ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **18**, Number 2, 2015, 109–125

HPC Application in Cloud Environment

M. PAUN[1], C. LEANGSUKSUN[3], R. NASSAR[2], T. THANAKORNWORAKIJ[3]

[1]National Institute for Research and Development of Biological Sciences, Bucharest, Romania
[2]Louisiana Tech University, Department of Mathematics and Statistics, Ruston, Louisiana, USA
[3]Louisiana Tech University, Department of Computer Science, Ruston, Louisiana, USA

E-mail: mihaela.paun@incdsb.ro

Abstract. High Performance Computing applications on Cloud are of significance because of cost-effectiveness and elasticity. Reliability analysis of HPC applications on Cloud is an important area of study to better utilize infrastructure while dealing fault tolerant issues in a Cloud environment. In this work, we present a reliability model of a Cloud system under four scenarios: 1) Hardware components fail independently and software components fail independently; 2) software components fail independently and hardware components are correlated in failure; 3) correlated software failure and independent hardware failure; 4) dependent software and hardware failures. Moreover, we propose an optimal checkpoint placement technique based on reliability information for each scenario. Results show that if failure of the nodes and/or software in the system possesses a degree of dependency, the system becomes less reliable, which means that the failure rate increases and the mean time to failure decreases. Also, an increase in the number of nodes decreases the reliability of the system. Moreover, the optimal checkpoint interval decreases when the reliability of the system decreases.

Key-words: Fault tolerance, reliability, cloud computing, cloud performance.

ACRONYMS

CDF	Cumulative distribution function
pdf	Probability density function
TTF	Time to failure
HPC	High performance computing
MTTF	Mean time to failure

1. Introduction

Cloud computing is increasingly popular because it is easy to deploy, flexible, and cost-effective. Cloud computing has not only been utilized in commercial but also in scientific communities such as by a High Energy Physics group in the research project CernVM, which utilizes Virtual machines for running the Large Hadron Collider (LHC) experiments [2]. Cloud computing also provides services for HPC applications. Amazon EC2 offers HPC services on their site [1]. Penguin computing [18] provides private, public, and hybrid HPC as-a Service [13] for their customers. Many studies have evaluated HPC applications on Cloud. Evangelinos [4] studied performance of HPC applications on Amazon EC2. He used Atmosphere-ocean climate models, which were implemented by using Message Passing Interface (MPI) for parallel computing. He also benchmarked several MPI implementations, such as LAM, MPICH, OpenMPI, on Amazon EC2. Results showed that Amazon EC2 is appropriate for small sized HPC applications. Gupta [6] compared Cloud performance based on HPC with high-end machines. He evaluated the cost-effectiveness of Cloud computing, the communication efficiency of HPC applications on Cloud, and the type of HPC applications that are appropriate to run on Cloud. Results showed that Cloud is appropriate for relatively low communication-intensive HPC applications. For MPI applications, a failure may interrupt an entire application. Hence, there are some fault tolerant mechanisms to mitigate failures. A checkpoint/restart mechanism is a technique that saves a computing state. When the job fails, one can restore the job from the last checkpoint. Live migration is another technique that moves a Virtual machine to another healthier machine before a failure occurs. To better deal with failures, one should know the failure behavior or the reliability of the system. In this paper, we propose a reliability model for a cloud computing system that considers software, Virtual Machine, Hypervisor, and hardware failures as well as correlation of failures within the software and hardware. Furthermore, we develop an optimal checkpoint placement technique from the Cloud reliability model. The paper is organized as follows: Section 2 introduces related work and Multivariate model, the TTF of Cloud system is discussed in Section 3. Cloud reliability model is presented in Section 4 and Section 5 presents Cloud reliability, MTTF, and failure rate. Section 6 illustrates the optimal checkpoint placement using reliability information from Section 5. Conclusions are discussed in Section 7.

2. Related work

Several researchers have studied cloud computing reliability. Vishwanath [21] studied the characteristics of hardware failure and hardware repair in cloud computing. Results showed that when a server failed, it has a better chance of failing again. Dai [3] conducted a study on cloud service reliability. He classified failures into two groups, request stage failures and execution stage failures. The request stage failures considered overflow and timeout issues. The execution stage failures are related to missing data and computing resources, software failure, database failure, hardware

failure, and network failure. The reliability of a cloud service is considered to be the reliability of request stage failures multiplied by those of the execution stage. Hacker [10] considered hardware reliability and assumed a virtual machine per server. The model Hacker proposed was based on the Weibull distribution. However, this work did not incorporate software reliability into the model. None of the work considers reliability of Cloud for HPC applications. In our work, we will consider the reliability of a cloud computing application that includes both hardware and software.

Studies have shown that the time to failure (TTF) of a computer system can be described by a Weibull distribution [10], [9], [23]. Hacker [9] studied the impact of reliability rates of individual components on the reliability of an HPC system. He also showed that the time between failures for an HPC system followed a Weibull distribution. Xu [23] also showed that TTFs of Windows NT servers follow a Weibull distribution. In our previous work, Gottumukkala et. al, [7] developed a reliability model of a k-node system for HPC applications when individual TTF follows a Weibull distribution. We considered the excess life or time since the last failure of an individual node in the reliability model to gain a more accurate estimation of the reliability of a system based on an assumption that nodes fail independently. However, time to failure of some computer systems may not be independent [23] , [19] . Xu [23] showed that there is failure dependency of Windows NT Servers across the network. Schroeder [19] presented a study of failures in an HPC system at Los Alamos National Laboratory (LANL) showing that nodes were correlated with regard to failure and that two or more nodes may fail at the same time. In this work, we consider excess life as well as correlation due to possible simultaneous failures of nodes in the system.

