

Develop and Deploy Multi-Tenant Web-delivered Solutions using IBM middleware: Part 2: Approaches for enabling multi-tenancy

Skill Level: Intermediate

[Carl Osipov \(osipov@us.ibm.com\)](mailto:osipov@us.ibm.com)

Software Architect

IBM

[Germán Goldszmidt \(gsg@us.ibm.com\)](mailto:gsg@us.ibm.com)

IBM Distinguished Engineer

IBM

[Mary Taylor \(marytaylor@us.ibm.com\)](mailto:marytaylor@us.ibm.com)

Consulting IT Specialist

IBM

[Indrajit Poddar \(ipoddar@us.ibm.com\)](mailto:ipoddar@us.ibm.com)

Software Architect

IBM

20 May 2009

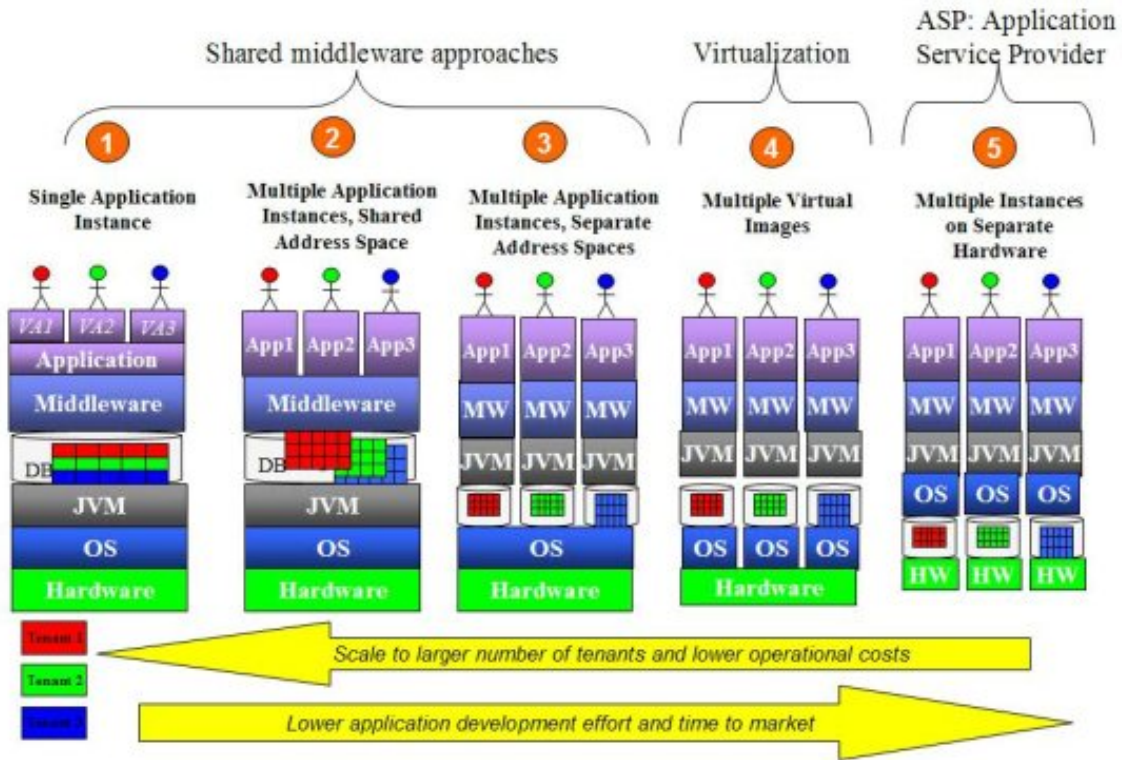
Part 1 of this series describes what multi-tenancy is and presents several technical challenges for building and deploying multi-tenant web-delivered solutions. In this article, we identify five representative approaches for enabling multi-tenancy in web-delivered solutions (also known as software-as-a-service) and compare their costs and benefits.

Introduction

Service providers contemplating web-delivery of existing or new multi-tenant services are often faced with a number of design choices. We have identified five major approaches as shown in Figure 1. Approach 1 is sharing a single application instance i.e. the same servers, middleware and applications, amongst all tenants.

Approach 5 is having tenant specific application instances run on separate servers (which is what many Application Service Providers or ASPs do currently). In between, we have identified at least three other major approaches which require varying degrees of resource sharing and development complexity. Each approach provides different benefits due to scale and operational efficiencies and costs due to development complexities and time-to-market.

