A Reliability Model for Cloud Computing for High Performance Computing Applications*

Thanadech Thanakornworakij[1], Raja F. Nassar[1], Chokchai Leangsuksun[1], and Mihaela Păun[1,2,**]

[1] College of Engineering & Science, Louisiana Tech University, Ruston, LA 71270, USA
[2] National Institute of Research and Development for Biological Sciences, Bucharest, 060031, Romania
mpaun@latech.edu

Abstract. With virtualization technology, Cloud computing utilizes resources more efficiently. A physical server can deploy many virtual machines and operating systems. However, with the increase in software and hardware components, more failures are likely to occur in the system. Hence, one should understand failure behavior in the Cloud environment in order to better utilize the cloud resources. In this work, we propose a reliability model and estimate the mean time to failure and failure rate based on a system of k nodes and s virtual machines under four scenarios. Results show that if the failure of the hardware and/or the software in the system exhibits a degree of dependency, the system becomes less reliable, which means that the failure rate of the system increases and the mean time to failure decreases. Additionally, an increase in the number of nodes decreases the reliability of the system.

Keywords: Fault tolerance, Reliability, cloud computing, cloud performance.

1 Introduction

Cloud computing allows businesses to rent computing resources from cloud service providers, such as Amazon, Rackspace, etc., instead of investing to acquire computing facilities. Users can increase or decrease the amount of resources -such as storage, memory, and central processing unit (CPU) – as they need and pay per usage according to the amount of services or resources they need, like in a traditional public utility. When companies decide to use a cloud service, they have in mind a Service Level Agreement (SLA), which involves the reliability and performance of the Cloud within a desired time frame. Reliability of cloud systems directly relates to its performance. When a system fails, applications that are running on the cloud can be interrupted. If the system does not have any fault tolerant mechanisms such as live migration and checkpoint/restart, failed jobs have to re-run. If the system provides fault tolerant mechanisms there is still some performance loss; re-computing time and the

* This work was supported in part by the CNCS TE 97/2011and NSF CNS-0834483.
** Corresponding author.

time it takes to restart the system to function again. Therefore, it is important to accurately estimate the reliability of a cloud computing system in order to better mitigate faults and therefore effectively utilize its performance to achieve the SLA of cloud users. The demand for High Performance Computing (HPC) is increasing for solving advance scientific researches and some mission critical applications. Many scientific and HPC applications are running on Cloud.

In Cloud environment, service providers must manage numerous computing components such as processors, memory modules, storage, network switches, etc. The more components, the failures are likely to occur. A failure may interrupt an entire application, for instance Message Passing Interface (MPI) applications. MPI has been used in most parallel scientific applications. MPI on Cloud is an active research area. Raihan [28] analyzed HPC applications and tested MPI performance on Cloud. If the service providers know the failure characteristic of the cloud computing components, they can better manage the computing resources to tolerate the failures and sustain better performance [7], [8].

Reliability information is one of the key factors to consider in a cloud computing environment. As such, many studies considered reliability as a main factor in their research on performance and usage cost. The work of Artur [5] focused on monetary versus reliability balance based on Spot Instances in the Amazon Elastic Computing Cloud (EC2). The Spot Instances are idle resources. Users can bid a price for idle resources. Whenever the price of resources is equal or less than the bid price, the Spot instances are allocated to the users. On the other hand, if the price goes above the bid price, the Spot instances are deallocated without warning. This leads to the question of how to bid the Spot Instances with given SLA constraints. Prasad [6] presented resource allocation method for data processing considering not just CPU speed, memory usage, data throughput, and network speed, but also reliability. In his work, Prasad presented algorithms for partitioning data in order to gain better performance and resource allocation for optimal pricing and for meeting SLA constraints. None of the work considers reliability of Cloud for HPC applications. Our work will consider the reliability of a cloud computing application that includes both hardware and software.

In this paper, we propose a reliability model for a cloud computing system that considers software, application, Virtual Machine, Hypervisor, and hardware failures as well as correlation of failures within the software and hardware.

2 Related Work

Many researchers have studied cloud computing reliability. Kashi [1] studied characteristic of hardware failure and hardware repair in cloud computing. The result showed that when a server failed, it has more chance of failing again. Also, successive failures are fit best to an inverse curve. Moreover, Dai [3] introduced cloud service reliability. The reliability of a cloud service is considered to be the reliability of request stage failures multiplied by those of the execution stage. Hacker [4] considered hardware reliability and assumed a virtual machine per server. The proposed model 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

Many studies have shown that the time to failure (TTF) of a computer system can be described by a Weibull distribution [4], [9], [11]. 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 [11] also showed TTF's of Windows NT servers follow a Weibull distribution. Gottumukkala et. al, [12] developed the reliability model of a k-node system for HPC applications when individual TTF follows a Weibull distribution. They considered the excess life or time since the last failure of an individual node in the reliability model to gain more accurate estimation of reliability of the system based on an assumption that nodes fail independently. However, time to failure of some computer systems may not be independent [11], [13]. Xu [11] showed that there is failure dependency of Windows NT Servers across the network. Schroeder [13] presented a study of failures in an HPC system at Los Alamos National Laboratory (LANL) 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 the excess life as well as correlation due to possible simultaneous failures of nodes in the system.

