Software as a Service

Haojie Hang
Ogheneovo Dibie

Executive Summary

- In this presentation, we go through the **Software as a Service** Methodology, examine its benefits and drawbacks and talk about two state-of-art SaaS systems– **Amazon Web Service** and **Google App Engine**
- We also look into **Service Oriented Architecture** powering SaaS applications and its impact on **modern web 2.0 applications**
- Finally, we examine **hybrids** of traditional and SaaS applications

Overview

- What is Software as a Service (SaaS)
- Background
  - Brief history
  - Concept
  - Big picture
  - Related terms
- Computing Today
  - SasS is everywhere
  - The SaaS Market
- Benefits of SaaS
- Drawbacks of SaaS
  - Robustness
  - Privacy
  - Security
  - Reliability
- Service Oriented Architectures (SOA)
  - Guiding principles of SOA
- Case studies
  - Amazon Web Services (AWS)
  - Google App Engine
- Influence of SOA on Web 2.0 development
  - Zend Framework
- Hybrids of Traditional and SaaS applications
  - Dropbox
  - Microsoft Office
- Summary
- References

What is SaaS?

- **Definition**: Software as a Service (SaaS), a.k.a. on-demand software, is a software delivery model in which software and its associated data are hosted centrally and accessed using a thin-client, usually a web browser over the internet. – Wikipedia

- Simply put, SaaS is a method for delivering software that provides remote access to software as a web-based service. The software service can be purchased with a monthly fee and pay as you go.

What is SaaS?

- Where does the term SaaS come from?
  - The *SAAS* acronym allegedly first appeared in an article called "Strategic Backgrounder: Software As A Service", internally published in February 2001 by the Software & Information Industry's eBusiness Division
- Multi-tenant architecture
  - Virtualization as a alternative
- Pricing model
  - pay as we go, relatively low cost for user provisioning
- Configuration and customization
  - Easy for application customization
- Accelerated feature delivery
  - It means a much shorter release cycle
- Open integration protocols
  - Typically based on HTTP, JSON,REST, SOAP

An example

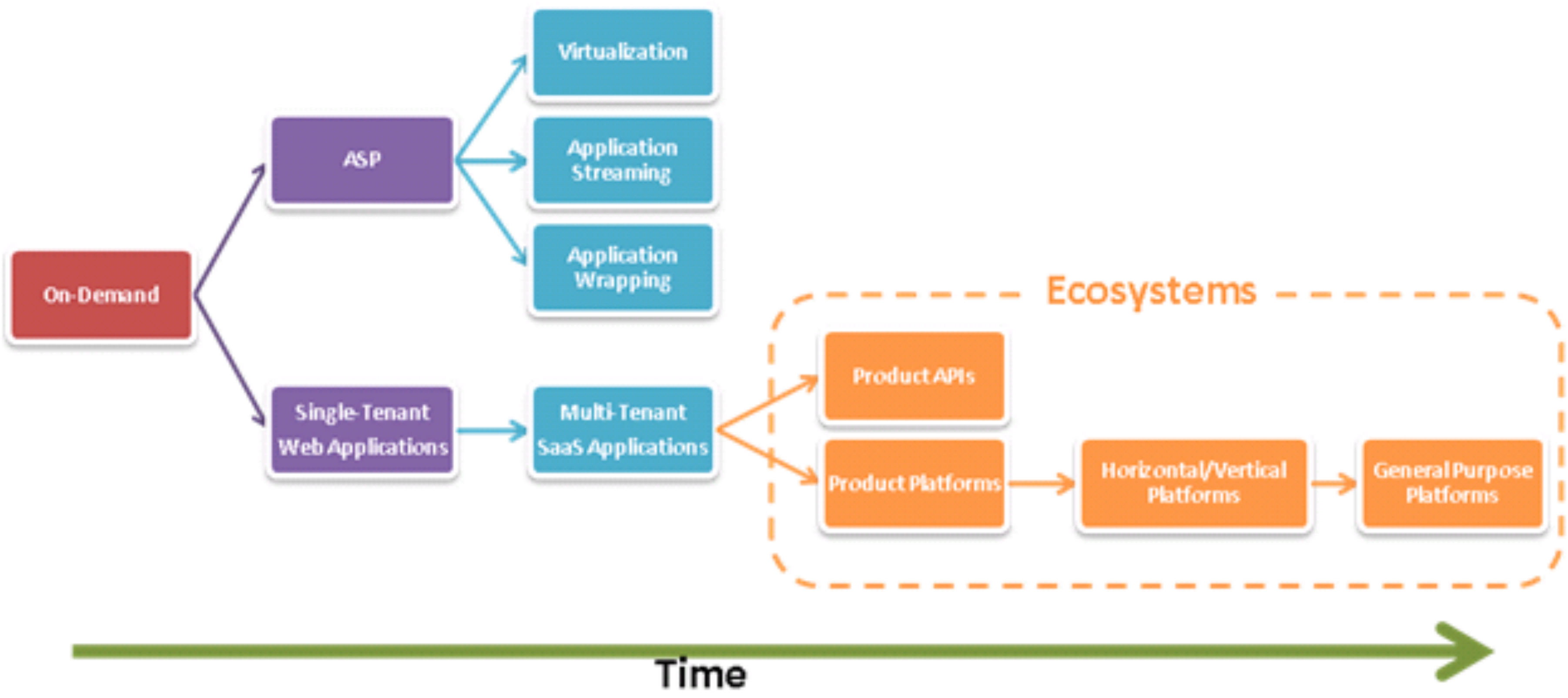
- **Imagine** you are the founder of a start-up company and you need to deal with tons of new customers
- Buying a full version Customer Relationship Management (CRM) Software is expensive
- With SaaS, you can buy a web-based CRM software that is pay as you go and scales to demand!

