On Virtualization

William Favorite

Maximizing compute efficiencies across the environment

Introduction

Across the IT industry, optimization of compute resources comes at a steep price in terms of man hours. The resulting trade off of human-time for compute-time skews our valuations of this optimization activity. The cost of computing resource waste is minuscule when weighed against the human-time required to optimize for efficiency. This is a reoccurring trend that we see in diverse areas such as application development, storage allocation, and systems design. The economics of corporate IT has little tolerance for perfectionists.

In most cases, development costs and project timelines are too tight to shave 10% off a sort algorithm.[1] Developers simply implement a *working* solution within the time constraints of the project, and throw relatively cheaper compute resources at the problem. As a result, code on modern business systems is rarely optimized for performance or size as we tend to opt for the significantly cheaper option of faster CPUs and more memory.

As a result we tend to search for optimization methods not in a manual human-time intensive micro-management of resources but in an automated method or tool that allows us to achieve the efficiency at a significantly lower amount of effort. Programmers look to pre-built libraries, storage administrators look to highly evolved storage management arrays and software, and systems administrators are now looking to highly evolved virtualization methods to achieve these efficiencies.

The storage administration[2] area is another good example of breaking the human-time intensive barrier to resource optimization. As the number and size systems grew, the amount of direct attached storage was spiraling out of control. Individual servers required large amounts of storage capacity and growth capability so that each system was laden with a significant number of spindles.

> Compute resource optimization is a human-time intensive endeavor. Highly automated and evolved tools are required to make it cost effective.

> Highly evolved and powerful storage administration tools are an example of how technology has overcome human-time intensive optimization.

---

[1] While profiling and optimization may be highly evolved in computationally intensive or tightly constrained environments where the economies of optimized development require it, the majority of corporate systems today do not optimize in this manner.

[2] Storage virtualization is such a long established concept that it is rarely used by the term (virtualization). "Storage virtualization" has come to describe a new family of products that offer entirely different capabilities for storage administrators that fit the same model of optimization costs overcome by automated methods. In this respect the storage industry has entered a second *round* of virtualization technology that I frequently refer to as "storage obfuscation".

Administrators and DBAs were willing to allow waste of large portions of the storage to insure that spindles were sufficiently independent and/or striped for performance. The problem was clearly not scalable and the economics eventually demanded a solution in the form of centralized storage arrays that were capable of slicing disks to the optimal size while maintaining the performance characteristics required by the applications. The ability to thin provision, or oversubscribe, the storage resource has further driven this efficiency so that now one of the key metrics of storage administrators is the utilization of the storage.

Server virtualization is a recent development [3] that has given mainstream corporate IT the ability to virtualize the system itself to enable greater efficiencies in CPU and I/O utilization. Like the storage administrator, the server administrator is reaching a point were the spiraling costs of a "scale-out" approach to server allocation are too expensive to ignore.

Virtualization of a server amounts to the same concepts that were previously mentioned (the automated optimization of resources) as applied to the traditional server resource contention points, namely CPU, memory, and I/O. Tools have been developed that allow these resources to be sliced and optimized to the needs of each individual system. The ability to pool and then share resources through a streamlined tool has resulted in a more efficient utilization of those resources and a more optimized and cost effective allocation method.

With products now widely available from a number of established vendors there a number of technologies that provide these capabilities. Even though the underlying technologies are quite different, the result is still largely the same. These tools take the formerly problematic and relatively difficult application stacking methods and made them completely manageable in the constraints of human-time.

Shades of green - making the argument

System virtualization is typically presented as part of a "green" argument that is in vogue today. My response to this is that while it is admirable to be green, the true motivation maps much closer to traditional economies of optimization that have ruled the IT industry for decades. That is, system virtualization now makes sense because a highly automated solution has overcome the human-time constraints of manually optimizing resource utilization. Now it is possible to deal with the formerly inefficient allocation of resources with a mechanism

---

[3]Server virtualization in the corporate open-systems / production computing space is less than a decade old. The concepts and actual implementations date back significantly longer in other environments, but it was not until the advent of VMware and IBM Power5 LPARs that it achieved a maturity and mainstream acceptance in open-systems.

that leverages set and forget algorithms rather than human micro-management of those allocations.