For software reliability, there are studies undertaken to understand the characteristics of a software failure [22], [17], [20] .Wood [22] was interested in understanding software reliability models and their utility. He evaluated 9 different software reliability models. He showed that a simple exponential model performs as well or better than complex models, and the simple model outperformed the other models in terms of both stability and predictive ability. Musa [17] evaluated 7 model groups on 15 sets of data. Based on his evaluation, the exponential and logarithmic models were recommended for modeling software reliability. Thirumurugan [20] studied software reliability modeling in the testing and operation phase. Failures occurred at a constant rate over time [20], [24]. He considers an exponential software reliability model. Results showed that, at any given time, the reliability in the operation phase was less than that in the testing phase. The majority of software reliability models make the assumption that failures occur independently. However, as evident from [8], the software failures are not always independent. Popstojanova [5] used Markov renewal modeling techniques in order to formulate software reliability models that considered dependent failures and time to failure following the exponential distribution.

2.1. Multivariate Exponential Model

As known from the literature [17], an exponential distribution is appropriate for modeling software reliability (Applications, VM's and Hypervisor). In this work, we will refer to Application, VM's and Hypervisors as software since they all assume an

exponential time till failure distribution. Software failure may be independent or dependent. Independent software failures, for example, can occur in the case of parallel jobs, such as biological sequence and video encoding. In case of dependent failure, system configuration and operation environment may cause dependent software failures [12]. Moreover, applications may fail at the same time, due to a certain situation such as communication outages between processes. For example, blocking communication on an MPI application may cause simultaneous process failures. Another example is simultaneous failures due to the fact that applications may have to wait on data to become available. To consider correlation between software failures, we use the Marshall-Olkin Multivariate Exponential Distribution (MOMED) [16] based on the following fatal shock model.

$$\overline{F}_{Y_1\ldots Y_k}(y_1...y_k) = \exp\{-\sum_{i=1}^{k}\gamma_i y_i - \sum_{i<j}\gamma_{ij}\max(y_i,y_j) - \sum_{i<j<l}\gamma_{ijl}\max(y_i,y_j,y_l) - ... - \gamma_{12...k}\max(y_1,y_2,...,y_k)\} \tag{1}$$

Suppose that a component fails after receiving a fatal shock. The occurrence of shocks is based on independent Poisson process , with $i = 1, 2 \ldots k$. In each Poisson process, $t > 0$ and lambda is the Poisson parameter. The events in the Poisson process, are shocks to component i, the events in the process, are simultaneous shocks to both components i and j, the events in the process are simultaneous shocks to components i, j and l and the events in the process are simultaneous shocks to all k components. Thus is a survival function of k components.

2.2. Multivariate Weibull

For node failure, we have evidence that nodes may fail simultaneously [19]. Many studies also showed that the time to failure of an individual node follows a Weibull distribution. To model nodes failing at the same time, we use the Multivariate Weibull distribution. One can derive a Multivariate Weibull model from the multivariate exponential model by a variable transformation technique used in [11]. In the survival function of the multivariate exponential Eq. (1), consider the transformation, $c > 0$, Using the transformation of variable technique in [14], one can obtain the survival function of the general multivariate Weibull (MVW) ,which is given by

$$\overline{F}_{X_1\ldots X_k}(x_1...x_k) = \exp\{-\sum_{i=1}^{k}\lambda_i x_i^c - \sum_{i<j}\lambda_{ij}\max(x_i^c,x_j^c) - \sum_{i<j<l}\lambda_{ijl}\max(x_i^c,x_j^c,x_l^c) - ... - \lambda_{12...k}\max(x_i^c,...,x_k^c)\}. \tag{2}$$

In what follows we extend the model in Eq. (2) to include an excess or conditional Weibull in order to determine, in conjunction with Eq. (1), the reliability of a cloud system for different scenarios that may occur in practice.

3. TTF considerations of a Cloud System

In this section, we consider reliability of an application that has ℓ processes (App1-Appℓ) in a cloud system composed of s virtual machines deployed in k nodes as shown in Fig. 1, where each node can have a different number of virtual machines. We make an assumption that it is a classic HPC application based on MPI. An application fails when any one of the k nodes fails or any software component fails. In the hardware case, when any node fails, it is replaced with a new node and the system is re-started. In the software case, the VM is re-started and the system operates again.

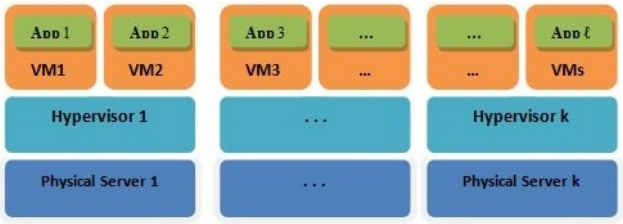

Fig. 1. Checkpoint Notation.

Our interest is in determining the probability density function (PDF) for the time to the first failure of the system after the j^{th} node replacement or a VM re-start due to a failure. This time is referred to as the time to failure (TTF). Therefore, we make the following assumptions concerning the failure properties of individual nodes in an HPC system, based on our discussion in the previous section.

- TTF of an individual node follows a Weibull distribution
- TTF of an application that is running on a particular VM has an exponential distribution.
- The first failure interrupts the entire application.
- After a failure, the node is replaced with a new node at the next time instant, and hence the system returns to operation.

We consider a k-node cloud system with n ($n = \ell + s + k$, see Fig. 1) software components (App, VM and Hypervisor), each with an exponential time to failure), where any failure in any one of the n components interrupts the entire system. If any of the software components fails the component (and hence the system) will be re-started. On the other hand, if a node fails, the node is replaced and the system is restarted.

In case of node failures, the distribution of the time to failure is Weibull for the node that is replaced and excess Weibull for the other nodes. Note that because of simultaneous failures; more than one node may be replaced at the same time. The excess life is defined as the probability that a node will fail at time x given that the

node has survived until time t. The excess life pdf for a Weibull distribution can be expressed as

$$f(t_{ij}+x|t_{ij}) = \lambda c(t_{ij}+x)^{c-1} e^{-\lambda(-t_{ij}^c+(t_{ij}+x)^c)} \tag{3}$$

with the CDF given as

$$P(X < t_{ij}+x|t_{ij}) = 1 - e^{-\lambda(-t_{ij}^c+(t_{ij}+x)^c)} \tag{4}$$

where i is the node that fails and j is the j^{th} system re-start.

