Virtual Machine is the right platform for deploying Internet services
CS 553 Position Paper-II

Abstract

Virtual machines provide isolated execution domain which can be a very useful property for hosting Internet services. In this position paper, I will outline the features offered by Virtual machines and argue for the benefits made available by them to applications in general and Internet services in particular when deployed on such a platform. In particular I will present the inefficiency problem faced by current hosting facilities (in-house and third party) and discuss how the use of Virtual machines can help solve it.

1 Background

The concept of *Virtual Machines (VM)* was born in mid 1960s, when IBM scientists proposed the partitioning of a single machine into multiple (virtual) platforms, to be able to concurrently test multiple versions of their operating system products. The idea was extremely successful and IBM included the concept in their lineage of multi-user server machines. These machines namely, IBM S/370 and the S/390 were a major hit among businesses and academic/research institutions as they provided a way to share resources among users with no interference among concurrent users, since each one ran within an isolated VM. Thus, VMs were originally proposed as protection domains to share resources of a single physical machine in a non-interfering manner.

Though this was the predominant notion of a VM prior to 1990s, a slightly different line of thinking started to emerge in the early 1980s. Between 1980 and 1985, the IEEE standards board pushed the POSIX(Portable Operating System) project to standardize the library functions Unix-like operating systems. This was required for programmers to be able to write portable applications that could easily be run on many platforms. However, its impossible to enforce developers to conform completely to the POSIX specifications and many products display the label of "POSIX *compliant*" which implies partial *conformance* [4]. So, while the standard lives on, the quest to achieve device and operating system independence was not satisfied. The phrase *"...write once, run anywhere"*, continued to evade application developers.

The above problem was the primary motivation in the conception of Java and the Java Virtual Machine(JVM) by Sun. Its success, in turn sparked a rejuvenation of the VM concept. Originally intended (in 1991) for running portable applications on various smart, consumer electronic devices (without having to rewrite the device and OS specific parts), it was later remoulded as an *"Internet ready"* platform that ran on any hardware-OS combination computers. Finally programmers could write something which ran anywhere, provided the runtime was supported by the respective platform. However, it still had the disadvantage of having to use a language specific platform to write applications. This brings us to the question: Which is the right abstraction to have a virtualization layer? Is it above the OS closer to the application, or is it below the OS and closer to the hardware?

Along with commercial products such as VMware [11] that provide complete emulation of underlying hardware functionality, a few research groups have also shown great interest in developing language-independent VMs. The research institutions have focussed on what is termed *Virtual Machine Monitors (VMM)* [7, 12] capable of running multiple, isolated, protection domains each with its own guest operating system, very much in the lines of the VM/CMS components in the IBM VM lineage of machines mentioned above. Figure 1 depicts the architecture of such a

