SOA and Virtualization: How Do They Fit Together?

A White Paper from BEA and VMware
Introduction

Industry discussion on Service-Oriented Architecture (SOA) and virtualization positions them as separate, though related, topics today. Because of the wealth of public discussion on each technology, IT managers may wonder whether they are in any sense connected to each other. In fact, the technologies that come with virtualization can be seen as the key enablers for the flexibility promised by SOA. This paper establishes the technical links between these two popular developments in enterprise computing.

Enterprises are pursuing both of these developments with great speed and at the same time, although perhaps in different departments. SOA is of interest mainly at the business and technical architecture levels, whereas virtualization is principally the concern of the datacenter management and applications systems deployment groups. These communities are often separated from each other in the organization and are frequently driven by separate goals. Business and IT want to achieve greater agility in their business processes and applications, whereas datacenter management wants to reduce costs by consolidating computing, storage and network infrastructure.

These seemingly disparate concerns meet at the enterprise architect’s, chief information officer’s or systems release manager’s desk, where the mapping of SOA-based applications to the virtualization platform needs to be quite clear. The deployment platform of choice is fast becoming a virtual one, so that enterprise architects have to think about mapping their newly-developed SOA services to that virtual platform. The enterprise architect’s position is often where the development teams, pursuing their SOA strategy, meet the deployment teams, who are busy implementing virtualization in the datacenter and consolidating. The big questions then center on how SOA-based systems can best be mapped to a virtualized implementation, what the connection points are between them, and how one can make optimal use of the other.

This paper lays out the basics for how these two developments in the industry, and the technologies underlying them, can complement each other, if implemented correctly.

What is SOA?

There are many definitions for Service-Oriented Architecture in current use. The most widely accepted definition is that SOA is a set of architectural principles that help build modular systems based on “services” or units of IT functionality. These services, either at the business or technical level, are offered by one party, the service provider, and consumed by another. This idea of a well-defined “contract” that is fulfilled by a provider and used by another consuming party is central to SOA principles. Providers and consumers can reside in the same organization or in separate ones – even in separate companies.

The definition given here does not prescribe a particular form of technology for implementation of SOA, and the SOA services should be independent of their underlying implementation. Figure 1 shows the contrast between the traditional “monolithic” application construction scheme and the move to applications as service consumers, where the commonly-used services are extracted from the applications into a service repository. A definite benefit of this change is that the application logic diminishes in size. The numbers of business applications that make use of a common service largely determine that success of that service – if it is used only once, then it does not qualify as a service at all. However, the more applications that use it, the more pressure there is on the service to perform well and to adapt in a robust way to changing use patterns over its lifetime.
At the technical level, the contract between the consumer and provider parties is described in a Web Services Definition Language (WSDL) document that is agreed upon by both sides. The service provider and consumer can both be parts of one business application or business process or they can be separated. One of the biggest benefits of adopting the SOA approach comes when services can be orchestrated together in new ways, perhaps by new consumers (such as new applications), to provide new business functionality. To do that, prospective service consumers can query one or more registries for existing services that fulfill attributes they require, based on their WSDL definitions.

In general, we can regard the different types of consumer of SOA-based services as either
- a business application,
- another SOA service, or
- an orchestration sequence that makes up a business process.

There are notations for expressing the business process in a language that is web-service aware, such as the business process execution language (BPEL). Using BPEL has become one of the most popular means of constructing SOA-based systems from sets of existing services – but of course the SOA services must exist first before that can be done. BPEL-based construction of SOA systems is sometimes referred to as the “orchestration” of services.

SOA services are knitted together in a flexible way using BPEL and business process management tools. Doing system construction in this way allows changes to be made to the business process without changing the implementation code for the services, when business rules change, for example. Instead, the business flow is changed in a higher level language, without affecting the services themselves.

The SOA services that support business applications or business processes are designed to be reusable components – with the intention that they will be leveraged across many service consumers. The services may be contacted and used by means of well-known end-points (such as a URL) or by using a more dynamic attribute-based lookup in a services registry, or through use of
flexible message routers. Service registries have appeared on the market as UDDI-conformant products, while intelligent message routers appear as components of an enterprise service bus, making up the backbone of a SOA.

Having a set of services is a good first step towards the SOA end goal. However, there are also several other concerns that need to be addressed to create order in the system. These have to do with the rules of service interaction, quality of service provided, performance, governance, security, availability and overall systems management. Without these functions in place, the SOA ecosystem becomes a free-for-all, where service consumers and service providers can do whatever they like and perhaps compromise the performance or integrity of the overall collection of services.

Initial proof-of-concept exercises and real deployments of SOA have discovered this fact very quickly and have put SOA policies and management tools in place in order to deal with it. SOA in this sense is as much about the policies that govern the services, as it is about the functionality of the services themselves. One of those important management policies has to do with how a new service is deployed into the ecosystem, or how to supplement an existing service with more processing power. This topic is discussed further later in this paper.

