Building multi-tenancy applications with IBM middleware
Agenda

Introduction: SaaS and Cloud Computing -
Benefits and challenges of SaaS
Multi-tenancy approaches
SaaS Technical challenges
Addressing technical challenges with IBM middleware
Provisioning, monitoring and metering SaaS applications
Conclusion
What is Software as a Service?

Software as a Service, SaaS is a software delivery method that provides access to software and its functions remotely as a Web-based service. SaaS allows organizations to access business functionality at a cost typically less than paying for licensed applications since SaaS pricing is based on a monthly fee. Also, because the software is hosted remotely, users don't need to invest in additional hardware. SaaS removes the need for organizations to handle the installation, set-up and often daily upkeep and maintenance.
What is Cloud Computing?

A user experience and a business model
- Cloud computing is an emerging style of IT delivery in which applications, data, and IT resources are **rapidly provisioned** and provided as **standardized offerings** to users over the web in a **flexible pricing model**.

An infrastructure management and services delivery methodology
- Cloud computing is a way of **managing** large numbers of highly **virtualized resources** such that, from a management perspective, they resemble a single large resource. This can then be used to deliver services with **elastic scaling**.
Layers of Cloud Computing

**Software as a Service**
- Collaboration
- Business Processes
- Industry Applications
- CRM/ERP/HR

**Platform as a Service**
- Middleware
- Web 2.0 Application Runtime
- Java Runtime
- Development Tooling
- Database

**Infrastructure as a Service**
- Servers
- Networking
- Data Center Fabric
- Storage

Shared virtualized, dynamic provisioning
Positioning SaaS and Cloud Computing

- Fixed, dedicated resources
- Elastic Internet resources
- Multi-tenant application
  - Tenant 1
  - Tenant 2
  - Tenant n
Agenda

Introduction: SaaS and Cloud Computing -

Benefits and challenges of SaaS

Multi-tenancy approaches

SaaS Technical challenges

Addressing technical challenges with IBM middleware

Provisioning, monitoring and metering SaaS applications

Conclusion
It is not easy being an Independent Software Vendor

- You build and sell software
- Your customers purchase and operate
- One time charge plus recurring maintenance and support
- CAPEX for infrastructure leaves less money for your software
- Market reach limited by the capacity of direct sales and channel. Building a channel is very expensive.
- Long sales cycles and even longer deployment timeframes
- Multiple versions of software in the field means multiple code bases and maintenance streams
Enterprises are now demanding cloud based software because of...

Easy accessibility by remote users....

....Deferral/avoidance of hardware purchases

Decrease in time/cost for software maintenance....
Software as a Service Can Help

SaaS Vendor

- You build and **RUN** your software; sell as service
- **Predictable revenue stream**: recurring (e.g. monthly) charges
- **Capture more of the customer spend**: no need for customers to spend on hardware/software/personnel
- **Shorter sales cycles**: dealing directly with line of business
- **Global market reach** leverages self-service
- **Lower cost of delivery**:
  - Lower resource per client due to economies of scale
  - Pay only for what you use allows alignment of IT expense with revenue
- **Drastically reduce ongoing support costs**: single version of software in the field
- **Gain competitive advantage** by rapidly evolving feature set:
  - Single version of the software to evolve
  - Better insight into actual use of the features/functions
- **Rapidly deliver and have customers adopt new features**
SaaS is a great addition to your current business model

- Most ISVs are looking to SaaS to supplement and not replace current business model
- Combined SaaS + ISV approach:
  - Mitigates concern with recurring charges and higher customer churn
  - Still allows for high level of customization for high-end customers
  - Provides flexibility for different customer sets
Four things to remember when adopting the SaaS model

- **There is no “Cloud,” but rather many “Clouds.”**
  - For maximum exposure and market reach, ISVs will do well to select a cloud partner with an accepted platform that supports multiple public and private environments.

- **It’s not just about the platform.**
  - Management tools, go-to-market strategy, and pricing are all part of the overall SaaS package, and these crucial business processes can be facilitated or hampered by the choice of a cloud partner.

- **Cloud adoption is a long-term investment.**
  - ISVs must develop a comprehensive cloud strategy that anticipates the future of software consumption and delivery.

- **Cloud adoption must contribute materially to business growth**
  - Cloud adoption should help ISVs achieve their business goals, by allowing them to enter new markets or attract more customers. The cloud should add value to software applications, by enabling greater functionality and integration with other software.
Agenda

Introduction: SaaS and Cloud Computing -
Benefits and challenges of SaaS

**Multi-tenancy approaches**

SaaS Technical challenges
Addressing technical challenges with IBM middleware
Provisioning, monitoring and metering SaaS applications

Conclusion
What is Multi-tenancy?