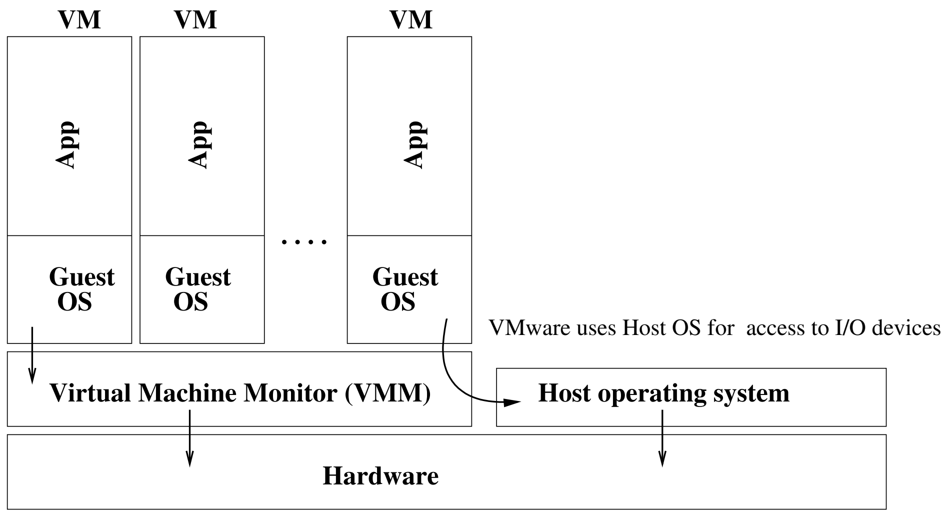

Figure 1: *Virtual machine platform architecture, showing a VMM running over the bare hardware, alongside a optional host-operating system (not necessary if all necessary functionality is handled by the VMM). The multiple VMs run over the VMM, and include their own guest operating system, which might run one or more applications within the protected domain*

system.

The two orthogonal approaches of virtualizing system resources, one above the host operating system (language specific) and the other just above the hardware (VMware and VMMs) are the two most common approaches. For reasons reserved until a later discussion, all further mention of VMs in this paper will refer to the complete isolated protection domains running over VMMs. However, towards the end in Section 5, I will present a brief discussion towards to bring out the positives and negatives of the language specific VMs. Also note that all benefits of language specific VMs can still be exploited while running them over guest operating systems running over virtual machine monitors.

1.1 Hosting Trends for Today's Internet Services

A large number of todays Internet Services run on clusters of commodity computers [6, 9]. A reasonably complex service involves multiple levels or tiers of processing to satisfy a client request. Typical services are architected as a 3-tier system with front-end nodes (handling load-balancing, and static content caching), application server nodes and back-end nodes (for persistent storage) [9]. In order to ensure resource guarantees and minimal load perturbations, components from different tiers in the system are usually run on separate nodes. Equipment allocation decisions have to be individually decided for each tier depending on resource requirements based on a estimated average and worst case load on the system. Thus it is often a horrendous task to come up with a equipment purchase and allocation plan for these services. It would be a great convenience for service providers in terms of management, if hardware and other resource allocation could be dynamically handled in a transparent and adaptive (to load dynamics) manner.

Rising to this requirement are third party web service hosting facilities [8], a trend which is increasingly gaining popularity among service providers. Such facilities maintain a large farm of mostly homogenous machines (unless clients demand a different hardware OS combination) and often host multiple services within the same facility. Service providers enter into service level agreements (SLAs) with such facilities to host their software and data and provide service to their clients with certain performance (e.g., throughput, response time) guarantees. The components of the different services are often run on separate set of nodes due to anxieties expressed by service providers regarding resource contention, security and failure propagation. Considering the distributions of accesses to various Internet services, it is common knowledge that static, independent allocation of resources leads to inefficient utilization within the hosting facility. One solution to this problem is to run several lightly loaded services on a single node, and dynamically deciding whether additional resources are required. However, one has to first deal with the anxieties mentioned above.

1.2 Position

In the previous subsection we presented a common problem associated with hosting today's Internet services. In this position paper we will argue for VMs to effectively solve this issue, improving the efficiency of such hosting facilities. We will also detail other benefits of using VMs to run Internet services such as improving the overall availability, manageability and security aspects. In a slight deviation from our focus I will first counter some arguments that advocate that VM is not the right platform, and then go onto present the various features rendered by the use of VMs and how applications in general and Internet services in particular benefit from it.

2 Countering the Antagonists

2.1 Virtual *adj.* : *alias* slow

The general perspective among antagonists of the VM is that running applications over these platforms consume excessive resources in terms of memory and computational cycles, resulting in slower applications. An environment running over a virtual platform, results in an extra level of indirection, for all services accessed directly through the VM monitor or using services implemented by a host operating system (see Figure 1). However, it turns out that this redirection overhead is not unbearable. Initial results published by a group at University of Washington, suggest that performance of a Web Server running over a light-weight VM, Denali [12], incurs a performance loss of around 2-15% for a widely varying workload [12]. I feel that this is a small cost, not a prohibiting one, a tradeoff, that one would exchange for the other benefits offered by VMs[1].