**Benefits of Adopting the SOA Approach**

The key benefits of adopting SOA architectural principles are:

1. More agile business processes, through faster composition of those processes from business-level services;
2. Quicker creation of new applications through re-use of libraries of existing services;
3. More flexibility in application construction through a single design approach that connects legacy systems to new systems using one design paradigm;
4. Composability of new functionality through the modularity that services offer as a design concept.

There are implementation technologies available from several vendors that can validly claim to be SOA technologies. Many of these technologies relate the concept of a service in SOA to the technical concept of a "web service" – often as a one-to-one mapping. A "web service" is a piece of software that offers its interface to its consumers through a Web Service Definition Language (WSDL) definition that expresses the contract with the consumer described earlier, and communicates using SOAP messages.

The technology that underlies web services and WSDL is called Simple Object Access Protocol (SOAP) which imposes some structure on the language constructs of XML (specifying formats for message headers, message body and protocol details). Using SOAP to build a web service is really just the starting point for a SOA, the most basic building block - and it is where most system developers use SOA technologies for the first time. Despite the fact that a web service is a lower order object in the overall business hierarchy, it is a useful starting point for bringing together virtualization and SOA technologies.

**Lessons Learned from Early SOA Implementations**

The expectation in the early days of SOA popularity was that once you have created a collection of web services, then you have implemented a full SOA-based system and you are done. However, there have been some key learning points in the industry as SOA-based implementations have begun to appear supporting production systems. Among those learnings are:
Lesson 1: Services can exist at several layers of an organization, from the simplest IT utility functionality (such as a service for checking a customer's identity or their valid address in a database) to services at a much higher business level (e.g., the "placing a purchase order" or "on-boarding a new employee" processes in an enterprise, represented as a service). These different layers of services in the organization must be categorized and their impacts on each other understood, before the SOA can really work well. There are a number of smaller, lower-level, single-function utility services that are useful to the higher business-level services in many organizations.

Lesson 2: A collection of lower-level technical web services, while useful, does not fulfill the promises of agility that were expected because these smaller services are too fine-grained a unit of re-use to have an effect on the agility of a business process.

SOA applied to business level processes is much more likely to give the re-use and process agility benefits that the industry wants. These are coarse-grained services – giving much higher value when they are re-used. Developers have exerted much effort to understand the orchestration and business process management (BPM) fields as they apply to SOA.

Lesson 3: Management, security and governance are key areas in getting control of the SOA. Imposing some quality of service standards on the collection of services that are published for use is a necessary measure for success.

Lesson 4: SOA service-based interactions must be measured and monitored just as they are in any client-server system. When the performance of a service is not up to its required levels, that event must be first noticed and remedial actions should be taken. Possibilities for remediation include shifting the load to another place or expanding the capacity of that service to deal with the traffic. Load balancing for performance is one of the key join-points between SOA and virtualization that will be discussed later in this paper.

Lesson 5: One approach for moving to SOA is to look at the current inventory of business applications, extract from them the common functions and "break the application up" into smaller pieces. The question then becomes how to re-assemble those pieces in the most flexible way possible – and then where to assign those smaller services to be deployed. That is precisely where the questions to do with the virtual infrastructure platforms begin.

What is Virtualization?

Virtualization is an abstraction layer that decouples the physical hardware from the operating system to deliver greater IT resource utilization and flexibility. Virtualization allows multiple virtual machines, with heterogeneous operating systems inside them, to run in isolation, side-by-side on the same physical machine. Each virtual machine has its own set of virtual hardware (e.g., RAM, CPU, NIC, disk, and so forth) upon which an operating system and applications are loaded. The operating system sees a consistent, normalized set of hardware regardless of the actual physical hardware components.

A virtual machine encapsulates an operating system and any applications that live within it into a single unit that is independent of the hardware on which it runs. The main unit of modularity, application isolation, and execution in virtualization is the virtual machine. Many virtual machines may execute at one time on a software platform called a hypervisor. The job of the hypervisor is to insulate the virtual machine from the hardware in the same way that an operating system insulates its resident applications from the hardware. Virtualization allows us to separate the
application further from the underlying hardware – and indeed to separate the operating system from the hardware also.

Virtual machines in VMware Infrastructure 3 are encapsulated into files, making it possible to save, copy and provision a new virtual machine from a template rapidly. Full running systems (fully configured applications, operating systems, BIOS and virtual hardware) can be moved within seconds, using VMware® VMotion™ technology, from one physical server to another with zero-downtime and continuous workload handling. This capability provides a significant benefit in maintenance and upgrade of systems.

VMware® ESX Server, a widely deployed hypervisor technology, runs directly on the hardware and schedules a set of virtual machines to execute. One can think of VMware ESX Server as the operating system for virtual machines. It gives each virtual machine a time-slice of the available processor cycles and a share of all of the other machine resources.

Built on top of this fundamental ESX Server component are several other features that make up the full VMware Virtual Infrastructure 3 suite of products. These are:

**VMware® VirtualCenter Management Server (VirtualCenter Server):** The central point for configuring, provisioning, and managing virtualized IT environments.