Figure 1. Five major approaches for enabling multi-tenancy



In this article, there are a number of roles we will use to describe the different approaches for multi-tenancy:

1. Service Provider: responsible for providing the services/solution in the web-delivery model
2. Service Developer: developer of the services/solution being offered
3. Tenant: Customers of the service provider
4. End-users: A tenant may have one or more end-users of the service/solution

The five major approaches that service providers can use are:

1. Shared middleware with a single application instance
2. Shared middleware with multiple application instances and shared address spaces
3. Shared middleware with multiple application instances and separate address spaces
4. Virtualization with tenant specific virtual images
 - a). Virtualization with a Mediation layer
5. ASP model with multiple instances on separate servers

We will describe each of these five approaches in more detail in the following sections. When determining which of these approaches to adopt, it helps to understand the costs and benefits associated with each approach. We provide such a cost/benefit analysis in a following section.

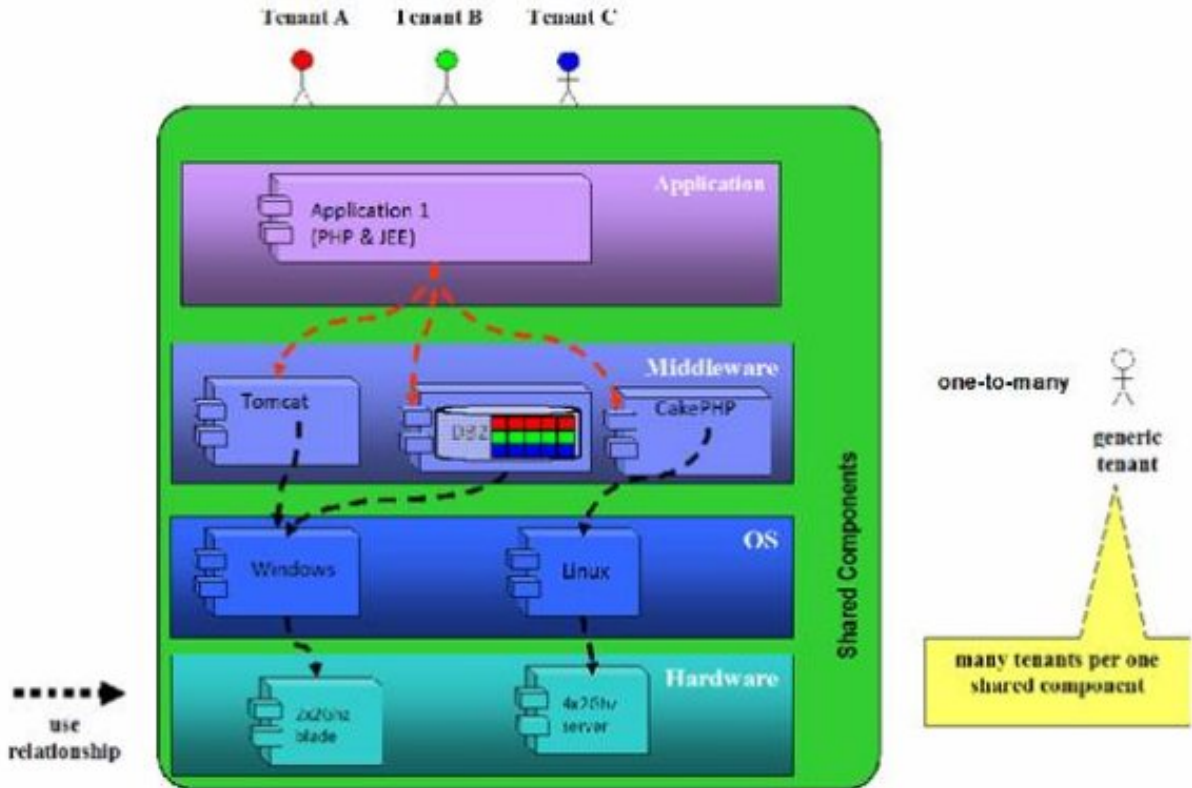
Shared middleware approaches

Approaches 1, 2 and 3 represent different degrees of sharing of the middleware and the application components between multiple tenants. In the following subsections, we show examples of each approach.

Approach 1: Shared middleware with a single application instance

All tenants share the operating system, servers, and a single instance of the middleware and application. This is accomplished by parameterizing a single instance of an application with a tenant identification parameter. For example, if the application has web service interfaces and implementations, then a tenant ID parameter is added to the operations and data objects in the interface. If the application uses database tables, then a new column designating the tenant ID is added to each table in the database. In this model, there are some configuration elements that are unique to each tenant. For example, a virtual portal is configured for each tenant with a different look-and-feel and unique database schema elements are offered to each tenant.