- **Benefits**: Save money on software license, cut cost on maintenance, and hardware purchase. Combined with lower start-up cost and a faster return on investment!

A Brief History

- In 1960s, IBM and other mainframe providers conducted time-sharing or utility computing services, offering computer power and database to banks and large organization
- In 1990s with the expansion of Internet, Application Service Providers (ASP) appeared. They provided small businesses with the service of hosting and managing specialized business application
- Starting from 2003, the true SaaS became popular due to the increased speed of internet connections. Ultimately, all software will be web-based and pay-as-go

Timeline

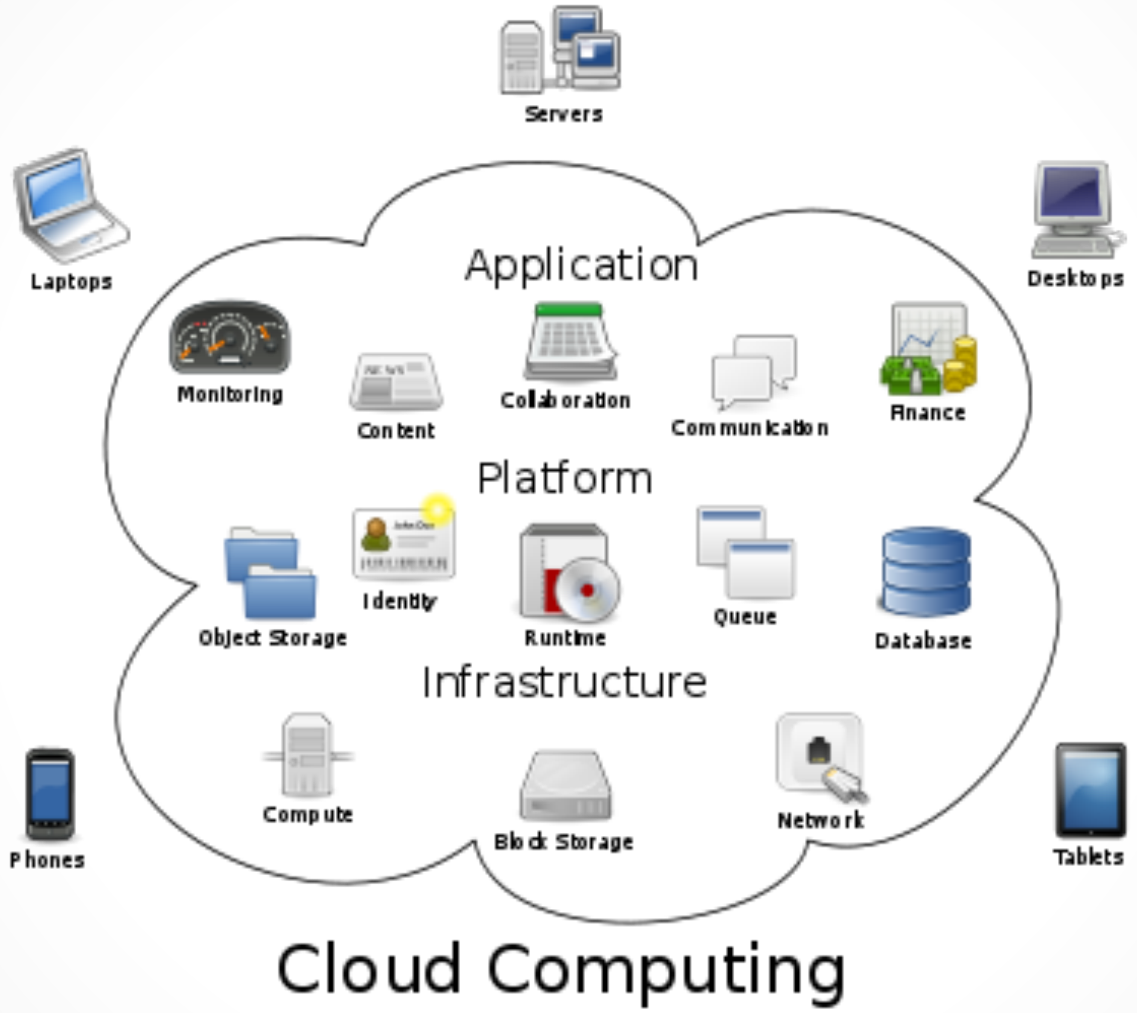

On-Demand
ASP
Single-Tenant Web Applications
Virtualization
Application Streaming
Application Wrapping
Multi-Tenant SaaS Applications
Ecosystems
Product APIs
Product Platforms
Horizontal/Vertical Platforms
General Purpose Platforms
Time

This diagrams shows the evolution of the Software as a Service and its ecosystem

Concept

- The idea of using software as a service first popped up in the late 1990s in order to allow sharing end-user licenses in a way that reduced cost and also shifted infrastructure demands from the company to the software provider.
- Does it merely save on the license cost?
- And more: upgrading, maintenance, hardware…

The Big Picture



Software as a Service is located in the application level of the stack

Related Terminology

- Cloud computing
  - Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility over a network
- Platform as a Service
  - Platform as a service (PaaS) is a category of cloud computing services that provide a computing platform and a solution stack as a service.
- Infrastructure as a Service
