Virtualizing Intelligent River® : A Comparative Study of Alternative Virtualization Technologies

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science
Computer Science

---

by
Aravindh Sampath Kumar
December 2013

Accepted by:
Dr. Jason O. Hallstrom, Committee Chair
Dr. Amy Apon
Dr. Brian A. Malloy

Abstract

Emerging cloud computing infrastructure models facilitate a modern and efficient way of utilizing virtualized resources, enabling applications to scale with varying demands over time. The core idea behind the cloud computing paradigm is *virtualization*. The concept of virtualization is not new; it has garnered significant research attention, in a race to achieve the lowest overhead compared to bare-metal systems. The evolution has led to three primary virtualization approaches- *full-virtualization*, *para-virtualization*, and *container-based virtualization*, each with a unique set of strengths and weaknesses. Thus it becomes important to study and evaluate their quantitative and qualitative differences. The operational requirements of the Intelligent River® middleware system motivated us to compare the choices beyond the standard benchmarks to bring out the unique benefits and limitations of the virtualization approaches.

This thesis evaluates representative implementations of each approach: (i) full-virtualization - *KVM*, (ii) para-virtualization - *Xen*, and (iii) container-based virtualization - *LXC*. First, this thesis describes the scalable and resilient design of the Intelligent River® middleware system used as a reference application to evaluate the virtualization platforms. Second, this thesis describes the deployment of the application components on each of the test environments. Third, this thesis assesses the benefits and limitations of each based on their virtualization overhead, resource entitlement and isolation facilities, operational flexibility, scalability, and security.

The study presented in this thesis provides an improved understanding of available virtualization technologies. The results will be useful to architects in leveraging the best virtualization platform for a given application.

Chapter 1

Introduction

The adoption of cloud computing paradigm has changed the way we look at server infrastructure for next-generation applications. The cloud computing model has catalyzed a change from the traditional model of deploying applications on dedicated servers with limited room to scale to a new model of deploying applications on a shared pool of computing resources with (theoretically) unlimited scalability [10].

The technology backbone behind the idea of cloud computing is virtualization. Virtualization is the process of creating virtual devices that simulate the hardware which are in turn mapped to the physical resources on an underlying server. Virtualization primarily addresses the problem of under-utilization of computing resources in the dedicated server model. Virtualization maximizes the utilization of the hardware investment by running an increased number of isolated applications simultaneously on a physical server or "*host*". The isolated applications are run in an operating environment referred as *Virtual Machines (VM)* [15] or *Containers* [11]. The VMs / containers abstract the hardware interfaces required by the applications they run and utilize as much computational resources as they need (or are available). Virtualization opens the door for several added benefits:

Improved Fault-tolerance :- Virtual machines can be statefully migrated to other physical machines in the event of a hardware failure [4]. The fail-over can also be automatically triggered in some cluster-aware virtualization platforms [13]. This facility effectively enables continuous availability for critical applications, which would require application-level awareness to achieve in the dedicated server model.

Operational Flexibility :- Virtual Machines and their associated virtual devices are usually represented as a few files, which make them easy to clone, snapshot, and migrate to other physical machines. Also, individual VMs can be dynamically started or stopped without causing any outage to other VMs or the host.

New Avenues for Tuning and Customization :- Since virtualization platforms introduce an additional layer of control between applications and hardware, they enable more options to customize the environment for individual VMs. That also provide more points of control to administer the resources accessible to individual virtual machines.

Granular Monitoring :- The physical server that hosts VMs provides deeper insights and visibility into the performance, capacity, utilization and the overall health of the individual VMs. Analysis of the monitoring data facilitates resource management and control.

The benefits offered by the virtualization platforms form the core of the cloud computing ecosystem. Commercial cloud infrastructure providers have built their solutions around these benefits and offer "pay as you go" services where end-users are billed only for the resources used by the virtual machines or containers. Cloud providers pass on isolation and granular control options for the virtualization platform to end-users with a flexible interface, letting users keep complete control of their operating systems, storage environment, and networking setup without worrying about the underlying physical infrastructure. Large scale cloud computing platforms that lease their servers to virtualize applications of several end-users over a large pool of shared resources are exemplified by Amazon Elastic Compute Cloud (EC2) [cite], RackSpace [cite], and Heroku [cite].They differ by their choice of the virtualization platform. Enterprises and academic institutions also run a private cloud platform and have driven the development of open source cloud computing toolkits like OpenStack [cite], CloudStack [cite] and OpenShift [cite].

