International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Performance Monitoring and Improvement by VMM in Cloud System

Shafali Gupta[1], Payal Kulkarni[2]

[1]Professor, Department of Computer Engineering, RMD Sinhgad School of Engineering, University of Pune, India

[2]Department of Computer Engineering, RMD Sinhgad School of Engineering, University of Pune, India

Abstract: ***Cloud computing is a promising paradigm able to rationalize the use of hardware resources by means of virtualization. Virtualization allows to instantiate one or more virtual machines (VMs) on top of a single physical machine managed by a virtual machine monitor (VMM). Similarly to any other software, a VMM experiences aging and failures. It was always possible that, because of overload, non-responsive applications, bulky applications, system get slowed down or get hanged. In this case, every user who was using physical machine in virtual mode gets affected by system crash. Might be, he loses his important data, work. Also, task given to printer, scanner, CD-ROM or writer get affected resulting in loss. The paper focuses on solution of above mentioned problem. Virtual machine monitor (VMM) monitors every virtual machine. It monitors, running applications, memory usage, resources used, etc. If it comes to found any problem with any application, then VMM stops the executions such that terminate the corresponding applications. In emergency, to recall system from harmful crashes, it restarts the application. Priority is given to application termination and to save user work in limited problems and priority is given to system in case of dangerous problems.***

Keywords: Time –based rejuvenation, cloud computing, dynamic availability, phase type distribution

1. Introduction

Cloud computing a form of ubiquitous computing deals with providing everything as a service. Cloud computing is mainly used in business and IT industry which offers heavy outsourcing model computational resource, where service availability, security and quality are essential features. In cloud computing High service availability is the most important requirement increasingly being demanded in commercial computer, and communication systems. In recent years many research efforts have been going to find the optimal infrastructure size and configuration that guarantee the desired availability level. Software fault tolerance is often found to be the bottleneck. That is increase of failure rate in computer system is more due to software failures than hardware failures. Software often shows degradation in performance level after using it for long time [1] Most of the techniques for modeling and assessing software aging phenomena and rejuvenation policies implement analytical approaches, specifying the problem in terms of stochastic processes. The characterization of such processes is driven by the need to incorporate the software age into the model. This prevents the use of Markov models and similar techniques that assume "memory-less" behaviors [2]. Nevertheless, Markov models are used to model software aging whenever software age is approximated by discretizing aging in phases or epochs. Regardless the accuracy that can be reached, Markovian models do not allow to represent some aspects and behaviors related to specific software aging and rejuvenation processes. A contribution of this work is a non-Markovian analytical technique that allows investigating time-based rejuvenation strategies. Such technique is able to manage the intrinsic non-Markovian nature of the software aging process as well as the influence of the workload on the software behaviors. More specifically, we assume the VMM aging phenomenon is characterized by an increasing failure rate (IFR) distribution and depends on the number of VMs it is managing. Then, we characterize the VMM time to failure through continuous phase type (CPH) distributions [3]. The system availability is, thus, modeled by an expanded process that allows to keep memory of the age reached by the VMM when the number of hosted VMs changes according to the conservation of reliability principle. The expanded process is symbolically represented in terms of Kronecker algebra. This allows to formally representing the workload-dependent system behavior in a way that is intuitive and easily implementable in a software tool [4]. We also address the state-space explosion problem affecting state-space models, especially when phase type (PH) expansion techniques are used. The main goal of time-based rejuvenation models is to find an optimal rejuvenation timer that allows minimizing some objective functions.

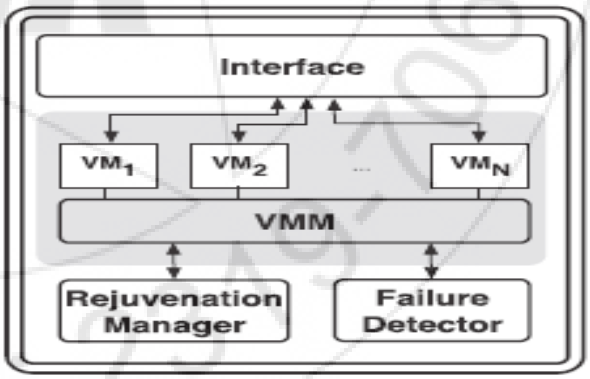

Figure 1: A rejuvenation enabled cloud node

2. Related Works

2.1Classification of Software Faults

Faults, in both hardware and software, can be classified according to their phase of creation or occurrence, system boundaries (internal or external), domain (hardware or software). Most modern studies on failure data have reported that a large percentage of software failures are transient in nature caused by phenomena such as overloads or timing and

exception errors. The revise of failure data indicate that 70% of the failures were transient failures, caused by faults similar to race conditions and timing problems. Aging related faults fall under Bohrbugs or Heisenbugs depending on whether the failure is deterministic (repeatable) or transient