For software reliability, there are many studies undertaken to understand the characteristics of a software failure [14], [15]. Alan [10] was interested in understanding software reliability models and their utilityHe 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 [14] evaluated 7 model groups on 15 sets of data. Based on the evaluation, the exponential and logarithmic models were recommended for modeling software reliability. Thirumurugan [15] studied software reliability modeling in the testing and operational phase. Failures occurred at a constant rate over time [15], [16]. The majority of software reliability models make the assumption that failures occur independently. However, evidently in [17] the software failures are not always independent. Popstojanova [18] extended the classic software reliability theory, Markov renewal modeling, 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 [14], 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. Moreover, applications may fail at the same time, due to 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) [19] based on the fatal shock model shown, as follows.

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

Equation (1) is the probability that the system components each survive beyond times $y_1, \ldots, y_k$, respectively.

Suppose that a component fails after receiving a fatal shock. The occurrence of shocks is based on independent Poisson process $Z_i(t, \lambda_i)$, with *i = 1 2… k*. In each Poisson process, *t >0* and lambda is the Poisson parameter. The events in the Poisson process, $Z_i(t, \lambda_i)$ are shocks to component *i*, the events in the process, $Z_{i,j}(t, \lambda_{ij})$ are simultaneous shocks to both components *i* and *j*, the events in the process $Z_{i,j,l}(t, \lambda_{ijl})$ are simultaneous shocks to components *i, j* and *l* and the events in the process $Z_{1,2,\ldots,k}(t, \lambda_{i12\ldots k})$ are simultaneous shocks to all *k* components. Thus $\overline{F}_{Y_1 \ldots Y_k}(y_1 \ldots y_k)$ is a survival function of k components.

2.2 Multivariate Weibull

For node failure, we have evidence that nodes may fail simultaneously [13]. 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 [20]. In the survival function of the multivariate exponential Eq. (1), consider the transformation $Y_i = X_i^c$, c>0, $i = 1, \ldots, k$. Using the transformation of variable technique, one can obtain the survival function of the general multivariate Weibull (MVW) ,which is given by

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

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 of a Cloud System

In this section, we consider reliability of an application that has ℓ processes (App1-App ℓ) in a cloud system composed of s virtual machines deployed in k nodes as shown in Figure 1, where each node can have a different number of virtual machines. We make an assumption that it is an MPI application. A system 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.

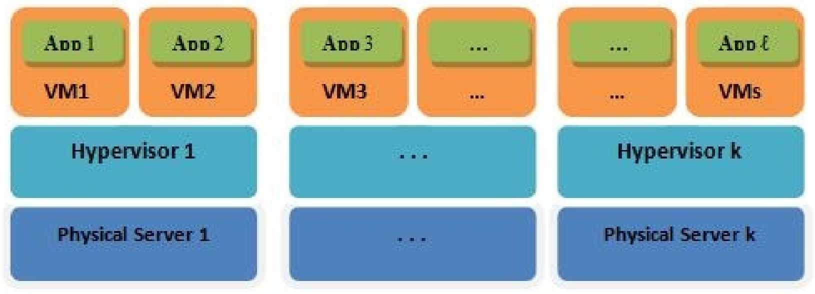

Fig. 1. Cloud Computing Architecture

Our interest is in determining the probability density function 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.

1. TTF of an individual node follows a Weibull distribution
2. TTF of an application that is running on a particular VM has an exponential distribution.
3. The first failure interrupts the entire application.
4. 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 = ℓ+s+k, see Figure 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

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

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.

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

From Eqs. (1) and (2) one can show that the reliability model of n software components running on k physical nodes can be expressed as follows:

$$R_j(x) = \overline{F}(x) = P(X_1 > x, X_2 > x \ldots X_{k+n} > x) = \exp\{-\sum_{i=1}^{k} \lambda_i x_i' - \sum_{v=1}^{n} \gamma_v x\}, \tag{4}$$

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 γ_i 's are not equal to zero.

Eq. (5) can be readily derived from Eq. (1) by using the transformation $Y_i = X_i^c$, 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.

