

# SaaS SCALABILITY





## TABLE OF CONTENTS

> 1. Introduction	1
> 2. Theory	1
2.1 Definition	2
2.2 Dimensions	2
2.3 Scalability Methods	3
> 3. SaaS Architecture Scalability	3
3.1 Distribution Tier	4
3.2 Application Tier	4
> 4. SaaS Administration Scalability	6
4.1 Billing	7
4.2 Operations	7
4.3 Support	7
> 5. Summary	8
> 6. References	8
> 7. Glossary of Terms	9



## 1. INTRODUCTION

This paper is one of a series of papers that explore and discuss the technical architectural components of the Software-as-a-Service (SaaS) model.

SaaS shares the distinction of being both a business model and an application delivery model. SaaS enables customers to utilize an application on a pay-as-you-go basis and eliminates the need to install and run the application on the customer's own hardware. Customers generally access the application via a web browser or thin client over the Internet. SaaS is most often subscription based and all ongoing support, maintenance, and upgrades are provided by the software vendor as part of the service. Application customization capabilities, if available at all, are generally provided to all customers in a consistent manner. From the perspective of the software vendor, the SaaS model provides stronger protection of its intellectual property, operational control of the environment running the software, and generally a repeatable revenue stream from the service subscription fees. Software vendors have varying capabilities and applications can come in varying flavors but SaaS applications most typically support many unique customers using a single instance of that application - also known as multi-tenancy.

Scalability is an important topic to be considered when planning a SaaS implementation. Since customers rely on the SaaS vendor to provide the solution reliably, it is critical that the application be able to handle the load of the current customer base and also be able to keep pace with the growth of that base. It is also critical that the SaaS vendor be able to handle growth from an administrative perspective or management of the business can quickly become chaotic.

This technical brief will address both the architectural aspects of load scalability and the organizational aspects of administrative scalability. Architectural scalability will be discussed in relation to the tiers defined in the SaaS Architecture Model presented in a prior paper on SaaS Architecture. Scalability of the data tiers was covered in a separate technical brief on SaaS Application Tenancy. This paper will focus on the remaining tiers.

## 2. THEORY

Before getting into the details of scalability as it applies to implementing a SaaS offering, we will first explore some theory. First we will provide a working definition of the term and then look at some dimensions of scalability to be discussed and some scalability methods.



## 2.1 Definition

Wikipedia provides an excellent definition that suits the purposes of this technical brief perfectly.

“Scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner, or to be readily enlarged. For example, it can refer to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added. An analogous meaning is implied when the word is used in a commercial context, where scalability of a company implies that the underlying business model offers the potential for economic growth within the company.”<sup>1</sup>

This definition provides a great framework for our discussion because it encompasses both the technical load and administrative business dimensions of the term. We will be covering both in this paper.

## 2.2 Dimensions

We will look at scalability from the aspects of load and administration as these are especially relevant to SaaS vendors..

### 2.2.1 Load Scalability

Load scalability deals with a system’s ability to effectively handle increasing levels of demand and throughput in a non-disruptive way. By non-disruptive we mean that additional resources can be added to the mix in a predefined way (i.e., servers, processors, RAM, disk space, bandwidth, etc.) so that as demand increases, the environment can be expanded to handle the increase. By contrast, a system that has not been designed to be scalable may need to go through a re-configuration or a disruptive re-architecture when demand reaches a certain point. So, it is good practice to think through how a solution will scale from the beginning to avoid unexpected future bottlenecks.

### 2.2.2 Administrative Scalability

Administrative scalability has to do with the SaaS vendor’s ability to manage multiple customers in a single environment. The ability to deliver consistent levels of service to each customer as the customer base grows is a function of how effective the administrative systems are that the staff uses to manage the customers. Therefore it is useful to think about how billing, operations and support will grow over time to meet customer demands.

## 2.3 Scalability Methods

From a theoretical standpoint, methods for addressing scalability generally fall into two categories: vertical and horizontal. Understanding the distinction between these two methods will help provide clarity when considering alternatives for scalable implementations.

### 2.3.1 Vertical

Scaling vertically, sometimes referred to as scaling *up*, means to add more resources to a node (a device connected to a network, such as a server or router) in a system. In terms of an application infrastructure, this might involve adding additional processors, RAM or disk space to a server. Planning for vertical scaling of hardware usually includes adding redundancy of all components (i.e., power supplies, fans, disk drives, etc.) to insure that machines remain running if a given component fails. Another way to look at vertical scaling might be to add more software servers to a single hardware server.

