A Comparative Analysis of Cloud Computing Environments

Prof. Dr. Andreas Polze
Operating Systems and Middleware
Hasso-Plattner-Institute for Software Engineering
University of Postdam
D-14440 Potsdam, Germany

andreas.polze@hpi.uni-potsdam.de

December 7, 2009

Table of Content

A Comparative Analysis of Cloud Computing Environments

Cloud is about a new business model for providing and obtaining IT services. Cloud computing promises to cut operational and capital cost. It lets IT departments focus on strategic projects rather than on managing the own datacenter. Cloud computing is building upon a number of ideas and principles, that have been established in context of utility computing, grid computing, and autonomic computing a couple of years ago. However, in contrast to previous approaches, cloud computing no longer assumes that developers and users are aware of the provisioning and management infrastructure for cloud services.

Cloud computing has established the notion of a service as basic unit of abstraction. Services are "running in the cloud" and can be consumed via standard WebService interfaces. Services can be on different levels of abstraction - comparable to standard applications (Software-as-a-Service - SaaS), comparable to frameworks and programming platforms (Platform-as-a-Service - PaaS), or comparable to an (virtualized) IT infrastructure (Infrastructure-as-a-Service - IaaS). From the standpoint of a programmer, cloud computing on each of these levels of abstraction has a number of technical design choices, which impose implications on developers.

Cloud computing environments can be offering their services via the Internet - thus forming the public cloud - or can be offering services only on a company's intranet - effectively implementing a private cloud. The usage of cloud technology with services both on the intranet and the Internet simultaneously is sometimes called "hybrid cloud".

Cloud computing is appealing for certain kind of applications, namely those imposing a variable load and needing massive scaling (such as Web 2.0 apps during peak hours), those with a short or unpredictable lifetime, and those doing parallel computing using huge amounts of resources. Cloud computing also can be seen as a system consolidation approach to be used to handle mergers and acquisitions of companies - allowing moving "exotic" applications into the cloud and out of the datacenter.

Within this paper, we want to focus on technical aspects of cloud computing. We will not discuss business benefits from outsourcing certain IT operations to external providers nor discuss preconditions and implications of efficient datacenter operation. Instead we will outline options available to architects and developers when choosing one or the other cloud-computing environment.

1 How it all began - Related Work

Cloud Computing is a relatively new term. Amazon's Elastic Compute Cloud (EC2) was the initial offering of a cloud computing platforms that has been made available in beta status in 2006. Since then, many solutions have been announced but only a few of them are commercially available as of end of 2009. Nevertheless, Cloud Computing has a few predecessors - approaches, where compute power has been pooled across the network and made available on demand.

1.1 Grid Computing

Grid Computing is an approach for pooling compute resources from multiple administrative domains on order to carry for massively parallel computations, such as scientific computing and simulation. Grid computing efforts have been sponsored through research programs of the European Union as well as national research agencies.

Grid computing applications typically follow a "divide-and-conquer" strategy, splitting tasks in (large) sets of subtasks and carrying out those subtasks on multiple (sometimes thousands) of nodes on the grid. Grids may be established and used inside one organization or being spread across organizational boundaries forming so called "Virtual Organizations".

Grid computing is a form of distributed computing forming a "virtual supercomputer". Following the SPMD (Same Program Multiple Data) approach, grid applications often run the same program with varying datasets on multiple nodes in the grid. Those coarsely granular parallel applications can be found in the areas of distributed simulation, climate predication, drug discovery, economic forecast, and the like.

One early example for grid computing efforts is the SETI@HOME project, where users all over the world voluntarily donate idle cycles of their computers to help with analyzing recorded radio data and support the search for extraterrestrial intelligence. Another well-known grid computing project is the World Community Grid - Berkeley Open Infrastructure for Network Computing (BOINC). Using again idle time computing, this effort helps conducting protein-folding experiments in order to create more durable rice crops. Besides those "special purpose" grid initiatives, efforts have been made to establish standard tools and environments to send out jobs to multiple grid nodes and collect results. Most notably is Sun Microsystems's Grid Engine that forms a cluster computing environment out of standard workstations. The Globus toolkit (GTK) is another de facto standard for setting up grid computing environments. Standardization efforts of the Open Grid Forum (OGF) focus on compilation of WebService standards for usage on the grid.

In contrast to cloud computing, grid computing never considered the interactive end user being part of the scenario. It rather follows a model where jobs are being submitted through portals and results are being retrieved hours later in form of log files and collected program output. Also, grid computing did not solve the problem of integrating administrative domains. To enable interoperability across sites, grid administrators typically have to install certificates, enable ssh (secure shell) login, and setting up mutual trust relationships across all partners. Most importantly, in contrast to providing virtualized compute resources to many users simultaneously (as cloud computing does), grid computing rather focuses on creating a huge, distributed virtual supercomputer, which is managed in a batch-job-processing scheme.

1.2 Utility Computing

The term utility computing refers to packaging of IT resources (such as CPU, memory, network bandwidth, storage) into a metered service similar to traditional utilities (such as the telephone network). With low initial costs, utility computing relies on a pay-per-use billing model and allows quick reaction to changes in demand of IT services.

Utility computing is not a new concept but has a rather long history in the world of ultra-reliable but expensive mainframe computers. IBM typically sells computing capacity rather than physical processor and storage resources. Virtualization techniques have been in place on the mainframe since multiple decades. They provide the underpinnings for effectively sharing compute resources across multiple users and organizations, thus establishing the notion of compute resources as a utility.

Client/server computing and the PC marked a departure from the traditional world of mainframe computing. However, with the massive increase in computing capacity in the PC architecture, the establishment of huge, underutilized datacenters, and the advent of virtualization support in Intel's and AMD's CPUs, all the prerequisites for managing computation as a pay-per-use service have been in place at the beginning of the new century. Cloud computing extends the notion of utility computing by including the client side in the picture.

1.3 Autonomic Computing

The term autonomic computing initially has been coined by the IBM. It describes the ability of systems to be more self-managed. Self-management is seen as a promising approach to cope with the ever-increasing complexity of computing systems and infrastructures.

In an autonomic system, the human operator does not control the system directly. Instead, he defines general policies and rules that serve as an input for the self-management process. The following four functional areas have been defined for autonomic systems:

- Self-Configuration: Automatic configuration of components;
- Self-Healing: Automatic discovery, and correction of faults;
- Self-Optimization: Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements;
- Self-Protection: Proactive identification and protection from arbitrary attacks.

In order to be manageable, cloud computing infrastructures (often called "the fabric") have to follow autonomic computing principles. Moving workloads transparently from one compute node to another (with little or no blackout time) can be seen as a self-healing mechanism in the cloud. The ability to instantiate certain pre-configured machine images (as with Amazon EC2) or certain roles (as with Windows Azure) can be seen as a step towards self-configuration. Replication in space (redundancy) or time (re-execution) for both, computation and data is the typical approach for self-optimization and self-protection found in the cloud.

1.4 Dynamic Datacenter Alliance

With its Dynamic Data Center Alliance, Microsoft pursues similar goals like the autonomic computing initiative.

Microsoft's Dynamic Data Center Toolkit enables data center managers to optimize resources, improve visibility of IT assets and issues, meet service levels, and increase efficiencies—all while decreasing costs. The System Center Data Center Management solutions provide:

- Configuration management: Automated provisioning and updating with server consolidation through virtualization.
- End-to-end monitoring: Application and service-level monitoring with proactive platform monitoring.
- Server compliance: Configuration controls and reporting plus centralized audit of system security.
- Data protection and recovery: Backup and restore in addition to business continuity through server virtualization.