[1] Appendix A presents an argument against claims that language specific VMs such as Sun's JVM are *fluffy and groggy!*

2.2 Too heavyweight....Do not scale to many VMs

This was certainly true of initial implementations of VMs. Though VMware claims their VM to scale to many per machine [2], they are considered to be to heavy-weight and scale to more than a few (order of 2-4) VMs on a node with 1GB memory, and 1.x GHz Intel processor. However, this restriction has spawned academic research efforts in building lightweight VMs which would scale to to the order of 10-100s of VMs per physical machine. The effort at University of Washington has shown upto 80 VMs to run on a machine with the above mentioned configuration [12]. Also inspite of VMware being considered too heavy-weight, the onset of powerful machines such as the 32 and 64 processor machines with gross amounts of memory, allows packing of many more VMs per physical node. There are indications that even such combinations are a commercial success, with IBM claiming to run upto 20 VMs on 4-way x360 series server machine [2].

3 Requirements of Internet Services

Before we spell out the benefits of virtualization, I will briefly provide some background on some broader features desired of these services to put things in context.

Cost Effectiveness Todays Internet services span huge client bases, and correspondingly involve large investments in hardware and software components. The costs include both the acquisition costs as well as the management costs (more copies of software to be installed and maintained, increased operator responsibilities, etc.). Though cost-effective solutions are always a priority, it is more so in current times of hardship in the software industry[3]. Thus any effort to reduce the number of hardware and software components is surely appreciated.

[2] Note: VMware is ofcourse talking about a 8+ processor IBM x 440 series machine

[3] One look at the Nasdaq index should convince one of this fact

High Availability Availability in these services is equivalent to money. Cost of downtime is huge as exemplified by the infamous Ebay outage [10]. Various techniques are applied by current Internet services to ensure high availability including hardware and software replication, automatic failover, multiple copies of persistent data, and geographic colocation services and data. Some of the clustering infrastructure used for such services have started to provide high availability features, such as watchdogs, failover and restart tools and automatic state backup for server applications. High availability requires an integration of mechanisms at all levels in the hierarchy of hardware and software. Virtualization can contribute significantly to this effort.

Fast Deployment Times A study of the software cycle in current Internet services [9] shows that most services update their software every few months with patches and new features. For services which run 24x7 (understandable since client bases are spread across the 24 timezones), speedy deployment is essential to reducing downtime during the switch. Though various techniques such as rolling-updates are employed currently, software (including operating systems) may need to be selectively updated with minimum interference to unaffected components of the service. The ease of the deployment process is also a major concern of service providers.

4 VMs: A good match for Internet services?

In this section, I will elaborate upon the features offered by VMs and detail how they enhance the cost effectiveness, availability, security, and manageability aspects of Internet services. The following discussion titles are in the template of: **VM feature :: Derived benefit**.

Isolated Domains :: Server Consolidation VMs offer resource and fault isolation which allows multiple applications to be run simultaneously on a single hardware platform with minimal or no visible interference. This takes us back to the hosting problems with current Internet services mentioned in Section 1.1. Being able to run multiple components belonging to one or more services allows efficient usage of resources. Especially in the case of a farm of nodes hosting multiple services, it is often the case that not all services are at peak at all times, and some services are generally less popular than others. Studies of web servers and other network services have shown that popularity is often driven by Zipf-like distributions [5]. The implication is that a majority of requests are directed towards a minority of popular services. Thus, allocating independent resources to unpopular services results in inefficient utilization. Being able to consolidate different services, as well as server components onto few hardware resources leads to substantial savings in equipment. However, issues associated with this setup are: How do u ensure components are protected from load and fault interferences arising from co-components? How do ensure data and computation within services are secure and not visible to other services? Paragraphs below will discuss resource and fault isolation, while security is dealt in the next subsection.