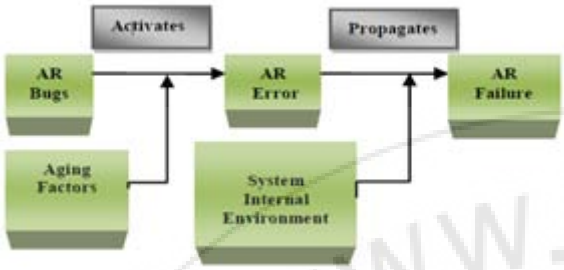

Figure 2: Chain of threats| for an (AR) aging-related failure

2.2 Software aging and rejuvenation

Many works have been reported on software aging and rejuvenation. Software aging is not a new topic still it lacks proper research background. Due to which recent safety critical applications also suffer from software aging. But, only few works have been reported on software aging in virtualized environment. In this chapter, we discuss the work done by previous researchers on many more other categories of memory related software aging both in physical machines and virtual machines. RivalinoMatias et al. [15] on 2012 has done research work on memory related software aging issues which causes aging related failures. They have mainly focused on memory leak problems. They have discussed important drawbacks of using well known system-wide and application-specific aging indicators, as well as propose effectual solutions for both cases. Memory-related aging effects are caused by memory leak and memory fragmentation problems. Memory leak is a software defect that is mainly caused by incorrect use of memory management routines. A memory leak occurs when an application process dynamically allocates memory blocks and, for some reason, does not release them back to the OS during its runtime. Here the authors have tried to find out memory leaks in a system both in user level and kernel level by the use of aging indicators. Aging indicators will detect the error in a system in running condition. System-wide aging indicators provide information related to subsystems components. With the help of system-wide aging indicators free/used physical memory and swap space they have conducted their experiment. But sometimes these indicators give false indication about memory consumption. So they have used aging free baseline to compare all the memory consumption with it for better result. Application-specific aging indicators provide specific information about an individual application process.

2.3. Software aging and rejuvenation in server virtualized System

Software aging in the virtual machine monitor (VMMs) as critical as server consolidation using virtual machines (VMs) is widely being carried out. A hypervisor or VMM is the piece of computer software that multiplexes physical resources such as CPU and memory to the VMs running on top of it. We review the three VMM rejuvenation techniques. When VMM rejuvenation needs to be performed on a host, the hosted VMs also need to be controlled because the execution environments of VMs are cleared by the VMM rejuvenation. VMM rejuvenation, we can perform VM shutdown (i.e., Cold-VM rejuvenation), Suspend (i.e., Warm-Rejuvenation), or VM migration (i.e., Migrate-VM rejuvenation). These approaches are presented in the next three subsections

2.3.1 Cold-VM rejuvenation

The easiest way to deal with the hosted VMs before triggering rejuvenation of VMM is to shut down all the hosted VMs regardless of the execution states of the VMs. The VMs are then restarted in clean states after the VMM rejuvenation. This approach is called Cold-VM rejuvenation. All the transactions running on VMs are vanished by the Cold-VM rejuvenation [6]. An advantage of the Cold-VM rejuvenation, however, is that the rejuvenation action cleans all the aging states of the VMs in addition to the aging states of the VMM.

2.3.2 Warm-VM rejuvenation

Instead of shutting down the hosted VMs, the hosted VMs are suspended prior to VMM rejuvenation is triggered and the executions of the VMs are resumed at the completion of the VMM rejuvenation. We call this technique Warm-VM rejuvenation [5]. Since the execution states of the hosted VMs are saved prior to VMM rejuvenation, the transactions running on the VMs are not lost due to the VMM rejuvenation. However, Warm-VM rejuvenation retains the aging states of VMs by VM suspend. The aging states in the hosted VMs are not cleared by VMM rejuvenation and hence we need to rely on rejuvenation for VM to clear the aging states of VMs.

2.3.3 Migrate-VM rejuvenation

Live VM migration is a technique to move a running VM to another host incur a short service interruption and is supported in most modern VMM implementations such as Xen and VMware. Although a shared storage system is required to store a VM image, the downtime overhead caused by a VM migration is less. Using live VM migration, hosted VMs are moved to another host prior to VMM rejuvenation and returned back to the original hosting server after the completion of the rejuvenation of the VMM, by a reverse live VM migration. We call this combined method as Migrate-VM rejuvenation [6]. The VM continues the execution even while the VMM on the original host is being rejuvenated. However, the aging states in the hosted VMs are not cleared by the VMM rejuvenation as in the case of Warm-VM rejuvenation. Live VM migration works only when the migration target server is running and it has a capacity to accept the migrated VM.