Most notably with Microsoft's approach is the attempt to form and define an ecosystem of partners including hosting companies, system integrators, hardware manufacturers, and software vendors who are working with Microsoft namely to offer differentiated value to customers based on the Dynamic Data Center Toolkit (for Hosters and Enterprises).

However, principles of the dynamic datacenter are applicable to the operation of cloud environments as well. In fact, VMware's vCloud initiative is exactly suggesting an ecosystem of partners (hosters) to build (private) clouds.

1.5 Hosting / Outsourcing

Just like cloud computing, traditional hosting/outsourcing offerings typically suggest the migration of line of business applications and related services, such as backup, from a company's data center onto an infrastructure owned, managed, and operated by an external provider. Typically, there are service-level-agreements (SLAs) associated with software and services operated by that provider.

However, there are two major differences between hosting and cloud computing: The deployment of applications on an off-premise hoster's infrastructure has to obey certain rules and practices. These may be time consuming. For test installation, development setups, or rapidly and suddenly changing applications these procedures may be just too inflexible. Cloud computing can shorten certain procedures. There are just fewer partner involved and the developer is basically able to manage his infrastructure on his own. This gives freedom but imposes also a burden.

The second difference between cloud computing and off-premise hosting is in the potential amount of savings in terms of operational cost, management (administrative) overhead, and even carbon footprint. While many hosters operate applications and services for multiple clients and thus are able to consolidate multiple workloads on fewer physical machines, the potential amount of sharing on the level of the hardware infrastructure is even bigger in the huge datacenters operated by cloud providers.

Cloud computing is a young field - with commercial offerings by some providers available for just over a year. Therefore, it is difficult to judge whether potential savings due to the economy of scale really will be reflected in the overall operational costs of clients. Depending on the size of an enterprise, cloud computing and traditional hosting will just fall in the same order of magnitude with respect to operational costs.

2 A Classification of Cloud Implementations

Cloud implementations can be sorted into three different categories according to their granularity and the level of abstraction provided to developers and clients.

As previously discussed, an initial motivation for cloud computing was the efficient use of hardware and server infrastructures within existing data centers. Typical units of abstraction

(building blocks for the cloud) on this level are servers, racks, storage, power supplies, etc. One major step from the traditional data center to the cloud was the introduction of virtualization techniques, leading to virtual compute and virtual storage (key/value, block storage). This level of service computing often is referred to as "Infrastructure-as-a-Service" (IaaS) or utility computing. Within this paper, we want to discuss Amazon Web Services and the VMware vCloud offering as representatives for the IaaS category.

Ease-of-use and developer efficiency are major motivations to consider cloud computing on a higher level of abstraction. In order to make a hosting infrastructure easy to use, one has to abstract away management and administration interfaces. This is achieved by introducing the concept of managed execution containers (or roles) and managed cloud data store. No longer is the developer nor the user responsible for maintenance, upgrades, or patching the software running in the cloud - but instead the cloud infrastructure (or fabric) will automatically perform these tasks. Developer efficiency is achieved by linking integrated development environments, such as Eclipse or Visual Studio directly with the execution environment on the cloud fabric. Obviously, there is a tradeoff between portability and ease-of-use: applications developed in a certain IDE most likely will only be able to run on the corresponding cloud fabric. This category of cloud computing often is referred to as "Platform-as-a-Service" (PaaS). Within this paper, we want to discuss Windows Azure and the Google AppEngine as prime candidates for this category.

Focusing on the usage of certain business applications, productivity applications, analytics applications, or even games, rather than on the developer efficiency is the "Software-as-a-Service" (SaaS) category of cloud computing. On this level of abstraction, it is not visible how a particular cloud application has been architected and implemented. The only interesting fact to the user is the application's feature set. Within this paper, we will very briefly touch Office Live and Salesforce.com's CRM as representatives for this category.

Finally, people distinguish private and public clouds. Many enterprises treat their data as valuable assets, which cannot be stored on the Internet. The notion of a private cloud has been coined to denote the applicability of cloud computing abstractions and implementation for the local, private datacenter. From a technical standpoint, requirements the public or a private cloud do not differ. We will briefly comment whether current commercial offerings for the cloud implementations discussed below do consider private and public clouds.

3 Amazon Web Services - IaaS

Since August 2006, Amazon Web Services (AWS) has provided an infrastructure web services platform in the cloud. By providing a suite of elastic IT infrastructure services, AWS enables access to compute power, storage, and other services. AWS operates on a pay-per-use basis, with no up-front expenses or long-term commitments. This makes AWS a cost-effective way to deliver applications to customers and clients. AWS operates on Amazon.com's global computing infrastructure. The term "cloud computing" initially has been coined for a few Amazon Web Services, namely the Elastic Compute Cloud (EC2) and the Simple Storage Service (S3).

Amazon Web Services is more than a collection of infrastructure services. It incorporates identity, payment, database, messaging, and other services. All AWS services are priced on a pay as you go model, with no up front expenses or long-term commitments.

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. EC2 can be categorized as an **Infrastructure-as-a-Service** (IaaS) offering. It allows configuring an Amazon Machine Instance (AMI) and loading it into the Amazon EC2 service. By running multiple AMIs simultaneously, EC2 allows to quickly scale capacity, both up and down, as computing requirements change.

Amazon Simple Storage Service (Amazon S3) is a simple web services interface that can be used to store and retrieve large amounts of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. Data on S3 is unstructured (blobs). There exists no interface to run structured queries or operations on the data on S3 directly from the client. Instead, SimpleDB adds an additional layer on S3 to provide such an interface.

With Amazon CloudFront, content delivery on the Web can be implemented. Even with thin clients, cloud services need a certain portion of user interface running on the client's desktop in order to be usable. CloudFront is the proposed solution for hosting those downloadable user interfaces - in form of a collection of JavaScript/Ajax code. Cloud Front integrates with other Amazon Web Services.

Amazon SimpleDB is a web service for running queries on structured data in real time. This service works in close conjunction with Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2), collectively providing the ability to store, process and query data sets in the cloud. Amazon SimpleDB provides the core functionality of a database—real-time lookup and simple querying of structured data.

Amazon Simple Queue Service (Amazon SQS™) is a reliable, highly scalable, hosted queue for storing messages as they travel between computers. By using Amazon SQS, developers can simply move data between distributed components of their applications that perform different tasks, without losing messages or requiring each component to be always available. Configuration rules for EC2 can be based on SQS attributes and dynamically start up or shutdown AMI instances as queue length vary. This allows for implementation of self-adaptive behavior in cloud services.

With EC2, S3, and SQS, Amazon has provided the first implementation for a public cloud. Public clouds are appealing for many use cases, where data security, privacy, and ownership are not of primary concern. This is true for use cases like scientific computing, simulation, image processing, video rendering, and even gaming. However, most enterprises view data as their primary asset. Private clouds - which are very much comparable to traditional hosting solutions (as discusses above), may be a promising approach to keep important data absolutely private.

Amazon Virtual Private Cloud (Amazon VPC) is an extension of the initial, purely public cloud. It provides a secure and seamless bridge between a company's existing IT infrastructure and the AWS cloud. Amazon VPC enables enterprises to connect their existing infrastructure to a set of isolated AWS compute resources via a Virtual Private Network (VPN) connection, and to extend their existing management capabilities such as security services, firewalls, and intrusion detection systems to include their AWS resources. Amazon VPC integrates today (beta status) with Amazon EC2, and will integrate with other AWS services in the future.

When developing for the Amazon cloud, one has to distinguish between APIs and tool support for deploying, managing and monitoring instances of Amazon Machine Images (AMIs) in the cloud and between APIs and tools for developing applications running on AMIs in the cloud. Being an IaaS solution, AWS tools mainly focus on managing AMIs for EC2 and setting up SimpleDB configurations for S3. Based on the Eclipse Web Tools Platform, the AWS Toolkit for Eclipse guides Java developers through common workflows and helps to configure Tomcat servers, run applications on Amazon EC2, and debug the software remotely through the Eclipse IDE. There exists also a .NET library that wraps AWS APIs and allows management of AMI instances (preferably running Windows and IIS) from .NET client programs. Similar wrapper libraries exist for Perl, PHP, Ruby and Python.