4. Cloud System Reliability

In general, there could be many possible combinations of failure dependencies among components. However, we practically focus on four major scenarios described as follows. We consider four possible scenarios.

(1) *Software failures occur independently. Also, hardware failures occur independently.* From Eqs. (1) and E(2) one can show that the reliability model of n software components running on k physical nodes can be expressed as follows:

$$\begin{aligned} R_j(x) = F(x) &= P(X_1 > x, X_2 > x, \ldots, X_k > x, X_{k+1} > x, \ldots, X_{k+n} > x) \\ &= \exp\{-\sum_{i=1}^{k} \lambda_i x_i^{'} - \sum_{v=1}^{n} \gamma_v x\} \end{aligned} \tag{5}$$

where
$x_i^{'} = x^c$,if the i^{th} node is replaced at the j^{th} re-start
$= -t_{ij}^c + (t_{ij}+x)^c$,if the i^{th} node is not replaced at the j^{th} re-start

Similarly, for $x_s^{'}, x_l^{'}, \ldots, x_k^{'}$. Note that for independence, only λ_i s and ν_i s are not equal to zero. Eq.(5) can be readily derived from Eq.(1) by using the transformation, $c > 0$ for the nodes that have a Weibull time to failure (failed and were replaced to restart the system), $Y_i = -t_i^c + (t_i + X)^c$ for nodes that have an excess Weibull distribution (did not fail when the system was re-started) and $Y = X$ for the n software components that have an exponential time to failure.

(2) *In case of correlated software failures, and independent hardware failure,* the reliability model can be expressed as in Eq. (6), which is a product of Eq. (1) and Eq. (2). Note that for independent hardware failure only λ_is are not equal to zero.

$$R_j(x) = \exp\{-\sum_{i=1}^{k} \lambda_i x_i^{'} - \sum_{v=1}^{n} \gamma_v x - \sum_{v,w=1}^{n} \gamma_{vw} x - \sum_{v,w,z=1}^{n} \gamma_{vwz} x - \ldots - \gamma_{12\ldots n} x\}, \tag{6}$$

where $v < w < z$.

In a similar manner, we derive Eq. (7) and Eq. (8) for the scenarios described below in case 3 and case 4.

(3) *For a system where software failures occur independently, but hardware failures are correlated,* the reliability model is given by

$$R_j(x) = \exp\{-\sum_{i=1}^{k}\lambda_i x_i^{'} - \sum_{i,s=1}^{k}\lambda_{is}\max\left\{x_i^{'}, x_s^{'}\right\} - \sum_{i,s,l=1}^{k}\lambda_{isl}\max\left\{x_i^{'}, x_s^{'}, x_l^{'}\right\} - \\ \ldots - \lambda_{123\ldots k}\max\left\{x_1^{'},, x_k^{'}\right\} - \sum_{v=1}^{n}\gamma_v x\}, \tag{7}$$

where $i < s < l$.

(4) *In the case of both software and hardware correlated failures*, the reliability model of the cloud system is given by

$$\begin{aligned} &R_j(x) = \\ &= \exp\{-\sum_{i=1}^{k}\lambda_i x_i^{'} - \sum_{i,s=1}^{k}\lambda_{is}\max\left\{x_i^{'}, x_s^{'}\right\} - \sum_{i,s,l=1}^{k}\lambda_{isl}\max\left\{x_i^{'}, x_s^{'}, x_l^{'}\right\} - \ldots \\ &- \max\left\{x_1^{'}, \ldots, x_k^{'}\right\} - \sum_{v=1}^{n}\gamma_v x - \sum_{v,w=1}^{n}\gamma_{vw}x - \sum_{v,w,z=1}^{n}\gamma_{vwz}x - \ldots - \gamma_{12\ldots n}x\} \end{aligned} \tag{8}$$

where $i < s < l$ and $v < w < z$.

In all four scenarios, we assume that hardware and software fail independently from each other. From Eq. (5), the pdf of an application on Cloud can be derived as

$$f(x) = [c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{v=1}^{n}\gamma_v] * \exp\{-\sum_{i=1}^{k}\lambda_i x_i^{'} - \sum_{v=1}^{n}\gamma_v x\}, \tag{9}$$

$x_i^{''} = x^{c-1}$, if the i^{th} node is replaced at the j^{th} re-start
$=(t_{ij}+x)^{c-1}$, if the i^{th} node is not replaced at the j^{th} re-start

From Eq. (6), the pdf for this case is

$$\begin{aligned} f(x) = &[c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{v=1}^{n}\gamma_v + \sum_{v,w=1}^{n}\gamma_{vw} + \sum_{v,w,z=1}^{n}\gamma_{vwz} + \ldots + \gamma_{12\ldots n}] \\ &* \exp\{-\sum_{i=1}^{k}\lambda_i x_i^{'} - \sum_{v=1}^{n}\gamma_v x - \sum_{v,w=1}^{n}\gamma_{vw}x - \sum_{v,w,z=1}^{n}\gamma_{vwz}x - \ldots - \gamma_{12\ldots n}x\}, \end{aligned} \tag{10}$$

where $v < w < z$.