Figure 2. Topology example for Approach 1, sharing a single instance of the application and middleware between multiple tenants



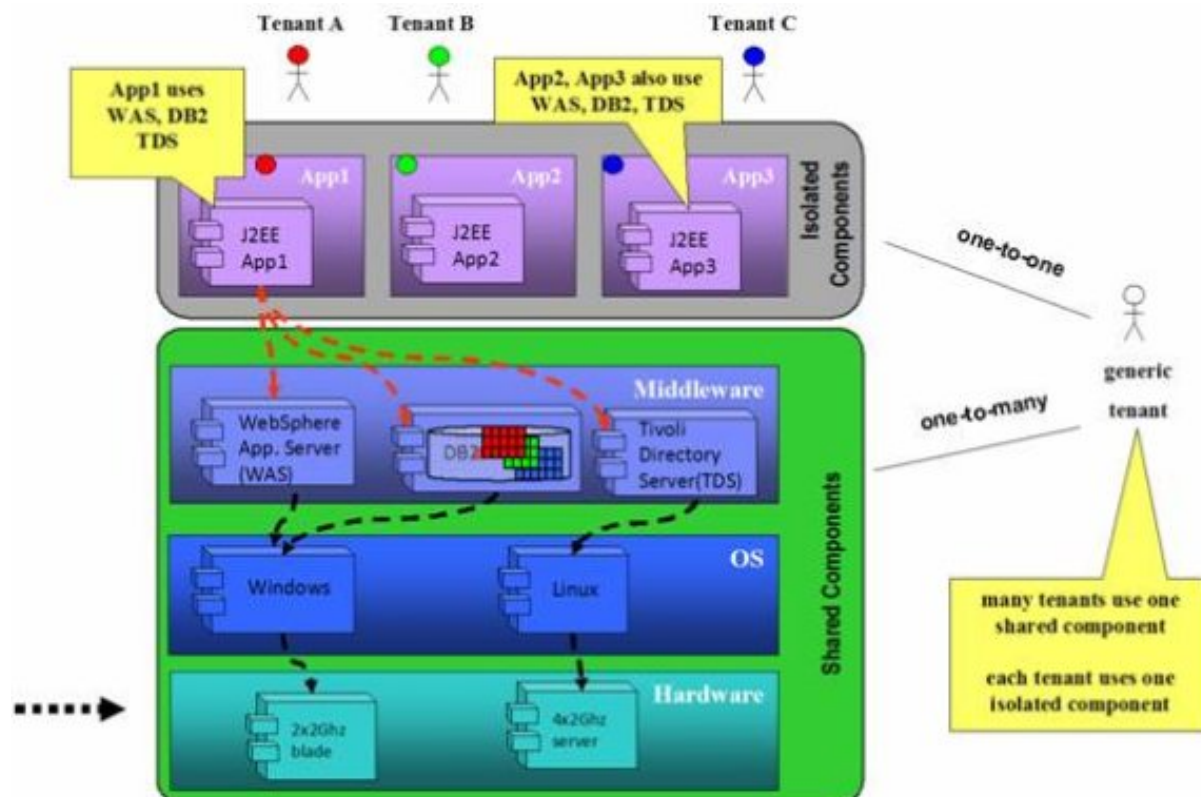
In Figure 2 we show a sample topology using this approach. There are three tenants: A, B and C, which share the same code for Application 1. Application 1 uses Tomcat and DB2 running on Windows on a Blade server, and Apache HTTP server running on Linux on another physical server. The middleware, OS and Servers are shared amongst all of the tenants. Sample code for an example multi-tenant application using this approach can be downloaded from [“Building Web delivered SaaS applications on open source and entry level IBM middleware”](#). When a tenant is on-boarded, tenant specific configurations (e.g., CSS files for each tenant) are created for that tenant, but there is only one instance of the application which is shared between all the tenants. Using this approach, the application must be designed to provide the capability to isolate each tenant’s data and customizations from the other tenants. Parts 3 and 4 of this series will present important architectural considerations for using this approach for enabling multi-tenancy with IBM WebSphere Application Server.

Approach 2: Shared middleware with multiple application instances in shared address space

Tenants use different instances of the application deployed on a single instance of the middleware which share a single operating system process (address space). Tenants share the operating system and servers. Figure 3 shows an example application shared by tenants A, B, and C using this approach.