In the remainder of the section we will take a closer look on EC2, S3, and SQS.

3.1 The Elastic Compute Cloud (EC2)

Amazon EC2 presents a virtual computing environment, that allows to use web service interfaces to launch instances with a variety of operating systems, to load them with custom application environments, to manage network's access permissions, and to run images on as many or few systems as desired. Amazon has reached licensing agreements with Microsoft and IBM in order to be able to support the Windows operating system (with or without SQL server) on Amazon Machine Images as well as the Lotus suite of applications. In addition, EC2 provides pre-configured Linux and OpenSolaris instances.

To use Amazon EC2, the following steps have to be performed:

- Creation of an Amazon Machine Image (AMI) containing applications, libraries, data and associated configuration settings. Alternatively, pre-configured, template images can be run immediately.
- Uploading the AMI into Amazon S3. Amazon EC2 provides tools to store AMI images in a safe and reliable manner.
- Amazon EC2 web services can be used to configure security and network access.
- Web service APIs or specialized management tools can be used to start, terminate, and monitor as many instances of an AMI as needed. Amazon EC2 distinguishes compute instances (virtual machines) of various sizes (types).
- Definition whether the service need to run in multiple locations, utilize static IP endpoints, or attach persistent block storage to your instances.

EC2 supports three standard types of compute instances. Unit of granularity is the so-called EC2 compute unit, which is supposed to be the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. The EC2 standard instance types are; small (1 virtual core with 1 EC2 Compute Unit, 32 bit, 1.7 GB memory), medium (2 virtual cores with 2 EC2 Compute Units each, 64 bit, 7.5 GB memory), and extra large ((4 virtual cores with 2 EC2 Compute Units each, 64 bit, 15 GB memory).

In addition, EC2 offers specialized compute instances for high computing needs and high memory needs.

3.2 The Simple Storage Service (S3)

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers. S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. The service aims to maximize benefits of scale and to pass those benefits on to developers.

Amazon S3 is built with a minimal feature set.

- S3 supports operations to write, read, and delete objects containing from 1 byte to 5 gigabytes of data each. The number of objects a client can store is unlimited.
- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
- A bucket can be located in the United States or in Europe. All objects within the bucket will be stored in the bucket's location, but the objects can be accessed from anywhere.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
- S3 uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.
- S3 is intended to be flexible so that protocol or functional layers can easily be added. The default download protocol is HTTP. A Bit Torrent protocol interface is provided to lower costs for high-scale distribution.

Data in S3 is backed by the AWS service level agreement (SLA) that promises to use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage of at least 99.9% during any monthly billing cycle. In the event Amazon S3 does not meet the Service Commitment, users are eligible to receive a service credits.

The basic units of abstraction for S3 are Buckets, Objects and Keys.

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata and can range in size from 1 byte to 5 gigabytes. The data portion is opaque to Amazon S3. The metadata is a set of name-value pairs that describe the object. The developer can specify custom metadata and standard HTTP metadata, such as Content-Type.

Objects are uploaded in buckets. There is no limit to the number of objects that can be stored in a bucket. The bucket provides a unique namespace for the management of objects contained in the bucket. Because the namespace for bucket names is global, each developer can own up to 100 buckets at a time. The creator of a bucket becomes its owner. Amazon charges for storing objects in buckets and transferring objects in and out. It is not possible to modify or append data to an existing object. Amazon S3 either replaces an existing object or does nothing to it at all.

A key is the unique identifier for an object within a bucket. In simple terms, the key is the bucket's name. Every object has exactly one key. Together, a bucket name and a key uniquely identify an object in Amazon S3. An object in S3 can be accessed by a combination

of the service endpoint, bucket name, and key. For example, in http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl, "doc" is the bucket name and "2006-03-01/AmazonS3.wsdl" is the key.

By default, Amazon S3 stores objects with "private" access control grants that allow only the bucket owner to read the object later. However, using access control instructions, flexible access grants can be specified. One of the most straightforward and common alternatives to the default access control is to set an object to be publicly readable. This means that any Internet user can download the object via a simple standard HTTP URL. For publicly readable objects, Amazon S3 bills the bucket owner for all data downloads. To make an object publicly readable, include "x-amz-acl: public-read" in the HTTP header of a PUT request.

3.3 The Simple Queuing Services (SQS)

Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers. By using Amazon SQS, developers can simply move data between distributed components of their applications that perform different tasks, without losing messages or requiring each component to be always available. Amazon SQS makes it easy to build an automated workflow, working in close conjunction with the Amazon Elastic Compute Cloud (Amazon EC2) and the other AWS infrastructure web services. Amazon SQS works by exposing Amazon's messaging infrastructure as a web service. Any computer on the Internet can add or read messages without any installed software or special firewall configurations. Components of applications using Amazon SQS can run independently, and do not need to be on the same network, developed with the same technologies, or running at the same time.

Developers can create an unlimited number of Amazon SQS queues with an unlimited number of messages.

- A queue can be created in the United States or in Europe. Queue names and message stores are independent of other regions.
- The message body can contain up to 8 KB of text in any format.
- Messages can be retained in queues for up to 4 days.
- Messages can be sent and read simultaneously.
- When a message is received, it becomes "locked" while being processed. This keeps other computers from processing the message simultaneously. If the message processing fails, the lock will expire and the message will be available again. In the

case where the application needs more time for processing, the “lock” timeout can be changed dynamically via the ChangeMessageVisibility operation.

- Developers can access Amazon SQS through standards-based SOAP and Query interfaces.
- Developers can securely share Amazon SQS queues with others. Queues can be shared with other AWS accounts and Anonymously. Queue sharing can also be restricted by IP address and time-of-day.

Amazon SQS can be used with EC2, as well as S3 and SimpleDB, to make applications more flexible and scalable. A common use case is to create an integrated and automated workflow, where multiple components or modules need to communicate with each other, but can’t all process the same amount of work simultaneously. In this case, SQS queues carry messages to be processed in an orderly fashion by the user’s application running on Amazon EC2 instances. The Amazon EC2 instances can read the queue, process the job, and then post the results as messages to another SQS queue (possibly for further processing by another application). Because Amazon EC2 allows applications to scale up and down dynamically, application developers can easily vary the number of compute instances based on the amount of work in the SQS queues, to ensure that jobs are executed in a timely manner.

4 VMware vCloud - IaaS

Virtualization was first developed in the 1960s to partition large, mainframe hardware for better hardware utilization. Today, computers based on x86 architecture are faced with the same problems of rigidity and underutilization that mainframes faced in the 1960s. Virtualization was effectively abandoned during the 1980s and 1990s when client-server applications and inexpensive x86 servers and desktops led to client/server architectures for distributed computing. The broad adoption of Windows and the emergence of Linux as server operating systems in the 1990s established x86 servers as the industry standard. In 1999, VMware introduced virtualization to x86 systems to address many of these challenges and transform x86 systems into a general purpose, shared hardware infrastructure that offers full isolation, mobility and operating system choice for application environments.

Virtualization is one key ingredient to cloud computing. Virtualization may dramatically improve the efficiency and availability of resources and applications in an organization's datacenters. However, virtualizing a single physical computer or even multiple computers in one datacenter is just the beginning. You can build an entire virtual infrastructure, scaling

across hundreds of interconnected physical computers and storage devices with VMware vSphere, a proven virtualization platform uses as the foundation for building private and public clouds. With appropriate management tools, there is no need to assign servers, storage, or network bandwidth permanently to each application. Instead, hardware resources are dynamically allocated when and where they're needed within an organization's private cloud. Connecting this private cloud to a public cloud to create a hybrid cloud, giving businesses the flexibility, availability and scalability they need to thrive.

