Virtualizing Pervasive Database Servers

A White Paper From

For more information, see our web site at
http://www.goldstarsoftware.com

Virtualizing Pervasive Database Servers

Last Updated: 03/06/2014

As servers continue to advance in power, adding more and more memory, disk space, processors and cores into each physical box, many system administrators are finding that their servers are woefully under-utilized. Using a server to only a small percentage of its capacity wastes resources such as electricity, cooling, and space.

For some time, the solution was server consolidation, in which multiple servers performing single tasks were consolidated onto a single server, performing each of those tasks separately. For example, setting up a single server to handle DNS, DHCP, Email, file services, print services, database services, and more was very common. This works well in smaller environments, but a failure of one process can often take down the entire server.

Virtualization has arisen as an alternate solution to the utilization problem. A virtualized environment takes one or more virtual servers and implements them on top of a single physical hardware platform. This is controlled by a component known as a hypervisor, which allocates resources (CPU, memory, etc.) to each virtual machine (VM) and translates I/O requests from the virtual machines to the physical hardware underneath. This translation layer make the virtualization almost invisible to the guest operating system and applications, which believe that they are running on dedicated hardware.

Before we talk about the details of running Pervasive in such an environment, let's review the core advantages and disadvantages of virtualization.

Advantages of Virtualization

The advantages or virtualization have been proven in many environments. Virtualizing your under-utilized servers can provide real "hard" savings, including many of the following:

- Lower physical hardware costs, since one piece of hardware can now provide resources to numerous virtual servers.
- Lower software licensing costs, since some software is priced differently for virtualized servers, and some operating systems come with licenses that offer VM access on multiple machines for "free".
- Lower electricity usage and costs, since there are fewer hardware servers that are more heavily utilized, the power loss due to power supplies and underutilized servers can be minimized.
- Lower cooling requirements and costs, since having fewer hardware boxes generates less heat, thus reducing the need to cool the server room.

- Less rack space needed, due to having fewer physical servers. This can make the data center or server room small, further cutting costs due to office space and cooling.
- Fewer (or smaller) hard disks are required, since the virtualized drives use up only as much physical drive space as they actually need at the time. Physical drive space can be assigned dynamically as the servers grow, reducing the need to over-buy disk space or plan for upgrades on a periodic basis.
- Less noise in the office or server room, due to having fewer servers and a less powerful cooling unit.
- Fewer switch ports required to connect all of the servers, which can offer cost savings on the network buildout and simplify cabling installations.

Additionally, you can see "soft" benefits as well:

- Easier remote access via dedicated remote access consoles to many servers, eliminating or reducing the need for expensive keyboard/video/mouse (KVM) hardware switches or other "lights-out" (ILO) solutions.
- Faster provisioning of new machines, since new "servers" can be brought on-line in a matter of minutes by copying an existing "base" server environment to a new one and making a few minor modifications.
- Easier splitting of services onto dedicated machines. Due to hardware and OS costs in the past, many sites used to consolidate their servers and have many services running on one box. This led to a problem if one service crashed and needed a reboot, it often impacted other services. Now, the services can be easily broken up into their own "sandbox", which can be rebooted without impacting anyone else.
- Dynamic resource management is gained by giving you the ability to add more memory or CPU resources to servers that need more, and to remove resources from servers that need less. This makes it much easier to respond to the changing needs of the servers in the data center over time, reducing the need to physically upgrade servers. The same can be said for disk space, since you can allocate disk space to servers from a large pool of physical disk space, and servers that will need lots of disk "eventually" don't need to suck up that space immediately.
- Easier system snapshots can now be taken, as VM's are stored in physical files on the host system or SAN. You can easily shut down a machine, take a snapshot of the machine or backup copy, and restore that snapshot very easily, using normal file copy tools instead of needing "bare metal" restore tools.
- Improved fault tolerance gained through the ability to move a VM from one physical platform to another, allowing hardware maintenance to occur without the users knowing that anything is happening. Advanced VM systems also offer the ability to migrate machines "in use", with zero downtime.
- Ability to run older applications and even unsupported operating systems on newer hardware, providing a migration path for legacy systems. This can be extended to non-production backups of data which must be retained due to legal

requirements, such as health care data. Physical machine configurations can be migrated to VM's for this purpose, and then shut down when they are not needed.