**VMware® Virtual Infrastructure Client (VI Client):** An interface that allows users to connect remotely to the VirtualCenter Server or to individual ESX Server hosts from any Windows PC.

**VMware® Virtual Infrastructure Web Access (VI Web Access):** A Web interface that allows virtual machine management and access to remote consoles.

**VMware® VMware Virtual Machine File System (VMFS):** A high-performance cluster file system for ESX Server virtual machines.

**VMware® Virtual Symmetric Multi-Processing (SMP):** This feature enables a single virtual machine to use multiple physical processors simultaneously.

**VMware VMotion:** This feature enables the live migration of running virtual machines from one physical server to another with zero down time, continuous service availability, and complete transaction integrity.

**VMware® High Availability (HA):** This feature provides easy-to-use, cost-effective high availability for applications running in virtual machines. In the event of server failure, affected virtual machines are automatically restarted on other production servers that have spare capacity.

**VMware® Distributed Resource Scheduler (DRS):** This feature allocates and balances computing capacity dynamically across collections of hardware resources for virtual machines.

**VMware® Consolidated Backup (Consolidated Backup):** This feature provides an easy-to-use, centralized facility for agent-free backup of virtual machines. It simplifies backup administration and reduces the load on ESX Server hosts.

**VMware®Infrastructure SDK:** This feature provides a standard interface for VMware and third-party solutions to access VMware Infrastructure.
The relationships among the various components of VMware Infrastructure are shown in figure 2.

Many benefits arise from moving systems to a virtualization platform, including:

1. the ability to consolidate the workload from multiple physical machines on to fewer servers, each executing several separate virtual machines and as a result reducing hardware costs;
2. enhanced business continuity, through the use of distributed resource scheduling and failover technology across clusters of ESX Server hosts;
3. enabling developers to control their environments in a more flexible way through self-provisioning of complex arrangements of virtual machines,
4. the ability to point another developer to a previously saved configuration of several virtual machines for testing and debugging purposes, thereby achieving time savings in the machine setup phase.

Figure 3 outlines the effect of adopting VMware Infrastructure as the basis for implementing business applications. Reducing the number of physical machines has had a significant positive impact on the cost of power consumption, availability of datacenter physical space, efficiency of administration management staff, and server deployment time — providing significant savings to
many organizations. In figure 3, six (or more) physical servers have been consolidated to two servers, each running VMware Infrastructure 3 and handling three virtual machines.

![Figure 3: The move from dedicated servers for single applications to a shared virtualization infrastructure](image)

**Key Connection Points between SOA and Virtualization**

Figure 4 below depicts a simple overlay of virtualization on to the SOA outline diagram. Here both the service consumer, i.e. the business application or the business orchestration script, and the service provider are candidates for hosting on virtual infrastructure. The BEA AquaLogic Service Bus, for example, can execute on VMware Infrastructure, as well as BEA WebLogic Server, both of which are platforms for service providers or consumers.
The separation of service consumers from service providers leads us to an interesting decision point. Where can you gain more from a move to virtualization in the dual world of service consumers and service providers that make up the SOA approach? The first party to move to the virtualization platform must be explored in the context of your own enterprise, with your own business applications and operating systems as the determining factor. Certain applications and operating systems lend themselves more easily to virtualization than others. There are tools and processes from VMware that help with this decision. Many enterprises will mix the physical implementations of certain functions with virtual implementations for some time to come. There is no reason why the SOA architecture should not span both.

Dynamic provisioning of the service provider side of the SOA, using virtualization techniques, offers significant gains. This suggests that it is advisable to build SOA services on a virtualization platform to start with. However, there are also benefits to be had from the virtualization of the application or service consumer side. You will notice in figure 4 that not all applications have an operating system supporting them, particularly the applications named App 4 and App 6. These are Java applications and services that make use of Java Virtual Machine (JVM) technology from BEA that is itself closely integrated with the ESX hypervisor.

The same principle can apply to the platforms on which the SOA services live. The BEA LiquidVM communicates directly with the ESX Server software beneath it, exchanging information about resource allocation and obviating the need for a separate full operating system layer between the two. Removing the traditional operating system reduces the footprint of this new type of virtual machine, allowing more of these virtual machines to be accommodated on a physical machine.

It is also possible to virtualize parts of the SOA services infrastructure without virtualizing all of them. Partial virtualization is a useful step if an enterprise is new to virtualization or to SOA. Parts of the service may reside on a virtual platform such as VMware Infrastructure, while other parts of it
may reside on a physical one. A J2EE application can communicate with legacy mainframe systems using older protocols, but at the same time present SOA interfaces to its consumers. The J2EE and web services implementation can live on VMware Infrastructure while the older legacy system stays on its original physical platform. This scheme is followed today with Java/J2EE applications that communicate with legacy systems, and it is equally applicable to the SOA world.