1.1 Motivation

The Intelligent River® is a large scale environmental monitoring platform that is being built by researchers at Clemson University [cite]. The Intelligent River® involves a large and distributed sensor network in the Savannah River basin and a real-time distributed middleware system hosted in Clemson University that receive, analyze and visualize the real-time environmental observation streams. The volume and unpredictable nature of the observation streams along with the increasing

scale of sensor deployments demanded a distributed middleware system that is flexible, fault-tolerant and designed for scale. Architecting he Intelligent River® middleware system posed an important design question,

Given the multitude of open source virtualization platforms, KVM, Xen, and Linux Containers, and the complex operational requirements of our middleware stack, which platform to choose ?

Our quest to find the answer to the question needed a detailed analysis of the common virtualization platforms beyond the available results of the standard benchmarks [cite]. Prior research work by Andew J.Younge et al. [cite] on analysis of the virtualization platforms for HPC applications shows that KVM performed better compared to Xen on the standard HPCC benchmarks whereas another research work published by Jianhua Che et al. [cite] using the standard benchmarks claims that OpenVZ, a virtualization platform based on Linux containers performed the best while KVM performed *significantly lower* than Xen which made clear that the comparison using the standard benchmarks were not enough. This motivated us to perform a detailed study of the principles behind the three major open source virtualization platforms, KVM, Xen, and Linux Containers and evaluate them based not just on the quantitative metrics like virtualization overhead and scalability but also on the qualitative metrics like operational flexibility, isolation, resource management, operational flexibility, and security.

1.2 Contributions

This research thesis builds on the prior work on comparing the open source virtualization platforms and brings out the advantages and disadvantages of each.The contributions of this thesis include (i) Description of the scalable and resilient design of our Intelligent River® middleware system. (ii) A study on the principles of operations behind the three chosen virtualization platforms which will be useful to the architects who design their applications around them. (iii) A quantitative comparison of the virtualization overhead (and scalability ?) exhibited by KVM, Xen and Linux containers as of date of writing. (iv) A discussion on the differences among the chosen platforms in terms of qualitative factors like ease of deployment, resiliency and security. (v) A discussion on the facilities to assure the resource entitlement and control with respect to CPU,Network bandwidth, Memory and I/O which is often overlooked and an area in need of improvement from the operational

perspective.

1.3 Thesis Organization

Chapter 2

Related Work

The core ideas of virtualization having stood the test of time have also prompted a variety of comparative studies by the academia as well as the industries. Andrew J. Younge et al. [21] compared Xen and KVM with virtual resources from the FutureGrid project [20] and claims, KVM performed significantly better than Xen, under the standard HPCC and SPEC benchmarks. Another benchmark oriented synthetic study [3] claims OpenVZ (the predecessor of Linux Containers) performed exceptionally well on almost all benchmarks. But, also claims that Xen performs significantly better than KVM contradicting the results of other publishers. A recent study by Miguel G. Xavier et al. [19] and an earlier study [16], again corroborates the fact that LXC performs significantly better than alternative virtualization platforms, Linux-VServer, OpenVZ, and Xen based on a variety of benchmarks including LINPACK, STREM, IOZONE, NETPIPE, and NPB. They also highlighted the shortcomings in isolation, and sharing in LXC which have already been addressed in the recent releases. A relatively old research work by Vincent Neri et al. [12] threw light on serious performance limitations on Xen under certain operational circumstances and also motivated the need for microbenchmarks.

Among the many new opportunities that were opened up by the relatively new and lightweight LXC, network virtualization or Software Defined Networking (SDN) attracted lots of research attention. Studies, [6], and [14] discussed the performance benefits of large scale virtual network emulation by implementing open Vswitch [18] on LXC based platforms. Mesos [8] , A clustered platform that provides fine-grained resource sharing among multiple frameworks like Hadoop [7] and MPI [5] utilizes LXC to isolate and limit the CPU, Memory, Network, and I/O resources. Also,

Mircea Bardac et al. [2] showcased the scalability of LXC by deploying a solution for testing large scale Peer-to-Peer systems.

Resource affinity, Isolation and Control are discussed in detail in this thesis. Vishal Ahuja et al. [1] provides an example of exploiting CPU isolation facilities to improve overall network throughput by pinning application, and protocol processing to the same processor core. The discussions in this thesis on providing resource affinity will enable Vishal Ahuja et al.'s work to be applicable under LXC. Another interesting attempt by Zhaoling Guo and Qinfen Hao [9] made use of cgroups (the isolation mechanism used by LXC) in combination with KVM to enable CPU affinity to achieve improved performance.

CoreOS [17], in its very early stages and being actively developed at the time of writing of this thesis is a very light weight operating system (Linux Kernel+systemd) that is built to run in containerized environments attempts to lower the overhead by eliminating redundant operating system functions.