Disadvantages of Virtualization

Like any other technology, there are disadvantages to virtualization. Here's a short list:

- While some hypervisors may be free, others may have a cost to acquire, depending on the features needed. Like with any other technology, the more features you want, the higher the cost.
- Even with free hypervisors, there may be a learning curve to understanding how to use the provided tools, so there may be required training or other time-related costs.
- Hardware failures can easily impact numerous servers all at the same time, and a failing hard disk array can completely shut down an entire environment.
- Administrative costs (managing servers, patching, etc.) is based on the number of VM's , not the number of physical servers. Making it easy to create a new VM in an environment means that the system administrators may have to continually maintain and patch those servers and their hosted applications.
- Higher licensing costs can be seen, especially in "necessary" tools such as anti-virus, server management and patching tools, and other such packages that are installed on a per-OS basis.
- Reduced performance can be seen on many VM's with an increased I/O path. Since any I/O request inside a VM has to pass through a translation layer and then get passed to the physical hardware, it takes longer to perform each operation.
- More difficult troubleshooting is often the norm in a VM environment, as the interplay or problems from one VM to another running on the same physical machine get much more complicated. In other words, a disk-intensive process running on one VM will have an impact on a disk-intensive process running on a completely independent VM hosted on the same physical hardware. With physical machines, you can always isolate the environment, so you would never see a similar issue.

Virtualization of your Pervasive Database Engine

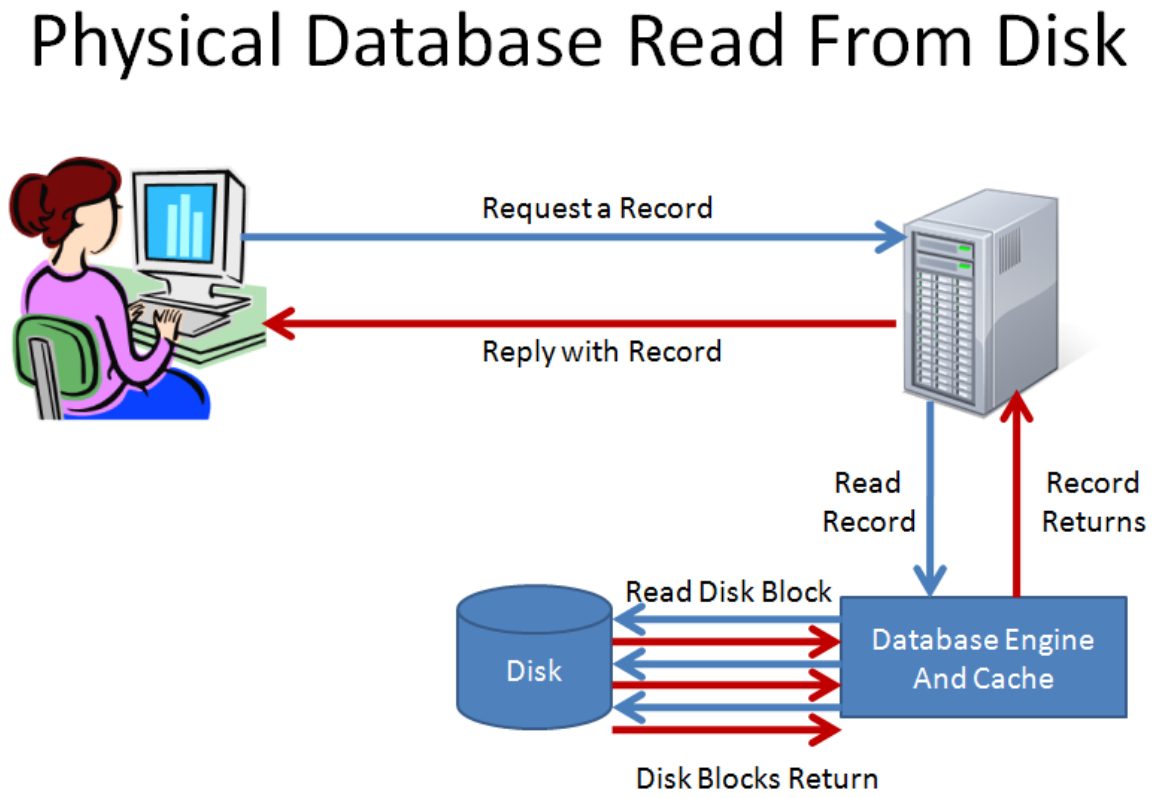
Officially, the Pervasive database engine is fully supported in a virtual server with respect to *functionality*. This means that Pervasive will help you troubleshoot any issues due to crashing or other problems on virtual servers.

