [33], [34]. Our derived models can be applied to measure the benefits of these approaches with regards to maintaining good computing efficiency.

In terms of node reliability, existing studies mainly focus on increasing node MTBF. In Fig. 11, we present the required node MTBF on future systems in order to maintain the same efficiency on Intrepid (Fig. 10). Here, we assume checkpoint overhead is the same as on Intrepid. As we can see, the shape value β plays a critical role with regard to node MTBF. There are three curves representing different shape values in the plot. It is shown that a smaller β (e.g., when $\beta = 0.6$) generally requires much faster growing node MTBF in order to maintain the efficiency. The main reason is that a lower shape value leads to a higher failure rate right after the occurrence of failure [49], thus a higher MTBF is expected to reduce the average number of failures during the application execution time. In other words, it is essential to increase both the shape value and the node MTBF to gain a good efficiency on future systems.

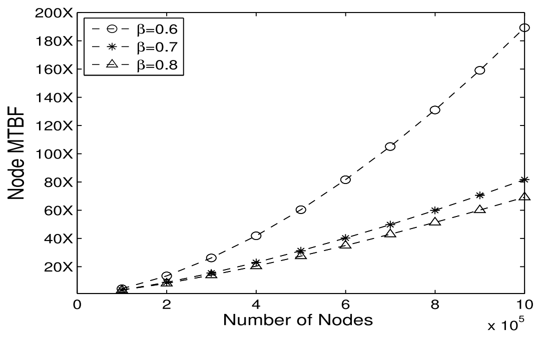


Fig. 11. The node MTBF is required on future exascale systems to maintain the same efficiency on Intrepid. Here $200\times$ represents 200 times over node MTBF in Intrepid. The parameters are $\alpha = 0.99$, $\mu = 0.2$ hours, $W = 168$ hours.

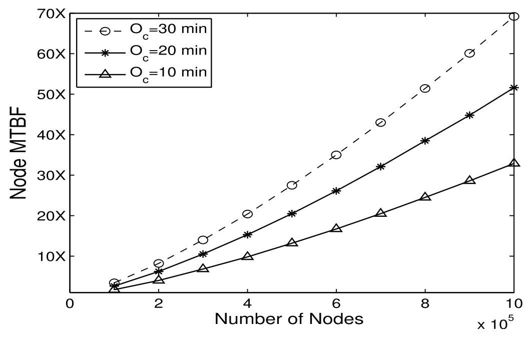


Fig. 12. The combined effect of node MTBF and checkpointing overhead for future exascale systems to maintain the same efficiency as on Intrepid. Here $70\times$ represents 70 times over node MTBF in Intrepid. The parameters are $\alpha = 0.99$, $\mu = 0.2$ hours, $W = 168$ hours.

Improving checkpointing performance is an active research area, and a wide range of hardware and software technologies are presented to reduce the checkpointing overhead [16], [26], [33], [52]. Our models indicate that reducing checkpointing overhead alone is insufficient to maintain computing efficiency due to inevitable work loss. Reducing checkpointing overhead can reduce the growth requirement on node MTBF, as shown in Fig. 12. As an example, for a future system composing of 10^6 nodes, if the system keeps the same checkpointing overhead (i.e., 30 minutes), its node MTBF should be increased 69.2 times longer than that observed on Intrepid, in order to maintain the same level of computing efficiency; nevertheless, if the checkpointing overhead can be reduced to 10 minutes, its node MTBF should be increased 32.9 times longer than that on Intrepid.