### 2.3.2 Horizontal

Scaling horizontally, sometimes referred to as scaling *out*, means to add more nodes to a system. This may look like adding another server to an application server cluster. A cluster is a group of systems that are coupled together, generally via a fast network, so they are aware of each other. Work is distributed to individual systems in the cluster and if one becomes unavailable, the others are able to handle the processing seamlessly.

## 3. SaaS ARCHITECTURE SCALABILITY

We will now turn to looking into scalability in the various tiers of the SaaS Architecture Model shown in Figure 1, focusing on the distribution and application tiers. Scalability of the data tiers was covered in a separate technical brief.

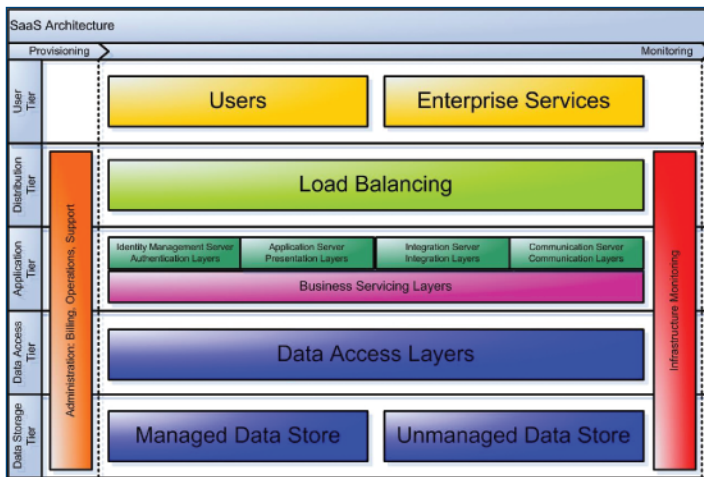


Figure 1 – SaaS Architecture Model



### **3.1 Distribution Tier**

The distribution tier in a SaaS implementation can serve multiple purposes. First, it can be used to distribute requests to different components of the application tier as appropriate. Second, it can also distribute load among different servers that might make up a cluster within a single component in the application tier. Third, it can be used to distribute traffic geographically so that different customers can be serviced at different locations.

It is possible to design a distribution tier that serves some or all of these purposes. This tier can be implemented from within the components in the application tier or as a completely separate physical tier. One advantage to separating the tiers physically is that they can be individually scaled, monitored and managed as the need arises.

### **3.2 Application Tier**

In the SaaS Architecture model we identified four distinct types of servers. There may be additional types of servers or services added at this tier depending on the needs of the overall solution, but the four identified can have general applicability across all SaaS implementations. In using the term server, we do not mean to imply a separate physical server. Some of these services can be run from the same server and scalability needs to be considered when thinking through implementation details.

#### **3.2.1 Identity Management Server**

A simple SaaS solution may contain its own authentication functionality whereby user ids and passwords are stored within the application. As these solutions mature and find more wide acceptance, the need for separating out identity management will naturally increase. Corporate users find it challenging to maintain multiple authentication credentials for different systems and therefore yearn for single sign on solutions. In this approach, once users sign on to their corporate network those same credentials can be presented to multiple applications to allow access. This not only makes it easier for users to get into the applications they need, but it also makes it easier to manage authorization centrally.

Identity management, often referred to as IdM, finds its roots in the X.500 directory services standards and one of the most widely used implementations of this is LDAP. Due to the nature of these services requiring fast access for many simultaneous users, many of the product offerings available today provide great scalability options both vertically and horizontally. They can easily handle tens of thousands of requests per second. Examples include the Sun Java Identity Management Suite<sup>2</sup> and Microsoft Active Directory<sup>3</sup>.



### 3.2.2 Application Server

The application server handles the main application functionality. For the purposes of this paper, we will distinguish the application server as the software engine that delivers dynamic application functionality to a client computer or device generally via the internet and generally using the HTTP protocol. This is different than a web server that generally delivers static content via HTTP. An application framework is a software framework designed to aid in the development of dynamic web applications.

The subject of application servers and frameworks is deep and broad enough to warrant a book to cover it extensively. There are many different products available and these are generally distinguished by the development languages they support, and of these environments provide their own way of separating data access from business logic and user interface code to facilitate the development of robust applications. The major application server environments also provide great scalability and can be expanded both vertically and horizontally.

### 3.2.3 Integration Server

Given that the importance of integration increases as the SaaS model matures, it is important to think about the way a SaaS solution will handle integration. The greatest challenge lies in making it as easy as possible for customers to tie into existing corporate data sources or services to avoid duplication of effort and to make sure data is accurate and up to date. With the emerging ubiquity of web services (see W3C definition<sup>4</sup>), it makes sense to at least provide a web services API so customers can add, delete, update or select data in the SaaS application. However, many customers find it challenging to provide the resources to write a program that will extract data from an existing corporate data source and then use this web services API to update the SaaS application.