  - Infrastructure as a Service is a provision model in which an organization outsources the equipment used to support operations, including storage, hardware, servers and networking components. The service provider owns the equipment and is responsible for housing, running and maintaining it.
- Multi-Tenancy
  - Multi-Tenancy refers to a principle in software architecture where a single instance of the software runs on a server, serving multiple client organizations
- Application Service Provider
  - provided businesses with the service of hosting and managing specialized business applications, with the goal of reducing costs through central administration and through the solution provider's specialization in a particular business application

Some key differences

- Clouding computing is the big application context (umbrella) covering SaaS and other related terms
- "… as a Service" are the buzz words used to specify various application scenarios.
    - E.g. Storage as a Service is an umbrella for SaaS applications that provide cloud storage.
- PaaS deals with whole computing platforms provided as a service such as operating system enviromnets
    - E.g. Google Chrome OS running on Google ChromeBooks
- IaaS aims to provide the whole computing power (computer clusters) for the application domain so we don't have to worry about the physical machines and how they are deployed

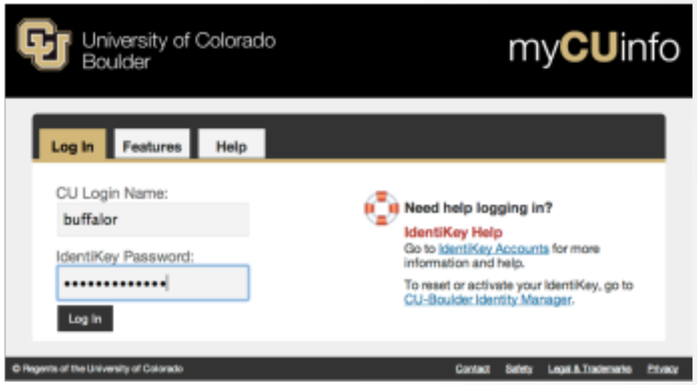
Computing Today

- We are in the era of Cloud Computing!
- Cloud computing stack
  - Infrastructure as a Service (IaaS)
  - Platform as a Service (PaaS)
  - Software as a Service (SaaS)
  - Monitoring as a Service (MaaS) - emerging
  - Communication as a Service (CaaS)
  - Anything as a Service – emerging
- So many buzz terms...but SaaS is one of the most widely used service model

SaaS is everywhere

YouTube Broadcast Yourself™

Gmail™ by Google


University of Colorado Boulder

myCUinfo

CU Login Name: buffalor

IdentiKey Password: ••••••••••••••

Need help logging in?

IdentiKey Help
Go to IdentiKey Accounts for more information and help.