The pdf that is derived from Eq. (7) is given as

$$\begin{aligned} f(x) = &[c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{i,s=1}^{k}\lambda_{is}x_{is}^{''} + \sum_{i,s,l=1}^{k}\lambda_{isl}x_{isl}^{''} + \ldots + \lambda_{123\ldots k}x_{123\ldots k}^{''} + \sum_{v=1}^{n}\gamma_v] \\ &*\exp\{-\sum_{i=1}^{k}\lambda_i * x_i^{'} - \sum_{i,s=1}^{k}\lambda_{is}\max\left\{x_i^{'}, x_s^{'}\right\} - \sum_{i,s,l=1}^{k}\lambda_{isl}\max\left\{x_i^{'}, x_s^{'}, x_l^{'}\right\} - \ldots \\ &- \lambda_{123\ldots k}\max\{x_1^{'}, \ldots, x_k^{'}\} - \sum_{v=1}^{n}\gamma_v x\}, \end{aligned} \tag{11}$$

where $i < s < l$, and

$$x_{is}^{''} = \frac{\text{dmax}\{x_i^{'}, x_s^{'}\}}{\text{d}x} \text{ and } x_{isl}^{''} = \frac{\text{dmax}\{x_i^{'}, x_s^{'}, , x_l^{'}\}}{\text{d}x} \text{ and } x_{12\ldots k}^{''} = \frac{\text{dmax}\{x_i^{'}, ..., x_k^{'}\}}{\text{d}x}.$$

The pdf from Eq. (8) can be expressed as

$$\begin{aligned} f\left(x\right) &= [c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{i,s=1}^{k}\lambda_{is}x_{is}^{''} + \sum_{i,s,l=1}^{k}\lambda_{isl}x_{isl}^{''} + \ldots + \lambda_{1\ldots k}x_{123\ldots k}^{''} + \\ &+ \sum_{v=1}^{n}\gamma_v + \sum_{v,w=1}^{n}\gamma_{vw} + \sum_{v,w,z=1}^{n}\gamma_{vwz} + \ldots + \gamma_{1\ldots n}] \\ &*\exp\{-\sum_{i=1}^{k}\lambda_i * x_i^{'} - \sum_{i,s=1}^{k}\lambda_{is}\max\left\{x_i^{'}, x_s^{'}\right\} - \sum_{i,s,l=1}^{k}\lambda_{isl}\max\left\{x_i^{'}, x_s^{'}, x_l^{'}\right\} - \ldots \\ &-\lambda_{1\ldots k}\max\left\{x_1^{'}, \ldots, x_k^{'}\right\} - \sum_{v=1}^{n}\gamma_v x - \sum_{v,w=1}^{n}\gamma_{vw}x - \sum_{v,w,z=1}^{n}\gamma_{vwz}x - \ldots - \gamma_{1\ldots n}x\}, \end{aligned} \tag{12}$$

where $i < s < l$ and $v < w < z$.

After we have the reliability and pdf of an application, we will use this information to compute failure rate and mean time to failure of an application on Cloud.

4.1. Failure rate and Mean time to failure

The failure rate "$\lambda_i(x)$" of independent failure of software and hardware after the j^{th} re-start of the system is given by

$$\lambda_j = \frac{f\left(x\right)}{R_j\left(x\right)} = c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{v=1}^{n}\gamma_v \tag{13}$$

For the case of dependent software failure and independent hardware failure, the failure rate is:

$$\lambda_j = \frac{f\left(x\right)}{R_j\left(x\right)} = c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{v=1}^{n}\gamma_v + \sum_{v,w=1}^{n}\gamma_{vw} + \sum_{v,w,z=1}^{n}\gamma_{vwz} + \ldots + \gamma_{12\ldots n}, \tag{14}$$

where $v < w < z$.

The failure rate for the case of independent software failure and dependent hardware failure is:

$$\lambda_j = c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{i,s=1}^{k}\lambda_{is}x_{is}^{''} + \sum_{i,s,l=1}^{k}\lambda_{isl}x_{isl}^{''} + \ldots + \lambda_{123\ldots k}x_{123\ldots k}^{''} + \sum_{v=1}^{n}\gamma_v, \tag{15}$$

where $i < s < l$.

In the case of dependent software and hardware failures, the failure rate is given as:

$$\lambda_j = \frac{f(x)}{R_j(x)} = c\sum_{i=1}^{k}\lambda_i x_i^{''} + \sum_{i,s=1}^{k}\lambda_{is}x_{is}^{''} + \sum_{i,s,l=1}^{k}\lambda_{isl}x_{isl}^{''} + \ldots + \lambda_{123\ldots k}x_{123\ldots k}^{''} + \sum_{v=1}^{n}\gamma_v + \sum_{v,w=1}^{n}\gamma_{vw} + \sum_{v,w,z=1}^{n}\gamma_{vwz} + \ldots + \gamma_{12\ldots n} \quad , \tag{16}$$

where $i < s < l$ and $v < w < z$.

The mean time to failure (MTTF) of an application on Cloud is given by:

$$E(x) = \int_0^{\infty} x * f(x)\,\mathrm{d}x, \tag{17}$$

where $f(x)$ can be the pdf of the any case presented in Eqs. (9), ((10), (11), and (12) for a cloud system.

5. Cloud System Reliability, Mean Time to Failure (MTTF), and Failure Rate

In this section we show by example, considering for easy understanding that $k = 3$ and $n = 12$, the effect of survival time (T) and parameter values for joint node failures $\lambda_{ij\ldots}$ on system reliability for the four cases described above in Eqs. (5), (6), (7), (8), and failure rates from Eqs. (13), (14), (15) and (16) and the MMTF from Eq. (17). For simplicity we choose only one joint parameter $\gamma_{12\ldots n}$ .The example has a job run length (x) equals 100 hours and node parameters $\lambda_1, \lambda_2, \lambda_3 = 0.00005$, the 2-node joint parameters $\lambda_{12}, \lambda_{23}, \lambda_{13} = 0.00005$, the 3-node joint parameter $\lambda_{123} = 0.00003, C = 1.5$, the 2 software components joint parameters $\gamma_{12}, \cdots = 0.00003$, the 3 software components joint parameter $\lambda_{123} = 0.0007$, and the hypervisor and VM parameter $\lambda = 0.00001$.