With virtualization offerings on the infrastructure level (compute & networking resources), VMware has taken a different approach to cloud computing than other cloud service providers. Instead of focusing on one particular implementation of a public cloud, VMware has specified a cloud computing API called vCloud. Services for vCloud have to be packaged as virtual appliances (basically virtual machine images) in a particular format. By standardizing the so-called Open Virtualization Format (OVF) it is possible to run virtual machines (or cloud services in VMware's terminology) on multiple cloud computing infrastructures, thus achieving portability and avoiding vendor locking. As always, this portability comes with a tradeoff - VMware's vCloud operates on the IaaS level and does not (yet?) focus on integration with existing development tools.

The vCloud API enables enterprises to build internal clouds based on VMware's technology stack. As part of their external cloud offerings, service providers can build **Infrastructure-as-a-Service** portals with a standard, consistent interface. ISVs can package existing or new applications, which can then be deployed in internal or external clouds or be federated between clouds.

The vCloud API is an interface for providing and consuming virtual resources in the cloud. It enables deploying and managing virtualized workloads in internal or external clouds as well as interoperability between clouds. Such workloads packaged as vApps (or virtual appliances) are pre-built software solutions optimized for the cloud. They consist of multiple virtual machines that are bundled and maintained as a single entity. The vCloud API enables the upload, download, instantiation, deployment and operation of vApps. The concept of a vApp is comparable to the Amazon Machine Image (AMI). However, there are differences: vApps can be comprised of multiple virtual machine images - and they can be combined with virtual networking resources, effectively enabling layer-2 connectivity across virtual machines.

The vCloud API is an open RESTful, pure virtual, API that supports multi-tenancy. It uses Open Virtualization Format (OVF) as a unit of distribution and storage for vApps. Because these vApps are uploaded, downloaded, and stored in OVF package form, the vCloud API

supports compatibility and deployment across a wide variety of applications. The vCloud API spans multiple vCenter Servers to provide available virtual resources. This offers immense scalability needed for peak loads or disaster recovery in cloud environments.
The vCloud API has been first published in September 2009 and is currently in Technology Preview mode.

4.1 vCloud Express

vCloud Express is an Infrastructure-as-a-Service (IaaS) offering delivered by VMware service provider partners. It provides reliable, on-demand, pay-as-you-go infrastructure that ensures compatibility with internal VMware environments and with VMware Virtualized services. This class of service allows IT to reduce both the capital expenditures and resource challenges associated with the fluctuating infrastructure requirements of development teams. Developers are able to use the vCloud Express service at their convenience to address various infrastructure and programming needs such as experimentation, prototyping and testing.
The list of service providers for vCloud Express currently includes hosting.com, bluelock, logica, melbourne IT, terremark – some of the offering cloud services in (limited) beta mode.

5 Google AppEngine - PaaS

Google App Engine allows running web applications on Google's infrastructure. This effectively allows clients to build their own **Software-as-a-Service** applications. App Engine includes the following features:

- 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 a client 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

Applications can run in one of two runtime environments: the Java environment, and the Python environment. Each environment provides standard protocols and common technologies for web application development. Applications can be made accessible either from a client's domain name (using Google Apps) or through Google's reserved domain

called appspot.com. Applications can be shared with the world, or within an organization. App Engine supports integrating an application with Google Accounts for user authentication.

Google App Engine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language—or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features a dedicated Python runtime environment, which includes a fast Python interpreter and the Python standard library. The Java and Python runtime environments are built to ensure that applications run quickly, securely, and without interference from other apps on the system.

Applications run in a secure environment that provides limited access to the underlying operating system. These limitations allow App Engine to distribute web requests for the application across multiple servers, and start and stop servers to meet traffic demands. The sandbox isolates an application in its own secure, reliable environment that is independent of the hardware, operating system and physical location of the web server.

Examples of the limitations of the secure sandbox environment include:

An application can only access other computers on the Internet through the provided URL fetch and email services. Other computers can only connect to the application by making HTTP (or HTTPS) requests on the standard ports.

An application cannot write to the file system. An app can read files, but only files uploaded with the application code. The app must use the App Engine datastore, memcache or other services for all data that persists between requests.

Application code only runs in response to a web request, a queued task, or a scheduled task, and must return response data within 30 seconds in any case. A request handler cannot spawn a sub-process or execute code after the response has been sent.

App Engine costs nothing to get started. All applications can use up to 500 MB of storage and enough CPU and bandwidth to support an efficient app serving around 5 million page views a month, absolutely free. When you enable billing for your application, your free limits are raised, and you only pay for resources you use above the free levels.

5.1 The Java Runtime Environment

Applications for the Java runtime environment can be developed using common Java web development tools and API standards. Applications interact with the environment using the

Java Servlets standard, and can use common web application technologies such as Java Server Pages (JSPs).

The Java runtime environment uses Java 6. The App Engine Java SDK supports developing apps using either Java 5 or 6. The restrictions of the sandbox environment are implemented in the JVM. An application can use any JVM byte code or library feature, as long as it does not exceed the sandbox restrictions. For instance, byte code that attempts to open a socket or write to a file will throw a runtime exception. Another severe restriction concerns the creation of additional threads, which is not supported by App Engine.

Applications access most App Engine services using Java standard APIs. For the App Engine datastore, the Java SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces. Applications can use the JavaMail API to send email messages with the App Engine Mail service. The java.net HTTP APIs access the App Engine URL fetch service. App Engine also includes low-level APIs (such as datastore, memcache, URL fetch, images, Google AccountAPIs) for its services to implement additional adapters, or to use directly from the application.

Typically, Java developers use the Java programming language and APIs to implement web applications for the JVM. With the use of JVM-compatible compilers or interpreters, it is also possible to use other languages to develop web applications, such as JavaScript, Ruby, or Scala. The units of deployment for applications on App Engine are Java archives (jar-files).

5.2 The Python Runtime Environment

With App Engine's Python runtime environment, you can implement your app using the Python programming language, and run it on an optimized Python interpreter. App Engine includes rich APIs and tools for Python web application development, including a feature rich data modeling API, an easy-to-use web application framework, and tools for managing and accessing your app's data. You can also take advantage of a wide variety of mature libraries and frameworks for Python web application development, such as Django.

The Python runtime environment uses Python version 2.5.2. Additional support for Python 3 is being considered for a future release. The Python environment includes the Python standard library. Of course, not all of the library's features can run in the sandbox environment. For instance, a call to a method that attempts to open a socket or write to a file will raise an exception. For convenience, several modules in the standard library whose core features are not supported by the runtime environment have been disabled, and code that imports them will raise an error.

Application code written for the Python environment must be written exclusively in Python. Extensions written in the C language are not supported. The Python environment provides rich Python APIs for the datastore, Google Accounts, URL fetch, and email services. App Engine also provides a simple Python web application framework called webapp to make it easy to start building applications.

5.3 The Datastore

App Engine provides a distributed data storage service that features a query engine and transactions. However, the App Engine datastore is not like a traditional relational database. Data objects, or "entities," have a kind and a set of properties (They are basically a typed set of attribute-value pairs). Queries can retrieve entities of a given kind filtered and sorted by the values of the properties. Property values have to be of a supported property value types.
Datastore entities are "schema-less." The structure of data entities has to be provided by and enforced by the application code. The Java JDO/JPA interfaces and the Python datastore interface include features for applying and enforcing structure within an application. Applications can also access the datastore directly to apply as much or as little structure as it needs.
The datastore is strongly consistent and uses optimistic concurrency control. An update of an entity occurs in a transaction that is retried a fixed number of times if other processes are trying to update the same entity simultaneously. An application can execute multiple datastore operations in a single transaction, which either all succeed or all fail, ensuring the integrity of your data. The datastore implements transactions across its distributed network using "entity groups." A transaction manipulates entities within a single group. Entities of the same group are stored together for efficient execution of transactions. An application can assign entities to groups when the entities are created.