9 Model Discussion

In this paper, we focus on global coordinated checkpointing given that it is the most popular fault tolerant mechanism used in practice [26], [56], [58]. Nevertheless, as demonstrated in Section 8.2, global coordinated checkpointing may be not a viable resiliency solution as we move toward exascale computing. Recently several new techniques are developed to address the potential problem associated with global coordinated checkpointing, and these include uncoordinated checkpointing with message logging [56] and hybrid checkpointing protocol [58], [60]. With uncoordinated checkpointing, each process can take its checkpoints independently and locally. Upon a failure, only the failed process rolls back to the previous state. Since uncoordinated checkpointing may lead to a domino effect that forces the entire application to restart from the beginning, message logging is adopted to replay the messages for recovery. However, extensive message logging consumes storage space and adds a significant overhead on communication bandwidth. To alleviate this issue, hybrid protocol is proposed to combine coordinated checkpointing and message logging. It conducts coordinated checkpointing inside clusters of processes and message logging between the clusters. Meanwhile, hybrid protocol leverages the common properties of parallel applications such as send-determinism [56] and channel-determinism [60] to reduce the amount of logging messages and the performance overhead during recovery.

Our models can be extended for the aforementioned fault tolerant techniques. Take hybrid checkpointing protocol as an example. Within each cluster of processes, the speedup model is identical to the coordinated checkpointing models. Upon a failure, only the impacted cluster of processes needs to roll back and replays the corresponding inter-cluster messages. As a result, the speedup of the application is determined by the slowest cluster and the recovery performance of the protocol. Suppose the cluster size is K, all the nodes have exponential failure distribution with the same failure rate λ, and the corresponding average recovery time is μ_K, we can derive the augmented Amdahl's model for hybrid protocol from Equation (19):

$$SA^{f}_{exp,hybrid} = \frac{K\lambda\tau_K}{e^{\mu_K K\lambda}\left(e^{\delta_K K\lambda}-1\right)\left(1-\alpha+\frac{\alpha}{N}\right)}, \tag{30}$$

where τ_K is the optimal checkpointing interval for the cluster. By studying the relation between μ_K and K, this model can be used to identify the optimal cluster size.

10 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 coordinated checkpointing. In particular, we have derived speedup models based on Weibull failure distribution, and analyzed the cases with and without checkpointing in these models. By means of real failure data from various production systems, we have demonstrated that these analytical models can better represent application performance and speedup in the presence of failures. Moreover, our results clearly show that Weibull based models outperform Exponential based models in terms of characterizing application speedup in the presence of failures.

The newly derived speedup models can quantitatively guide the community in terms of evaluating, optimizing, and predicting application performance in realistic failure-present environments. One of our future work is to combine these analytical models with our empirical log analysis studies [51], [19], [55] to promote performance and resilience of extreme scale computing. Furthermore, we will extend our models to study the impact of other fault tolerance techniques such as uncoordinated checkpointing with message logging [56] and hybrid checkpointing protocol [58], [60].

Acknowledgments

Zhiling Lan was supported by US NSF grant CNS-0834514.

References

[1] A. Oliner and J. Stearly, "What supercomputers say: A study of five system logs," in *Proc. IEEE Annu. Int. Conf. Dependable Syst. Netw.*, 2007, pp. 575–584.

[2] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance-computing systems," in *Proc. IEEE Annu. Int. Conf. Dependable Syst. Netw.*, 2006, pp. 337–350.

[3] G. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," in *Proc. AFIPS Spring Joint Comput. Conf.*, 1967, pp. 483–485.

[4] J. Gustafson, "Reevaluating Amdahl's law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, 1988.

[5] Y. Tanaka, H. Takemiya, S. Sekiguchi, S. Ogata, A. Nakano, R. Kalia, and P. Va shishta, "Adaptive grid-enabled SIMOX simulation on Japan-US grid testbed," in *Proc. TeraGrid*, 2006.

[6] (2013). Top500 supercomputing sites. http://top500.org/

[7] D. Reed, C. Lu, and C. Mendes, "Big systems and big reliability challenges," in *Proc. Parallel Comput.*, 2003, pp. 729–736.

[8] T. Lin and D. Siewiorek, "Error log analysis: Statistical modeling and heuristic trend analysis," *IEEE Trans. Rel.*, vol. 39, no. 4, pp. 419–432, Oct. 1990.

[9] X. Sun and L. Ni, "Another view on parallel speedup," in *Proc. Supercomput.*, 1990, pp. 324–333.

[10] R. Ge and K. Cameron, "Power-aware speedup," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2007, pp. 1–10.