To reset or activate your IdentiKey, go to CU-Boulder Identity Manager.

© Regents of the University of Colorado

facebook®

amazon.com®

ebay®

The SaaS market

- SaaS sales in 2010 reached $10billion
- In 2011, its sales is up 20.7% from 2010
- SaaS revenue will be more than double its 2010 numbers by 2015 and reach a projected $21.3
- Business SaaS is the major market – Customer Relationship Management (CRM) is the largest market with 18.8% annually growth worldwide

Business's perspective

SaaS has a lot of appeal to businesses. Here are a few reasons why:

- Multi-tenant software architecture
- Low cost, fast investment, shared license
- High manageability
- Free of deployment and support
- Cost-effective: pay as we go
- Customization is easy
- Can scale well – commercialization

Advantages of SaaS

- Easy to use – Most SaaS applications do not require more than a web browser to run
- Cheap- The pay as you go pricing model of SaaS makes it affordable to small businesses and individuals.
- Scalability: SaaS application can be easily scaled up or down to meet consumer demand. Consumers do not need to worry about additional computing infrastructure to scale up.
- Applications are less prone to data loss since data is being stored in the cloud.
- Compared to traditional applications, SaaS applications are less clunky. They do not require users to install/uninstall binary code on their machines
- Due to the delivery nature of Sass through the internet, SaaS applications are able to run on a wide variety of devices.
- Allows for better collaboration between teams since the data is stored in a central location.
- Velocity of change in SaaS applications is much faster.
- SaaS favors a Agile development life cycle.
    - Software changes and frequent and on-demand. Most Saas services are updated about every 2 weeks and users are most time unaware of these changes.

Drawbacks of SaaS

- Robustness:
  - SaaS software may not be as robust (functionality wise) as traditional software applications due to browser limitations. Consider Google Doc & Microsoft Office.
- Privacy
  - Having all of a user's data sit in the cloud raises security & privacy concerns. SaaS providers are usually the target of hack exploits e.g. Google servers have been the target of exploits purportedly from China in the last several years
- Security
  - Attack detection, malicious code detection
- Reliability:
  - In the rare event of a SaaS provider going down, a wide range of dependent clients could be affected. For example, when Amazon EC2 service went down in April 2011, it took down FourSquare, Reddit, Quora and other well known applications that run on it.
  - We shall discuss each of these issues in more details in the next section

Robustness

- SaaS applications may not be able to provide the same level of functionality as traditional applications. This is partly due to current limitations of the web browser. Consider Google doc and Microsoft Office
- Most SaaS applications are intolerant to slow internet connections and this can lead to erratic behavior
  - Google doc may not be synchronized well between teams in a low internet connection

Privacy

- Lots of issues arise with sensitive data stored in the cloud. Common privacy questions include:
  - Who has the access to the data? How to distribute the rights?
  - What type of data can be saved on the cloud, and locally? What about the confidential data?
  - Don’t we really have to worry about data sharing? Who is viewing our data, modifying the data, and re-distributing our data? With or without permission?
  - Data sharing between private and public clouds