Chapter 3

Virtualization Technologies

General introduction about virtualization. What types of virtualization we are going to talk in this thesis. Introduce LXC, KVM, Xen. Write about the technical background behind all three. specifically, LXC is an userspace wrapper over the linux containers facility merged into the Linux kernel. The idea of KVM and its isolation benefits as it allows running individual kernels for the VMs. The fundamental idea of para virtualization on which Xen is based on.

3.1 Linux Containers

Refer the notes and th eman pages of LXC for a solid technical discussion on LXC+ Containers

Add a block diagram that shows all the components used in the Linux Containers... cgroups, namespaces, individual filesystems etc..

Explain the block diagram in detail.

The device works in two modes:

1. Solar Powered: The device performs two tasks in solar powered mode. First, it supplies power to the circuit. Second, it charges the Li-Ion battery.

2. Battery Powered: The circuit operates in battery powered mode whenever there is insufficient solar power. The circuit uses power from the Li-Ion battery in this mode.

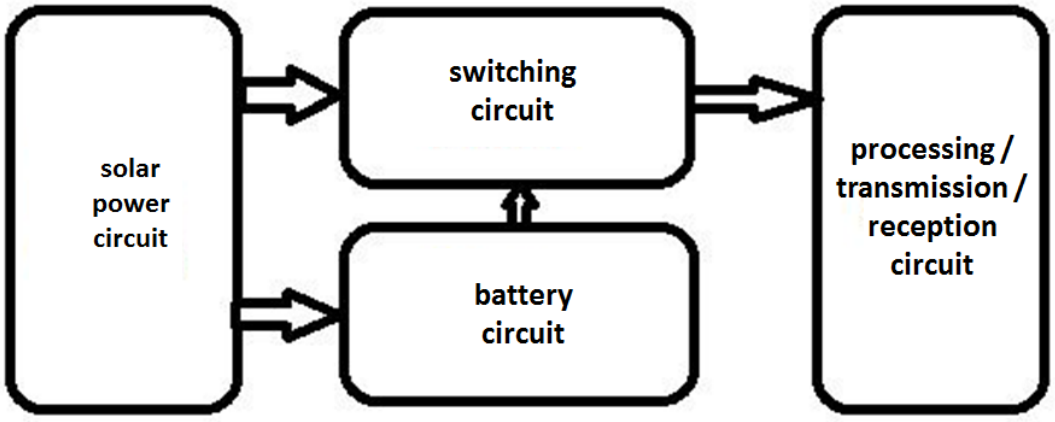

Figure 3.1: Block Diagram of the Mote Platform

3.2 Kernel based Virtual Machines (KVM)

A good technical discussion on how KVM works ... Refer the papers that already did this .... Add more info. Google it. Add a nice block diagram of KVM..

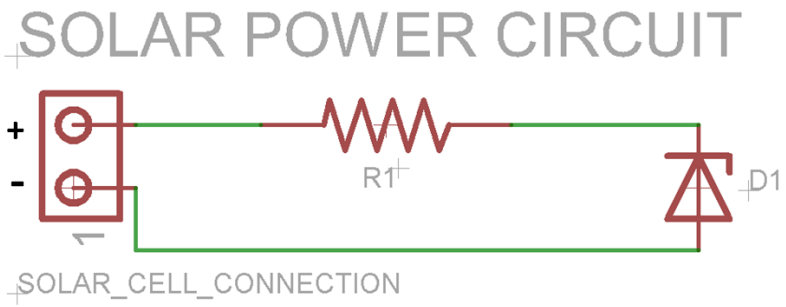

Figure 3.2: Solar Power Circuit

As shown in Figure 3.2, Explain the block diagram in detail ..... As shown in Figure 3.2,

Add more detailed tech discussions ... See anatomy of KVM, Developerworks article by Tim Jones ...

where,

R = Resistance ($ohms$)

V_s = Output voltage of solar cell ($volts$)

V_z = Breakdown voltage of zener diode ($volts$)

I_r = Amount of current required by circuit (mA)

3.3 Xen

Same as the above two .. A good technical introduction, followed by a block diagram and more detailed explanation.