Products are emerging from the EAI (enterprise application integration) space that help to simplify this process, and these products are designed to scale to handle the load of a major SaaS application. The SaaS vendor may need to consider both hosting an integration server layer as part of the offering as well as allowing the customer to host the integration layer internally. This is a function of each customer's individual corporate security policies and comfort levels with applications accessing their corporate data. Either way, the SaaS vendor has to be prepared to handle the load of programmatic access to data in addition to access via a user interface.



### 3.2.4 Communication Server

As SaaS applications become more, full featured and part of the overall corporate workflow, the need for handling inbound and outbound communication may naturally increase. Notifying users that data has changed, that they may need to approve changes, that certain work needs to be done, or any number of other communications can make a SaaS application all the more useful and user friendly. The SaaS vendor therefore may need to consider providing a scalable communication infrastructure.

Most electronic communication technologies are designed to scale so they can handle tremendous volumes. People often think of SMTP (email) initially when discussing communication, but other protocols that are also popular include SMPP (native SMS or text messaging) and SNPP (pagers), and these should be considered also.

### 3.2.5 Services

The SaaS Architecture Model defines Business Servicing Layers in the Application Tier. This refers to the fact that there may be additional functionality that is provided via various services. The term Enterprise Service Bus (ESB) is generally used to describe this abstraction layer. Further definition of ESB can be difficult because it is neither a single product nor a formal standard. In general, ESB functionality is built around XML messages, and while web services are not used exclusively, their use is certainly common (web services leverage XML wrapped in the SOAP protocol).

An example of some SaaS functionality that might be provided this way might be a workflow mechanism. Acknowledging the fact that a given SaaS vendor's product might represent a set of features that are a subset of a larger overall corporate process, it would be useful to provide customers with a simple way to pass in information from systems that are used earlier in the process and then pass data on to systems that are used later in the process. Workflow functionality such as this is being utilized with increasing frequency and SaaS vendors may consider providing services such as this as part of their overall product offering. Making sure that the implementation of this functionality allows for adequate scaling is critical.

## 4. SaaS ADMINISTRATION SCALABILITY

The ability of a SaaS vendor to efficiently scale its business as the customer base grows can be greatly enhanced by the administration functionality built into the SaaS environment. Some of the issues around scaling the billing, operations and support functions will be discussed.



## 4.1 Billing

As a SaaS customer base grows, managing invoice generation and payment processing can become challenging. If usage needs to be measured or metered, this adds complexity to the problem. This aspect of administration is sometimes referred to as billing mediation and ideally handles the automated generation of invoices based on customer contract and usage, forwards invoice data to a payment processing system and also updates the accounting system for financial reporting. These three components of metering, invoicing and payment processing can be handled within a single billing system within the SaaS application or as separate systems. If separate, some may even be outsourced to third parties, with invoicing and payment processing being the most likely candidates.


## 4.2 Operations

The operations side of administration also needs to think about how it will scale as the business grows. It is one thing to keep track of a few dozen customers in a multi-tenancy environment but quite a different exercise to manage a few thousand. Issues such as keeping track of resource usage (including disk space), data backup and restoration and current account status (has the customer been suspended for lack of payment?) are just some of those that can become unwieldy if the administration system is not designed accordingly. It is also important to consider how the provisioning of additional services might be handled.

## 4.3 Support

The more dependent a customer becomes on a SaaS solution or the more critical the solution becomes to the overall workflow of a business, the more the customer will look to the SaaS vendor for support. The ability to deliver satisfactory support can become a function of how well the support system scales. There are many layers that make up a world class support offering and these may include any or all of the following.

- > 24/7 live Help Desk
- > Integrated ticketing system allowing users to submit support requests and track status
- > Online help
- > Online video training
- > Forums or threaded discussion groups
- > Searchable knowledge base
- > User groups



Understanding the needs of users and then providing the support that will leave them feeling taken care of is one of the keys to retaining customers. Satisfied customers also become a sales asset as their testimonials can become a powerful marketing tool. Thinking through how to provide these various layers of support and how to do this efficiently as the business grows is a critical aspect of SaaS administration.

## 5. SUMMARY

In this technical brief we have covered the topic of scalability both as it applies to aspects of the architecture of a SaaS application as well as to efficient administration of the business. We began with some theory around scalability to provide a basis for the discussion. Then the various tiers of the SaaS Architecture Model were reviewed from the perspective of scalability. Finally, the administrative topics of billing, operations and support were discussed and scalability issues were noted.

The very nature of a SaaS application and the SaaS business model, calls for inherent scalability. Addressing the issues before they become problems can make the difference between success or failure and this paper provided many of these issues for consideration.