Security

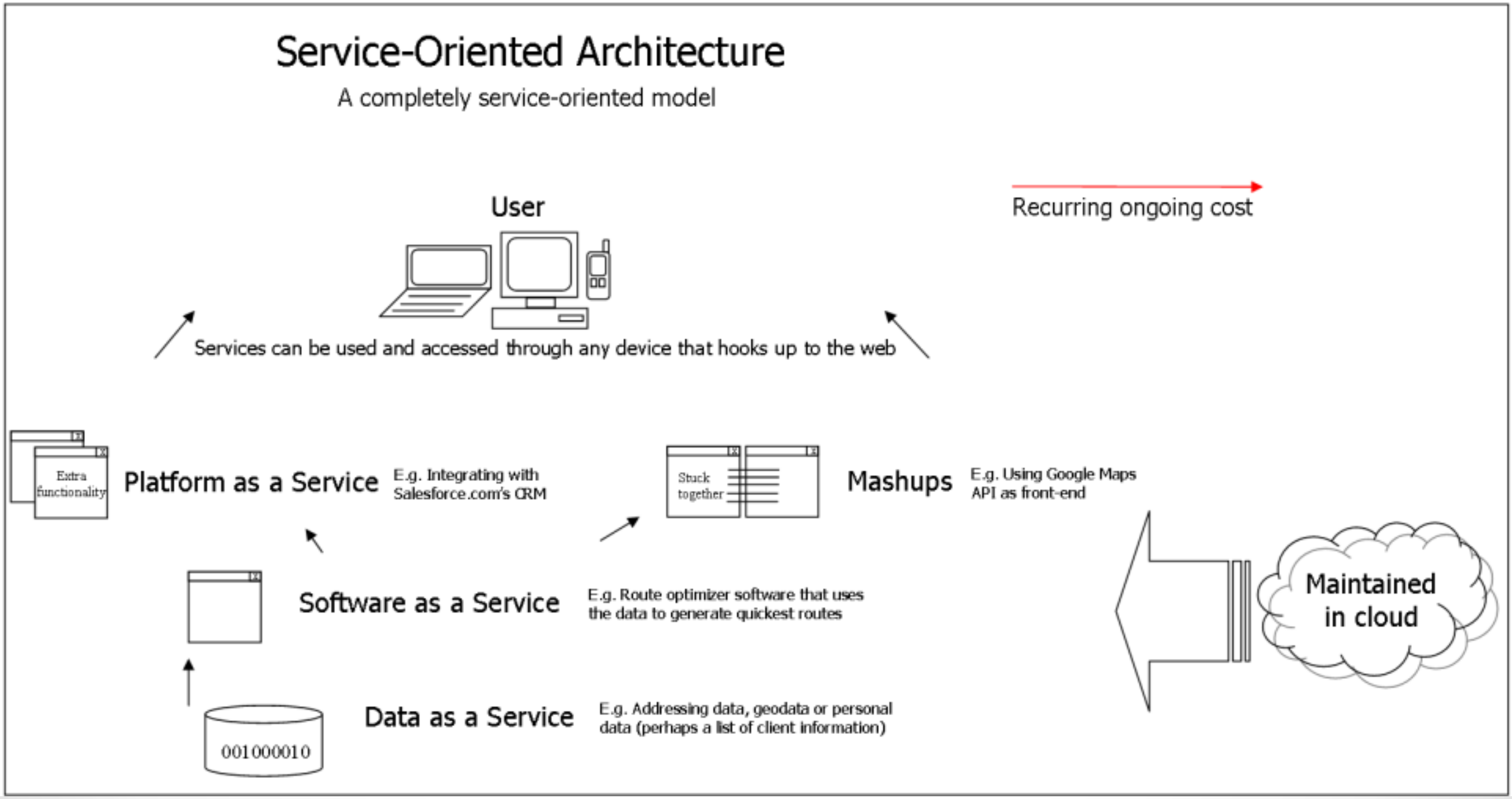
- SaaS applications are prone to attack because everything is sent over the internet
- Data encryption and decryption
- Communication protocols
- Virtualization versus Multi-tenant architecture: which one is better in terms of the security?
- Transaction processing, networking issues

Reliability

- Although most SaaS applications are highly reliable, down time is still inevitable and can be very expensive – commercial SaaS software
- The application, data, backups, everything are in the cloud, thus making it hard to recover from the server down time.
    - You don't physically own the code, they are in the cloud

Service-Oriented Architecture

- SaaS is the methodology for providing computing service over the Internet
- SOA is the software architecture that powers SaaS application
  - One of the most commonly seen practices for SaaS and cloud computing
- Definition: a set of principles and methodologies for designing and developing software in the form of interoperable services
  - It provides a way for consumers of services, such as web-based applications to be aware of available SOA-based services.


Service-Oriented Architecture
A completely service-oriented model
User
Recurring ongoing cost
Services can be used and accessed through any device that hooks up to the web
Extra functionality
Platform as a Service
E.g. Integrating with Salesforce.com's CRM
Stuck together
Mashups
E.g. Using Google Maps API as front-end
Software as a Service
E.g. Route optimizer software that uses the data to generate quickest routes
Maintained in cloud
001000010
Data as a Service
E.g. Addressing data, geodata or personal data (perhaps a list of client information)

The diagram above shows the role of SOA in SaaS

Guiding Principles of SOA

- Standardized service contract
  - Services adhere to a communications agreement, as defined collectively by one or more service-description documents
- Service abstraction
  - Beyond descriptions in the service contract, services hide logic from the outside world.
- Service loose coupling
  - Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.
- Service autonomy
  - Services have control over the logic they encapsulate.
- Service reusability
  - Logic is divided into services with the intention of promoting reuse.
- Service granularity
  - A design consideration to provide optimal scope and right granular level of the business functionality in a service operation.
- Service statelessness
  - Services minimize resource consumption by deferring the management of state information when necessary
- Service composability
  - Services are effective composition participants, regardless of the size and complexity of the composition.