The following are a few arguments for the use of virtualization that while may be green, they are as much about cost reductions and improving efficiencies as they are about carbon footprints.

1. Reduce power
   The ability to reduce power is often argued in terms of the regular power *consumption* cost, but rarely takes into account the infrastructure costs and the disruptive nature of constantly running higher and higher amperage to more and more cabinets throughout the data center. The "scale-out" approach of system allocation is running head on into the *massive* power requirements of today's rack dense servers.

2. Reduce HVAC
   Cooling costs scale directly with power costs as the energy is pumped into the racks it will escape largely in the form of heat. Like power, this is frequently seen as a re-occurring cost but must also account for the massive incremental chiller upgrades that are the direct result of the "scale-out" method of satisfying compute needs. The cost of expanding cooling in the data center is taking new impact as we begin to run chilled water back to the racks to cool rack dense servers and blades. Like the *hidden* power cost, these upgrades to capacity have disruptive effects upon data center operations.

3. Corporate waste & responsibility
   The practice of discarding 80% of your work product in any other industry would be grounds for shareholder revolt, yet it is done every day in data centers across the industry. The *hard* argument is "Would it be acceptable to GM[4] shareholders if the auto manufacturer crushed and dumped 80% of its manufactured product in the Detroit river?" The softer, and more realistic argument, is "Would it be acceptable to GM shareholders if GM ran all plants at 20% capacity, then built out more plants, to run at 20% capacity, for each new years model?" The answer to each of these questions is a resounding no.

4. Development platforms
   Developers want as close an approximation to the production environment

---

[4]General Motors is used as an example of a relatively mature engineering and manufacturing industry. IT, being a young industry by comparison, seems to simultaneously lag in these measurements, but also be driven by slightly different motivators. As technology offers IT managers the *ability* adapt they then become *bound* by the same factors that drive more traditional industries.

as possible. This requires that we dedicate a specialized environment to develop and test an application against each incremental OS change. Development systems often represent the worst offenders in terms of lost compute capacity as they are numerous and tend to be considerably under utilized when compared to their corresponding production systems. The benefit of virtualized environments is that they allow you to deploy these systems at reduced cost.

Despite the high cost of these systems it is imperative that we continue to have proper development environments. Virtualization allows us to take advantage of both the cost benefit of having a full development environment as well as the ability to save costs in those deployments. Low cost virtualized systems allow a developer to be free to experiment with radical code changes prior to moving them into the main development branch. It is in this case where virtualization reduces waste but at the same time enables more of the activity that was the source of the waste.

5. System personality

Using the term "personality" is a bit misleading as the desired trait is a *lack* of *unique* personality. It is desirable for virtualized systems to appear from a hardware perspective to be the same, and to utilize simple yet robust interfaces (drivers) with the virtualized hardware. Most *mature* virtualized environments offer a unified system image that is supported with optimized drivers. The idea is that when I/O is virtualized, it can take the form of a generic driver that has a fair degree of hardware independence. For example, the complexities of Etherchannel/802.3ad or multipath I/O can be established for the I/O partitions and then leveraged through general interfaces on each virtualized client. Many virtualized clients get the benefit of redundant I/O without the complexities of multiple configurations. The ultimate advantage is that virtualized systems can be moved from one host to another without changes to the client host install. This feature is a fundamental requirement for live migration / motion capabilities that allow the client host to be moved *while running* to another system. Currently this is provided by the VMware, IBM System P, and HP platforms.

6. Provisioning

The ability to provision hardware resources has been simplified in many environments by pre-ordering and racking various hardware configurations. This speeds the process but exacerbates the waste of power and port utilization. Virtualization on the other hand allows resources to be leveraged while acting as a capacity pool for future deployments. (Hardware) Systems capable of virtualization will effectively require the same deployment methods for the OS but offers much faster hardware provisioning options.

Systems that offer CoD[5] expand upon this by electing to "pay as you go" with hardware resources. With virtualized systems, provisioning activity has *literally* been reduced from weeks to minutes.

7. Port count
   The acquisition and power costs of modern (Ethernet / SAN) ports are quite expensive. The "scale-out" approach to procuring systems requires that each system to have at least one production port and frequently more to meet the needs of backup networks, dedicated iSCSI connections, fault tolerant (ie: Etherchannel) ports, and lights out access ports. Virtualization allows these to be condensed into fewer ports that are shared amongst the virtualized hosts.