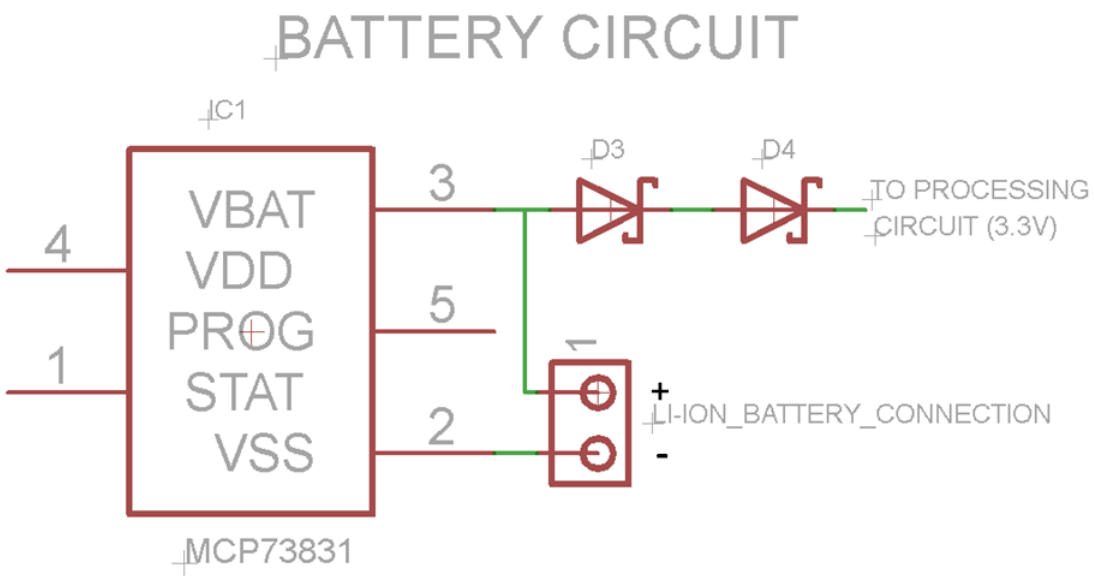

Figure 3.3: Battery Circuit

As shown in Figure 3.3,

Bibliography

[1] Vishal Ahuja, Matthew Farrens, and Dipak Ghosal. Cache-aware affinitization on commodity multicores for high-speed network flows. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, ANCS '12, pages 39–48, New York, NY, USA, 2012. ACM.

[2] M. Bardac, R. Deaconescu, and A.M. Florea. Scaling peer-to-peer testing using linux containers. In *Roedunet International Conference (RoEduNet), 2010 9th*, pages 287–292, 2010.

[3] Jianhua Che, Yong Yu, Congcong Shi, and Weimin Lin. A synthetical performance evaluation of openvz, xen and kvm. In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pages 587–594, 2010.

[4] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[5] MPI Commitee. Message passing interface. `http://www.mcs.anl.gov/research/projects/mpi/`, August 2013 (Last Accessed).

[6] C.N.A. Correa, S.C. de Lucena, D. de A.Leao Marques, C.E. Rothenberg, and M.R. Salvador. An experimental evaluation of lightweight virtualization for software-defined routing platform. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 607–610, 2012.

[7] Apache Software Foundation. hadoop. `http://hadoop.apache.org/`, August 2013 (Last Accessed).

[8] Apache Software Foundation. Mesos. `http://mesos.apache.org/`, August 2013 (Last Accessed).

[9] Zhaoliang Guo and Qinfen Hao. Optimization of kvm network based on cpu affinity on multicores. In *Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on*, volume 4, pages 347–351, 2011.

[10] Todd Hoff. Building super scalable systems. `http://highscalability.com/blog/2009/12/16/building-super-scalable-systems-blade-runner-meets-autonomic.html`, August 2013 (Last Accessed).

[11] Linux. Linux containers. `http://sourceforge.net/projects/lxc/`, August 2013 (Last Accessed).

[12] Benjamin Qutier, Vincent Neri, and Franck Cappello. Scalability comparison of four host virtualization tools. *Journal of Grid Computing*, 5(1):83–98, 2007.

[13] Inc Red Hat. Automatic virtual machine migration. https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Virtualization/3.0/html/Administration_Guide/Tasks_RHEV_Migration_Automatic_Virtual_Machine_Migration.html, August 2013 (Last Accessed).

[14] L. Sarzyniec, T. Buchert, E. Jeanvoine, and L. Nussbaum. Design and evaluation of a virtual experimental environment for distributed systems. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 172–179, 2013.

[15] J.E. Smith and R. Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, 2005.

[16] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, 41(3):275–287, March 2007.

[17] Open Source. Coreos. http://coreos.com/, August 2013 (Last Accessed).

[18] Open Source. Open vswitch. http://openvswitch.org/, August 2013 (Last Accessed).

[19] M.G. Xavier, M.V. Neves, F.D. Rossi, T.C. Ferreto, T. Lange, and C.A.F. De Rose. Performance evaluation of container-based virtualization for high performance computing environments. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 233–240, 2013.

[20] XSEDE. Futuregrid. https://portal.futuregrid.org/, August 2013 (Last Accessed).

[21] Andrew J Younge, Robert Henschel, James T Brown, Gregor von Laszewski, Judy Qiu, and Geoffrey C Fox. Analysis of virtualization technologies for high performance computing environments. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 9–16. IEEE, 2011.