[11] M. Wu, X. Sun, and H. Jin, "Performance under failure of high-end computing," in *Proc. SuperComput.*, 2007, pp. 48:1–48:11.

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

[13] S. Garg, Y. Huang, C. Kintala, and K. Trivedi, "Minimizing completion time of a program by checkpointing and rejuvenation," in *Proc. Int. Conf. Measurement Modeling Comput. Syst.*, 1996, pp. 252–261.

[14] J. Plank and M. Thomason, "Processor allocation and checkpoint interval selection in cluster computing systems," *J. Parallel Distrib. Comput.*, vol. 61, no. 11, pp. 1570–1590, 2001.

[15] J. Plank and W. Elwasif, "Experimental assessment of workstation failures and their impact on checkpointing systems," in *Proc. 28th Annu. Int. Symp. Fault-Tolerant Comput.*, 1998, pp. 48–57.

[16] E. Elnozahy and J. Plank, "Checkpointing for Peta-scale systems: A look into the future of practical rollback-recovery," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 2, pp. 97–108, Apr.–Jun. 2004.

[17] L. Wang, K. Pattabiraman, Z. Kalbarczyk, and R. Iyer, "Modeling coordinated checkpointing for large-scale supercomputers," in *Proc. Int. Conf. Dependable Syst. Netw.*, 2005, pp. 812–821.

[18] Z. Lan and Y. Li, "Adaptive fault management of parallel applications for high performance computing," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1647–1660, Dec. 2008.

[19] 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," in *Proc. Int. Conf. Parallel Process.*, 2008, pp. 57–164.

[20] Z. Zheng and Z. Lan, "Reliability-aware scalability models for high performance computing," in *Proc. IEEE Int. Conf. Cluster Comput. Workshops*, 2009, pp. 1–9.

[21] 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," in *Proc. High Availability Perform. Comput. Workshop*, 2008.

[22] J. Young, "A first order approximation to the optimal checkpoint interval," *Commun. ACM*, vol. 17, no. 9, pp. 530–531, 1974.

[23] Y. Zhang, M. Squillante, A. Sivasubramaniam, and R. Sahoo, "Performance implications of failures in large-scale cluster scheduling," in *Proc. 10th Int. Workshop JSSPP, SIGMETRICS*, 2004, pp. 233–252.

[24] A. Oliner, R. Sahoo, J. Moreira, and M. Gupta, "Performance implications of periodic checkpointing on large-scale cluster systems," in *Proc. Parallel Distrib. Process. Symp.*, 2005, p. 8.

[25] S. Arunagiri, J. Daly, P. Teller, S. Seelam, R. Oldfield, M. Varela, and R. Riesen, "Opportunistic checkpoint intervals to improve system performance," Tech. Rep. UTEP-CS-08-24, University of Texas at El Paso, 2008.

[26] A. Bouteiller, P. Lemarinier, G. Krawezik, and F. Cappello, "Improved message logging versus improved coordinated checkpointing for fault tolerant MPI," in *Proc. Int. Conf. Cluster Comput.*, 2003, pp. 115–124.

[27] V. Kumar and A. Gupta, "Analysis of scalability of parallel algorithms and architectures: A survey," in *Proc. 5th Int. Conf. Supercomput.*, 1991, pp. 396–405.

[28] D. Woo and H. Lee, "Extending Amdahl's law for energy-efficient computing in the many-core era," *IEEE Comput.*, vol. 41, no. 12, pp. 24–31, Dec. 2008.

[29] A. Oliner, L. Rudolph, and R. Sahoo, "Cooperative checkpointing: A robust approach to large-scale systems reliability," in *Proc. Int. Conf. Supercomput.*, 2006, pp. 14–23.

[30] R. Oldfield, "Investigating lightweight storage and overlay networks for fault tolerance," in *Proc. High Avail. Perform. Comput. Workshop*, 2006.

[31] F. Petrini, K. Davis, and J. Sancho, "System-level fault tolerance in large-scale parallel machines with buffered coscheduling," in *Proc. Int. Parallel Distrib. Process. Symp.*, 2004, p. 209.