However, they do NOT guarantee *performance* in these environments. This means that if the system is not performing well, the official reply from Pervasive may just be "oh well". Honestly, I cannot say as I blame them, as troubleshooting this type of environment is VERY difficult, and you have to look at the combined load of numerous virtual machines all at the same time to understand what is happening on the physical box.

To "V" or not to "V" -- that is the question. To get a better understanding of the issues, let's look at how your Pervasive client/server application works today. For this discussion, we're going to concentrate on the use of applications built on the Btrieve interface, since this comprises the bulk of applications running on Pervasive PSQL today.

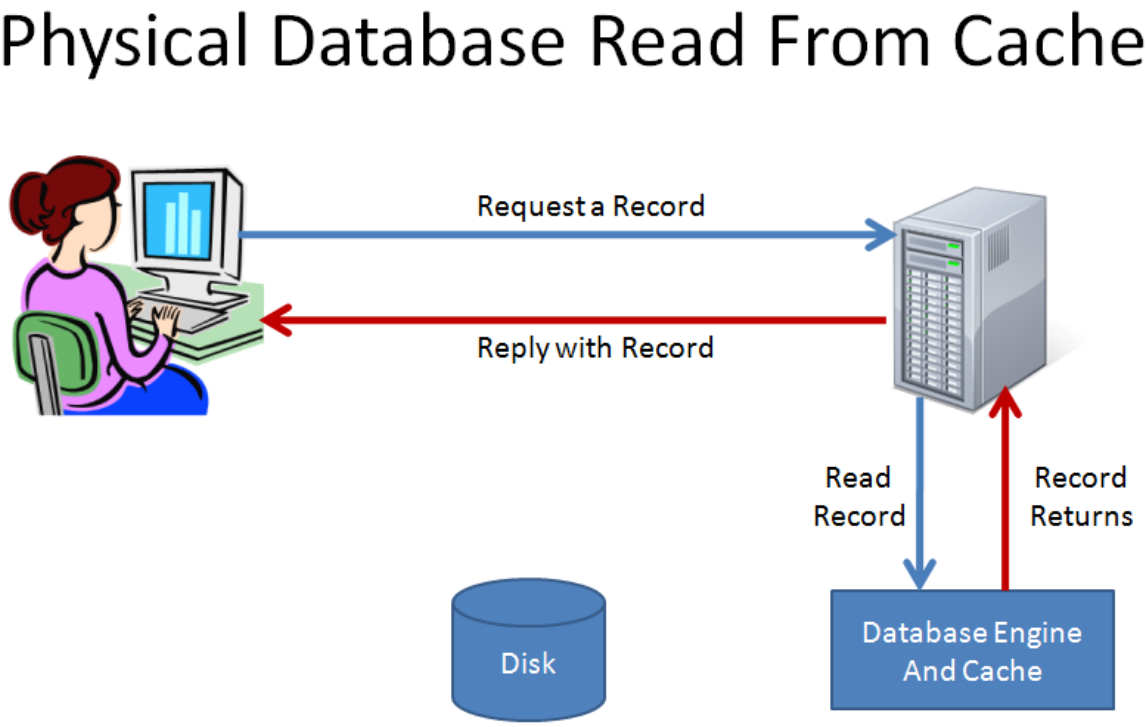
What Does My Application Look Like on a Physical Server?