8. Port utilization
   Port utilization is different from port count in that the port count argument is an attempt to reduce port count, while port utilization is the idea that if you are going to pay the acquisition and power costs of a modern switch port then you should use as much of the bandwidth as possible. This is a continuation of the wasted resource argument but extended to the communication networks that connect the systems. Here, once again, we are looking to drive out waste from the system by using virtualization to properly utilize resources we have.

9. Rack dense
   Blades are frequently seen as the leading argument for rack dense computing, where they are really just condensing the "scale-out" problem into a smaller space. The problem is that they are rigid in allocation units, suffer from the same power/cooling problems as individual servers and are still not a rack dense as a virtualized environment. The IBM blade center H offers 14 (Intel/AMD/Power) servers in 10u of space. The IBM p570 offers up to 160 virtualized systems[6] in 8u of space. The strange side effect of blade servers is that they are exacerbating the heat issue and are the leading reason for bringing chilled water (back) into the racks. Virtualized hosts solve this problem the same way they do with *normal* servers, they lower the resource requirements through more effective utilization of those resources. The end result is *less* resources *despite* the form factor they come in.

---

[5]Capacity on Demand: The ability to immediately enable staged resources on a system and then pay in a subscription, single use, or outright purchase form. CoD is usually a premium option on higher end systems.

[6]This is a CPU based limitation that is way beyond what would normally be considered practical. The point is that the blade is not the most rack dense object in the data center. Two IBM systems are used as a simple illustration, this phenomenon is not unique to the IBM blade chassis.

10. Hardware monitoring & maintenance
    Every aspect of managing large numbers of hardware devices is human-time intensive. From managing firmware levels to hardware monitoring, the more systems you manage the more resources it takes to monitor it. The virtualization of systems allows you to continue with independent operating system installs while reducing the different hardware (physical) systems on site.

11. Interconnect speed
    Clients virtualized on the same hardware tend to have I/O interconnect speeds that are typically in an order of magnitude faster than the traditional physical Ethernet speeds that would normally connect them. I/O transfers between most types of partitions happen at a mixture of CPU and memory speeds rather than that of the physical Ethernet interface while maintaining seamless compatibility with an Ethernet device / networking stack.

12. Analogous technology - storage
    As disks grew it became increasingly difficult to maintain spindle independence yet not throw away too much of the disk. In some cases the storage became *too large* for the task at hand. Storage arrays allowed us to slice these spindles to more appropriate sizes while remaining independent *enough* to satisfy the performance requirement.
    We now face a similar situation with CPUs and I/O interfaces that are simply *too fast* for the work at hand. While it is possible for a modern CPU to fill a modern pipe, in most cases we are seeing applications that are not, by themselves, filling these interfaces, and when they do it is only for short bursts. The ability to virtualize them, much like the storage analogy, is what will allow us to drive out the waste.

13. NP-Hard
    Like a computing problem that cannot be scaled beyond a limit, our data centers simply cannot continue to "scale-out" with physical hardware. Every addition of compute resources adds to the eventuality that the data center will physically need to be expanded, the next chiller will be required, the next UPS will need to be installed, or the next infrastructure upgrade will be required, all at significant tangible and indirect costs.

14. Service contracts
    The maintenance on hardware must be factored into the cost of the "scale-out" approach. As the server count increases so do the costs associated with this approach.

15. Per-CPU and per-system licensing costs
    Items licensed per (physical) system are scaled with the "scale-out" approach, while anything licensed per processor takes an increased hit in the non-virtualized environment as the effects inherent to CPU wastage are borne out on the license tally. In short, reducing physical systems reduces license based upon physical systems, improving CPU efficiencies improves licensing costs as CPU wastage is directly proportional to the licenses incurred.

16. Purchasing power
    Single, larger, infrastructure spends give the purchaser greater leverage than many smaller spends. Additionally, these single, or *fewer*, spends can be synchronized with vendor product announcements and sales cycles. This allows the customer to leverage end of quarter / year sales instead of paying less optimized prices based upon each project's acquisition schedule.