5.4 Development Workflow

The App Engine software development kits (SDKs) for Java and Python each include a web server application that emulates all of the App Engine services on a local computer. Each SDK includes all of the APIs and libraries available on App Engine. The web server also simulates the secure sandbox environment, including checks for attempts to access system resources disallowed in the App Engine runtime environment.
Each SDK also includes a tool to upload applications to AppEngine. Once an application's code, static files and configuration files are created, the tool can be used to upload the data. The tool prompts for a Google account email address and password. New major release of

an application that is already running on AppEngine can be uploaded as a new version. The old version will continue to serve users until the developer switches to the new version. New version on App Engine can be tested while the old version is still running.

The Java SDK runs on any platform with Java 5 or Java 6. The SDK is available as a Zip file. With the Eclipse development environment, the Google Plug-in for Eclipse can be used to create, test and upload App Engine applications. The SDK also includes command-line tools for running the development server and uploading an application.

The Python SDK is implemented in pure Python, and runs on any platform with Python 2.5, including Windows, Mac OS X and Linux. The SDK is available as a Zip file, and installers are available for Windows and Mac OS X.

The Administration Console is the web-based interface for managing your applications running on App Engine. It can be used create new applications, configure domain names, change which version of an application is live, examine access and error logs, and browse an application's datastore.

6 Windows Azure platform - PaaS

The Windows Azure platform is a set of cloud computing services that can be used together or independently. The Windows Azure platform enables developers to use existing skills and familiar tools to develop cloud applications. It is based on a pay as you go cost model and allows to dynamically adjusting compute and storage resources available to applications without adding application complexity. Windows Azure has been first announced in Fall 2008 and is commercially available since November 2009.

The Windows Azure SDK can be used with Visual Studio to develop applications for Windows Azure. Those applications can be developed, tested and debugged on the Local Fabric - a runtime environment on the desktop that emulates the behavior of Windows Azure in the cloud. Windows Azure clearly falls in the category of a **Platform-as-a-Service** (PaaS). However, a particular strength of Windows Azure consists in the set of mechanisms for interconnecting applications in the cloud with existing on-premise applications hosted locally.

The Windows Azure platform consists of three components:

- **Windows Azure** provides a Windows-based scalable environment with compute, storage, hosting, and management capabilities. It links to on-premises applications with secure connectivity, messaging, and identity management.
- **SQL Azure** is a full relational database in the cloud. Its implementation is based on SQL server.

- **AppFabric** provides Network Services for the Cloud. It offers identity management and firewall friendly messaging to protect your assets by enabling secure connectivity and messaging between on-premises IT applications and cloud-based services. (AppFabric integrates the previously known .NET services).

6.1 Windows Azure

Windows Azure is a cloud services operating system that serves as the development, service hosting and service management environment for the Windows Azure platform. Windows Azure provides developers with on-demand compute and storage to host, scale, and manage web applications on the Internet through Microsoft datacenters. Windows Azure is a flexible platform that supports multiple languages and integrates with your existing on-premises environment. To build applications and services on Windows Azure, developers can use their existing Microsoft Visual Studio expertise. In addition, Windows Azure supports popular standards and protocols including SOAP, REST, XML, and PHP.

6.1.1 Compute Service

The Windows Azure compute service is based on Windows APIs and built from one or more roles. A role defines a component that may run in the execution environment. Within Windows Azure, a service may run one or more instances of a role. Windows Azure supports web roles and worker roles. Developers can create Web applications, applications that run as independent background processes, or applications that combine the two.

A web role is customized for web application programming, as supported by IIS 7 and ASP.NET. Web roles run in integrated pipeline mode on an Internet Information Services (IIS) 7.0 Hosted Web Core. A worker role is useful for generalized development, and may perform background processing for a web role. Besides running managed .NET code, it also supports unmanaged code. In addition, Windows APIs can be used to create multiple threads in Windows Azure services. Services running on Windows Azure can use technologies such as ASP.NET and Silverlight to expose remote client-side user interfaces.

A service deployment is configured with two XML files: a service definition file and a service configuration file. The service definition file defines the roles available to a service, specifies the service endpoints, and establishes configuration parameters for the service. The service definition settings cannot be changed after a service is deployed. The service configuration file specifies values for the configuration settings for one or more role instances within the

running service. You can dynamically modify service configuration settings without redeploying the service.

6.1.2 Storage Service

The Windows Azure storage services provide persistent, durable storage in the cloud. The fundamental storage services include blob, queue, and table service. The Windows Azure SDK offers a REST API and a managed API for working with the storage services. You may access the storage services from within a service running in Windows Azure or directly over the Internet from any application that can send and receive data over HTTP/HTTPS.
To let its customers create, configure, and monitor applications, Windows Azure provides a browser accessible portal. A customer provides a Windows Live ID, then chooses whether to create a hosting account for running applications, a storage account for storing data, or both. An application is free to charge its customers in any way it likes: subscriptions, per-use fees, or anything else.

6.1.3 Development Environment

The Windows Azure SDK provides a simulated environment for developing and testing services on the developer's local computer. The development environment includes local storage services that simulate the Blob, Queue, and Table services available in Windows Azure. The development storage UI provides a means to view the status of the local storage services and to start, stop, and reset them.
The development fabric simulates Windows Azure on the user's local computer. The development fabric user interface provides a means to view service deployments and role instances, start and stop a service, and test logging levels. The CSPack command-line tool prepares a service for deployment, either to the development fabric or to the Windows Azure fabric. Finally, the CSRun command-line tool runs a service in the development fabric.

6.2 SQL Azure

Microsoft SQL Azure Database is a cloud-based relational database service built on SQL Server technologies. It provides a highly available, scalable, multi-tenant database service hosted by Microsoft in the cloud. SQL Azure Database helps to ease provisioning and deployment of multiple databases. Developers do not have to install, setup, patch or manage any software. High availability and fault tolerance is built-in and no physical administration is required. SQL Azure Database supports Transact-SQL (T-SQL).

The benefits of using SQL Azure include manageability, high availability, scalability, a familiar development model, and a relational data model. SQL Azure offers the scale and functionality of an enterprise data center without the administrative overheads that are associated with on-premise instances of SQL Server. This self-managing capability enables organizations to provision data services for applications throughout the enterprise without adding to the support burden of the central IT department or distracting technology-savvy employees from their core tasks in order to maintain a departmental database application.

SQL Azure replicates multiple redundant copies of data to multiple physical servers to maintain data availability and business continuity. In the case of a hardware failure, SQL Azure provides automatic failover to optimize availability for your application. A key advantage of SQL Azure is the ease with which you can scale your solution. After partitioning your data, the service scales as your data grows.

When developers create on-premise applications that use SQL Server, they use client libraries that use the tabular data stream (TDS) protocol to communicate between client and server. SQL Azure provides the same TDS interface as SQL Server so that you can use the same tools and libraries to build client applications for data that is stored in SQL Azure. SQL Azure will seem very familiar to developers and administrators because data is stored in SQL Azure just like it is stored in SQL Server, by using Transact-SQL. Conceptually similar to an on-premise instance of SQL Server, a SQL Azure server is logical group of databases that acts as an authorization boundary.