Figure 3. Topology example for approach 2, running different instances of the

application for different tenants in a single instance of the middleware



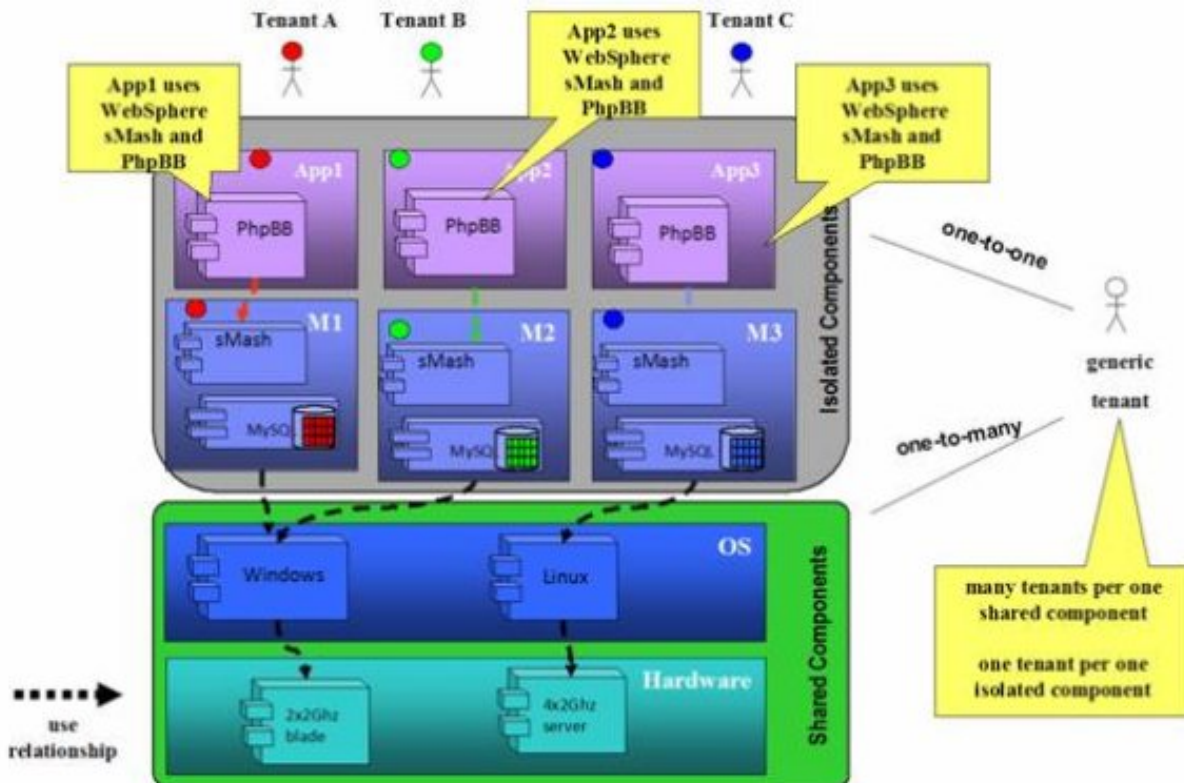
When a new tenant is on-boarded a separate copy of the application is created and given a name that includes a tenant identifier and deployed to a shared instance of the application server. For example, the ear file for Application App (App.ear) would be copied and named App1.ear and App2.ear respectively. Similarly, in the database tier, an application table called App_table would be copied into two tables App1_table and App2_table for tenants A and B respectively. Tenant specific customizations (e.g. CSS files and table schemas) are added to the tenant specific copy of the application and table. This model requires that tenant isolation be maintained at the middleware layer.

Approach 3: Shared middleware with multiple application instances in separate address space

Tenants use different instances of the application deployed on different instances of the middleware. Tenants share the operating system and servers. Since the middleware instance is different, each tenant is allocated its own set of operating system processes (i.e. address space). Thus this model requires that tenant isolation be maintained at the operating system layer. This approach supports fewer tenants than approaches 1 and 2 on the same physical server. This approach also offers more isolation amongst tenants within the three shared middleware approaches. But isolation concerns still exist at the OS and servers layers, e.g., it is possible for one tenant's users to consume all the CPU or memory in the physical

server.

Figure 4. Topology example for approach 3, running different instances of the application for different tenants in different instances of the middleware



In the example in Figure 4, we see that each tenant (A, B and C) is running its own physical copy of the same application (App1, App2 and App3 respectively), in its own physical copy of the middleware (M1, M2 and M3 respectively). Tenants A and B are running their applications on Windows, while Tenant C is running their application on Linux. Of the three shared middleware approaches, this approach requires the least amount of changes to an existing application, possibly enabling faster deployment. In an upcoming demo in the SaaS Blueprints series will show an example of this approach using WebSphere Smash and MySQL as the middleware running a multi-tenant phpBB bulletin board application.

Virtualization approaches

Virtualization technology is used to run multiple operating system partitions with dedicated application and middleware instances for each tenant on shared servers.

Approach 4: Virtualization with multiple VM images

Tenants use different virtual images with different instances of application, middleware and operating system while sharing physical servers. In recent years,