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_i's are not equal to zero.

$$R_j(x) = \exp\{-\sum_{i=1}^{k} \lambda_i x_i' - \sum_{v=1}^{n} \gamma_v x - \sum_{\substack{v,w=1 \\ v<w}}^{n} \gamma_{vw} x - \sum_{\substack{v,w,z=1 \\ v<w<z}}^{n} \gamma_{vwz} x - \ldots - \gamma_{12\ldots n} x\} \tag{5}$$

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

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

$$\begin{aligned} R_j(x) = \exp\{&-\sum_{i=1}^{k} \lambda_i x_i' - \sum_{\substack{i,s=1 \\ i<s}}^{k} \lambda_{is} max\{x_i', x_s'\} - \sum_{\substack{i,s,l=1 \\ i<s<l}}^{k} \lambda_{isl} max\{x_i', x_s', x_l'\} - \ldots \\ &- \lambda_{12\ldots k} max\{x_1', x_2' \ldots x_k'\} - \sum_{v=1}^{n} \gamma_v x\} \end{aligned} \tag{6}$$

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

$$R_j(x) = \exp\{-\sum_{i=1}^{k} \lambda_i x_i^{'} - \sum_{\substack{i,s=1\\ i<s}}^{k} \lambda_{is} max\{x_i^{'}, x_s^{'}\} - \sum_{\substack{i,s,l=1\\ i<s<l}}^{k} \lambda_{isl} max\{x_i^{'}, x_s^{'}, x_l^{'}\} - ...$$
$$- \lambda_{12...k} max\{x_1^{'}, x_2^{'} ... x_k^{'}\} - \sum_{v=1}^{n} \gamma_v x - \sum_{\substack{v,w=1\\ v<w}}^{n} \gamma_{vw} x - \sum_{\substack{v,w,z=1\\ v<w<z}}^{n} \gamma_{vwz} x - ... - \gamma_{12...n} x\} \quad (7)$$

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

$$f(x) = [c\sum_{i=1}^{k} \lambda_i x^{''} + \sum_{v=1}^{n} \gamma_v] * \exp\{-\sum_{i=1}^{k} \lambda_i x' - \sum_{v=1}^{n} \gamma_v x\}, \quad (8)$$

where $x_i'' = x^{c-1}$ $if\ the\ i^{th}\ node$ 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

$$f(x) = [c\sum_{i=1}^{k} \lambda_i x^{''} + \sum_{v=1}^{n} \lambda_v + \sum_{\substack{v,w=1\\ v<w}}^{n} \lambda_{vw} + \sum_{\substack{v,w,z=1\\ v<w<z}}^{n} \lambda_{vwz} + ... + \lambda_{12...n}]$$
$$* \exp\{-\sum_{i=1}^{k} \lambda_i x' - \sum_{v=1}^{n} \lambda_v x - \sum_{\substack{v,w=1\\ v<w}}^{n} \lambda_{vw} x - \sum_{\substack{v,w,z=1\\ v<w<z}}^{n} \lambda_{vwz} x - ... - \lambda_{12...n} x\}, \quad (9)$$

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.

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

In this section, we show by examples, for k = 3 and n=12, the effect of survival time (T) and parameter values for joint node failures ($\lambda_{ij...}$) on system reliability for the four cases, Eqs. (5), (6), (7) and (8), and on the failure rate Eqs. (13), (14), (15) and (16) and MTTF, Eq. (17). For simplicity, we choose the example where there is only one joint parameter ($\gamma_{12...n}$). We use an example where job run length (x) equals 100 hours and 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 $\gamma_1, \ldots, \gamma_{12} = 0.00003$, $\text{the 3 software} - \text{component joint parameter}\ \gamma_{123} = 0.0007$, and the hypervisor and VM parameters λ=0.00001 . It is obvious from the reliability equations (5), (6), (7) and (6) 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. This implies that a system with independent nodes ($\lambda_{ij\ldots}, \gamma_{ij\ldots} = 0$) is more reliable than one where the nodes are correlated in failure. Figure 2 shows the reliability for the four scenarios. Is it seen, as expected, that scenario 1 is most reliable and scenario 4 is least reliable. For all scenario, 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. (5-8).

Figure 3 shows the Cloud system MTTF. When the number of nodes increases,, mean time to failure decreases. Also, when adding more components with correlated failure, the system becomes less reliable.

