Clipper Notes™
VENDOR-AGNOSTIC EXPLANATIONS AND ADVICE FOR THE INFORMATION TECHNOLOGY BUYER
Navigating Information Technology Horizons℠

New in 2007 Report #TCG2007028 February 27, 2007

Server Virtualization Made Real

Part 1 of a Multi-Part Series on Server Virtualization

Analyst: Mike Kahn

Management Summary

Server virtualization is as confusing as the chatter of political wannabees prior to the presidential primaries. The politicos all use the same vocabulary to talk about the critical issues of the day, however, their use of the same words don't all mean the same, just like all of their promises are not equal. You need to be a good listener and note taker to sort it all out; a healthy amount of curiosity and skepticism is a big plus. The same is true for server virtualization. To sort this all out, you need a few concepts, which have been intentionally phrased not to use the overused buzzwords, for now.

What is Virtualization?

In the simplest terms, applications run within an operating system, which creates the environment for executing applications for a specific computer architecture (system). For example, your browser (the application) runs on Windows, MacOS, or Linux (each an operating system) on an x86 computer (a well-defined computer architecture). Unfortunately, several forms of virtualization are possible, even in this simple example.

Before going any further, we need a simple definition of virtualization, to kick-start this discussion. Here's one, albeit in two parts.

- *Virtualization is the intermediation of a physical architecture and/or components with a more-human-understandable logical view, and*
- *With the objective of simplifying the effort required to use the underlying physical asset.*

Most of us are old enough to know that telephone area codes once were linked to specific geographic areas, like 617 was for all of eastern Massachusetts (once upon a time). When you called someone in area 617, you knew that the phone was going to ring in a specific physical place, further identified by the exchange (the next 3 digits). This "hardwiring" was required because the switching equipment was designed to connect physical locations. Step forward a couple of decades. With telephone number portability (a kind of virtualization, as we will soon see), you could assign your home phone number to a cell phone, which could be used, generally speaking, at any physical location. Additionally, if you moved across the country, depending on your carrier, you could keep your 617 telephone number, even if you lived in Los Angeles. The caller could no longer make a safe assumption on where you are or even where you live. The same is true with Internet (VOIP) phone calls. Someone (actually, some thing) knows where you physically are located, but you and your caller really don't need to know and, probably, don't care to know.

IN THIS ISSUE

Server virtualization - of many kinds and variations - is about the application not knowing (or caring) about which piece or kind of hardware it are using, as long as it can get the work done, i.e., run your application. This brings us to a second level of complication that must be understood, in order for it to be "virtualized away". The fundamental question is "what is an application?", to be followed by "how does it (or several applications) relate to getting work done on a server (or several servers), whether real or virtualized?" These can be a deeply philosophical questions.

What is the scope of the virtualization?

Consider your desktop or laptop computer. Most of us are running many applications at what appears to be at the "same time". Mail is being fetched, anti-virus software is scanning it for evil and spam, while you are composing a document and referencing old mail and your calendar. What we have here is many applications coexisting and executing within a single environment, which can be visualized as a "container" of sorts. Unless you are very sophisticated, you neither want to know or can even influence how the operating system manages all of this, and sorts out how to best do this with varying underlying hardware configurations, while maintaining integrity and security.

What this means is that **the operating system itself is an agent of virtualization**. The philosophical question is whether you want your container to do one thing (think of a specific financial analysis, i.e., a specific application) or whether your container should envelop many applications simultaneously, just as a talented juggler can keep many differing objects in the air. These all lead to the question of "workload management" and where pending work to be done is queued and then executed. **Workload management is another kind of virtualization.**

Like so many things in life, one's perspective influences how one sees the kinds or vehicles of virtualization, i.e., what needs to be virtualized. End-users may see none of the virtualization, except for what appears on his or her screen. Everything else might be viewed as a virtualization (or stack of virtualizations that are beyond the scope of caring). This is seeing an application or business solution as a service and not caring how it all is made to happen. For those inside the IT infrastructure, the underlying vehicles to be virtualized are more readily apparent. The IT manager knows that there is not single entity that delivers the application to the users. This is akin to a single-celled organism. While it may be interesting in itself, what is really interesting is what is going on inside the cell, at the functional, component, biochemical, and DNA levels.

It is easier to think about the single organism but this may not be sufficient for optimizing its effectiveness and efficiency. **And that is the goal of virtualization, to make the IT delivery vehicle (and its components) more effective (faster, better, safer, more secure, etc.), and more efficient (lower total cost of ownership, by sharing of resources - IT assets of all sorts - and simpler ways of managing them).**

Virtual What?