Thinking Green

For those driven by the green argument[7] that are trying to reduce their carbon footprint, virtualization offers promise that green servers do not. Less servers and ports means less packing material, less manufacturing impact and less waste as the technology is aged out of service. Driving resources to the levels they were designed is the most straightforward approach to reducing the carbon footprint.

When a low carbon footprint is a motivating factor, virtualization offers the most green of green alternatives.

Green development of new hardware is attempting to design low power modes into CPU and networking interfaces so that they drop into a power saving state when not driven at full utilization. Discussions of full-power-up latency effects on performance should be considered at this point, but are not necessary when compared against the virtualized system. The virtualized system is simply eliminating many of the CPUs and I/O interfaces so low power or not, virtualization simply bypasses a significant portion of the power consumption caused by the scale-out method by not deploying the extraneous hardware in the first place.

Virtualization adoption problems

Many IT acquisition methods are structured financially to the scale out approach and find it difficult to bill back *capacity* rather than individual servers. The way in which hardware is purchased needs to adjust from purchasing individual servers in incremental steps to laying out a larger portion for "infrastruc-

IT must adjust to the new environment that virtualization presents.

---

[7]Data centers in California are starting to see government involvement in development of more power efficient data centers.

ture" then charging back for capacity and possibly enabling / purchasing more capacity. A problem that arises out of this is the idea that the server is a physical entity that can be moved and repurposed for other things such as deployment to another business unit in a remote location, DR site, or sold/depreciated as an individual asset.

Most vendors are aware of virtualization technology and have structured per-CPU software licensing costs appropriately. Despite this, there are some that have not structured CPU licensing costs to the reality of sub-CPU virtualization.

Ideally virtualization will make the hosted environments appear as similar as possible to a stand-alone system. Most operators and all users should not notice that they are on a virtualized system. That said, some training will be required for the IT staff managing the virtualized systems as they learn the tools and methods used to control the technology.

Virtualization technologies will require that system administrators perform configuration tasks that were formerly the domain of network or SAN administrators. Virtualization deployments typically have to deal with concepts like VLANs and zoning / LUN masking of disks to client partitions[8].

It will be necessary to overcome some concepts that are prevalent in "scale-out" deployments. For example, it is common practice to separate production and development when they are physically independent systems, but when virtualized it makes sense to combine these two platforms so that thin (CPU) provisioning will allow you to deploy larger (or more) development environments that can take a lower priority to production. By doing this you allow development to utilize unused production cycles and production to steal from development during peak periods.

Technology

Systems virtualization implementations are defined by a number of factors such as the granularity of resource allocation, the independence of partitions, and features that are used to manage and monitor these divisions of resources. Some of the first implementations were simple partitioning that were focused on dividing the system into sub-sections on rather *coarse* electrical boundaries. The trend in more recent implementations has been to offer more granular divisions of resources along with more advanced methods of sharing those resources between the divisions.

From the original crude partitioning, vendor implementations began to take widely different approaches to providing more granular divisions of resources and

Virtualization has evolved from simple partitioning of resources to a finely controlled, granular division that is satisfied through many different vendor implementations

---

[8]The Cisco Nexus 1000V switch is an example of an attempt to rectify this problem on the Ethernet networking side, meanwhile NPIV capable HBAs are making the problem more difficult for storage admins.

methods to manage them. The methods that were chosen fall in several different, yet non-exclusive, camps:

- Hard (coarse) partitioning
  This is the division of systems on electrical boundary points in the system. This is typically done on a *bank* of CPUs, memory, or I/O subsystem. The most popular example of this is the Sun/Fujitsu M-Series systems and HP nPars.

- Hard (granular) partitioning
  This division is like the coarse version but is less restrictive in resource allocation by allowing individual CPUs, *blocks* of memory, and individual I/O cards to be selected. This is most typified by the IBM Power4 (LPAR and DLPAR) method as well as HP vPars and even Sun LDoms.

- Paravirtualization
  Paravirtualized operating systems are modified to run in the virtualized environment[9] and are hosted by fundamentally general purpose systems (both OS and hardware). The key difference from a type 2 hypervisor is that the paravirtualized client OS is a modified version of that OS that is *aware* that it is running in a virtualized environment. It is different from a container in that the hosted OS is a complete OS including kernel and drivers where the container method relies on the single underlying kernel and drivers. One popular example of this method is the xen product.