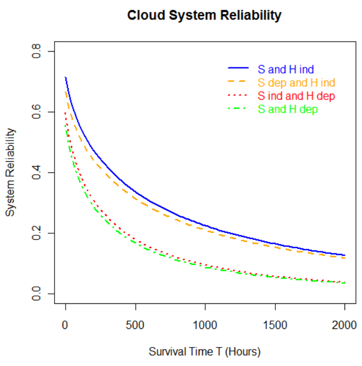

Fig. 2. Cloud System Reliability, with k=3 and n=12, for the four scenarios in Eqs. (5) – (8)

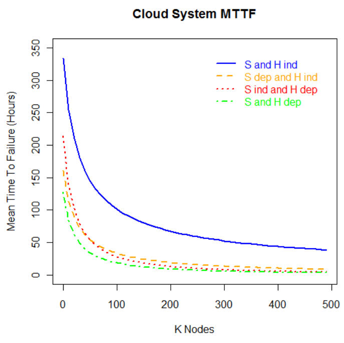

Fig. 3. Cloud System MTTF as function of the number of nodes (*k*)

6 Conclusions

Reliability information is important for improving Cloud computing system utilization. In order to mitigate the interruption of an application because of a failure, it is of importance to know how the reliability, failure rate, and mean time to failure of a Cloud computing system are affected by joint node failures. Many studies have shown the existence of failure correlations among nodes in computer systems [11] and [12]. In this paper, we proposed reliability models that consider software and hardware components, which account for joint failures among nodes as well as VMs. We also developed the reliability, failure rate, and mean time to failure of a system based on k nodes and s VMs. We consider four major scenarios that are combinations of software and hardware failure correlation. We also discussed the reliability model for the case where VMs and Hypervisors exhibit an aging effect. Results showed that if failures in the system possess a degree of dependency, the system becomes less reliable. Future work will focus on extending the present models to include component redundancies.

References

1. Vishwanath, K.V., Nagappan, N.: Characterizing Cloud Computing Hardware Reliability. In: International Conference on Management of Data, pp. 193–204 (2010)
2. Yi, S., Kondo, D., Andrzejak, A.: Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud. In: IEEE Cloud Computing, CLOUD, pp. 236–243 (2010)
3. Dai, Y.S., Yang, B., Dongarra, J., Zhang, G.: Cloud Service Reliability: Modeling and Analysis. In: The 15th IEEE Pacific Rim International Symposium on Dependable Computing (2009)
4. Hacker, T.J.: Cloud Computing and Software Services: Theory and Techniques, pp. 139–152. CRC Press (2011)
5. Andrzejak, A., Kondo, D., Yi, S.: Decision Model for Cloud Computing under SLA Constraints. In: International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (2010)
6. Prasad, K.H., Faruquie, T.A., Subramaniam, L.V., Mohania, M., Venkatachaliah, G.: Resource Allocation and SLA Determination for Large Data Processing Services over Cloud. In: Services Computing, SCC, pp. 522–529 (2010)
7. Bonvin, N., Papaioannou, T., Aberer, K.: Cost-efficient and Differentiated Data Availability Guarantees in Data Clouds. In: International Conference on Data Engineering, pp. 980–983 (2010)
8. Gueyoung, J., Joshi, K.R., Hiltunen, M.A.: Performance and Availability Aware Regeneration for Cloud Based Multitier Application. In: Dependable Systems and Networks, DSN, pp. 497–506 (2010)
9. Hacker, T.J., Meglicki, Z.: Using queue structures to improve job reliability. In: Proceedings of the 16th International Symposium on Hig-Performance Distributed Computing, HPDC-16 2007, Monterey, CA, pp. 43–54 (2007)
10. Wood, A.: Software Reliability Growth Models: Assumptions vs. Reality. In: The Proceedings of the Eighth International Symposium on Software Reliability Engineering, ISSRE 1997, pp. 136–141 (1997)
11. Xu, J., Kalbarczyk, Z., Iyer, R.K.: Networked Windows NT system field failure data analysis. In: Proceedings of the 1999 Pacific Rim International Symposium on Dependable Computing, pp. 178–185 (1999)
12. 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 59(1), 162–169 (2010)

13. Schroeder, B., Gibson, G.A.: A large-scale study of failures in high-performance computing systems. In: Proceedings of International Symposium on Dependable Systems and Networks, DSN, pp. 249–258. IEEE Computer Society (2006)
14. Musa, J.D.: Software Reliability Engineering. Osborne/McGraw-Hill (1998)
15. 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 1(1), 284–289 (2007)
16. Yang, B., Xie, M.: A study of operational and testing reliability in software reliability analysis. Reliability Engineering & System Safety Journal 70(3), 323–329 (2000)
17. Hamlet, D.: Are we testing for true reliability? IEEE Software 9(4), 21–27 (1992)
18. Goseva-Popstojanova, K., Trivedi, K.S.: Failure Correlation in Software Reliability Models. IEEE Transactions on Reliability 49(1), 37–48 (2000)
19. Marshall, A.W., Olkin, I.: A multivariate exponential distribution. Journal of the American Statistical Association 62, 30–44 (1967)
20. Hanagal, D.D.: A multivariate Weibull distribution. Economic Quality Control 11, 193–200 (1996)