So the object(s) of and the degrees and complexities of what is being virtualized is critical to defining virtualization of any specific sort. This brings us to the topic of the day and this report - "virtual machines", whatever that term really means. When folks see the letters VM today, many will think of "Virtual Machine". Those of us who have been around the computer industry for more than a couple of decades might have said that VM used to mean "Virtual Memory". Even when we all say "Virtual Machine" (when we see "VM"), we may be envisioning something different or varying degrees of the same thing.

For this reason, in this Clipper Note, the term "Virtual Environment" will be used instead of "Virtual Machine", because the latter is either too broadly defined or a misnomer. As you will see, **the Virtual Environment also is in the eye of the beholder, with respect to domain of control and scope of resources**. Accordingly, one virtual environment might contain one or more other virtual environments. OK, that is a truly confusing thought. We will come back to it in a while.

Let's think beyond the personal computer example to a more complicated enterprise environment. There is a lot of work to be done; think about it as many users running some of a collection of applications, with

varying degrees of scope (e.g., how much processing power is required to get it done) and priority (e.g., how much more important is this application or user, etc., from others). What we have here (in the collective IT sense) is a diverse and varying set of work to get done, i.e., a *workload*.

What we want is to be able to do that workload without specifying where and how it is to be accomplished. We want to "decompose the workload" into tasks, dependencies, priorities and the like, without having to think about the complexities of doing this for (potentially) thousands of users, activities, and objects, many coming and going unpredictably in real time. Additionally, we want to virtualize the resources, so that we can think about them in the abstract, without caring about the physicality of each resource. (Now you see why it is easier to think about running only one application on a computer (server).)

We could solve the first part of this problem by rigorously scheduling the work to be done when the IT assets were available, like saying the only time you could use an ATM was when the bank scheduled you to do so. We could solve the second part of this problem by having infinite resources (assets) to deploy against the many, continuing needs, but that wouldn't be efficient. These, and many other methods, now are known to be less effective or too costly. We no longer try to solve all application needs with a single program. We divide the problem into components (objects) and develop each object separately. We then put in a level of "integrative abstraction" that allows the "system" to assemble what is needed to meet a specific use or request.

If this sounds like SOA (Service Oriented Architecture), with its greater use/reuse of fewer components, then you now are thinking virtually in another dimension. SOA not only saves money but also standardizes business processes so that they are more easily combined and extended and, more importantly, easier to use as building blocks; sort of a Legos philosophy – get the interfaces consistent and anything can be built. Dependencies are handled by making software more declarative and application processes more discrete (a.k.a. "services").

Really, there is virtualization under every IT stone. No wonder we have so much trouble understanding the meaning of "virtual machine". One person's machine is another's set of many components, each with the potential to be another virtual machine!

Where You Put the Point of Control

Back to reality, for a moment. Having virtualized the IT workload (at least intellectually), no easy task, leads us to want to virtualize the physical resources on which the application(s) will run, i.e., what might have been called the "systems environment". This is what many today consider to be a "virtual machine". That works, sort of, until you realize that the systems environment actually consists of may "virtualizable components". Here we go again, spiraling down the virtualization hierarchies.

Nonetheless, we all have been "CPU fixated" for a long time, so it is easy to start with the desire to subdivide (somehow) the processing "horsepower"[1], especially in the new era of multi-core processors and multi-processor computers. Some computers can be partitioned physically and others only logically, like getting 10% of the total (available) horsepower, usually by a time-based (round-robin) sharing of the CPUs' cycles. Doing this "time-slicing" is what most folks consider "server virtualization" to be. However, it is only the beginning and not an end to itself.

There are other hard components that need to be shared, including network bandwidth, memory, cache, I/O controllers, and peripherals, such as storage and printing devices. To only slice and manage the CPU solves only part of the problem and, in many cases, just moves the bottleneck to a lesser-managed or less-manageable component. To do it right, you need to virtualize all of the resources used by the applications.

How Does Time Affect to Virtualization?

OK, this does sound like is an Einsteinian question on the theory of relativity. Nonetheless, Einstein was right – everything is relative to a point in time and in the eyes of

[1] Obviously, there are no "horses" here, but the "horsepower concept" seems to aid common understanding of the power of a server, whether physical, partitioned, or virtualized. "Throughput" might be more accurate, but the word's meaning is dependent on the application being run.

the observer. **Thus, time is the (and, potentially, most important) extra dimension in the virtualization equation.**

While it is incredibly amazing that everything involved with launching the Space Shuttle comes together to make it all happen, this is a very simplified case of what is really desired and needed for 21st Century space exploitation, to wit, many cargos and crews and missions for many shuttles being launched simultaneously and continually. So, while doing it "once in a while" is great, **doing it frequently, and also on demand is far better but much harder.**

Using a cargo distribution model as an example, we need to virtualize the cargo being picked up and shipped into manageable pieces (where the size and weight and volume are key determinants) and then we need to take the manageable pieces and assign and prioritize them on different conveying vehicles (the fleet of carts, trucks, planes, etc.), according to the policies and priorities (of a pricing model).