- The ability to deliver software to multiple client organizations (or tenants) from a single, shared instance of the software
  - Customizations made for one tenant are contained within metadata
    - e.g. Cascading Style Sheets for UI branding
  - Each tenant runs the same application code

- Consumer applications are usually excluded from discussions of multi-tenancy
### Types of multi-tenancy

<table>
<thead>
<tr>
<th>I</th>
<th>Physical level multi-tenancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant</td>
<td>Tenant</td>
</tr>
<tr>
<td>App</td>
<td>App</td>
</tr>
<tr>
<td>Platform</td>
<td>Platform</td>
</tr>
<tr>
<td>OS</td>
<td>OS</td>
</tr>
<tr>
<td>Infras</td>
<td>Infras</td>
</tr>
<tr>
<td>Data center floor</td>
<td>Data center floor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>II</th>
<th>Hypervisor level multi-tenancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant</td>
<td>Tenant</td>
</tr>
<tr>
<td>App</td>
<td>App</td>
</tr>
<tr>
<td>Platform</td>
<td>Platform</td>
</tr>
<tr>
<td>OS</td>
<td>OS</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>Data center floor</td>
<td>Data center floor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>III</th>
<th>Operating System level multi-tenancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant</td>
<td>Tenant</td>
</tr>
<tr>
<td>App</td>
<td>App</td>
</tr>
<tr>
<td>Platform</td>
<td>Platform</td>
</tr>
<tr>
<td>Operating System</td>
<td>Operating System</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>Data center floor</td>
<td>Data center floor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IV</th>
<th>Platform level multi-tenancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant</td>
<td>Tenant</td>
</tr>
<tr>
<td>App</td>
<td>App</td>
</tr>
<tr>
<td>Platform</td>
<td>Platform</td>
</tr>
<tr>
<td>Operating System</td>
<td>Operating System</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>Data center floor</td>
<td>Data center floor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
<th>Application level multi-tenancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tenant</td>
<td>Tenant</td>
</tr>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td></td>
</tr>
<tr>
<td>Data center floor</td>
<td></td>
</tr>
</tbody>
</table>
Approach selection is a factor of up-front cost vs. scalability & administration costs
Type I - Physical Level Multi-tenancy
Application Service Providers host each tenant’s application in dedicated hardware, middleware and operating system.

- Presentation Layer
- Business Layer
- Data Layer
- HW & OS Layer

Tenant 1’s users: Presentation Layer
Tenant 2’s users: Presentation Layer
Tenant 3’s users: Presentation Layer
Type I - Physical Level multi-tenancy

Characteristics

- Better performance than virtualization because using native hardware
- Runs non multi-tenant applications w/o changes
- Poorest scalability (number of tenants per server = 1)
- Highest operational costs
- Provisioning cannot be fully automated

When to use this model

- Generally hard to justify use of this model except if customers require dedicated servers because of regulation / standards
Type II - Hypervisor Level multi-tenancy

Tenant 1’s users

Tenant 2’s users

Tenant 3’s users

Pres. Layer

Business Layer

Data Layer

App

Virtual-OS

OS Virtualization/Cloud

OS

Server Hardware
Type II - Hypervisor Level multi-tenancy

Characteristics

- Runs non multi-tenant applications w/o changes
- Drop off in performance from Type I because applications have 2 Operating Systems to traverse to get to hardware
- Better scalability than physical level (number of tenants per server > 1)
- Operational costs on par with Type I minus TCO of dedicated hardware
- Provisioning can be fully automated

When to use this model

- Application is not multi-tenancy aware and need to GTM quickly
- Unknown demand for SaaS model and/or lack of resources to modify application
Type III – Operating System Level multi-tenancy
Type III – Operating System Level multi-tenancy

Characteristics

- Runs non multi-tenant applications w/o changes
- Can improve software license costs over Types I and II if middleware supports multiple instances (e.g. database, application server) with a single license
- Smaller footprint than Types I or II
- Provisioning can be fully automated

When to use this model

• Application is not multi-tenancy aware and need to GTM quickly
• Unknown demand for SaaS model and/or lack of resources to modify application
• Middleware supports multiple instances on single OS
Type IV – Platform level multi-tenancy
Hardware, OS and Application server layers are shared, to varying degrees, across users. Multiple instances of the same application deployed. Data Layer comprises Shared Databases with either shared or separate tables.
Type IV – Platform Level multi-tenancy

Characteristics

- Cannot run non multi-tenant applications w/o changes
- Can improve software license costs over Types I, II
- Smaller footprint than Types I, II or III
- Provisioning can be fully automated

When to use this model

- New applications or resources exist to modify existing non-multi-tenant applications
- Market for type of application is extremely competitive driving prices down
- Middleware addresses technical challenges of true multi-tenant applications
Type V – Application level multi-tenancy
Single application instance shared by all tenants. A mediator determines tenant id in each request so each application request can be handled properly.