It is obvious from the reliability equations (5), (6), (7) and (8) that when the system has failures that are correlated it becomes less reliable. Moreover, when the joint failure rates $\lambda_{ij\ldots}$ increase, the reliability of the system decreases. Figure 2a shows the reliability for the four scenarios. It is seen, as expected, that scenario 1 is most reliable and scenario 4 is least reliable. For all the four scenarios, as survival time (T) increases, the reliability decreases.

It is worth mentioning that as the number of nodes or VMs increases, the reliability of the system decreases, as seen from Eqs. (7–10). Figure 2b shows the failure rate for the four scenarios. Is it seen, as expected, that scenario 1 has the smallest failure rate and scenario 4 has the largest failure rate. Because of hardware aging (Weibull and conditional Weibull time to failure) in the model, failure rate increases with survival time.

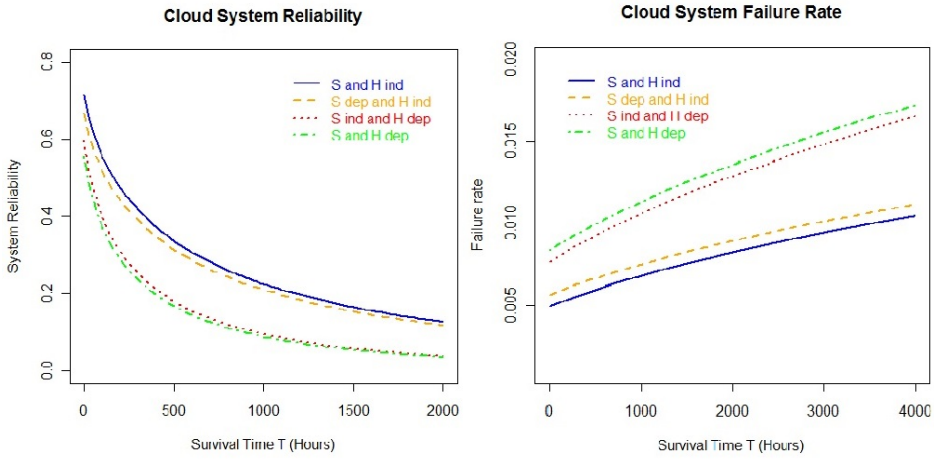

Fig. 2. a) Cloud System Reliability, k=3 and n=12, the four scenarios in Eqs. (5) to (8); **b)** Cloud System Failure Rate, with k=3 and n=12, the four scenarios represented by Eqs. (13) to (16).

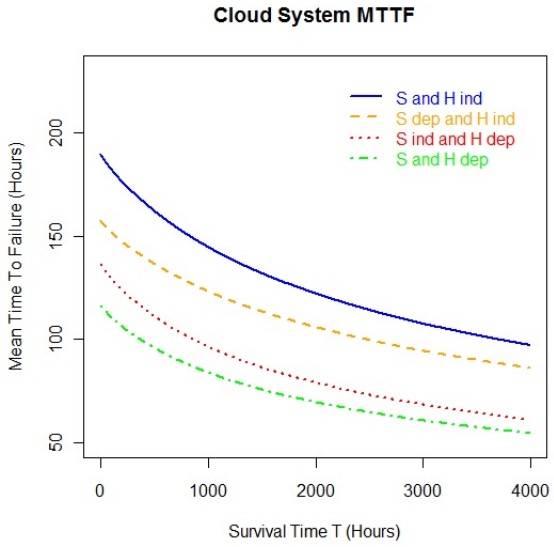

Fig. 3. Cloud System MTTF, with k=3 and n=12, the four scenarios.

Figure 3 shows the MTTF for the four scenarios. Is it seen, as expected, that scenario 1 has the largest MTTF and scenario 4 has the smallest MTTF. As expected, the MTTF decreases with an increase in survival time. Figure 4a shows the Cloud system reliability as function of nodes. We assume that each physical node has 4 VMs. As the number of nodes increases, the reliability decreases. Figure 4b shows

the Cloud system failure rate when the number of nodes increases. The failure rate increases when the system has more nodes.

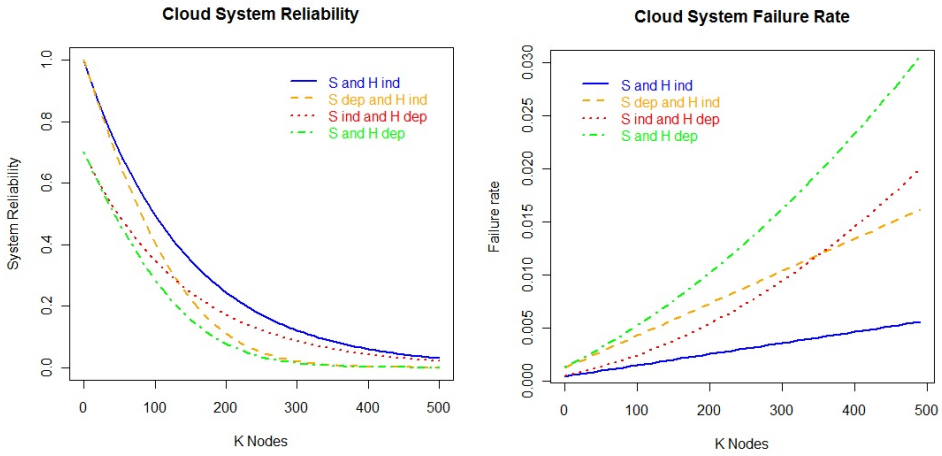

Fig. 4. a) Cloud System Reliability as function of the number of nodes (k); **b)** Cloud System Failure Rate as function of the number of nodes (k).

Figure 5 shows the Cloud system MTTF. When the number of nodes increases, Mean time to failure decreases. Also, when adding more components and the failures of components have correlation, the system becomes less reliable. That means less reliability, more failure rate, and shorter MTTF.