3. Proposed Work

The proposed statement of the system is to propose a new innovative approach to model software aging in cloud system and also in LAN network. Hence, we propose a technique to model and evaluate the VMM aging process and to investigate the optimal rejuvenation policy that maximizes the VMM availability under variable workload conditions.

Starting from dynamic reliability theory and adopting symbolic algebraic techniques, we investigate and compare existing time-based VMM rejuvenation policies. We also propose a time-based policy that adapts the rejuvenation timer to the VMM workload condition improving the system availability.

3.1 Module

These main two main modules are,

1. VMM
VMM is a virtual machine monitor, having higher level of physical machine. But in actual implementation, VMM is a virtual machine residing on a physical machine. Hence our application can consist of a VMM application located on physical machine, for which, here after, we are called as VMM Manager or only as Manager.

2. Node
The Node is nothing but a machine which is under monitoring by VMM Manager. Paper stated that it might be workstation or any other device connected in network. But as paper deals with only software failures, it must be workstation such that a physical machine connected in network. Hence, nodes are physical machines, might be slaves or clients connected in network, either internet or intranet (LAN), and Manager becomes their server.

4. Conclusion

The work can be carried out in two forms On the one hand an analytic technique that allows to represent any generic failure and repair distributions, adequately modeling changes in the workload through the conservation of reliability principle; on the other hand, a variable timer rejuvenation policy aiming at optimizing the software (VMM) availability in case of workload changes. The obtained results show that the proposed variable timer policy outperforms the fixed timer one, also considering different impact of the workload on the aging process

5. Acknowledgement

We would like to thank the principal and staff members of RMD Sinhgad School of Engineering, University of Pune, friends and family members for their support their valuable reviews and support to bring this article.

References

[1] L. Bittencourt, C. Senna, and E. Madeira, “Scheduling Service Workflows for Cost Optimization in Hybrid Clouds,” Proc. Int’l Conf. Network and Service Management, pp. 394-397, 2010.
[2] S. Pearson and A. Benameur, “Privacy, Security and Trust Issues Arising from Cloud Computing,” Proc. IEEE second Int’l Conf. Cloud Computing Technology and Science, pp. 693-702, 2010.
[3] R. Ghosh, K. Trivedi, V. Naik, and D.S. Kim, “End-to-End Performability Analysis for Infrastructure-as-a-Service Cloud:An Interacting Stochastic Models Approach,” Proc. IEEE 16th Pacific Rim Int’l Symp. Dependable Computing, pp. 125-132, 2010.
[4] S. Distefano, F. Longo, and M. Scarpa, “Availability Assessment ofHA Standby Redundant Clusters,” Proc. IEEE 29th Symp.Reliable Distributed Systems, pp. 265-274, 2010.
[5] M. Grottke and K.S. Trivedi, “Fighting Bugs: Remove, Retry, Replicate, and Rejuvenate,” Computer, vol. 40, no. 2, pp. 107-109, Feb. 2007.
[6] Y. Huang, C. Kintala, N. Kolettis, and N. Fulton, “SoftwareRejuvenation: Analysis, Module and Applications,” Proc. 25th Int’l Symp. Fault-Tolerant Computing (FTCS), pp. 381-390, 1995.
[7] K. Vaidyanathan and K.S. Trivedi, “A Comprehensive Model for Software Rejuvenation,” IEEE Trans. Dependable and Secure Computing, vol. 2, no. 2, pp. 124-137, Apr.-June 2005.
[8] S. Garg, A. Puliafito, M. Telek, and K. Trivedi, “Analysis of Software Rejuvenation Using Markov Regenerative Stochastic Petri Net,” Proc. Sixth Int’l Symp.Software Reliability Eng., pp. 180- 187, 1995.
[9] K. Vaidyanathan and K. Trivedi, “A Measurement-Based Model for Estimation of Resource Exhaustion in Operational Software Systems,” Proc. 10th Int’l Symp.Software Reliability Eng., pp. 84-93, 1999.
[10] S. Garg, A. van Moorsel, K. Vaidyanathan, and K. Trivedi, “A Methodology for Detection and Estimation of Software Aging,” Proc. Ninth Int’l Symp. Software Reliability Eng., pp. 283-292, 1998.

Author Profile

Shafali Gupta,Professor, Department of Computer Engineering, RMD Sinhgad School of Engineering, University of Pune, India

Payal Kulkarni, Research Scholar RMD Sinhgad School of Engineering Warje, Pune, University of Pune.