- Type 1 Hypervisor
  A type 1 hypervisor is a special purpose virtualization manager that controls / arbitrates access to system resources. Type 1 hypervisors are typically implemented in firmware[10] and typically allow for a special purpose client that can re-share I/O resources to other client partitions. Client OSs hosted by a type 1 hypervisor are full / stock kernel implementations of a client OS that are designed to resemble a normal system install. This type of hypervisor is typified by the IBM System P architecture as well as VMware's ESX implementations.

---

[9]The distinction of paravirtualized is blurred as virtualization environments provide *smarter* drivers and utilities that *are* aware of, and optimized for, the virtualization layer.

[10]The VMware ESX hypervisor typically loads from some sort of disk but generally meets this description. The subtleties between implementations allows for hours of technical debate, but a hypervisor that launches without loading a general purpose OS with it should be considered type 1. I argue that the key difference is the fact that type 1 hypervisors simply *cannot* run general purpose software directly on the hypervisor. More specifically, you are unable to install and run a third party driver or utility on a type 1 hypervisor.

- Type 2 Hypervisor
  Type 2 hypervisors are typified by the fact that they run on a general purpose OS. The idea is that OS resources are re-presented to client OS partitions through an integrated virtual I/O system. Client OSs hosted by a type 2 or a type 1 hypervisor are full / stock kernel implementations of a client OS that are designed to resemble a normal system install. Popular examples of this virtualization environment are VMware GSX (VMware Server), HP Integrity VMs, or the Parallels environment on OS-X.

- Containers
  Containers are not virtualized operating systems but are more akin to compartmentalized process groups that (may optionally) have resource controls. Because the compartmentalization happens at the (init) process level[11], the underlying system does not need to *re-present* I/O to the containers as they will simply use existing mount points and file systems[12]. Popular implementations of this technology are Solaris zones and AIX WPARS.

- Hardware Assist
  As the name suggests, this is not a virtualization method, but it *is* an important and growing factor in virtualization implementations. This is hardware (typically I/O devices) that are designed to enable or enhance a virtualization capability. An early example of this is a VLAN capable Ethernet adapter that allows for multiple distinct networks to be represented by a single physical card. This concept has been extended to NPIV (N Port ID Virtualization) for HBAs, converged networking adapters, and the hardware based virtualization of devices like the HEA (Host Ethernet Adapter) provided on IBM System P.

Each method is non-exclusive and may be used interchangeably. For example as containers serve as a good application encapsulation method they may be used to serve that purpose on a partitioned or (hypervisor) virtualized system. Several hypervisor based solutions allow physical resources to be added directly to the partition. Some vendor implementations tend to be more pure examples of one or the other while some may leverage from each description.

Choosing the right virtualization solution may involve a number of criteria such as application compatibility, the ability to run particular vendor's hardware or OS. For the sake of this paper we shall assume all things being equal and we

---

[11]This is based on the *reference* Solaris implementation. IBM application WPARs function above the init level at the process (group) level.

[12]There are variations on this, but the underlying point is that the single, *global*, kernel provides the underlying device access and logical to physical translations.

are looking to deploy a Java or similar web based application that is entirely independent of the OS or platform. The goal is to evaluate entirely upon capabilities of the virtualization type.

Each method leverages different design considerations that provide distinctive deployment capabilities and issues. The partitioned solution provides the greatest fault tolerance as it strives for the maximum independence of each partition, but this solution requires very specialized hardware. On the other end of the spectrum the paravirtualized and container solutions provide the capability for their implementation on virtually any hardware.

I have included the partitioned system primarily for historical and completeness purposes. Partitioned systems typically offer the ability to move resources from one partition to another but little more. The main benefit of systems of this type were to allow companies to *adapt* the partitioned systems to a changing environment. On these systems it is possible to add CPU, memory, and I/O resources to a partition. The coarse granularity of the "building blocks" of these systems were approximate to entry level servers in that generation of hardware. As a result, it made little sense to heavily leverage partitioning because systems of comparable performance were *considerably* cheaper. When compared to the number of single image systems, early partitioned environments were not routinely deployed in most corporate environments.