The discussion so far has focused on the application and business level of the SOA at deployment time. The impacts of SOA and virtualization together can be felt in a much broader way than just in that phase of the software lifecycle. These two movements certainly impact developer and QA/test personnel, for example. Let’s examine the SOA development and deployment lifecycle to see where the two moves can together contribute value.

The SOA Software Lifecycle
This section explains how a service oriented architecture is developed, deployed, and managed.

1. Development of the Service-Oriented Architecture
Although the basic unit that makes up the SOA implementation can be seen as a simple business or web service, the technical infrastructure that underlies the service can be complex and multi-tiered. Data to support the service interface is retrieved and updated, often in a database access tier of the application. The business rules logic can either make use of the service or those rules may alternatively be implemented by the service itself – or both. These business rules are typically built on an application server or “middle” tier. Presentation logic is often implemented on a separate, third tier, perhaps using portal technology. Sitting between these tiers may be internal and external firewalls, accelerators and load balancers, either in software or in hardware form.

In fact, SOA services that support business applications are often implemented today as thin SOAP wrappers around existing functionality. That functionality is still hosted on .NET or J2EE application architectures or even on older legacy systems. This means that the development of SOA services encompasses and builds further on the development and testing of .NET and J2EE applications. Virtualization technologies have supported these types of developer and testing communities for a number of years, providing ease of provisioning of multi-machine configurations to the developer, as well as the capability to take snapshots of virtual machines in various states and to revert to previous snapshots of systems at will.

SOA developers can now have the service provider, service consumer components, and any intermediaries such as a service registry or security checker, all hosted on different virtual machines. This makes for an attractive environment for the developer. It produces great productivity gains for developers, who can now provision those complex SOA systems in a self-service fashion, without worrying about where to get the physical machines or the effects on other developers.

Many development organizations have pools of hardware available for the deployment of such multi-virtual machine systems. The mapping of a multi-tiered configuration of virtual machines to physical servers is no longer a concern for the developer, whose job is to get the software working in a robust way, not to worry about what component goes where. This synergy between SOA and virtualization at development time has had significant benefits in software organizations. New virtual machines can be brought rapidly into play to host the new services that the developer needs to carry out his/her work.
SOA, Virtualization and System Testing
With developer-focused virtualization technologies such as VMware® Workstation and VMware® Lab Manager, the developer team can leverage a library of useful configurations of multi-machine systems in a much more orderly fashion. VMware Lab Manager is a lab automation environment for managing complex systems of virtual machines. "Management" here is defined as storing and sharing complex configurations of multiple virtual machines, along with the ability to independently check them in and out of a library and make changes to them.

In a fast-moving development environment, it is easy to construct many configurations of multi-tier systems without paying much attention to already deployed configurations and the clean-up required when testing is done. Systems managers can find themselves with many redundant three-tier configurations that are no longer in active use. Separating those configurations, or sets of virtual machines, that are vital to current day development work from those that are not can be a significant task – Lab Manager helps the systems manager with that task by cataloging all current configurations.

Helping SOA infrastructure developers to rapidly provision, tear down, and control their multi-tier environments is one of the key values that the VMware Lab Manager solution provides. It also allows quality assurance personnel to communicate information about complex three-tier SOA systems back to the original developers with ease, giving them a URL as a pointer to a whole SOA setup of virtual machines, rather than having to re-create the entire system that exhibits the problem.

2. Deployment of the SOA

Release Management
Getting a set of services that make up an entire SOA implementation from the development phase into the production phase is not the same as doing this for traditional non-SOA enterprise applications. There are more moving parts in the SOA ecosystem, because some of the services may be outsourced to another department in the company or even to another, external service provider. Coordination of several parties in a web services conversation is required. The SOAP protocol between the service consumer and service provider also requires more processing at a more basic level than was the case in earlier client-server systems, so more computing power is needed in the infrastructure.

Both a functional and integration testing phase is needed before rolling out any new SOA services into production. The benefits of using virtualization technologies for these phases of the development lifecycle are now well understood. Testing and QA departments gain significant time savings by being able to provision and replicate collections of virtual machines containing the application and services at will, without having to go through IT service departments to get that work done.

Management of an SOA-based System in Production
A new kind of entity is under management in the datacenter – the virtual machine. Along with determining the status of physical machines, networks and storage, operations personnel must now also be aware of their virtual equivalents. This can mean that there are many more objects to manage in their world – making their lives harder. Now consider the imposition of fine-grained services across these two categories of infrastructure. SOA services may be deployed across the physical systems as well as the virtualized ones at the same time. All of this new complexity leads
to a heightened need for further automation in the management technologies that ensure that the entire system will run well.

As the new technology is introduced, the portion of the IT budget that is used up in just keeping business systems up and running is increasing, so that the portion that can be used for innovation and improvement projects suffers. We really want the opposite to occur – that is, more emphasis on innovation and less effort consumed in “just keeping it running.” Hence the more pronounced need for automated management systems in SOA, so that systems management talent can be used for innovation projects.