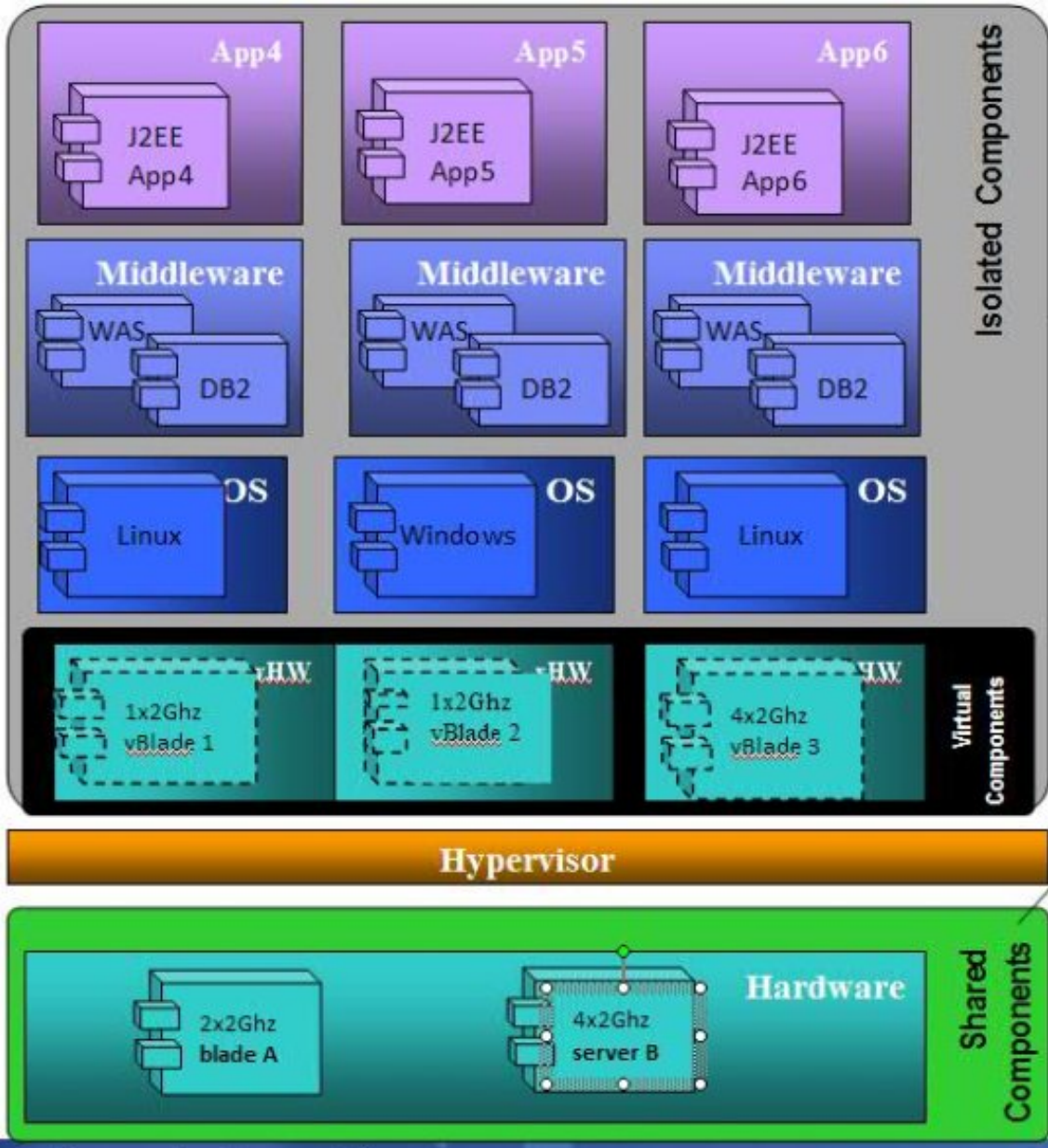
server virtualization has gained widespread adoption in x86 based servers and is rapidly becoming a low cost, commodity technology.

In contrast to the shared middleware approach, server virtualization does not require significant code development to enable multi-tenancy. Once server virtualization is installed on a physical server (host), a service provider provisions each additional tenant by instantiating a virtual server (guest) to host tenant specific software, including middleware and applications.

One of the technical challenges in multi-tenancy is provisioning new tenants. For provisioning new tenants service providers have to work with potentially lengthy and complex installation and configuration steps. Virtual appliances (e.g. VMWare Virtual Appliance for WebSphere Application Server Network Deployment V7.0 Open Beta) with pre-configured tenant specific operating system, middleware can help to address this challenge.

Figure 5 shows an example of this approach where a native hypervisor such as VMWare ESX™ or Xen is installed on physical servers. In this example, a 2 CPU, 2 Ghz physical blade server A in the green box at the bottom is partitioned into 2 single CPU 2 Ghz virtual blade servers: vBlade 1 and vBlade 2 in the black box. Virtual blade server: vBlade 3 is composed of a single 4 CPU 2Ghz server B. The virtual servers also share other resources of the physical server, such as memory, disk space and the network connection. Applications: App4, App5 for two different tenants are deployed in into vBlade 1 and vBlade 2 and App6 is deployed into vBlade 3. Note, that tenants are allowed to have different operating systems.