As mentioned earlier, the paravirtualized hypervisors provide the most platform agnostic implementations available. This means that we are not required to purchase specialized, expensive hardware to host multiple client operating systems (or application groups in the container example).

Because the "generic" adaptability of the container and/or paravirtualized environment allows us to deploy on the most basic hardware it also exposes us to the RAS issues of that hardware. So if we choose to deploy on a PC, we can expect the same degree of RAS for our virtualized hosts as we would from the underlying PC. This is true for any implementation, but presents a greater problem for these environments only because they will *allow* us to deploy to a sub-standard piece of hardware.

Another issue of the container and paravirtualized environment is that they represent a larger *target* for failure as they consist of significantly more things to go wrong in the complex hosting environment. This environment, by nature of being an standard OS, potentially contains other vendors code at the kernel level in the form of drivers and loadable modules. The larger the virtualizing environment, the more exposure the implementation has to failure.
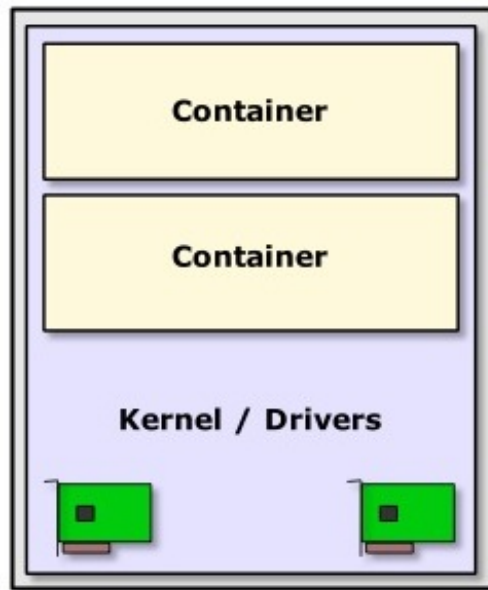
Figure 1. - Containers

One arguable benefit of virtualized environments is that they do not require special considerations as they *appear* to the virtualized OS to be running on real hardware. Containers bypass this completely by running only a group of processes that have a degree of application compatibility by replicating the OS interfaces (ie: file systems and network stacks) that the application is most likely to utilize. In the case of our example Java application we should see no discernible differences between the container and a full OS implementation. Applications requiring specific kernel access (such as NFS) will find that the container is not applicable as it does not have its own kernel.

The paravirtualized environment acts much like a type 2 hypervisor virtualized environment shown in figure 2, but does not qualify as either a container or a proper type 2 hypervisor because it has its own kernel and drivers which differentiates it from the container implementation in figure 1 but does not offer a full and unaltered kernel as typically seen in a fully virtualized environment.

Type 2 hypervisors, like containers and paravirtualized environments use OS scheduling for CPU arbitration and rely upon OS knowledge of the hardware for optimum memory configuration and CPU affinity to optimize memory access. The hypervisor implementation is also similar to the two as it drives all client I/O through the normal kernel I/O interfaces. The type 2 hypervisor is different from the two in that it offers a fully independent client OS that *appears* to the OS to be running on normal, although highly generic, hardware.
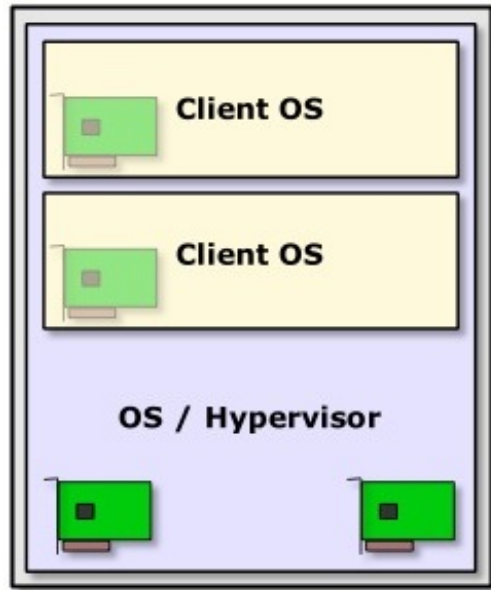
Figure 2. - Type 2 hypervisor with I/O

All three implementations are exposed to the limitations borne by the OS that they run on. Should an administrator make a mistake of some sort in the parent / hosting OS, all client OS's are suspect to failure.