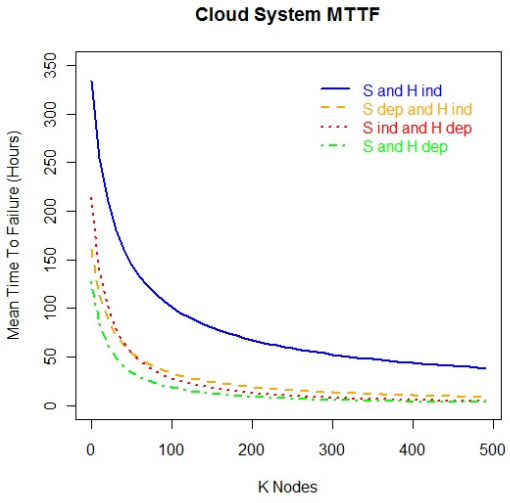

Fig. 5. Cloud System MTTF as function of the number of nodes (k).

6. Optimal Checkpoint Interval for HPC Applications in Cloud environment

In this section, we develop optimal checkpoint models for four scenarios. We first illustrate the checkpoint notation as shown in Figure 2. The checkpoint placement time (t_p) is the time to execute the checkpoint ($0 = t_1 < t_2 < \cdots < t_n$), we have illustrated t_p for $p = 1, 2$. Checkpoint overhead (T_s) is the time spent for saving the computing stage. Recovery Time (T_r) is the time spent for recovering the system after failure. Checkpoint Interval (T_c) is the time between two consecutive checkpoints. Rollback time (T_b) is the time to rollback from failure to the last checkpoint.

Fig. 6. Checkpoint Notation.

To find the optimal checkpoint interval, we consider the optimal checkpoint frequency [15].

$$n^*(t) = \sqrt{\frac{k * f(t)}{T_s * R(t)}}, \tag{18}$$

where f(t) is the system failure pdf, R(t) is the system reliability function and k is the rollback coefficient ($0 < k < 1$). The rollback coefficient (k) is the ratio between the rollback time and the checkpoint interval where the failure occurs.

$$k = \frac{T_b}{t_{p+1} - t_p} \tag{19}$$

To estimate k, one has to find the expected rollback time (T_b). We first consider the conditional probability that the system survives until time $t + s$ given that the system survived until time t. We denote the CDF, the pdf, and the expected value of the conditional probability as follows:

$$\begin{aligned} F(t+s) &= P(t+s|\mathrm{t}) \\ f(t+s) &= \frac{\mathrm{d}F(t+s)}{\mathrm{d}s} \\ E(S) &= \int_0^\infty s f(t+s|\mathrm{t})\,\mathrm{d}s \end{aligned} \tag{20}$$

The conditional CDF of each reliability scenario is as follows:

1) Software failures occur independently. Also, hardware failures occur independently

$$F(t+s) = 1 - \exp\{-\sum_{i=1}^{k} \lambda_i x_{i_{p+1}}^{'} - \sum_{v=1}^{n} \gamma_v x_{p+1} + \sum_{i=1}^{k} \lambda_i x_{i_p}^{'} + \sum_{v=1}^{n} \gamma_v x_p\} \tag{21}$$

$x_{i_{p+1}}^{'} = (x+s)^c$, if i^{th} node replaced at j^{th} re-start, $(p+1)^{th}$ checkpoint
$= -t_{ij}^c + (t_{ij} + x + s)^c$, if i^{th} node not replaced, j^{th} re-start, $(p+1)^{th}$ checkpoint

and

$x_{i_p}^{'} = x^c$, if the i^{th} node replaced at j^{th} re-start, p^{th} checkpoint
$= -t_{ij}^c + (t_{ij} + x)^c$, if i^{th} node not replaced at j^{th} re-start, p^{th} checkpoint
Similarly, for $x_{s_{p+1}}^{'}, x_{l_{p+1}}^{'}, \ldots, x_{k_{p+1}}^{'}$ and $x_{s_p}^{'}, x_{l_p}^{'}, \ldots, x_{k_p}^{'}$.

2) Correlated software failures, and independent hardware failure

$$\begin{aligned}
&F(t+s) = \\
&= 1 - \exp\{-\sum_{i=1}^{k} \lambda_i x_{i_{p+1}}^{'} - \sum_{v=1}^{n} \gamma_v x_{p+1} - \sum_{v,w=1}^{n} \gamma_{vw} x_{p+1} - \sum_{v,w,z=1}^{n} \gamma_{vwz} x_{p+1} \\
&- \ldots - \gamma_{1\ldots n} x_{p+1} + \sum_{i=1}^{k} \lambda_i x_{i_p}^{'} + \sum_{v=1}^{n} \gamma_{v_p} x_p + \sum_{v,w=1}^{n} \gamma_{vw} x_p + \sum_{v,w,z=1}^{n} \gamma_{vwz} x_p \\
&+ \ldots + \gamma_{1\ldots n} x_p\}
\end{aligned} \tag{22}$$

where $v < w < z$.

3) Software failures occur independently, but hardware failures are correlated

$$\begin{aligned}
&F(t+s) = \\
&= 1 - \exp\{-\sum_{i=1}^{k} \lambda_i x_{i_{p+1}}^{'} - \sum_{i,s=1}^{k} \lambda_{is} \max\left\{x_{i_{p+1}}^{'}, x_{s_{p+1}}^{'}\right\} \\
&- \sum_{i,s,l=1}^{k} \lambda_{isl} \max\left\{x_{i_{p+1}}^{'}, x_{s_{p+1}}^{'}, x_{l_{p+1}}^{'}\right\} - \ldots - \lambda_{1\ldots k} \max\left\{x_{1_{p+1}}^{'}, \ldots, x_{k_{p+1}}^{'}\right\} \\
&- \sum_{v=1}^{n} \gamma_v x_{p+1} + \sum_{i=1}^{k} \lambda_i x_{i_p}^{'} + \sum_{i,s=1}^{k} \lambda_{is} \max\left\{x_{i_p}^{'}, x_{s_p}^{'}\right\} \\
&+ \sum_{i,s,l=1}^{k} \lambda_{isl} \max\left\{x_{i_p}^{'}, x_{s_p}^{'}, x_{l_p}^{'}\right\} + \ldots + \lambda_{1\ldots k} \max\left\{x_{1_p}^{'}, \ldots, x_{k_p}^{'}\right\} + \sum_{v=1}^{n} \gamma_v x_p\},
\end{aligned} \tag{23}$$

where $i < s < l$.