Within each SQL Azure server, you can create multiple databases that have tables, views, stored procedures, indices, and other familiar database objects. SQL Azure servers and databases are virtual objects that do not correspond to physical servers and databases. By insulating developers from the physical implementation, SQL Azure enables them to spend time on your database design.

6.2.1 SQL Azure Architecture

From an architectural perspective, there are four distinct layers of abstraction that work together in SQL Azure to provide a relational database for client applications to use: the client layer, the services layer, the platform layer, and the infrastructure layer. The client layer resides closest to an application, and is used by the application to communicate directly with SQL Azure. The client layer can reside on-premise in your datacenter or be hosted in Windows Azure.

The services layer functions as a gateway between the client layer and the platform layer, where the data resides. The services layer provides three functions: provisioning, billing and

metering, and connection routing. It provisions the databases that you specify with a Windows Azure Platform account. The billing and metering aspect of the services layer enables multi-tenant support by providing monitoring and billing for database usage based on individual Windows Azure Platform accounts. SQL Azure is built on a scalable platform involving numerous physical servers; this layer handles all the connections routing between your application and the physical servers where your data resides.

The platform layer includes the physical servers and services that support the services layer. The platform layer consists of many instances of SQL Server, each of which is managed by the SQL Azure fabric. The SQL Azure fabric is a distributed computing system composed of tightly integrated networks, servers, and storage. It enables automatic failover, load balancing, and automatic replication between physical servers. Management services monitor the health of individual servers and enable automated installation of service upgrades and software patches.

The infrastructure layer represents the IT administration of the physical hardware and operating systems that support the services layer.

6.3 Windows Azure AppFabric

AppFabric provides the foundation for executing .NET applications in Windows Azure (Windows Azure AppFabric). However, it may also play the role of an application server on traditional Windows Server systems (Windows Server AppFabric). With that respect, AppFabric is comparable to the well-known concept of a Java-application server. Besides being a Windows application server, AppFabric integrates the distributed cache "velocity" as well as the Windows Azure access control services and the .NET Service Bus.

The Windows Azure platform AppFabric provides secure connectivity-as-a-Service to help developers to bridge cloud, on-premises, and hosted deployments. One can use AppFabric Service Bus and AppFabric Access Control to build distributed and federated applications as well as services that work across network and organizational boundaries. From simple eventing scenarios to complex protocol tunneling, AppFabric Service Bus gives developers the flexibility to choose how their applications communicate, and to address the challenges presented by firewalls, NATs, dynamic IP, and disparate identity systems. AppFabric Access Control enables simple, secure authorization for RESTful web services that federate with a variety of identity providers.

AppFabric Access Control Service implements authentication and authorization mechanisms for web applications and services. A simple and familiar programming model keeps application code clean and allows transition to the declarative model of rules and claims.

These rules can be easily configured to meet your applications' current and future access control needs. AppFabric Access Control is based on a claims-based authorization model, which alleviates the need to develop and support a variety of identity providers and architectures. The AppFabric Service Bus allows to expose application's or service's functionality across a variety of network-related constraints. Once AppFabric Service Bus has established connectivity among applications, it provides flexibility on how applications can communicate with each other. Developers are enabled to build solutions with various communication patterns such as relayed, buffered, bidirectional, publish-subscribe, multicast, streaming and direct-connect. AppFabric Service Bus provides each service a stable Internet-accessible Uniform Resource Identifier (URI) that can be accessed by any authorized client application. Powered by AppFabric Access Control, AppFabric Service Bus is able to control services accessibility with heterogeneous identity systems.

6.4 Additional Online Services

Not directly part of the Windows Azure platform are three additional sets of services, namely the Bing Services, the Live Services SDK, and the Microsoft Advertising Services. All three classes of services are of particular interest when developing frontends for services hosted on the cloud (the Windows Azure platform). Part of the Live Services is the Silverlight Streaming SDK - which can be seen as a novel approach to extend cloud services with rich-internet-application-style user interfaces.

7 Salesforce.com - SaaS / PaaS

Salesforce.com was founded in 1999 as a company specializing in **Software-as-a-Service (SaaS)** offering for Customer Relationship Management (CRM). It was really one of the pioneers of the SaaS model of distributing software on the Internet. Salesforce.com promises availability of 99.9% for its cloud infrastructure. This figure is comparable with the service levels reached by the competitors. However, the trust.salesforce.com site is truly unique. It shows both recent and historical performance plus uptime data for the Salesforce.com cloud infrastructure.

Salesforce.com's CRM solution is broken down into several modules: Sales, Service & Support, Partner Relationship Management, Marketing, Content, Ideas and Analytics. In order to ease customization and allow for extension of the CRM SaaS solution, Salesforce.com has built up AppExchange. Launched in 2005, AppExchange is a directory of applications built for Salesforce by third-party developers, which users can purchase and

add to their Salesforce environment. As of September 2008, there are over 800 applications available from over 450 ISVs. Applications include services from Google, Constant Contact, Vertical Response, Good Data, and Box.net. The Force.com platform is a further generalization of the initial SaaS solution.

Salesforce users can customize their CRM application on the platform- or the tab-level. In the system, there are tabs such as "Contacts", "Reports", and "Accounts". Each tab contains associated information. For example, "Contacts" has standard fields like First Name, Last Name, and Email. Customization can be done on each tab, by adding user-defined custom fields. Customization can also be done at the "platform" level by adding customized applications to a Salesforce.com instance, that is adding sets of customized / novel tabs for specific vertical- or function-level (Finance, Human Resources, etc) features.

In addition to the web interface, Salesforce offers a Web Services API that enables integration with other systems and has wrapper libraries for programming languages such as Java, VB.NET, C#, and other .NET languages. In April 2009, Salesforce released a slimmed down version of their application for subscribers with Blackberry, iPhone, and Windows mobile devices.

7.1 Force.com

By providing **Platform-as-a-Service (PaaS)** product, Salesforce.com opens their cloud-computing infrastructure for other SaaS applications. The Force.com platform allows external developers to create add-on applications that integrate into the main Salesforce application and are hosted on salesforce.com's infrastructure. These applications are built using Apex (a proprietary Java-like programming language for the Force.com Platform) and Visualforce (an XML-like syntax for building user interfaces in HTML, AJAX or Flex). Apex has strong ties with the Force.com persistence layer, the database. Visualforce-designed user interfaces link to logic and workflow intelligence implemented in Apex Code. Visualforce relies on the concept of mashups for integration of user-interface components on the client side.

The Force.com cloud computing platform includes a number of connectors built on top of the Force.com Web Services API that help to reduce the effort associated with integrating cloud computing applications. It supports integration with enterprise systems, including SAP, Oracle, Microsoft, and other third-party solutions via native ERP connectors for Oracle and SAP. Desktop connectors provide for integration between cloud computing applications and Microsoft Outlook, Lotus Notes, Microsoft Excel, and Microsoft Word.

The Force.com IDE is a client application for creating, modifying, testing and deploying Force.com applications written in Apex. It is based on the Eclipse platform and includes

components such as a unit test (Apex TestRunner), a source code control system and a schema explorer for the database and metadata components, of the system under development. The IDE also manages the deployment of cloud applications from the development environment onto a staging and production environment on the Force.com cloud.

However, the Apex- and Visualforce-based set of tools and interfaces available for developing, deploying, and managing software on the Force.com cloud is proprietary. In contrast to other PaaS approaches which provide a local runtime environment (such as Google's AppEngine and Microsoft's Windows Azure), applications developed for Force.com are tightly linked to the database model inherent to Salesforce.com and restricted to be executed on this platform only.

Adobe Flash Builder for Force.com is a new approach towards cloud-based rich Internet applications (RIAs). The tool - which is currently in developer preview status - supports end-user deployment through the browser via the Adobe Flash Player, or directly to the desktop using the Adobe AIR runtime. Applications constructed using Flash Builder will run online or offline while taking advantage of the security, scalability, and reliability of Force.com.