In our first picture, we're looking at a typical "first" request from a user, where the data is not yet in the database cache. This happens when a server is first started and the first users are signing on.



In this environment, the user submits a request to the database across the network. The request gets to the database server and generates a network interrupt. The operating system fields the network interrupt, then passes the request to the controlling application, which is the Pervasive Communications Manager. The request is queued waiting on a Communication Thread to be available in the engine. When a communications thread is available, the request gets processed. While processing, the engine determines that the data is not yet in the cache. So, it must build up a disk request to find the exact right page to load. In many cases, this takes multiple disk requests, since we must read file control records, page allocation tables, index pages, and data pages to read any given record into memory. Luckily, the database engine has a low-latency, high-speed connection to the disk drives, so the time to complete multiple reads is usually minimized here.

Once the data is in memory, subsequent database reads can be handled directly from the database cache memory, and the need for the disk accesses disappears. This is what we gain by having a suitably large database cache, as shown in this picture.



We still have the round trip time from the user to the server and into the engine, but since the data is already in RAM, we can avoid the disk I/O and return data immediately. Since memory access easily provides 1000x the performance of disk access, this simple change can give us a tremendous performance gain, usually by a real factor of 100x or higher. This gain is why careful consideration of your database cache memory is essential to getting the best performance in your Pervasive environment.

What About the Network Round Trips?

If you look carefully, you'll see that there is some additional overhead here. The request has to get from the user workstation to the server and the reply has to get back. This round trip time is called the network latency. The amount of latency is determined by the network links, cabling, number of intervening devices, and more. In a decent environment, you can expect a round trip time of 0.05ms. Slower servers and networks may see a round trip time (RTT) of 0.3ms. Wide area networks often have RTT latency of 10ms or more, because it simply takes that long to get the packet from the client to the server. You can get a rough idea of latency with the PING command, but for more accuiate values, you'll need a network analyzer.

Now, 10ms doesn't seem like a long time to most people. One thing to keep in mind about latency is that it impacts EVERY SINGLE REQUEST, regardless of the network