The type 1 hypervisor solves this largely by making the hypervisor largely off limits to most types of access. It cannot run OS commands, you cannot log into it, and it does not have an extensible interface that accepts plug in modules or drivers. This simplicity of design tends to make the type 1 hypervisor significantly more stable and appropriate for enterprise level workloads. Most hypervisors meeting this description are run as firmware based services and are virtually invisible to the operator of the system.

Because the type 1 hypervisor is so small and lacking a driver interface, a method is required for hosted client OS's to have access to I/O. Some implementations allow for each client to have direct I/O access to individual physical devices, but the most popular method is for a specialized client partition to also be a *server* of I/O devices. This is frequently called a VIO (Virtual I/O) Server and is illustrated in figure 3.
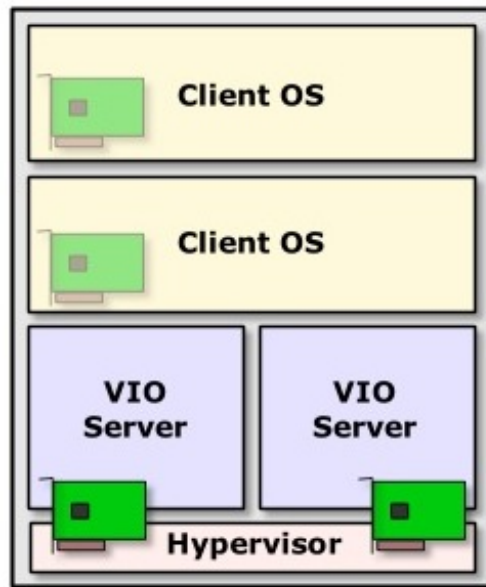
XIII

Figure 3. - Type 1 hypervisor with VIO

VIO servers are specialized partitions that re-serve I/O resources to the virtualized clients. This is the same as with the type 2 hypervisor in that we are re-presenting a disk or network device that the client OS can load a driver for and access, it is different in that the type 1 hypervisor is separate from the VIO server. Because they are not implemented in the same *space* they become more specialized, simple, and reliable. This independence also allows us to implement two or more VIO servers to provide redundant paths to network and disk. This ability to simplify and strengthen components while duplicating others makes this type of implementation exceptionally independent and reliable in comparison to other solutions.

Each of these implementations are not exclusive. Most vendor implementations offer a mixture of options that defies placing them neatly into a single category. Some type 1 hypervisors allow for physical devices such as network cards and HBAs to be assigned to a client OS while still virtualizing sub-CPU access. Some container implementations have been extended to allow modified kernels to run in the container.[13]

---

[13]Sun "Brand Z" zones are modified containers that emulate Solaris 8 kernels running on Solaris 10 systems. While this is not a full kernel implementation, it clearly blurs the lines of what a container is.

Summary

Virtualization is part of the natural evolution of the IT environment. As complexity and scalability issues have risen in the "scale-out" approach to growth, numerous system virtualization tools have been developed to negate these issues. Vendor provided solutions in IT respond in expected market driven patterns to human-time intensive activity to allow us to do more with less.

When evaluating virtualization technologies the system architect should read and consider carefully each vendors implementation and how they will meet the specific needs at hand. As with most choices in IT, each virtualization option presents strengths and weaknesses in price, performance, and reliability. One size does not fit all, and the consumer must decide based upon requirements and tolerances for each.

Early partitioning required specialized and expensive hardware that was divisible only on *crude* boundaries. The economics of early systems of this type provided little incentive to leverage these features. Virtualization has grown in capabilities to provide sub-CPU allocation / oversubscription, memory re-use, I/O interface normalization and optimization to name a few. These features have been expanded to provide live migration of running systems from one physical system to another.

The ability to host large amounts of virtualized systems on a single set of CPUs and I/O devices has allowed us to fully leverage the power of these new resources. The more complete utilization of each resource, with the same workload, has allowed us to deploy less of each, and therefore present a footprint that is "green" optimized - both environmentally green and saving on the financial green.

About