Commonly-used Protocols

- JSON
  - The JSON format is often used for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application, serving as an alternative to XML
- XML
  - The design goals of XML emphasize simplicity, generality, and usability over the Internet. It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services
- SOAP
  - originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks
- ATOM
  - The name Atom applies to a pair of related standards. The Atom Syndication Format is an XML language used for web feeds, while the Atom Publishing Protocol is a simple HTTP-based protocol for creating and updating web resources

SaSS case studies

- We would now consider the services provided by two of the biggest SaaS providers today: Amazon & Google.
- We look into the categories of their SaaS offerings and how they improve modern application development & deployment.

Amazon Web Services(AWS)

- Beginning in 2006, Amazon web services provides a wide range of services and solutions for powering applications. They fall under the following categories:
  - Storage
    - Amazon simple storage services(S3)
    - Amazon Elastic Book Store(EBS)
  - Networking
    - Amazon Virtual Private Cloud (VPC)
    - Amazon Route53
  - Database
    - Amazon Dynamo DB
    - Amazon Relational Database Service (RDS)
  - Compute
    - Amazon Elastic Cloud Compute (EC2)
    - Amazon Elastic Map Reduce (EMR)

AWS: Compute

- Amazon's compute web services provide users with raw computation power to meet application needs and scale accordingly. AWS have two core web services for computation.

- Amazon Elastic Cloud Compute (Amazon EC 2)
  - Amazon EC2 web service allows for resizable compute capacity in the cloud. With Amazon EC2, developers are able to easily scale their computation needs up or down to meet demand.

AWS: Compute

- Amazon Elastic Map Reduce

  Amazon Elastic Map Reduce (EMR) provides developers and researchers with compute power for processing data intensive tasks. It is based on Apache's Hadoop framework. As with EC2, users can easily provision how much compute resources they need to process data intensive tasks such as web mining, data warehousing, log file analysis, scientific calculations and so on. It allows users focus on the task at hand rather than worry about setting up computational frameworks to handle these tasks.

AWS: Storage

- Amazon’s storage web service provides a cost effective solution for storing and retrieving data easily. Amazon’s simple storage service (S3) and Elastic Book Store (EBS) are the main services in this area.

Amazon Simple Storage Service (S3): Amazon S3 makes it easy for developers to store and retrieve any amount of data, at any time from anywhere in the web. Using hashing technology where data is stored in key value pairs. The value being data objects and key a unique identifier assigned to a developer, it provides a fast, efficient and durable way for storing data.

AWS: Storage

- Amazon Elastic Book Store(EBS): Amazon EBS provides block storage instances that work independently of Amazon EC2 instances. They can act as backup data stores for EC2 instances by providing file-system like volumes that can be mounted on the machine.

AWS: Networking

- Amazon Virtual Private Cloud (Amazon VPC): Amazon VPC allows users to provision an isolated section of the AWS cloud for launching AWS resources. With Amazon VPC, you have complete control over the amount of resources within your private space, including the range of IP addresses, web servers and other compute resources.

AWS Networking

- Amazon Route53: Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. It enables developers to easily route domain names to AWS resources.

AWS Database

- Amazon Dynamo DB: Amazon's Dynamo DB is a fully managed NOSQL database service that provides fast and predictable performance with seamless scalability.

AWS Database

- Amazon Relational Database Service (Amazon RDS): Amazon RDS is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. Amazon's RDS has similar to MySQL and Oracle database systems. Amazon RDS automatically backs up data stored on instances and provides an easy way to scale up to meet application needs

Google App Engine

- **Google App Engine** enables you to build web applications on the same scalable systems that power Google applications, which is great!
- **App Engine applications** are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow.
- With App Engine, there are **no servers to maintain**. You just upload your application, and it's ready to serve to your users.

Benefits of App Engine

- Easy to get started
  - With App Engine you write your application code, test it on your local machine and upload it to Google with a simple click of a button or command line script. Once your application is uploaded to Google we host and scale your application for you
- Free and risk-free development
  - You can create an account and publish an application that people can use right away at no charge, and with no obligation. When you need to use more resources, you can enable billing and allocate your budget according to your needs
- Automatic Scalability
  - No matter how many users you have or how much data your application stores, App Engine can scale to meet your needs.
- The reliability, performance and security of Google infrastructure
  - Trustable: The same security, privacy and data protection policies we have for Google's applications applies to all App Engine applications.