4) Both software and hardware correlated failures

$$
\begin{aligned}
&F(t+s) = \\
&= 1 - \exp\{-\sum_{i=1}^{k} \lambda_i x_{i_{p+1}}^{'} - \sum_{i,s=1}^{k} \lambda_{is} \max\left\{x_{i_{p+1}}^{'}, x_{s_{p+1}}^{'}\right\} \\
&- \sum_{i<s<l}^{k} \lambda_{isl} \max\left\{x_{i_{p+1}}^{'}, x_{s_{p+1}}^{'}, x_{l_{p+1}}^{'}\right\} - \ldots - \lambda_{123\ldots k} \max\left\{x_{1_{p+1}}^{'}, \ldots, x_{k_{p+1}}^{'}\right\} \\
&- \sum_{v=1}^{n} \gamma_v x_{p+1} - \sum_{v,w=1}^{n} \gamma_{vw} x_{p+1} - \sum_{v,w,z=1}^{n} \gamma_{vwz} x_{p+1} - \ldots - \gamma_{12\ldots n} x + \sum_{i=1}^{k} \lambda_i x_{i_p}^{'} \\
&+ \sum_{i,s=1}^{k} \lambda_{is} \max\left\{x_{i_p}^{'}, x_{s_p}^{'}\right\} + \sum_{i,s,l=1}^{k} \lambda_{isl} \max\left\{x_{i_p}^{'}, x_{s_p}^{'}, x_{l_p}^{'}\right\} + \ldots + \\
&+ \lambda_{123\ldots k} \max\left\{x_{1_p}^{'}, \ldots, x_{k_p}^{'}\right\} + \sum_{v=1}^{n} \gamma_v x_p + \sum_{v,w=1}^{n} \gamma_{vw} x_p + + \ldots + \gamma_{12\ldots n} x_p\}
\end{aligned}
\tag{24}
$$

where $i < s < l$ and $v < w < z$.

The expected rollback time of each checkpoint time (t_p) can be calculated as follows.

$$
E(T_{bp}) = \frac{\int_0^{t_{p+1}-t_p} s f(t_p + s|t_p)\, \mathrm{d}s}{\int_0^{t_{p+1}-t_p} f((t_p + s|t_p)\, \mathrm{d}s} \tag{25}
$$

Then, we can compute the expected k of the p^{th} checkpoint interval as follows.

$$
\overline{k_p} = \frac{E(T_{bp})}{t_{p+1} - t_p} \tag{26}
$$

Therefore, the expected k can be calculated as

$$
\overline{k} = \frac{\sum_{p=1}^{N} P_p \overline{k_p}}{\sum_{p=1}^{N} P_p} \tag{27}
$$

where N is the number of the checkpoint and $P_p = P(t_p + s \mid t_p)$.

Algorithm to estimate k

1. Assume $\widehat{k} = a$, where $a \in (0, 1)$.
2. Calculate the checkpoint time (t_p) from $\int_{t_p}^{t_{p+1}} n^*(t)\, dt = 1$
   where $t_0 = 0$, $p = 1, 2, 3, \ldots$ and $\widehat{k}$ from step 1.
3. Calculate $\overline{k}$ from Equations (5), (6) and the checkpoint time from step 2.
4. If $\overline{k} = \widehat{k}$, then $k = \overline{k} = \widehat{k}$ END
   else repeat step 1.

7. Numerical Result for the Optimal Checkpoint Interval

In this section, we demonstrate the optimal checkpoint interval for each reliability case. We assume the node parameters $\lambda_1, \lambda_2, \lambda_3 = 0.00005$, the 2-node joint parameters $\lambda_{12}, \lambda_{13}, \lambda_{23} = 0.00005$, the 3-node joint parameter $\lambda_{123} = 0.00003, C = 1.5$, the 2 software-component joint parameters $\gamma_1, \ldots, \gamma_{12} = 0.00003$, the 3 software-component joint parameter $\gamma_{123} = 0.00003$ with the number of nodes (N) equals 3 and each node has 2 VMs. Figure 7 shows the optimal checkpoint interval versus checkpoint time for all four reliability case. As expected, the scenario where both software and hardware failures are independent has the longest checkpoint interval since it is the most reliable case. Also, the least reliable case (both software and hardware failures are dependent) has the shortest checkpoint interval.

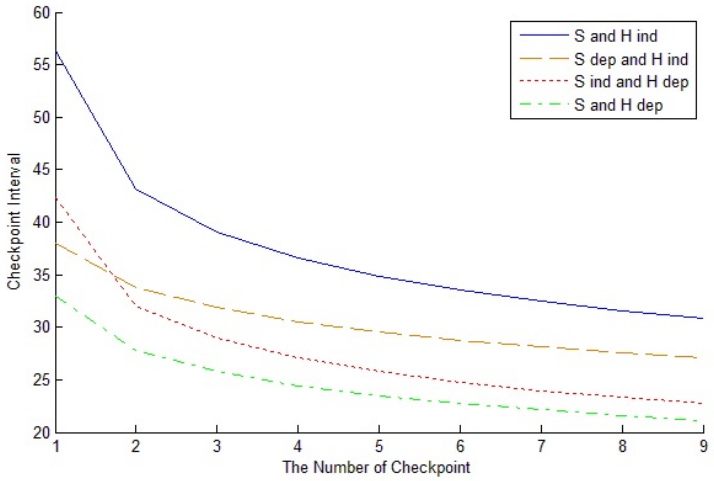

Fig. 7. Checkpoint Interval T_c as a function of the number of checkpoints for each reliability scenario.