Performance of SOA Services in Production

Since they are used by a range of different applications and business processes, it is hard to predict with certainty whether the services making up an SOA will be heavily or lightly loaded. This is particularly true when those services they are used by a range of different consuming applications or business processes. This issue is similar to that which was experienced in the initial days of successful consumer websites, where the capacity of the computing system supporting the site itself often had to be expanded rapidly to meet demand. Websites that made it through that performance crunch were provisioned with varying numbers of small, intact units of processing to support the critical web servers. This strategy is directly analogous to the provisioning considerations for SOA-based services and systems.

It is not hard to measure a single business application’s use of the SOA services during the load testing and quality assurance phase, provided those services are only used by that one application. What is more difficult to assess is what will happen to SOA services as they are employed in new use-cases or when their internal implementations are modified. What will the effect be on SOA services when we add another application or business process that uses their functionality? Will they be overloaded – or capable of handling the new requests?

One of the major motivations for implementing the SOA in the first place – modularity of services and the potential for re-use across different areas - now has implications for the service performance. This phenomenon gives rise to a need for automation of the service provisioning process.

Rapid Provisioning

SOA-based implementations comprise services that can be made up of fine-grained, smaller functions as well as larger, more business-oriented services. A fine-grained (“small” service) might be one that checks a customer’s address in the master customer information database, for example, with only one back-end system dependency. A larger service would be one that implements the processing of a customer order, for example, and touches multiple backend IT systems for any one transaction. The order management system will need to update the inventory and shipping systems to carry out the order. Larger business services usually make use of smaller utility services that perform one function well. The “handle customer order” service can make use of the lower-level “check customer address” service, in our example. These two services might also be deployed at some distance from each, over the network.

When SOA is implemented across the enterprise, a large collection of services is likely to be present. At a minimum, each SOA service requires a copy of itself to be running on a separate platform, to achieve a level of fail-over and load balancing. Although they may have started with small web services projects, several enterprises have already seen their population of services grow into the thousands. As the numbers of virtual machines and their physical hosts grows, and
more SOA services are spread across those machines, it becomes necessary to move parts or all of the SOA service provisioning responsibility from human hands to a more automated form.

The automated provisioning or management system must first have a means of measuring the current load on the whole system as well as on each of the important machines and services within it. It must also understand the levels of performance expected of each service. The system takes the appropriate steps based on management policies to bring the SOA system back within its performance constraints. These policy-driven functions are typically to be found in a management toolkit. The newer management toolkits have a detailed awareness of what is happening at the virtualization levels, allowing them to adjust the resources in the virtual infrastructure in order to achieve a business service level agreement (SLA). The Virtual Center management server in VMware Infrastructure, for example, delivers data about performance and availability of the virtualized components by means of a web services interface. That information is used by the BEA Liquid Operations Console to adjust the resource consumption at the JVM level.

When a set of services undergoes an unexpected increase in demand, such as may occur seasonally when a business application suddenly experiences higher user traffic, the whole system must be capable of flexing its processing power to meet that demand. This responsiveness is only possible when the SLAs that the business application system must meet are captured in the resource management tool – and when that tool has the capability and authority to expand the pool of resources to meet the demand. This capability exists within VMware® Distributed Resource Scheduler, part of VMware Infrastructure, and the new management tools from BEA that understand SLAs in the SOA and virtualization context. These BEA tools can make requests of the VirtualCenter server to add a new virtual machine to the collection, for example, to get more processing power into the SOA.

Adding new software and hardware infrastructure as required (and subtracting it when it is not required) is a key management function in the combined virtualized SOA world. The management tool typically works with a template for a deployable system. That template contains the necessary infrastructure software to support the application, such as a JVM hosting BEA WebLogic Server, and the application itself, correctly configured. As we have seen, there may or may not be an operating system in such a template.

The management tool instructs the virtual infrastructure (managed by VMware® VirtualCenter Server) to instantiate a new virtual machine and start it up. Application traffic may then be redirected to the new virtual machine and service implementation. When the need for extra processing power has dropped, the management tool can decommission the extra virtual machine from the pool, allowing hardware resources to be re-shared among the remaining live virtual machines. This capability is a key linkage point between implementations of SOA and virtualization.

The strategy for handling this kind of automation must be worked out at the enterprise architecture level. SLAs are first identified and described. The level of appropriate reaction to breaches of these SLAs is then laid down. When that reaction is to add more processing capability, then the best solution is to add the smallest unit that will clear the problem. For this purpose, a lightweight virtual machine configured with the minimal service implementation to handle the load may be the ideal approach.
Provisioning on the Hardware

Up to this point we have talked about the provisioning of a new service in terms of providing a virtual machine containing a resident operating system or a JVM of its own, some infrastructure software such as .NET framework or a J2EE application server – and finally the new SOA service. This discussion assumed that there was an existing physical ESX Server host on which the new virtual machine could be placed. However, it is possible to go one step further than that. The ESX Server software itself can be dynamically provisioned on to the available hardware, where no system existed before, and subsequently the provisioning system can create the virtual machine container to support the new service. This type of functionality is available in modern management tools.