- Tenant 1’s users
- Tenant 2’s users
- Tenant 3’s users

Meditation Layer
Presentation Layer
ID=1
ID=2
ID=3
Metadata Layer
Business Layer
Directory Services
Security
Provisioning
Management Layer
Server HW and OS Layer
Server Hardware and OS Layers are shared across users

App Instance
Shared Tables
Separate Tables
Type V – Application Level multi-tenancy

Characteristics

- Cannot run non multi-tenant applications w/o changes
- Smaller footprint than Types I, II, III or IV
- Provisioning can be fully automated
- Mediator allows for advanced features like dynamic QoS adjustments

When to use this model

• New applications or resources exist to modify existing non-multi-tenant applications
• Market for type of application is extremely competitive driving prices down
• Spikes in demand and/or SLA’s require dynamic load balancing
Agenda

Introduction: SaaS and Cloud Computing -
Benefits and challenges of SaaS
Multi-tenancy approaches

SaaS Technical challenges
Addressing technical challenges with IBM middleware
Provisioning, monitoring and metering SaaS applications

Conclusion
Technical challenges for Type IV and Type V

- **Access Control**
  - Sharing application resources (e.g. database tables, web services, J2EE artifacts) so that users for one tenant only see data/services for that tenant

- **Customization**
  - **Database**
    - Additional fields for Business object for each tenant (e.g. Purchase Order)
  - **User Interface**
    - Look and feel changes via configuration only
    - Enable additional fields when displaying business objects
  - **Business Logic**
    - Different business rules for processes in the application (e.g. calculating discounts)

- **Isolation**
  - **Data**
    - A tenant should not have any access to the data belonging to any other tenant
  - **Management**
    - A tenant can only configure, monitor and manage his own instance without interfering with other instances
Agenda

Introduction : SaaS and Cloud Computing -
Benefits and challenges of SaaS
Multi-tenancy approaches
SaaS Technical challenges

Addressing technical challenges with IBM middleware
Provisioning, monitoring and metering SaaS applications
Conclusion
SaaS Enablement Blueprints

A HOWTO guide for addressing multi-tenancy challenges with IBM middleware

Sample Banking Application Using IBM Technology to solve SaaS technical challenges

For the Enterprise
- DB2 V9.x
- WebSphere Portal and Process Server
- Tivoli Security

For the SMB ISV
- DB2 Express
- WAS CE

• Running Demos
• Whitepapers and Briefs
• Enterprise Approach
  • Customize through configuration
  • Code Snippets
• SMB Approach
  • Open Source Architecture
  • Development Techniques
  • Download real code

Multi-tenant Bank Environment
- Retail Banking
- Multi-role
- Web channel

Blueprints: Multi-tenant bank environment

Multi-tenant Banking Application
- Retail Banking
- Multi-role
- Web channel

Bank Administration
- Add/Delete Bank Customers
- View Bank Customers
- Update customer profile

Bank Employee Service
- Perform Teller operations
- Approve Loans

Bank Service Provider Administration
- On-board Bank
- Manage Master Bank Administrators
- Configure with dynamic profiles
- Configure look and feel

Bank Customer
- Account transfers
- View Interest Rates
- Apply for Bank Loan
- Review Loan Status
Addressing Access Control with IBM middleware

- **WebSphere Application Server v7.0**
  - Multiple security realms supported for an application
    - Mapped to separate user repositories
  - Multiple virtual hosts supported for an application server
    - For example [www.tenant1.com](http://www.tenant1.com) [www.tenant2.com](http://www.tenant2.com)
    - Same application can be deployed to different virtual hosts
  - Role based access control based on JEE security

- **WebSphere Portal Server v6.1**
  - Can create multiple virtual portals
    - Mapped to separate user repositories
  - Role based access control based on WebSphere Member Manager

- **Tivoli Directory Server**
  - Enterprise class LDAP server with support for multi-tenant directory trees
Access Control Example