## 6. REFERENCES

<sup>1</sup>Wikipedia Scalability definition – <http://en.wikipedia.org/wiki/Scalability>.

<sup>2</sup>Sun Java Identity Management Suite – <http://www.sun.com/software/products/identity/offerings.jsp>.

<sup>3</sup>Microsoft Active Directory – <http://www.microsoft.com/windowsserver2003/technologies/directory/activedirectory/default.aspx>.

<sup>4</sup>W3C Working Group Note 11 February 2004, “Web Services Glossary,” – <http://www.w3.org/TR/ws-gloss/>.

## 7. GLOSSARY OF TERMS

Term	Description
Cluster	A <b>Cluster</b> is a group of systems that are coupled together, general via a fast network, so they are aware of each other. Work is distributed to individual systems in the cluster and if one becomes unavailable, the others are able to handle the processing seamlessly.
Customer/Tenant	<b>Customer/Tenant</b> is a term in this document that refers to the company or business that subscribes to the SaaS offering.
HTTP	<b>Hypertext Transfer Protocol</b> is a communications protocol used to transfer data on the web.
IdM	<b>IdM</b> , or Identity Management, involves the management of the identity life cycle of entities including the establishment, description and destruction of the identity. IdM is often generically used to refer to the management of user authentication.
LDAP	<b>LDAP</b> stands for Lightweight Directory Access Protocol is an application protocol for querying and modifying directory services running over TCP/IP.
Multi-Tenancy	<b>Multi-Tenancy</b> is an architectural design concept used to maximize customer and user scalability. As well as to provide maximum operational scalability and lower support costs for the service provider by leveraging common user interface, business logic, and/or databases for all customers and users. Multi-tenancy at all layers of the application is considered an architectural ideal but not required at all layers to realize benefits. The cost savings often associated with multi-tenancy are often considered key to building and maintaining lower cost service offerings.

SaaS	<p><b>Software-as-a-Service (SaaS)</b> shares the distinction of being both a business model and an application delivery model. SaaS enables customers to utilize an application on a pay-as-you-go basis and eliminates the need to install and run the application on the customer's own hardware. Customers generally access the application via a web browser or thin client over the Internet. SaaS is most often subscription based and all ongoing support, maintenance, and upgrades are provided by the software vendor as part of the service. Application customization capabilities, if available at all, are generally provided to all customers in a consistent manner. From the perspective of the software vendor, the SaaS model provides stronger protection of its intellectual property, operational control of the environment running the software, and generally a repeatable revenue stream from the service subscription fees. Software vendors have varying capabilities and applications can come in varying flavors but SaaS applications most typically support many unique customers using a single instance of that application - also known as multi-tenancy.</p>
SaaS Offering	<p>A <b>SaaS Offering</b> in this document is defined as a turnkey service offering, which includes the application license, maintenance, application support, subscription pricing, and hosting and associated delivery infrastructure.</p>
SaaS-Enabled Application	<p>A <b>SaaS-Enabled Application</b> is defined as an application that has been designed and built to be deployed and consumed in a service model, and incorporates many of the attributes previously defined above.</p>
SMPP	<p>The <b>short message peer-to-peer protocol (SMPP)</b> is a telecommunications industry protocol for exchanging SMS messages between SMS peer entities such as short message service centers. It is often used to allow third parties to submit messages, often in bulk.</p>
SMS	<p><b>Short Message Service (SMS)</b> is a communications protocol allowing the interchange of short text messages between mobile telephone devices.</p>
SNPP	<p><b>Simple Network Paging Protocol (SNPP)</b> is a protocol that defines a method by which a pager can receive a message over the Internet.</p>
Users	<p><b>User</b> in this document is defined as an employee of the customer, as is the ultimate user of the service.</p>

Web Application Framework	A <b>Web Application Framework</b> is a software framework designed to aid in the development of dynamic web applications.
Web Application Server	A <b>Web Application Server</b> is the software engine that delivers dynamic application functionality to a client computer or device generally via the internet and generally using the HTTP protocol.
Web Server	A <b>Web Server</b> generally delivers static content via HTTP.
Web Service	A <b>Web Service</b> is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.
X.500	<b>X.500</b> refers to a series of computer networking standards covering electronic directory services that was developed by the International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T).



**Worldwide Headquarters**

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA  
Tel: +1 781 280-4000 Fax: +1 781 280-4095  
[www.progress.com](http://www.progress.com)

**For regional international office locations and contact information, please refer to [www.progress.com/worldwide](http://www.progress.com/worldwide)**

© Copyright 2008 Progress Software Corporation. All rights reserved. Progress is a registered trademark of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. or other countries. Any other trademarks contained herein are the property of their respective owners.