Along with that, new Java Virtual Machine functionality, such as BEA’s WebLogic Server Virtual Edition (WLS-VE) allows the elimination of one step from the provisioning task, that of the installation of an operating system into a virtual machine, making the provisioning task both simpler and faster, since there are fewer components to install and configure (and fewer points of failure).

With BEA’s virtualization management tool, called Liquid Operations Control, IT departments can provision Java applications directly on the virtual infrastructure. This approach puts control into the BEA Liquid Operations Control resource broker, which supports policy-driven automation of provisioning tasks, relieves applications of the details needed to run, and frees infrastructure to be dynamically reconfigured and re-optimized. Without this capability, IT staff would have to take the time to configure each application and SOA service manually.

This new functionality allows us to think in terms of a pool of hardware and a pool of Java Virtual Machines that can be adjusted dynamically according to the demands of a new service or its users. This essential capability facilitates the success of the SOA deployment. Without it, overloading of a service or a back-end supporting application will lead to problems and user dissatisfaction, once a service is operating well enough to be in higher demand.

Provisioning Virtual Systems versus Physical Systems to Support a Service

There is a cost to provision any new entity in the computing infrastructure. This can be the cost of the hardware, the cost of installing appropriate software, the people and time consumption and the cost of space, power, lighting and other essentials. Some of the costs that apply to physical systems are not relevant to virtual ones, and the time to provision new machines has been significantly reduced in the latter case. This fact leads us to the conclusion that virtualized platforms are a key piece of the infrastructure making up an SOA.

However, there are occasions when the current physical compute power is fully allocated – or the required spare capacity is not available. On those occasions, provisioning a new physical system is the most appropriate action. A virtualization management tool – or capacity planner – can minimize these occurrences.

SOA Service Initial Placement

When a new service is first deployed into production, a suitable home must be found for it to execute. The same principle holds for copies or separate instances of an existing service. In the past, new application or service functionality has called for new hardware to be added to the datacenter. This is no longer the case in a virtualized world. In a dynamic environment, possibly with larger numbers of services already present, the need arises for automatic placement of new
services as they are released to production. Which component would make the best choice of location for such a new service?

The answer to this question lies in the knowledge that VMware Distributed Resource Scheduler (DRS) has of the consumption of various pools of virtual resources, along with the knowledge that the SOA management tool has of the infrastructure needs for any new service. A virtual machine that has known minimum and maximum CPU and memory requirements can be automatically placed on to one ESX Server host within a DRS cluster today, to achieve a good performance balance. This placement will be more accurate if the performance effects of the service on that virtual machine have also been characterized.

**SOA and Infrastructure Orchestration**

The act of coordinating SOA services at the business level maps down to coordinating the separate utility services at the infrastructure level. This lower level is called the Service-Oriented Infrastructure (SOI). The following discussion explores the infrastructure itself in order to clarify that mapping.

**The Infrastructure as a Service**

When the automated provisioning system described above sends a request to the VirtualCenter Server in VMware Infrastructure to provide a new virtual machine, it is consuming an infrastructure-level service. It uses SOAP APIs to carry out that request. Management tools from third party vendors request information from the VirtualCenter Server today in a service-oriented way, in order to populate their management consoles with that information. These two management tools are exhibiting SOI behavior – treating the infrastructure itself as a set of services.

When we assign the virtual machine containing a new service instance to a particular resource pool with certain shares of the overall resources, we are using a service-based approach to doing so. With the newer lightweight OS virtual machines, such as those using BEA LiquidVM, it becomes feasible to provision new instances of services as virtual machines of their own. New uses of these lower level services will arise over time, such as the automatic cloning of a service – or the movement of a service from one location to another.

This treatment of the computing infrastructure as a service provider means that we are now applying the modularity and composability concepts that apply in SOA to the infrastructure at the virtual machine level. In this way, management of the virtualized infrastructure is opened up so that other parties, such as enterprise management system vendors, can participate. Enterprise management tools have the low-level details of traffic levels at the network layer in a multi-machine system. This factor may influence the placement of a new service on a particular host in the system, for example.

**Applying Shares and Resource Pools to SOA Services**

In the virtualization infrastructure (VMware Infrastructure 3) from VMware today, all the computing resources of the pool of physical ESX server hosts can be joined together into a cluster and then separated and re-assigned into individual resource pools. This operation can be done with CPU capacity, memory or even with disk bandwidth. Resource pools allow the placement of limits on resource consumption by a pool’s participants. More resources (memory, CPU) can be given to one pool to give it better performance characteristics than its competitors.
VMware DRS technology allows virtual machines to participate in using the resources of those pools to which they belong – and to compete for them based on their own shares allocation. Shares assigned to a particular resource pool determine how often that pool will win the competition against its peers for resources when those resources are in short supply. System management personnel determine the shares allocation for each pool.