When running multiple applications on a single hardware resource, unregulated contention for resources often degrades overall performance. While it is theoretically feasible to guarantee equal sharing, there also exists evidence to support that such an implementation is possible [12]. Such sharing mechanisms and policies can be implemented at the VMM layer. While it is well understood how processor cycles, physical memory, and disk space can be shared, disk and network bandwidth sharing are not trivial. One can think of simple queue-based scheduling policies implemented at the VMM layer or in the host operating system. In fact, a recent study explores network bandwidth scheduling for service differentiation in network intensive servers [13], which can be almost directly applied for our purposes. However, it is to be agreed that more extensive quantification studies are required to show the implementation feasibility.

Resource isolation can be viewed as a subset of the larger issue of fault isolation. Load related perturbations is a manifestation of *deviation from normal behavior*. VMs provide a safe domain, whose internal perturbations should not affect the functioning of other co-VMs. For example software faults within application code for service S1, should at the most hang or crash the VM it was running on, but shouldn't cause other VMs to be affected. While this is straightforward for application faults, its not obvious for faults that might affect components below the VM. In order to ensure isolation, all interactions of VMs with shared resources should be carefully monitored to prevent fault triggering or error propagation. This however is achievable since VMM presents exactly such an abstraction where all such interactions can be monitored.

VMM Level Services :: Trustworthy, Portable, Deployment++ As mentioned in the previous section, VMM presents a layer where services can be implemented independent of the guest operating system or the end-application. These services will run external to all VMs and very close the hardware itself, thus have to trust no other component except the VMM. Discussion in [7] makes a case that the VMM is much smaller in comparison to the operating systems, hence its bindings with the VMM would be more reliable and secure. This statement is a generally accepted principle in software engineering. Apart from being more trustworthy, and invulnerable to loop-holes within OSs[4], the services are inherently portable. It also decouples changes within these services from OS updates, thus also helping in faster update deployments. Examples of such services in the context of Internet services are secure storage, network data encryption, intruder detection modules, as well as the fair (or prioritized) resource sharing services mentioned above.

Encapsulation :: Replication, Migration, Fault-Tolerance The entire service component being embedded in the JVM allows easy replication or migration of the computation. Migration of computational contexts is plagued by state that is persistent, such as files bound to certain disks (i.e., not network mounted). However, this is a problem even with the real environments. In fact, the presence of a virtual layer can virtualize these resources (e.g., using iSCSI) to render the environment for easy migration. However, one can implement replication of VMs within the same machine with a minimum number of issues. Service overloads can be handled by quick replication of services on current or other nodes in the cluster, and the load re-balanced across the increased set of service handlers.

Replication of the computational context also allows for implementation of fault-avoidance and fault-tolerance schemes. Operations thought to be risky (e.g., processing a data payload of an IP packet whose hostname cannot be looked up), can first be executed on the dummy replicated VM to see if it is safe and when cleared can be executed on the original VM. This might require certain interaction guidelines to be specified at the VMM layer by the service component or the guest operating system.

A natural question is: Can't these benefits be derived by using language specific VMs? For completeness, I will briefly discuss the important differences/tradeoffs in using below the OS virtualization versus language specific VMs such as Sun's JVM.

5 Discussion: Software Vs Hardware VMs

Language specific VMs are referred to a software VMs, while below-the-OS VMs, which we have focussed on are the hardware types. Software VMs (SVM) offer various benefits to the application programmer as we discussed briefly in the background section. Portability is primary among them. The programmer need not worry about the interactions with the underlying operating system. The runtime provided with each operating system handles the compatibility issues. It also provides a layer of abstraction, which simplifies the design of applications considerably. However, as with hardware VMs they have their banes as well. Since SVMs run above the operating system, the program code is intermediate to machine code and requires interpretation. Performance being one of the serious drawbacks, SVMs like the JVM are often criticised and not considered as a serious platform for high performance services. However, the overhead has reduced over the years considerably with number optimizations. Section 6 provides a summary of the developments in the JVM performance history, and the results are certainly encouraging. Another issue with using SVMs as a platform for Internet services is that, running multiple applications within a single SVM presents major problems with pro-