Figure 5. Native hypervisor based server virtualization for multi-tenancy



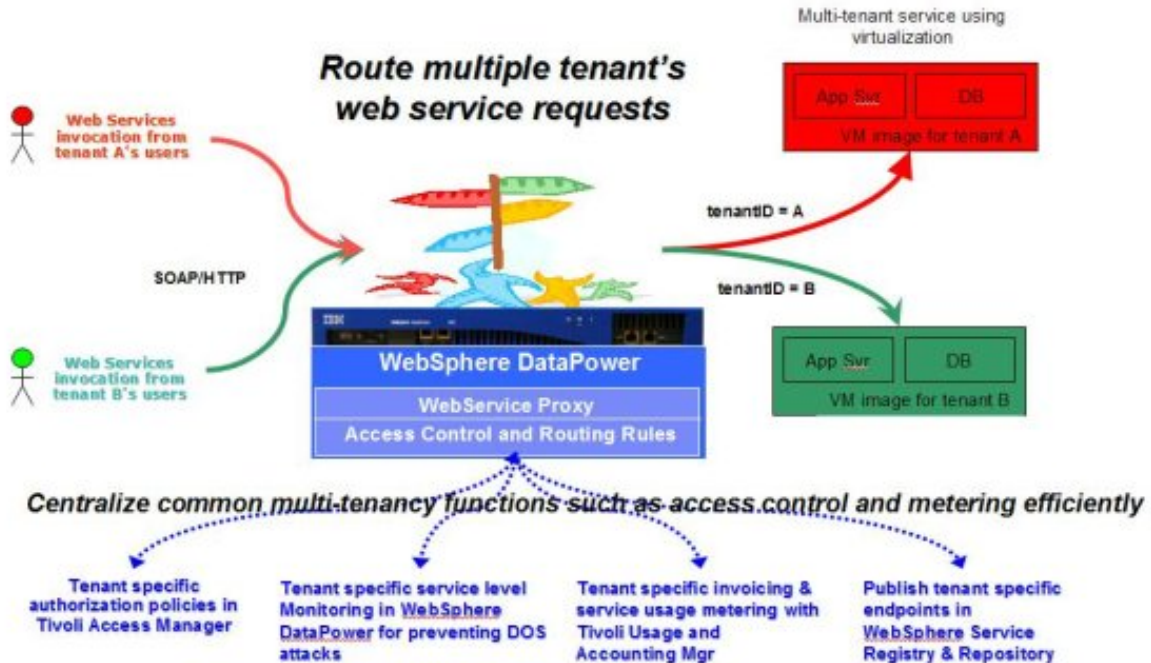
Approach 4a: Approach 4a: Virtualization with a Mediation layer

A service provider centralizes common multi-tenancy functions such as routing, access control and metering in a mediation proxy layer. A mediation layer can be used in conjunction with a virtualization approach by integrating it with tenant specific service instances running in separate virtual image partitions. The mediation proxy layer sits between the end user of the application and the services within the application and dynamically binds or routes service requests from a tenant’s users to tenant specific instances of the services as shown in Figure 6.

Figure 6 also describes an example of the mediation approach where the service

provider uses a WebSphere DataPower SOA appliance (WDP) to implement a mediation proxy layer. In this example, routing rules are configured in a WDP web service proxy to route requests from users for two different tenants (A and B) to the respective tenant-specific instance of an application. In addition, multi-tenancy functions such as metering, access control and auditing are added by integrating other middleware components such as Tivoli Access Manager and Tivoli Usage and Accounting Manager.

Figure 6. Example of a mediation approach for multi-tenancy using WebSphere DataPower SOA appliances



In parts 5, 6, 7 and 8 of this series, we will present three alternative options for implementing this approach using the following alternative combination of products:

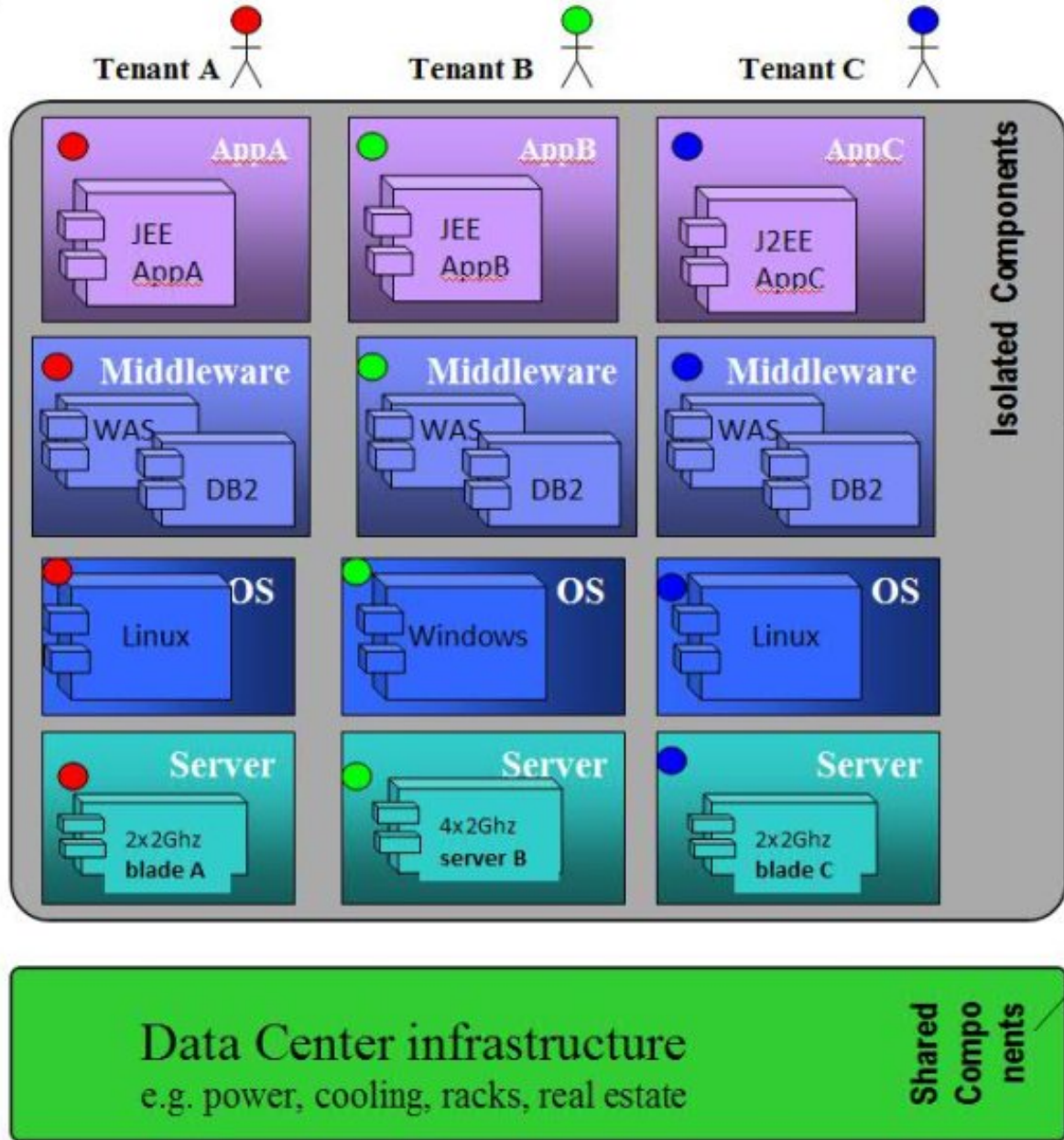
- WebSphere Business Services Fabric
- WebSphere DataPower SOA appliances with Tivoli Access Manager
- WebSphere Enterprise Services Bus with WebSphere Service Registry and Repository

Approach 5: ASP model with multiple instances on separate servers

Tenants share only the infrastructure of the data center e.g. power, cooling but use different instances of the application, middleware, operating system and servers. Figure 7 shows an example where tenants A, B and C have three different application instances AppA, AppB and AppC running on tenant specific middleware instances, operating system instances and physical servers. This approach is most

suitable for workloads and scenarios where a high degree of isolation and customization is required for different tenants.

Figure 7. Example of an ASP model for enabling multi-tenancy with multiple instances on separate servers



Cost-benefit comparison of the multi-tenancy approaches

The cost benefit analysis presented in this section is considered from the perspective of the service provider and the service developer.

Approach 5 (i.e. the ASP approach) is the most expensive for the service provider when scaling to large number of tenants. In the other four approaches there is a higher degree of sharing of resources which leads to improved cost efficiencies and economies of scale as shown in the two yellow arrows in Figure 1. The following types of tenant specific costs can be incurred:

- **Physical Server:** Each tenant requires dedicated servers, disk space and ancillary equipment.
- **Administration:** servers, and in particular software, requires continuous upkeep such as installation of security patches, upgrade of the OS and middleware, user account management and so on. These tasks have to be conducted separately on each tenant’s infrastructure.
- **Operation:** running a data center requires electricity, cooling and many real estate related expenses which grow significantly when more tenants are added.

However, this approach is the least expensive approach for a service developer because no application changes are necessary to enable multi-tenancy. Also this approach offers the highest degree of isolation between and customizability for tenant specific instances.

Approaches 1 through 3 (i.e. the shared middleware approaches) have the lowest operational cost of the three sharing models because of increased sharing of resources (servers, middleware and OS) leading to a fewer number of components to manage. This approach also offers efficiencies of scale because of the highest density of tenants hosted on a single physical server. Approach 1 (i.e. the single application instance) could be the most expensive option for the service developer because it may require significant redesign of existing applications with high development time and costs. In addition, it requires knowledge of advanced application server features and techniques and skilled developers.

Approach 4 (i.e. multiple virtual images) is less expensive than the shared middleware approaches (approaches 1-3) for the service developer because little or no application changes are required. This allows for faster, more expedient time-to-market. Adding a mediation layer (approach 4a) in the virtualization approach also makes it less complex and time consuming to integrate other multi-tenancy enabling functions such as access control and metering.