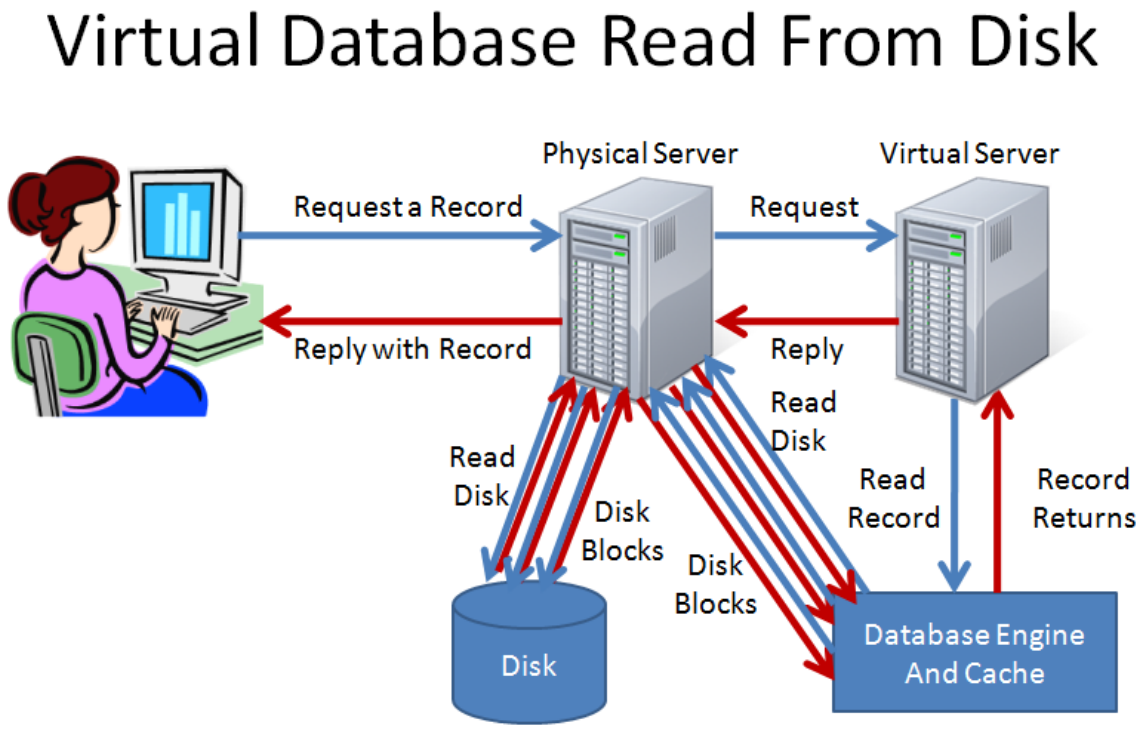
bandwidth. The net effect is that it acts as a multiplier on the time it takes to complete any set of operations. For example, let's say that your application has to read 5,000 records to display a screen of data. If your network environment has 0.05ms RTT latency, then you would expect to complete this process in 0.05ms * 5000 = 250ms, or 1/4 second. To the user, anything under 1/4 second will seem to be "instantaneous", so this is considered good response time.

However, run that same application over a high-latency link (with 10ms RTT) and do the math. You'll see that the process now takes 10ms * 5000 = 50000ms = 50 seconds, or almost a full minute! Now THAT is an obvious speed problem that your users will complain about!

What Does My Application Look Like on a Virtual Server?

Now, let's take and move our database engine from a physical server to a virtual server. In an ideal world, nothing would change -- the application wouldn't know about the change, the workstation wouldn't know about the change, the server OS wouldn't know about the change, and the database engine wouldn't know about the change -- so it should run identically.

Too bad we don't live in an ideal world. Check out this picture:



What REALLY happens is that every network request is first sent to the physical machine, where it gets intercepted by the hypervisor. The packet is then processed and

forwarded on to the virtual machine. After passing through the OS, the request finally gets to the database engine. If disk I/O is required, the I/O requests are sent to the "local hard disk" of the VM -- but this request again gets intercepted by the hypervisor, translated to a physical request, and finally passed along to the physical machine.

Remember our analysis of latency on the network? Well, we just increased the network latency, because the hypervisor is now "in the way" of the database engine, and the hypervisor takes a finite amount of time to process each packet and forward it along to the virtual server. Even if the additional network latency is only 0.05ms, this could easily double the round trip times to the server, and double the response time of the application itself, too.

What about those disk I/Os? Again, we are introducing additional latency in every single disk I/O here. We may be making three disk I/O's, but they are going through one at a time, not all at the same time. The net effect is like putting up stop signs on a highway. You may still have three lanes of traffic with plenty of capacity, but each vehicle must keep stopping periodically, making it take longer to run the round trip.

Why Does Increased Latency Impact Database Servers?

Put simply, almost everything a database server does involves I/O operations, which are all slowed down by the virtualization layer. Network requests are coming in for records, the database engine has to read one or more disk blocks to satisfy that request, and then the reply can get sent back. Since the network requests and replies are slowed down as they pass through the hypervisor, network latency increases. Further, since many applications are designed to read many small records one at a time, the effects or network latency are multiplied by the number of individual requests, making the problem worse.