Application Environment

- Dynamic web serving, with full support for common web technologies
- Persistent storage with queries, sorting and transactions
- Automatic scaling and load balancing
- APIs for authenticating users and sending email using Google Accounts
- A fully featured local development environment that simulates Google App Engine on your computer
- Task queues for performing work outside of the scope of a web request
- Scheduled tasks for triggering events at specified times and regular intervals

Google App Engine: Language support

- Currently, Google App Engine supports two application environments: **Java** and **Python**.
- Additionally, your website templates can include **JavaScript** along with your HTML which, among other things, allows you to write AJAX-enabled web applications.
  - App Engine applications can also be written in Java or any JVM-compatible language (e.g. JRuby, Groovy, Scala, etc.) and run in a Java 6 runtime environment.
  - App Engine's Python runtime supports Python 2.5 – newer versions of Python, including Python 2.6, are not currently supported. For security reasons, some Python modules written in C won't run in App Engine's sandbox.

SaaS and web 2.0 development

- Software as a service methodology has had a significant impact on the development of modern web 2.0 applications. This is especially evident in modern web development frameworks.
- Consider the following common use cases:
    - Most secure modern web applications use captcha images as a way of preventing form submissions by web bots and other automated entities.
    - Facebook, twitter, Google+ site integrations are common social integrations. This could be simple things such as like, tweets or +1 buttons or more complex such as identity notification and single sign on.
    - Many web applications make use of URL shortening services to enable easy-to-read, transferable URLs.
    - Many news sites, online magazines and blogs power their commenting features by integrating web services from service providers such as Disqus, IntenseDebate and Facebook comments API
    - Google Maps, Calendar and Youtube are now very common as embeds in many websites. Given their dynamic structure, they are more appealing compared to static map images or calendars.

Saas and web 2.0 development

- In the following section, we examine the impact of SaaS methodology on the Zend Framework for PHP. Other popular web development frameworks such as Ruby on Rails share a similar influence

Zend Framework(ZF)

- The Zend Framework is a full stack, object oriented, web development framework .
  - Began in late 2005
  - It consists of a library of components covering most of the used functionalities on the web such as form creation, authentication, access control lists, input validation, search and so on.
  - It is primarily a Model View Controller (MVC) architecture.
  - It also features a Use-at-will architecture: Users can use the framework out of the box or just use components of the framework as needed.
  - Sponsored by Zend, the official PHP company.
  - Very active developer community.
  - All code developed go through rigorous testing before being deployed in a release.

ZF and SaaS

- Zend Framework promotes the consumption and publishing of feeds via the Zend_Service and Zend_Rest APIs. We also look at the Zend_Cloud_Api which allows easy access to platform-type SaaS services such as Amazon Web Services.

Consuming feeds with Zend_Service

- The Zend_Service is an abstract class that serves as a foundation for REST & SOAP web service implementations.
- Zend_Service has a host of concrete implementations that act wrappers to popular web service APIs. They include:
  - Zend_Service_Twitter: Implements a client for Twitter's REST based APIs.
  - Zend_Service_Yahoo: A simple API for accessing many of Yahoo's REST based web services
  - Zend_Service_Ebay: A group of APIs for accessing Ebays web services
  - Zend_Service_ReCaptcha
  - Zend_Service_ShortUrl: Provides an API for accessing a number of different URL shortner services.
  - Zend_Service_Flickr: A simple API for using the Flickr REST Webservice and many more..

Accessing platform cloud services: Zend Framework Simple Cloud API

- Starting in 2009, the Zend Framework included a simple cloud API called Zend Cloud.
- Zend Cloud provides a single unified API for all the major Sass cloud providers such as Amazon, Rackspace, Windows Azure and Nirvanix.

Zend Cloud: Storage Service

- The Zend Cloud Storage Service provides a simple API for file storage on the cloud.
- The service abstracts the internal structure of files and they are only identifiable by a string key. Right now it supports Amazon S3, Nirvanix and WindowsAzure.

Zend Cloud’s Storage service

- Example : Instantiating an Amazon S3 adapter

```
$storage = Zend_Cloud_StorageService_Factory::getAdapter(array(
Zend_Cloud_StorageService_Factory::STORAGE_ADAPTER_KEY =>'Zend_Cloud_St,
orageService_Adapter_S3',
Zend_Cloud_StorageService_Adapter_S3::AWS_ACCESS_KEY   =>$amazonKey,
Zend_Cloud_StorageService_Adapter_S3::AWS_SECRET_KEY   =>$amazonSecret,
));
```

- Storing an item with Zend Cloud:

```
$data = file_get_contents('/my/local/dir/picture.jpg');
$returnedData = $storage->storeItem('/my/remote/path/picture.jpg',
$data);
```

One just has to modify the adapter to work with different providers.

