CHAPTER 2

LITERATURE REVIEW

2.1 Literature Review

Several researches have been previously done on the field of performance analysis of virtual machines across various virtualization platforms, and many performance tests have also been carried out. Lingfang Zeng, Yang Wang, Wei Shi and Dan Feng [3] have done a work in improving the performance of credit scheduler by making three improvements: load balancing of BOOST domains, prevention of premature preemption, and dynamic time slicing to boost the performance, and obtained an improvement in mean response time from the VCPUs running in the guest domain.

Chia-Ying Tseng and Po-Chun Huang, in their research work [4], have worked on modifying the Simple Earliest Deadline First (SEDF) algorithm, which is also a scheduler used by the Xen, by using Deadline-Monotonic scheduling strategy, in order to improve the performance under CPU-intensive and memory-intensive workloads in overloaded condition.

There has been one interesting attempt at enhancing the Xen hypervisor and enable it to handle the challenges imposed by real-time scheduling. Carried out as a joint project between different schools at Shanghai Jiao Tong University, it improves the default scheduler in Xen, tuning it for real-time execution. The result is proven to be a 20% improvement of handling real-time tasks [5].

A similar attempt has been carried out at Beijing University. Here it is also argued that the Xen hypervisor is not adapted to real-time tasks and changes or additions to it must be made in order to accommodate real-time guests. This method is also proved to be successful through experimental measurements [6].

Also from Shanghai comes a suggestion of an embedded real-time virtualization architecture based on the Kernel-based Virtual Machine hypervisor. Here the execution of the real-time tasks is handled by VxWorks, a real-time operating system, while the general operating system Linux is used to run general purpose tasks. Both these operating systems are installed as virtual machines on a KVM and Linux combination hypervisor. Tests were run to see how the real-time tasks were affected by running on a virtual machine together with other guests, and what kind of

latencies KVM introduced to the real-time operating system. An architecture based on the results were then proposed, where modifications to the host operating system was the solution [7].

Li et al. proposes a real-time scheduling mechanism for virtual machines based on list scheduling. Two algorithms have been designed: The Virtual Machine Management algorithm and the Processor Selection algorithm. According to the authors these designs have the potential to be successful [8].

Investigations of the Xen hypervisor's capabilities as a media server have been made by Lee et al. at the Georgia Institute of Technology and Avaya Labs respectively [9]. Classifying music and video playback as soft real-time tasks, they found that Xen is unable to support these kinds of task without notable performance loss. The problem was discovered to be because the virtual machines running the soft real-time tasks were not given enough CPU time. The Xen hypervisor uses a mechanism where tasks that are waiting for input or an event have their priorities boosted when one such arrives. The interesting thing was that the Xen host (known as dom0) spent almost all its time in this boosted state because it is almost always waiting for some input, effectively stealing the guest virtual machines CPU time. A guest can only be boosted from a blocked state, and when it is running a media application it is considered to be in a running state, making it impossible for them to ever compete with dom0 for execution time. Modifications were made and tests made with an IP telephony program showed that this new setup performed well and that non-real-time tasks were not affected by the changes.

A new real time multicore virtual machine scheduling framework in Xen VMM, RT-Xen 2.0 has been designed and implemented by Sisu Xi, Meng Xu, et.al, [10] in their work, which realizes global and partitioned VM schedulers, and each scheduler can be configured to support dynamic or static priorities, and to run VMs as periodic or deferrable servers. They showed that both global and partitioned VM scheduling can be implemented at moderate overhead, with observed improved performance of the virtual machines.