8. Conclusion

Reliability information is important for real-world applications that may face failure or service interruption issues. In order to mitigate the interruption, it is of importance to estimate how the reliability, failure rate, and mean time to failure of a Cloud computing system may be affected by various outages. Our studies have shown the existence of failure correlations among nodes in computer systems. In this work, we proposed reliability models that consider software and hardware components and account for joint failures among nodes as well as VM's. We also developed the reliability, failure rate, and mean time to failure of a cloud-based system with k nodes and s VM's. We considered four major scenarios that are combinations of software, VM and hardware failure correlation. We also discussed the reliability model for the case

where VM's and Hypervisors exhibit an aging effect. In addition, we also developed the optimal checkpoint placement for each four reliability scenarios. Results showed that if failures in the system possess a degree of dependency, the system becomes less reliable. Moreover, the optimal checkpoint interval decreases when the reliability of the system decreases. Future work will focus on extending the present models to include component redundancies.

Acknowledgement. This research has been supported by the Romanian National Authority for Scientific Research, *Nucleu National Research Program* PN 09-36, BIODIV.

References

[1] *Amazon AWS Case Studies: High Performance Computing*, `http://aws.amazon.com/solutions/case-studies/`, last accesed January 2014.

[2] BUNCIC P., AGUADO SANCHEZ C., BLOMER J., HARUTYUNYAN A., MUDRINIC M., *A practical approach to virtualization in HEP*, The European Physical Journal Plus, 2011, **126**(1) pp. 1–8.

[3] DAI Y.S., YANG B., DONGARRA J., ZHANG G., *Cloud Service Reliability: Modeling and Analysis*, The 15th IEEE Pacific Rim International Symposium on Dependable Computing, 2009.

[4] EVANGELINOS C., HILL C.N., *Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2*, 1st Workshop on Cloud Computing and its Applications (CCA), 2008.

[5] GOSEVA-POPSTOJANOVA K., TRIVEDI K.S., *Failure Correlation in Software Reliability Models*, IEEE Transactions on Reliability, 2000, **49**(1), pp. 37–48.

[6] GUPTA A., MILOJICIC D., *Evaluation of HPC Applications on Cloud*, Technical Report, HPL-2011-132.

[7] GOTTUMUKKALA N. R. NASSAR R. PAUN M. LEANGSUKSUN C. B. SCOTT S. L., *Reliability of a System of k Nodes for High Performance Computing Applications*, IEEE Transactions on Reliability, 2010, Volume **59**, Issue 1, pp. 162–169.

[8] HAMLET D., *Are we testing for true reliability?*, IEEE Software, 1992, **9**(4), pp. 21–27.

[9] HACKER T.J., MEGLICKI Z., *Using queue structures to improve job reliability*, Proceedings of the 16th International Symposium on High-Performance Distributed Computing (HPDC-16 2007), 2007, Monterrey, CA, 2007, pp. 43–54.

[10] HACKER T.J., *Cloud Computing and Software Services: Theory and Techniques*, CRC Press 2011, pp. 139–152.

[11] HANAGAL D.D., *A multivariate Weibull distribution*, Economic Quality Control, 1996, **11**, pp. 193–200.

[12] HEATH T., MARTIN R.P., NGUYEN T.D., *Improving cluster availability using workstation validation*, SIGMETRICS Perform. Eval. Rev. **20**, 1, Jun 2002, pp. 217–227.

[13] *HPC-as-a-Service*, `http://www.mellanox.com/blog/2009/04/high-performance-computing-as-a-service-hpcaas/`, last accesed January 2014.

[14] HOGG R.V., MCKEAN J. W., CRAIG A.T., *Introduction to Mathematical Statistics*, 6th edition, Pearson, 2005.

[15] LIU Y., NASSAR R., LEANGSUKSUN C., NAKSINEHABOON N., PAUN M., SCOTT S.L., *An optimal checkpoint/restart model for a large scale high performance computing system*, IPDPS 2008, pp. 1–9.

[16] MARSHALL A.W., OLKIN I., *A multivariate exponential distribution*, Journal of the American Statistical Association, 1967, Volume **62**, pp. 30–44.

[17] MUSA J.D., *Software Reliability Engineering*, Osborne/McGraw-Hill, 1998.

[18] *Penguin computing*, `http://www.penguincomputing.com/`, last accesed January 2014.

[19] SCHROEDER B., GIBSON G.A., *A large-scale study of failures in high-performance computing systems*, Proceedings of International Symposium on Dependable Systems and Networks (DSN), IEEE Computer Society, 2006, pp. 249–258.

[20] THIRUMURUGAN S., PRINCE WILLIAMS D.R., *Analysis of Testing and Operational Software Reliability in SRGM based on NHPP*, International Journal of Computer and Information Engineering, 2007, **1**(1), pp. 284–289.

[21] VISHWANATH K.V., NAGAPPAN N., *Characterizing Cloud Computing Hardware Reliability*, International Conference on Management of Data, 2010, pp. 193–204.

[22] WOOD A., *Software Reliability Growth Models: Assumptions vs. Reality*, Proceedings of the Eighth International Symposium on Software Reliability Engineering (ISSRE '97), 1997, p. 136.

[23] XU J., KALBARCZYK Z., IYER R.K., *Networked Windows NT system field failure data analysis*, Proceedings of the 1999 Pacific Rim International Symposium on Dependable Computing, 1999, pp. 178–185.

[24] YANG B., XIE M., *A study of operational and testing reliability in software reliability analysis*, Reliability Engineering and System Safety Journal, 2000, **70**(3), pp. 323–329.