On the disk I/O side, we need to look at the typical Btrieve database file. In many cases, the page size of the file is 4K or less. (You can check your own page size with BUTIL -STAT or by creating a Stat Report from the Maintenance Utility.) However, when starting a new record request, the engine doesn't know which pages it needs to read. So, it starts with the first page of a file, the FCR. Once it has this page, the engine determines that it needs to read a 4K index page. However, it first has to read a page allocation table to find out where in the file the index page resides BEFORE reading the index page. After reading the first 4K index page, the engine may decide that it needs a second 4K index page (and possibly a second PAT page), and possibly more. Then, finally, it can read a 4K data page (and maybe a PAT page) to find the needed data.

Note that each of these requests must be serviced in order -- there is no way to run these requests in parallel, because the engine doesn't know what page it needs next. As a result, you can be sitting waiting for 5 to 10 different page reads from the disk. Even at a 5ms read time from disk, this can still take up 50ms of time. Increase the latency to the disk by forcing it to go through a virtualization hypervisor, and you increase the time required accordingly.

Why Does Increased Latency NOT Impact File Servers?

Obviously, people are virtualizing servers all the time and not running into major performance problems. Why is this? Let's examine how a typical file server or Email server works.

Typically, users make a request for a file, and then start reading that file 64K at a time. The first request gets sent to the virtual server, which has to undergo the same latency issues as it passes through the hypervisor on its way to the virtual server.

The VM finally receives the request and issues a single 64K read from the disk. The request also has the same latency as a single disk request in the database example. The main difference from a database server, though, is that we are requesting the entire 64K block at once, instead of doing individual reads. The OS can do this entire block in a single function call quite efficiently, whereas reading 64K of data from a database at a rate of 4K per call takes 16 times the latency!

Once the data has been read into memory, we then have to get the data to the user. Again, file servers know exactly what data is being sent, and the typical requests ask for 64K of data at a time. The server sets up a streaming buffer and sends out the first 1500 bytes in a data packet. This packet, like in the database server, is intercepted by the hypervisor and processed out to the physical network, adding latency to the round trip time and increasing the delay in the reply.

However, instead of waiting for that reply to get to the workstation and waiting for another request from the client (thus incurring a round trip time delay), the server instead streams the remaining data packets across the wire as rapidly as it can. This streaming effectively eliminates the multiplying factor of the increased network latency, since we only have to worry about the high latency on the first packet. The net result is that latency is minimized substantially, and the overall percentage drop in performance is also minimal.

Minimizing Latency for Virtual Pervasive Servers

Based on this analysis, we can easily see that virtualizing a Pervasive database server will not be for everyone. However, if you can avoid the high latency operations, then perhaps you won't have any issues, and you CAN reap the benefits of virtualization.

How can we avoid latency? Simple:

- Eliminate Disk I/O Operations
- Eliminate Network Requests

OK, sounds simple, but it's really not so easy.

Minimizing Latency By Eliminating Disk I/O Requests

Let's start with the disk requests, because this is the easy one. As previously indicated, if we store data in the database cache, we can eliminate the need to pull data from the disk. What would happen if we stored the ENTIRE database in cache? Essentially, the server would NEVER need to go to disk, and performance wouldn't suffer due to disk latency!

If your database is small, and if your server has enough memory, then you can simply add up the sizes of your disk files, set the "Cache Allocation" to that amount, restart the server, and go. Eventually, as users access partsof your database, the database pages will be fully populated in RAM, and your disk I/Os will be limited to database writes (which are batched together by system transactions). You can even run a process like Goldstar Software's BtrvLoad to pre-load data into memory at very high speeds, eliminating the need to wait for the data to load normally.

However, if you are on a 32-bit operating system, you must remember that there are limits to the available memory addressing space -- 2GB is the max, and if you try to go much over a 1GB cache, you may run into this maximum, making your system unstable. As such, this only works for the smallest of databases.