Having the data persistency layer tightly integrated into the cloud-computing platform surely is one of the key benefits of the force.com environment. This concept is particularly interesting, as it allows for integration of elaborate security concepts such as object / record level access control and role-based access control into the cloud.

7.2 Force Database - the persistency layer

Data persistence lies at the heart of many applications. The Force.com platform provides a powerful and intuitive data persistence layer, referred to as the Force.com Database. The database provides not only a mechanism for creating persistent objects, but also a way of automatically generating a user interface around these objects. Reporting, tagging and much additional related functionality can also be added to applications, all out-of-the-box Force.com platform features.

The developer can create, configure and deploy persistent objects using the web-based Force.com Setup menu environment. However, database services are also tightly integrated with the Apex programming language, which has a dedicated syntax for invoking searches and iterating over results.

The Force.com Database uses objects to store data. Objects contain the functionality you expect in a table and more, but you should be aware that the difference also points to a difference in functionality. The object comprises a number of fields. Objects can be related to

other objects, with relationship fields mapping records in one object to records in another. All attributes of an object are described with metadata. Information is stored in records of the object.

7.3 Data Security

The Force.com platform provides a range of security features to ensure that your organization and its data are protected. This security foundation ranges from user authentication features such as SAML through to IP range restrictions on logons, session security and auditing.

Administrative permissions are used to grant or deny access to some areas of the Force.com platform functionality for particular sets of users. Profiles are the primary security feature here, and they provide a way to group users together for easier administration. A user can only belong to a single profile, and only administrators can change profile membership for a user. These profiles determine which objects users in that profile has access to. For example, you can determine which combination of read, create, edit or delete permissions are assigned to the profile for each object. You can also determine which fields within an object are accessible to that profile.

These permissions are fairly coarse - you have permission for a type of access to all the records in an object, or to a set of fields. However, the Force.com platform provides a way to implement different access to different data records stored in a single object. This type of security is based on individual rows of data - which opens up a host of interesting data security options.

A core concept in this type of security is record ownership. The owner of a record has all privileges for that record—including the ability to share the record with other users or even transfer ownership of the record. A single user can own a record, or a group of users (identified by a collection of users called a queue). The platform provides a rich set of features to determine whom a record should be shared with. For example, one can set up public groups (perhaps all bloggers in a company) and grant this group access to a record. One may also create role hierarchies, which represent an organization's reporting structure. The sharing features of the platform can be set up so that all superior roles have all the sharing permissions granted to all roles below them in the hierarchy for example.

8 Microsoft Office Live - SaaS

Microsoft Office Live is a set of Internet-based services designed for consumers and small

businesses interested in creating a website or storing and sharing documents online. As of 2009, it consists of two services, Office Live Workspace and Office Live Small Business. Office Live primarily is a document store. It supports a number of simple operations on documents in the cloud (such as creation of schedules). From that standpoint, Office Live can be characterized as a **Software-as-a-Service (SaaS)** offering. However, for complex operation such as document editing, it follows the "Software + Services" philosophy coined by Microsoft and relies on applications installed locally on the user's client computer.

Office Live Workspace requires web access and a compatible browser. Use of a workspace can be enhanced by installing Silverlight, a plug-in that allows for execution of rich user interfaces written in .NET and makes it easier to upload multiple documents and collaborate with others on a workspace. In order for workspaces to be accessed directly from Office, users of Word, Excel and PowerPoint must install an Office Live Update. Files in Office Live cannot be edited from within workspace, but clicking on "edit" will open them up in Microsoft Office. The workspace does not offer offline collaboration — instead documents are "checked out" and "checked in". However, the Office Live integration with SharedView allows for real-time screen sharing as a means of online collaboration.

Office Live Small Business is an Internet-based service designed to assist non-technical users with the creation of a professional-looking website. Office Live Small Business provides access to free online web design tools and templates for non-technical users' website development. Site Designer is a product feature used for customizing page layouts, colors, navigation, and other site elements. Users can also add modules such as PayPal buttons, blogs, and calendars to pages. Within Office Live Small Business, domain names may be selected and registered. Customers who already have a domain name with another provider can redirect it to Office Live Small Business. The Contact Manager is designed to organize customer information, contact histories, and sales information in one place and make it accessible via the web to their entire organization. Document Manager and Team Workspace are additional collaboration tools in Office Live.

8.1 LiveMesh.com

Live Mesh is another Software + Services platform from Microsoft. It enables PCs and other devices to ‘come alive’ by making them aware of each other through the Internet, enabling individuals and organizations to manage, access, and share their files and applications seamlessly on the Web and across their world of devices. Live Mesh builds on the cloud storage, management, service and provisioning and computational fabric that other Microsoft Live services use. On top of that, Live Mesh uses the same identity, synchronized storage

and connectivity services that Microsoft uses for other Live offerings. The “platform” services (a.k.a. the “developer stack”) include the new Mesh Framework, as well as both a cloud and a client software run-time Mesh Operating Environment (MOE). Live Mesh “experiences” from Microsoft and third-party providers will build on top of these layers.
The Live Mesh software, called Mesh Operating Environment (MOE), is currently available for Windows, Windows mobile and Mac OS X. It can be used to create and manage the synchronization relationships between devices and data. Live Mesh also includes a cloud storage component, called Live Desktop. It is an online storage service that allows synchronized folders to be accessible via a website. Live Mesh also provides remote desktop software called Live Mesh Remote Desktop that can be used to remotely connect to and manage any of the devices in a synchronization relationship. Live Mesh Remote Desktop allows users to control their devices from the Live Mesh application, as well as from any other Internet connected PC.

The extension of the **Software-as-a-Service (SaaS)** model to a Software + Service platform has up- and downsides. On one hand side, functionality, such as transparent access to the file system, change tracking, and write notification would not be possible without running client software natively. On the other hand side, ease of use and portability suffer dramatically, if users have to install additional software prior to being able to use a SaaS offering.

9 Google Apps - SaaS

Google Apps is a **Software-as-a-Service (SaaS)** offering from Google. It features several Web applications with similar functionality to traditional office suites. Usually, those applications would be accessible through docs.google.com for a single user. Google Apps allows clients to use their own domain name with the service and to customize the interface to reflect the branding of that client. Users can simply log in to the service to access their files and the tools to manipulate them. The offerings include communication tools (Gmail, Google Talk, and Google Calendar), productivity tools (Google Docs: text files, spreadsheets, and presentations), a customizable start page (iGoogle), and Google Sites (to develop web pages).
All of the applications in Google Apps work through a web browser. They are plain Software-as-a-Service offering with little room for configuration or extension. Users must have a Google account and, once logged in, can access their documents. Each file has a creator/owner, who determines who is allowed to access the file, either as a viewer (with

read-only rights) or a collaborator (who can change the file). Because Google stores all of the files and content centrally, collaboration and document management become simple. For a text document, for example, the application maintains the file, allowing authorized users to see or edit the text while keeping track of all the changes and who makes them. This version history is available and allows comparing any two versions of the document. These joint editing capabilities are a differentiator among the various Software-as-a-Service offerings for office tools.

Similarly, the content in a user's calendar can easily be shared with other users, and Google Sites provides a simple tool for groups to collaborate on developing web pages or whole websites. When a file is complete, it can be "published," which gives it a unique URL, or it can be exported. In the case of a text document, export options include PDF, Word, RTF, OpenOffice, and others. A spreadsheet can be exported as a PDF, a text file, or an .xls file, among others. Much of the Google Apps functionality is available on mobile devices.