[32] V. Nicola, *Checkpointing and Modelling of Program Execution Time. Software Fault Tolerance*. Hoboken, NJ, USA: Wiley, 1995.

[33] Y. Li and Z. Lan, "A fast recovery mechanism for checkpointing in networked environments," in *Proc. Int'l Conf. Dependable Syst. Netw.*, 2008, pp. 217–226.

[34] C. Wang, F. Mueller, C. Engelmann, and S. Scott, "Proactive process-level live migration in HPC environments," in *Proc. Supercomput.*, 2008, pp. 43:1–43:12.

[35] Los Alamos National Laboratory. Operational Data to Support and Enable Computer Science Research. (2006) [Online]. Available: http://institute.lanl.gov/data/lanldata.shtml

[36] 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," in *Proc. IEEE Int. Symp. Cluster Comput. Grid*, 2008, pp. 795–800.

[37] S. Amarasinghe, D. Campbell, W. Carlson, A. Chien, W. Dally, E. Elnohazy, M. Hall, R. Harrison, W. Harrod, K. Hill, J. Hiller, S. Karp, C. Koelbel, D. Koester, P. Kogge, J. Levesque, D. Reed, V. Sarkar, R. Schreiber, M. Richards, A. Scarpelli, J. Shalf, A. Snavely, and T. Sterling, "Exascale software study: Software challenges in extreme scale systems," DARPA IPTO, Air Force Research Labs, Tech. Rep 2009.

[38] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. S. Williams, K. Yelick, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Keckler, D. Klein, P. Kogge, R. S. Williams, and K. Yelick, "ExaScale computing study: Technology challenges in achieving exascale systems," Dept. Comput Sci., Univ. Notre Dame, Notre Dame, IN, USA, Tech. Report TR-2008-13, 2008.

[39] E. Yero and M. Henriques, "Speedup and scalability analysis of Master-Slave applications on large heterogeneous clusters," *J. Parallel Distrib. Comput.*, vol. 67, no. 11, pp. 1155–1167, 2007.

[40] N. Gottumukkala, R. Nassar, M. Paun, C. Leangsuksun, and S. Scott, "Reliability of a system of k nodes for high performance computing applications," *IEEE Trans. Reliability*, vol. 59, no. 1, pp. 112–169, Mar. 2010.

[41] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. Scott, "An optimal checkpoint/restart model for a large scale high performance computing system," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2011, pp. 1–9.

[42] P. Jogalekar and M. Woodside, "Evaluating the scalability of distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 6, pp. 589–603, Jun. 2000.

[43] W. Jones, J. Daly, and N. DeBardeleben, "Impact of sub-optimal checkpoint intervals on application efficiency in computational clusters," in *Proc. ACM Int. Symp. High Perform. Distrib. Comput.*, 2010, pp. 276–279.

[44] N. Taerat, N. Naksinehaboon, C. Chandler, J. Elliott, C. Leangsuksun, G. Ostrouchov, S. Scott, and C. Engelmann, "Blue Gene/L log analysis and time to interrupt estimation," in *Proc. Int. Conf. Avail., Rel. Security*, 2009, pp. 173–180.

[45] H. Jin, Y. Chen, and H. Zhu, X. Sun, "Optimizing HPC fault-tolerant environment: An analytical approach," in *Proc. Int. Conf. Parallel Process.*, 2010, pp. 525–534.

[46] T. Hacker, F. Romero, and C. Carothers, "An analysis of clustered failures on large supercomputing systems," *J. Parallel Distrib. Comput.*, vol. 69, no. 7, pp. 652–665, 2009.

[47] H. Naik, R. Gupta, and P. Beckman, "Analyzing checkpointing trends for applications on the IBM Blue Gene/P system, in," in *Proc. Int. Conf. Parallel Process.Workshop*, 2009, pp. 81–88.

[48] R. Oldfield, S. Arunagiri, P. Teller, S. Seelam, and M. Varela, "Modeling the impact of checkpoints on next-generation systems," in *Proc. Int. Conf. Mass Storage Syst. Technol*, 2007, pp. 30–46.