Table 1 provides a summary of the cost benefit comparison between the shared middleware approaches (1-3) and the virtualization approach (4 and 4a).

Table 1. A cost-benefit comparison of three major multi-tenancy approaches

	Shared Middleware approaches	Virtualization and Virtualization with Mediation
Benefits		

- Ability to *SCALE* to additional tenants quickly
- *COST EFFECTIVE* since the infrastructure is shared by all tenants
- *LESS OVERHEAD* than a virtualized or mediated approach
- No application *RE-DESIGN* is necessary
- Some minor integration code changes may be necessary to efficiently enable *ADDITIONAL COMMON MULTI-TENANCY FEATURES* such as:
 - Access control
 - Metering
 - *HETEROGENEITY* and transformation in protocol & interfaces in tenant specific applications
- *FASTER* time to market and *LOWER* up front cost
- Isolation for better *SECURITY* and *AVAILABILITY* for tenants
- Higher degree of HW and OS *CUSTOMIZATION* is provided than in a shared environment
 - Supports *HETEROGENEITY* (OS, HW) for tenant applications
- Simpler to *RELOCATE* the application to another platform provider
- Simpler *BACKUP* and *DISASTER RECOVERY* for each tenant

Costs

- Requires application *RE-DESIGN* or *CODE CHANGES* of existing single tenant code in approach 1
 - *TIME-TO-MARKET* impact of re-architecting applications for multi-tenancy
 - *HIGHER UPFRONT COST* when code changes are necessary
 - *SKILLED PROGRAMMERS REQUIRED* to implement approach 1
 - *ADDED COMPLEXITY* in providing features such as backup and restore customized for each tenant
- LOWER SCALABILITY* in number of tenants per server
GREATER PERFORMANCE OVERHEADS HIGHER OPERATIONAL COSTS and *MANAGEMENT COSTS* because of multiple instances of virtual machine images
HIGHER DEPLOYMENT COSTS when deploying to a mediation layer

Conclusion

The journey from a traditional application service provider model to a web delivery model has many entry points/approaches with differing costs and benefits for the service provider and the service developer. We have presented five such entry points which differ in the degree of resource sharing and development complexity. The goals of this journey are to keep improving cost efficiencies and keep lowering total cost of ownership. This article offers various pros and cons to help select the right entry point and prepare architectures suitable for incremental improvements towards these goals.

Acknowledgements

The authors would like to acknowledge the contributions of Ajay Mohindra and

Suresh N. Chari from IBM Research to Figure 1 describing the major multi-tenancy approaches.

Resources

- Develop and deploy multitenant Web-delivered solutions using IBM middleware [Part 1: Challenges and architectural patterns](#).
- IBM [SaaS Blueprint demo](#), series.
- Sample code for an example multi-tenant application using this approach can be downloaded from "[Building Web delivered SaaS applications on open source and entry level IBM middleware](#)".
- Check out the [WebSphere Application Server Community Edition](#).
- More about [WebSphere sMash](#).
- View the [DB2 V9](#) website.
- More about [phpBB](#).
- Check out [WebSphere DataPower SOA appliances](#).
- View the [WebSphere Virtual Enterprise](#) page.
- VMWare Virtual Appliance for [WebSphere Application Server Network Deployment V7.0 Open Beta](#).

About the authors

Carl Osipov

Carl Osipov is an experienced software architect with the Strategy and Technology organization in IBM Software Group. His skills are in the area of distributed computing, speech application development, and computational natural language understanding. He has published and presented on Service-Oriented Architecture and conversational dialog management to peers in the industry and academia. His current focus is on design of reuse techniques for composite business services.

Germán Goldszmidt

Dr. Germán Goldszmidt is an IBM Distinguished Engineer working in Software Group, Technical Strategy. He is responsible for the SWG Advanced Technology program, including incubation projects and the cross Software Group Research Joint Program. He has lead multiple projects in the area of Composite Business Services and Software as a Service. Before 2003 he was a Research Staff Member and Manager at the IBM Research T.J. Watson labs. He led the team that developed the

first compute provisioning e-utility that efficiently handles varying workloads and infrastructure changes by autonomically adjusting resource allocations. Earlier, he designed and developed Network Dispatcher, a high performance load balancer which provides reliability and scalability for multiple IBM products and deployments. He has published more than 50 papers in refereed journals and conference proceedings, and holds 15 patents.

Mary Taylor

Mary works in the SWG Technical Strategy and Incubation team and administers the SWG Incubation program. She is co-lead of a SaaS Community of Practice and is a contributor to the SaaS Blueprints. She is currently focused on the positioning of IBM middleware in the financial markets front office.

Indrajit Poddar

Indrajit Poddar (IP) is a member of the Strategy, Technology, Architecture, and Incubation team at IBM Software Group Strategy, where he leads several integration PoCs for building composite business services delivered in the Software-as-a-Service (SaaS) model.