In computer terms, the virtualization of the cargo is the classification and containerization of the applications (i.e., virtualizing the workload) and the virtualization of the fleet into varying conveyance (execution) vehicles is the virtualization of the resources (infrastructure) that makes it all happen (i.e., the virtualization of the systems components; collectively the virtualization of the environment). You need to think about both (and more) – changing continually and rapidly in real time – in order to think about optimizing the solution from end-to-end.

Conclusion, for Now

If you feel like I have sucked you into a black hole from which no light (and, thus, no clarity) can escape, I apologize. We are all trapped in here together. Now that we have a common understanding and, maybe, some common vocabulary for communicating with each other, we can talk about more productively about virtualization, in general, and server virtualization, in particular.

As the title to this bulletin indicated, this is the first of several in a series on Server Virtualization. For Part 2, please see the February 28, 2007, issue of **Clipper Notes** entitled *Virtual Machines — Three Things to Consider + Three Business Uses*.[2]

Also, please check back for additional discussions on server virtualization, since **virtualization is the common theme for much of what needs to be done**.

[2] Available at http://www.clipper.com/research/2007029.pdf.

About The Clipper Group, Inc.

The Clipper Group, Inc., is an independent consulting firm specializing in acquisition decisions and strategic advice regarding complex, enterprise-class information technologies. Our team of industry professionals averages more than 25 years of real-world experience. A team of staff consultants augments our capabilities, with significant experience across a broad spectrum of applications and environments.

- ***The Clipper Group can be reached at 781-235-0085 and found on the web at www.clipper.com.***

About the Author

***Mike Kahn* is Managing Director and a cofounder of The Clipper Group.** Mr. Kahn is a veteran of the computer industry, having spent more than three decades working on information technology, spending the last third at Clipper. For the vendor community, Mr. Kahn specializes on strategic marketing issues, especially for new and costly technologies and services, competitive analysis, and sales support. For the end-user community, he focuses on mission-critical information management decisions. Prior positions held by Mr. Kahn include: at International Data Corporation - Director of the Competitive Resource Center, Director of Consulting for the Software Research Group, and Director of the Systems Integration Program; President of Power Factor Corporation, a Boston-based electronics firm; at Honeywell Bull - Director of International Marketing and Support; at Honeywell Information Systems - Director of Marketing and Director of Strategy, Technology and Research; with Arthur D. Little, Inc. - a consultant specializing in database management systems and information resource management; and, for Intel Corporation, Mr. Kahn served in a variety of field and home office marketing management positions. Earlier, he founded and managed PRISM Associates of Ann Arbor, Michigan, a systems consulting firm specializing in data management products and applications. Mr. Kahn also managed a relational DBMS development group at The University of Michigan, where he earned B.S.E. and M.S.E. degrees in industrial engineering.

- ***Reach Mike Kahn via e-mail at MikeKahn@clipper.com or via phone at (781) 235-0085 Ext. 121. (Please dial "121" when you hear the automated attendant.)***

Regarding Trademarks and Service Marks

The Clipper Group Navigator, **The Clipper Group Explorer**, **The Clipper Group Observer**, **The Clipper Group *Captain's Log***, **The Clipper Group Voyager**, ***Clipper Notes***, and *"clipper.com"* are trademarks of The Clipper Group, Inc., and the clipper ship drawings, *"Navigating Information Technology Horizons"*, and *"teraproductivity"* are service marks of The Clipper Group, Inc. The Clipper Group, Inc., reserves all rights regarding its trademarks and service marks. All other trademarks, etc., belong to their respective owners.

Disclosure

Officers and/or employees of The Clipper Group may own as individuals, directly or indirectly, shares in one or more companies discussed in this bulletin. Company policy prohibits any officer or employee from holding more than one percent of the outstanding shares of any company covered by The Clipper Group. The Clipper Group, Inc., has no such equity holdings.

Regarding the Information in this Issue

The Clipper Group believes the information included in this report to be accurate. Data has been received from a variety of sources, which we believe to be reliable, including manufacturers, distributors, or users of the products discussed herein. The Clipper Group, Inc., cannot be held responsible for any consequential damages resulting from the application of information or opinions contained in this report.