[49] H. Rinne, *The Weibull Distribution: A Handbook*. Boca Raton, FL, USA: CRC Press, 2008.

[50] S. Lang, P. Carns, K. Harms, and W. Allcock, "I/O performance challenges at leadership scale," in *Proc. High Perform. Comput. Netw., Storage Anal.*, 2009, pp. 1–12.

[51] Z. Zheng, L. Yu, W. Tang, Z. Lan, R. Gupta, N. Desai, S. Coghlan, and D. Buettner, "Co-Analysis of RAS log and job log on Blue Gene/P," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2011, pp. 840–851.

[52] K. B. Ferreira, J. Stearley, J. H. Laros III, R. Oldfield, K. T. Pedretti, R. Brightwell, R. Riesen, P. G. Bridges, and D. Arnold, "Evaluating the viability of process replication reliability for Exascale systems," in *Proc. High Perform. Comput. Netw., Storage Anal.*, 2011, pp. 1–12.

[53] M. Bougeret, H. Casanova, M. Rabie, Y. Robert, and F. Vivien, "Checkpointing strategies for parallel jobs," in *Proc. High Perform. Comput. Netw., Storage Anal.*, 2011, pp. 1–11.

[54] Z. Zheng and Z. Lan, "Reliability-aware scalability models for high performance computing," in *Proc. Cluster Comput. Workshops*, 2009, pp. 1–9.

[55] L. Yu, Z. Zheng, Z. Lan, T. Jones, J. Brandt, and A. gentile, "Filtering log data: Finding the needles in the haystack," in *Proc. Int. Conf. Dependable Syst. Netw*, 2012, pp. 1–12.

[56] A. Guermouche, T. Ropars, E. Brunet, M. Snir, and F. Cappello, "Uncoordinated checkpointing without domino effect for send-deterministic MPI applications," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2011, pp. 989–1000.

[57] A. Moody, G. Bronevetsky, K. Mohror, and B. Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proc. High Perform. Comput. Netw., Storage Anal.*, 2010, pp. 1–10.

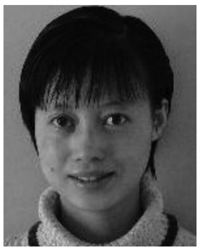
[58] R. Riesen, K. Ferreira, D. Silva, P. Lemarinier, D. Arnold, and P. Bridges, "Alleviating scalability issues of checkpointing protocols," in *Proc. High Perform. Comput. Netw., Storage Anal.*, 2012, pp. 18:1–18:11.

[59] G. Cao and M. Singhal, "Checkpointing with mutable checkpoints," *Theor. Comput. Sci.*, vol. 290, no. 2, pp. 1127–1148, 2003.

[60] T. Ropars, T. Martsinkevich, A. Guermouche, A. Schiper, and F. Cappello, "SPBC: Leveraging the characteristics of MPI HPC applications for scalable checkpointing," in *Proc. High Perform. Comput. Netw., Storage Anal.*, 2013, pp. 8:1–8:12.

Ziming Zheng received the BS and MS degrees from the University of Electronic Science and Technology of China in 2003 and 2006, respectively, and the PhD degree in computer science from Illinois Institute of Technology in 2012. He was a postdoctoral scholar at the University of Chicago in 2013. He is currently a software engineer in HP Vertica. His research focuses on fault tolerance in large-scale computer systems. He is a member of IEEE Computer Society and the IEEE.

Li Yu received the BS degree from Sichuan University in 2004, and the MS degree from Rochester Institute of Technology in 2009, respectively. He is currently working toward the PhD degree in computer science at the Illinois Institute of Technology since 2010. His research interests include HPC data analytics and performance modeling in large-scale systems. He is a student member of the IEEE.



Zhiling Lan received the PhD degree in computer engineering from Northwestern University in 2002. She is currently an associate professor of computer science at the Illinois Institute of Technology. Her research interests include fault tolerance, resource management and scheduling, energy efficiency, and performance analysis and modeling. She is a senior member of IEEE Computer Society and the IEEE.