Publishing feeds

- Apart from consuming web services, the Zend Framework makes it very easy to expose your application's services. It includes the Zend_REST_Server and Zend_Json_Server that enables the creation of web services which return XML and JSON responses.

Publishing web services: Zend_REST

- Zend_REST features two core implementations: Zend_REST_Client and Zend_REST_Server.
    - Zend_REST_Client provides a simple programmatic API for consuming RestFul web services while Zend_REST_Server provides a simple interface that makes publishing your application data provides a very simple interface for making class methods and functionality publicly accessible. We show a brief example of publishing a web service with Zend_REST_Server in the following section

Zend_Rest_Server: code example

- Working with Zend's Zend_Rest_Server component is very straightforward. To illustrate this, we show a simple code sample below:
- We begin by creating a class entity called Greetings with a single function ***sayHello***

```
<?php
Class Greetings {
/**
*@param  string $user_ name
*@return string
*/
 public function sayHello($user_name){
                return “Hello $user_name. How is your day going?”;
            }
}
```

Zend_Rest_Server: example

- We now create a controller class to handle incoming requests. The controller class features an action which instantiates the Zend_Rest_Server class

```
class RestController extends Zend_Controller_Action
{
protected $_server;
public function init()
{
$this->_server = new Zend_Rest_Server();
$this->_helper->viewRenderer->setNoRender();
}
public function indexAction()
{
require_once ‘Greetings.php';
$this->_server->setClass(‘Greetings');
$this->_server->handle();
}
}
```

With the Zend_REST_SERVER in place, a RESTful call such as http://mysite/rest?method=sayHello&name="Jack would return a result as follows:

Zend_Rest_Server: code example

```
<Greetings generator="zend" version="1.0">
<result>
<value>Hello Jack. How are you doing today? </value>
</result>
</Greetings>
```

Results can also be returned in JSON format if desired.

Hybrid of Traditional & SaaS applications

- Although most of our discussions have highlighted the advantages of SaaS architectures, we’ve also illustrated some of the drawbacks of SaaS most of which relate to security, privacy and reliability.
- To address these issues while taking advantages of SaaS, hybrids have developed recently
- SAP is one of the companies leading the way in this endeavor

Hybrid of Traditional & SaaS :

- Many SaaS providers today take such an approach. A good example of which is Dropbox.
- Dropbox creates a virtualized directory containing user files on their machine thus enabling offline access.
- Users are also able to access their data online through a web browser.

Tradition & SaaS Hybrid

- Microsoft Office also uses a hybdrid traditional SaaS software approach. Office desktop suite can be installed locally but documents can also be accessed and shared online through Microsoft Sky Drive.

Benefits of Traditional & SaaS Hybrid

- Applications and data can be accessed offline. This is especially useful in situations where users have limited or no internet access such as on airplanes.
- Applications can be better secure as users may decide which applications to upload to the cloud or remain with locally.
- Having local copies of data serve as rain-coat from complete data loss in situations where the SaaS provider is experiencing downtime.

Summary

- SaaS greatly enhances the ability of developers to scale their application on demand and better suite customer needs
- It encourages Agile practices by enabling providers deliver frequent updates/patches without waiting for major release cycles as in traditional applications.
- SaaS applications however are susceptible to privacy, security and reliability concerns
- Hybrid environments combining both SaaS and traditional application methodologies may be useful in scenarios of extremely sensitive data or where constant up-time must be maintained.

Summary

- Use of SaaS services is pervasive in the development and deployment of modern applications
- In conclusion, the SaaS methodology is very mature and would play a central to the future of computing.

Reference

- [1] http://en.wikipedia.org/wiki/Cloud_computing
- [2] http://aws.amazon.com/
- [3] http://ezinearticles.com/SaaS---History-and-a-Look-Ahead&id=2246590
- [4] http://www.service-now.com/knowledge.do?sysparm_document_key=kb_knowledge,4c8e15b90a0a3cc800e559d37a644090
- [5] Cloud business trends. http://www.cloudbusinesstrends.com/2011/05/19/analysis-saas-and-cloud-computing-the-future-of-software-development-stir-saas-cloud-computing-s.html
- [6] Allen, Rob et al. *Zend Framework in Action*. Manning Publications. December 2008.
- [7] The Zend Framework manual http://framework.zend.com/manual/1.11/en/manual.html
- [8] Pope, Keith. Zend Framework 1.8 Web Application Development. Packt Publishing. September 2009
- [9] http://code.google.com/appengine/