Extending this concept further, a web service that lives in a virtual machine of its own can be placed into a resource pool and assigned a number of shares. One service per virtual machine becomes feasible now with the lower footprint virtual machine technology seen in BEA’s WebLogic Server Virtual Edition. The resource sharing strategy for services can be implemented by allocating resource pools to a type of web service, for example a “mission critical” pool of services, a “non-critical” pool of services and a “commodity” pool of services, all with different share allocations. This allocation produces a balancing effect across SOA services using DRS, such that when competition arises for the resources in the whole collection, the resource pool with the most shares wins. The service resource pool with the highest number of shares is favored by the DRS algorithms for better access to the contested resource. This feature gives more power to certain services and removes it from others as the system is tuned to suit the load. This functionality is at the core of a “service-oriented infrastructure,” which is described next.

**Defining Service Oriented Infrastructure (SOI)**

“An SOI is a shared pool of infrastructure resources that can be dynamically manipulated to align with application requirements” (IDC).

SOI applies the SOA concepts of service provider and service consumer to the computing, networking and storage infrastructure themselves. When an application (or the management tool that is monitoring it) needs more CPU power or I/O bandwidth, or an even higher level service than those two, it makes a request in the SOA style (using web services APIs) to the SOI management system, which is the VirtualCenter management server in this case. That request is encapsulated in a SOAP request between the two management systems. Based on the available supply in the relevant resource pool, DRS will allocate more power to the requester or deny it. If the result is successful, then the required CPU capacity (or another infrastructure service) can be used by the application-level service. Otherwise the requester will need to be re-scheduled for a later time when the necessary resources become available. The interesting part of all this activity is the synchronization of the needs at the business service, or true SOA level, with the availability of resources at the infrastructure or SOI level. The business level essentially makes a demand for more resources, and if the resource balancing is correct, it gets what it wants from the SOI layer at the lower level.

**SOA and SOI Interoperating Together**

Today IT staff makes an educated guess at how much computing and networking infrastructure to provision for a new application that they want to deploy. They take a series of application load measurements, under different conditions, do some capacity planning exercises and carry out some pre-production staging. This staging or testing is done to prove the hypothesis that, for example, a 4-CPU, 6 GB of RAM and a 100 Mb of disk setup will support the new application when it is in use. This capacity planning exercise is often a sizeable effort in itself and has an impact on release schedules at a critical time - just before the application is ready to ship.

Consider a system that takes in certain business parameters, such as the number of customer orders that need to be placed over a time period, the number of SAP transactions per second required or the number of web shopping users that must be supported. This management system is also supplied with a description of the performance characteristics of the target SOA system.
(such as the typical application resource consumption figures from the testing phase). It then automatically provisions just enough infrastructure, at the right time, to support the SOA application.

During its lifetime, the system would dynamically shift resources into the resource pool being used by the application as load increases, or shift resources away from the application’s pool as the load diminishes. The new business functionality would be deployed purely at the SOA level (without worrying about the hardware) and the SOI would react to it in an appropriate way. The systems orchestration dialog between the SOA level and the SOI level would be characterized by a request such as “To run this new SOA service X, I request infrastructure services A, B and C from the total pool under management by SOI from 6am till 11pm Pacific time and half of those resources capacity for the remainder of the time.”

Another example would be a payroll batch job that is required to run between 2 a.m. and 4 a.m. on a certain day of the month, twice a month. It requires services from the PeopleSoft personnel database, along with some CPU capacity and I/O bandwidth to 100Mb of free disk space for storing its outputs and log files along with other data. Knowing that this job is coming up, the SOA management system requests the SOI management layer to pre-provision enough computing and network capacity to get the job done, monitor the payroll run’s progress while it is executing, and then de-commission the configuration of infrastructure services that were allocated once the job is done. Those resources would return to the free pool for consumption by another, perhaps unrelated, job.

In these examples, underlying infrastructure (virtual machines, resource pools, clusters) is treated as a set of services that can be called on, just like any other service. The job to be deployed or completed is submitted to the SOA management infrastructure and it, along with the SOI, does the job of allocating the necessary resources. This kind of functionality is one of the main goals of moving to an SOA and further to a Service-Oriented Infrastructure as its supporting mechanism.

To achieve such automation, there must be a free pool of infrastructure services from which to draw, and a coordinator to manage them, as well as a mapping from the higher business application levels down to the infrastructure services, based on application consumption. This new mapping concept synchronizes the SOA and SOI together by making use of virtual infrastructure platforms to fulfill the need expressed at the SOA level. The DRS functionality and the rapid provisioning features described above are necessary components towards making this work.

For SOA deployments that are in production today, the key step towards a combined SOA and SOI approach is the use of resource pools for services and the semi-automated or fully automated management of them. VMware DRS, along with the use of rapid service provisioning (for dynamic data center management) moves the virtualized infrastructure towards the SOI model. SOA and virtualization technologies are taking us in this direction; for those starting right now on this work, or who are already somewhere into it, here are some practical points of advice as initial steps on the journey.