[4] Outdone by Microsoft

viding resource and fault isolation. Running multiple SVMs will require complex runtime monitors which will administer the allocation of resources fairly among all the VMs. Experience with the JVMs have shown this can result in various vulnerabilities. Moreover SVMs can still be used in combination with Hardware VMs, to derive all language and abstraction benefits, while implementing all critical services that need to be portable at the VMM layer in the architecture promoted in this paper.

6 Conclusion

The above discussion has shown that, Internet services would benefit considerably if they were deployed over the VM platform. Though several implementation problems issues arise, the benefits are a good motivation to pursue this deployment platform. All said and done, we certainly need more quantitative characterization of the benefits of using such a platform. However, certain apparently qualitative benefits such as portability, manageability, and ease of deployment make this quantification a difficult task. This presents a good argument why the above issues should be considered valid research topics within the systems community, since they eventually affect the overall architecture, behavior and performance of the entire system.

References

[1] With HotSpot, the JVM is faster than a speeding bullet! http://www.javaworld.com/javaworld/javaone98/j1-98-hotspot.html, Mar. 1998.

[2] IBM eServer and VMware to Consolidate Multiple Virtual Servers on Single Server to Lower Costs for Businesses. http://www.vmware.com/news/releases/ibm_jda.html, Feb. 2002.

[3] Java HotSpot Technology:Java HotSpot Means Performance. http://java.sun.com/products/hotspot/, Apr. 2003.

[4] What is POSIX? http://www.lynuxworks.com/products/posix/posix.php3, Apr. 2003.

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *INFOCOM*, pages 126–134, Mar. 1999.

[6] E. Brewer. Lessons from Giant-Scale Services. *IEEE Internet Computing*, July/August 2001.

[7] P. M. Chen and B. D. Noble. When virtual is better than real. In *8th Workshop on Hot Topics in Operating Systems(HOTOS-VIII)*, pages 20–23, Elmau/Oberbayern, Germany, May 2001.

[8] IBM. The Ocano project, Apr. 2003. Available at http://www.research.ibm.com/oceanoproject/.

[9] D. Oppenheimer, A. Ganapathi, and D. Patterson. Why do internet services fail, and what can be done about it? In *4th Usenix Symposium on Internet Technologies and Systems*, Seattle, Mar. 2003.

[10] SiliconValley.internet.com. Ebay Outage Twice This Week. http://siliconvalley.internet.com/news/article/0,,3531_435741,00.html, Aug. 2000.

[11] VMware, Inc. VMware, Apr. 2003. Available at http//www.vmware.com/.

[12] A. Whitaker, M. Shaw, and S. Gribble. Denali:Lightweight Virtual Machines for Distributed and Networked Applications. Technical report, University of Washington, Seattle, Feb. 2001.

[13] X. Zhou and C.-Z. Xu. Transcoding-enabled Network-I/O Bandwidth Scheduling for Service Differentiation on Streaming Servers. Technical Report WSU-ECE-CIC-02-05, Wayne State University, May 2002.

APPENDIX A

Software VMs are redefining SLOW! This is undeniably true for early Java byte code interpreters where applications in Java ran 30-50 times slower than their native C counterparts. However, with the inception of Just-in-Time(JIT) compilation technology, the gap narrowed to 5-10 times or lower [1]. The HotSpot [3] technology in Sun's JVM includes various optimizations, using which some Java gurus claim that performance comparable to native code versions can be achieved. Thus, a reasonable performance comparison today would place the interpreted code to run 2-4 times slower than native versions. Considering that large Internet services already over-provision, while keeping their systems underloaded, and the dropping prices for computational and memory resources, it would be reasonable to incur the resource inefficiency for the benefits offered by VMs.