The next step, of course, is to move to a 64-bit server environment. With x64, we have access to a whopping 8TB of addressing space, and an L1 cache maximum size of 4TB. Although nobody is building servers with that much memory yet, some companies have built servers with 128GB of RAM, and have allocated 100GB to the database engine. In other words, it is NOT impossible for your virtual server to be granted 48GB of memory to host your 40GB database environment -- allowing you to cache everything.

Minimizing Latency By Eliminating Network I/O Requests

What about network calls? Unfortunately, unless you are a developer, this is out of your hands. Eliminating network calls would require a complete redesign of the application, using either the SQL interface (which retrieves entire sets of data) or perhaps the Btrieve GetNextExtended function call, which can read hundreds (or thousands) of records with a single network round trip. (If you ARE a developer, give us a call -- we'd be happy to discuss the benefits of these options for your users and help you develop a working solution with them.)

Luckily, this does NOT mean that you are completely without recourse. As many people with WAN links are already aware (including you, if you've read this entire paper), the high latency of WAN links causes performance problems for running applications. However, it is possible to eliminate this latency by running the applications on Terminal Services or Citrix machines, such that the applications run on the local network with the server, and only screen shots and key-presses cross the WAN link.

Well, can we do the same thing for out virtual box? Certainly! What if we ran the application directly on the database server itself, via a terminal services session? This is

a dangerous concept and one that you want to forget about from the get-go, since if a user does something stupid, the entire system can come crashing down. However, for time-sensitive processes with limited access requirements, this can be a great option. Beyond that, though, this options gets dicey in terms of server stability, and the increased memory loads from the terminal services users may impact the database even further.

Another possibility is to use a second virtual machine as a terminal server, and then host it on the SAME hypervisor/physical server as the database engine. With this layout, the network packets stay inside the hypervisor and never hit the physical network, which can reduce network latency even further. Of course, you can lose some important troubleshooting capabilities, and the increasing demands of the terminal services box can suck memory and CPU away from the database engine, slowing it down further, too.

Are There Any Ideal Solutions?

I think you already know the answer to that -- no. Everything that we do involves trade-offs. In short, we can boil it down into a few key concepts:

1. If you must have the absolute best performance, run on physical servers, and run time-sensitive processes directly on the server console itself.
2. If you run non-performance-sensitive systems (like test systems), then these are ideal virtualization candidates -- go ahead and virtualize them all.
3. If you want to virtualize for administrative gains (like backups, portability, etc.) instead of hardware utilization gains, then you should have just one VM hosted on your physical server -- that of the database server. A one-to-one virtualization retains some of the benefits of virtualization (but not all), as well as retaining the improved troubleshooting of a physical environment.

Beyond these simple cases, you're in a whole new world. You'll need to simply try it out in your own environment and see if performance is suitable for your users. If not, then database server virtualization is not for you.

Other Resources

Here are a few web links that provide information about virtualization:

- http://www.vmware.com/
- http://www.microsoft.com/hyper-v-server/en/us/default.aspx
- http://www.microsoft.com/windows/virtual-pc/
- http://www.microsoft.com/windowsserversystem/virtualserver/

And here are a few discussions about the validity of virtualizing in the real world:

- http://blogs.techrepublic.com.com/datacenter/?p=1595

- http://www.brentozar.com/archive/2009/03/why-your-sysadmin-wants-to-virtualize-your-servers/
- http://www.brentozar.com/archive/2009/03/why-would-you-virtualize-sql-server/
- http://www.brentozar.com/archive/2009/03/reasons-why-you-shouldnt-virtualize-sql-server/

If you are running vSphere 5.x, these VERY helpful papers may help you achieve better results:

- http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.5.pdf
- http://www.vmware.com/files/pdf/techpaper/VMW-Tuning-Latency-Sensitive-Workloads.pdf
- http://www.vmware.com/files/pdf/techpaper/latency-sensitive-perf-vsphere55.pdf

Note: Some of these papers are quite a technical read, so I recommend printing them out and going through them slowly and deliberately when you have a clear head!

Need some help with your Pervasive engine performance? Contact Goldstar Software and let us work with you!