**Key Questions to Ask When Embarking on SOA and Virtualization**

The goal of this paper is to describe the technical links between SOA and virtualization and help you to think about your virtualization design as you go about implementing your SOA and vice versa – each has an effect on the other. To combine SOA with virtualization properly, it is wise to conduct an early design-phase conversation with the designers and implementers of SOA services,
the enterprise architects, and the virtualization deployment team on the overlaps among their points of interest.

Here are some of the key questions to discuss at the enterprise architecture level as you go forward with these new technologies. These questions are topics that you should also discuss with your SOA and virtualization technology providers, so that you can gain a common understanding of how the two movements can complement your IT consolidation and modernization work.

1. How will the SOA artifacts, such as web services, be located within virtual machines? Will they be co-located or separated? Co-location of services can be a better approach for those services that frequently communicate with each other. Separation of services into different containers or virtual machines can be better for two services that frequently compete for one type of resource. These configurations should be tested as early as possible in the lifecycle.

2. Are there one or more services offered in a single virtual machine?

3. How do we replicate web services and load balance across the set of identical services?

4. How do we govern the quality and responsiveness of web/SOA services?

5. What are the infrastructure components that are needed for implementing SOA (registry of services, management policies, tools, etc)?

6. Can these infrastructure components also be virtualized?

7. How much automation do we need in provisioning new services and infrastructure?

8. Can we provide suitable measurement of new service requirements, such that the infrastructure changes it will need, if any, are predictable?

9. Is there any reason why this new service may not be virtualized?

10. Can the virtualized services be dynamically provisioned based on SLAs?

11. Can the SOA management tools use the Virtual Infrastructure APIs in a SOA-style to control those resources?

12. Can resource pools for different layers of services be created that reflect their relative importance to the business applications?

**How to Get Started on SOA, SOI and Virtualization**

SOA and virtualization technologies are linked together in several ways. Ignoring one while working on the other is not recommended and the following roadmap through the adoption process for SOA and virtualization will help you to navigate them and gain the most benefit from these technologies. The recommended steps for getting started are as follows:

1. Begin designing SOA-based systems with a view to re-using business services or, at a lower level, re-using utility services and making business processes more flexible. It is a good idea to start by categorizing the key services that will be needed within each of these communities of services. This review can be driven from the standpoint of having business applications or processes as the consumers of the new services. How many of the business applications in the company make use of a customer identity checking service, for example?

2. Understand the SOA service interactions at all levels – map these interactions out in the design phase.

3. Set in place a management structure and management policies for your SOA – consider purchasing the right tools for this phase. The questions to ask here pertain to the conditions that must be met by a candidate service in order to participate in the SOA ecosystem.

4. Design the infrastructure that will support the SOA at the same time as you design the services that will reside in it. Use a service-oriented approach to the infrastructure layout,
as well as to the application. Make use of DRS and structure your services into resource pools and clusters early on in your deployment design. This is the key to a successful SOI.

5. Give careful consideration to the placement, provisioning, load balancing and failover of services in this phase.

6. Give your SOA developers the greatest flexibility by allowing them to rapidly provision SOA infrastructure components. Developer lab management tools such as VMware Lab Manager present the right types of interfaces for this purpose. Integrate this rapid provisioning toolkit with your testing tools for your SOA services – there are SOAP APIs for doing this.

7. Use the SOA hosting platforms from BEA that are proven and supported on virtual infrastructure today. Many of your SOA services will go live on virtual machines in production – use the same environments for your development and testing phases.

8. Test the current reduced footprint virtualized application platform (such as BEA WebLogic Server Virtual Edition) for rapid provisioning of your SOA services. This platform, along with its management tool, the Liquid Operations Console, will be essential for the automated SLA-driven deployment phase of your SOA that will become necessary for success as your SOA grows in use. This will also allow your SOI to flex its power as the SOA level demands more in the future.

**Conclusion**

Adopting a service-oriented infrastructure (SOI) based on virtualization technology is a key step towards deploying a successful application-level SOA, both in the service development and deployment phases. SOI based on virtualization provides key mechanisms (such as distributed resource scheduling, clustering, resource pools, rapid service provisioning and de-provisioning) for first measuring and then responding to the behavior of a virtual system.

SOA deployment projects can now make use of an application or service container that is fully integrated with a virtual platform (such as the WebLogic Server, Virtual Edition integrated with VMware Infrastructure 3). The two main tiers described in this paper, SOA and SOI, interoperate in a service-oriented way through their respective management tools (Liquid Operations Console and VirtualCenter server) when capacity, load balancing and optimal service placement needs arise. SOA/SOI management decisions can be policy-based and depend on the computing resource needs of the services making up the SOA and their consuming applications. This paper explores some of the options available today for achieving this level of flexibility, using SOA and SOI technologies on virtualized platforms together, and gives the reader a roadmap for adoption of them.