- Multi-tenant user-registry using:
  - Tivoli Directory Server and the Member Manager feature in WebSphere Portal Server
  - Virtual Portals in WebSphere Portal Server with multi-realm security
- Role based WebSphere Portal access control

User: b1a1 can only log into bank1’s virtual portal

User: b2a1 can only log into bank2’s virtual portal
Utilizing the pureXML capabilities of DB2 v9.x to enable tenant data tier customization

Multi-tenant Banking Application

Shared Environment

Traditional SQL databases

CustomerProfile – Bank 1
  - customerid: Integer
  - bankid: Integer
  - firstname: String
  - lastName: String
  - cellPhone: String
  - homePhone: String

CustomerProfile – Bank 2
  - customerid: Integer
  - bankid: Integer
  - firstname: String
  - lastname: String
  - homePhone: String

Utilizing DB2 v9 XML Capabilities

CustomerProfile
  - Shared by both banks
  - bankid: Integer
  - customerdoc: xml

Customizable for each tenant
Utilizing the pureXML capabilities of DB2 v9.x to enable tenant data tier customization (cont’d)

- Support for multiple XML schemas for a single XML column
- Each tenant can provide tenant specific schemas for XML column values
- Typically would use default schema and allow tenants to customize as needed

### Step 1: Register the schema

```
REGISTER XMLSCHEMA 'http://www.tenant1.com/customer'
FROM FILE 'file:///schemas/tenant1-customers' AS TENANT1.CUSTOMER
```

### Step 2: Validate using schema’s SQL name

```
INSERT INTO CustomerProfile(BANKID, CUSTOMERDOC) VALUES(?, XMLVALIDATE(?
ACCORDING TO XMLSCHEMA ID TENANT1.CUSTOMER))
```
Addressing UI customization with IBM middleware

- WebSphere Application Server
  - Supports Java Server Faces
    - “Smart” UI components that can automatically adapt to changes in the model can be developed
    - See the following SaaS blueprint for an example

- WebSphere Portlet Factory/WebSphere Portlet Server
  - Portlet Factory IDE allows different profiles to be defined
    - UI fields can be profile dependent
      - Only show when specific profile is enabled at runtime
      - See the following SaaS blueprint for an example
Implementing a Configurable User Interface using WebSphere Portlet Factory Dynamic Profiles

Using Dynamic Profiles, you can simplify administration of configuration parameters for multiple tenants using a single Administrator portlet
Addressing business logic customization with IBM middleware

- **WebSphere Process Server**
  - BPEL based server for SOA
  - Contains Business Rule engine that allows business rule customization for each tenant

- **WebSphere Portlet Factory/WebSphere Portlet Server**
  - Portlet Factory IDE allows different profiles to be defined
    - Business logic can be profile dependent
      - Only run when specific profile is enabled at runtime
      - See the following SaaS blueprint for an example
Agenda

Introduction : SaaS and Cloud Computing -
Benefits and challenges of SaaS
Multi-tenancy approaches
SaaS Technical challenges
Addressing technical challenges with IBM middleware

Provisioning, monitoring and metering SaaS applications

Conclusion
Provisioning, monitoring and metering SaaS applications

- **Tivoli Provisioning Manager**
  - On board and off board applications and users
  - Allow self-service for new server requests
  - Automate provisioning of servers

- **Tivoli Monitoring**
  - For monitoring application performance

- **Tivoli Usage and Accounting Manager**
  - For metering usage of a SaaS application
  - Provides answers to the following question by collecting data throughout the IT infrastructure
    - Which tenant is consuming which resources?
    - What is the cost of the those resources (including shared resources)?
    - How should a SaaS provider allocate costs in order to bill tenants based on actual cost of providing the SaaS application?
Agenda

Introduction: SaaS and Cloud Computing -
Benefits and challenges of SaaS
Multi-tenancy approaches
SaaS Technical challenges
Addressing technical challenges with IBM middleware
Provisioning, monitoring and metering SaaS applications

Conclusion
Conclusion

- Five major approaches for enabling multi-tenancy in Software-as-a-Service
  1. Physical level
  2. Hypervisor level
  3. Operating System level
  4. Platform level
  5. Application level
- IBM software, systems and services can help to build and deploy SaaS applications
- The SaaS blueprints show how to address many of the technical challenges in developing multi-tenant applications
Thank you for attending...

All session materials from today’s virtual event will be posted in the CC4D MydeveloperWorks community. We want to continue hearing from you…. share your feedback from today’s session and stay connected through the Cloud computing for developers community