The greatest concern about Google Apps and similar services is the loss of control. Because access rights are shared across the service, users rely—to some extent—on how carefully others protect their login credentials. Even though providing e-mail and other applications is complex and expensive, many see this as a core responsibility of the IT department. Given concerns about long-term availability, security, and privacy, storing files, e-mail, calendar entries, and other content on external servers is a deal-breaker for some companies and institutions. From an administrative standpoint, Google doesn't offer as much granularity in managing user accounts as many institutions want and need.

10 A Comparison of Cloud Computing Platforms

In the previous sections we have discussed eight different approaches to cloud computing, which can be seen as representatives for a growing group of offerings. We have presented **Infrastructure-as-a-Service (IaaS)** offerings, namely Amazon Web Services (AWS) and VMwares vCloud. We have discussed three **Platform-as-a-Service (PaaS)** solutions, namely Google AppEngine, Microsoft's Windows Azure, and Salesforce.com's force.com. Finally, we have given an overview over three **Software-as-a-Service (SaaS)** solutions, namely Salesforce.com, Microsoft's OfficeLive, and Google Apps.

From a software architect's and developer's standpoint most interesting are the IaaS and PaaS solutions. Those solutions allow for architecting own applications that may be hosted on the cloud. In contrast, SaaS offerings typically only lend themselves to some limited configuration steps.

10.1 Common Building Blocks

All cloud computing platforms build on a similar set of building blocks. These are:

- Virtualization: In order to dynamically increase and decrease computing capacity in response to changing workloads or administrative triggers, physical computers have to be abstracted away. Virtualization of compute resources can be achieved on many different layers of abstraction (application server, operating system processes, virtual machines, logical partitions of physical hardware). For cloud computing environments, the logical machine level of abstraction has proven to work well. Virtual machines can easily be replicated when load increases. They can be moved across physical machines for consolidation purposes or to allow for system maintenance.
- WebServices: All cloud-computing platforms allow for access to cloud resources via WebService APIs. These APIs typically are hidden behind wrapper libraries in the common programming languages, such as Java, C#, C++ (and other .NET languages) for the client as well as "web languages" such as Python, Ruby, and PHP.
- Orchestration / Service Bus: All cloud-computing platforms provide some means for integrating services in the cloud. These services include compute services, data stores, and messaging middleware. Platforms differ with respect to their support for integrating external services (hosted on premise or at partner sites) with cloud services.
- Clients-side user interfaces: Cloud computing applications need to expose a client-side user interface. Thus, all cloud-computing platforms provide some means for interacting with the user. This may be as simple as a portal that integrates mashups shown in a web browser. Often, user interfaces can be based on Ajax and JavaScript. However, the most elaborate platforms rely on a fully fledged component model such as JavaBeans/Applets or Silverlight/.NET for programming user interfaces that are downloadable and can be executed on the client's computer.

10.2 Which Cloud to choose

Software-as-a-Service (SaaS) solutions have been the initiators of the cloud computing excitement. There are far more SaaS solutions available than the ones we have discussed here. From an end user's perspective, there are a few questions that need to be answered when it comes to using SaaS systems. The most important ones are: "Who owns the data?"

and "Can I work offline?" Systems like Salesforce.com CRM and Google Apps ask the user to fully trust the provider for keeping data secure, consistent, and available. Both services are of no use when the computer is not connected to the Internet. On the other hand, they provide unprecedented ease-of-use without any need to install and maintain software on the client's computer.

Microsoft's OfficeLive follows a different route. While still allowing (read) access to the data from anywhere on the Internet, it will download the data to a client's computer in order to implement modify or write operations. This puts a small burden on the user - as he needs to maintain some software on his computer. However, it provides the big advantage of having the data accessible even when the computer is not connected. Also, it is far easier to keep backups of data if local copies are maintained anyway. Another aspect is the richness of the interface: Software on a local computer today still provides for a better user experience than Web 2.0 interfaces.

Amazon's Elastic Compute Cloud (EC2) was the first cloud solution on the market that was appealing not only to the administrator and end-user but also to developers. Infrastructure-as-a-Service (IaaS) historically presented the second step in cloud computing. Amazon Web Services (AWS) define the de facto standard in IaaS. They have defined all the building blocks, like the concept of an Amazon Machine Image and the use of virtual machines, the Simple Storage Service, and the message queuing middleware. AWS also has a long-standing record in providing high levels of service availability. The AMI concept is very flexible - developers can basically pre-configure their virtual machine with all necessary tools, such as application servers, databases, and middleware solutions prior to deploying them on the cloud (via S3). The biggest drawback of the concept is the burden of maintenance - which is still on the developer, as he has to keep the software stack on his AMI in an updated and well-maintained fashion. Amazon is certainly working on this issue and tries to ease this burden by providing certain pre-configured machine images. One may argue, that the AWS cloud will slowly move from an IaaS offering to a PaaS offering.

VMware's vCloud, the other IaaS offering discussed here is the most recent addition to the cloud-computing spectrum. In contrast to all other cloud platforms discussed here, VMware envisions a whole cloud-computing ecosystem where independent software vendors and hosters may provide competitive offers for running virtual data centers in the cloud. Having multiple providers may circumvent the risk of vendor lock in and also mitigates the business risk of being dependent on just a single cloud operator. VMware tries to standardize on the interfaces necessary to configure, load, instantiate, move, and replicate virtual machine instances in the cloud. It is suggesting using the Open Virtualization Format (OVF) as the

standard for cloud computing instances. VMware has a long-standing record on virtualization. Playing just on the OVF level will allow VMware to build upon its strengths and implement monitoring and system administration interfaces for cloud instances on a generic level. Improved monitoring and self-adaptive management capabilities would be a most desirable feature for all the cloud platforms currently available.

Platform-as-a-Service (PaaS) solutions are - from an architect's and developer's standpoint - the wave of the future. The integration of elaborate development tools on the client side with a standardized and well-maintained execution environments in the cloud provides for ease-of-use and high efficiency. PaaS platforms differ in the set of supported programming languages and the set of additional infrastructure services available with the execution environment in the cloud.

Salesforce.com's Force.com PaaS offering is tightly integrated with a database in the cloud. It is best suited for a specialized set of cloud applications, where customer data are in the center of operation. However, developers have to use the proprietary Apex language in order to code for the cloud. Force.com applications therefore are hardly portable. There is little support for running cloud applications on-premise.

Microsoft's Windows Azure and Google's AppEngine are quite comparable in their basic approach to cloud computing. Both are PaaS offerings that promise ease-of-use and simple maintenance of cloud applications to the architect and developer. Both platforms support integration between powerful client-side integrated development environments - namely Visual Studio for Windows Azure and Eclipse for AppEngine - and the cloud. Both providers have a long-standing trust record for providing services on the Internet.

Microsoft supports the usage of all .NET languages (C#, VB.NET, C++, and many more) with Windows Azure. Google provides a tight integration of Java (and languages supported by the JVM) with AppEngine. Both platforms allow for installation of additional software and there are interoperability stories explaining how to run code written in other languages on Windows Azure and AppEngine.

The biggest difference between AppEngine and Windows Azure is the integration of SQL Azure in Microsoft's cloud computing platform. The availability of a database in the cloud that supports structured queries and complex operations on data may be important for a wide range of applications. Another specialty of Windows Azure is the Service Bus integrated in AppFabric. Not only does the service bus support orchestration of services in the cloud, it also allows reaching out from the cloud onto services running locally in a client’s or partner's datacenter. Obviously there is a tradeoff: accessing off-cloud services allows for easier

system integration on one hand side, it makes service availability directly dependent on those external services on the other hand side.

10.3 Beyond Scope

We have tried to discuss the most prominent cloud computing solutions available today. Our discussion has focused on technical characteristics and tried to differentiate the cloud computing platforms. We have not evaluated business perspectives, potential savings, or economic opportunities opened up by cloud computing. There are more aspects, which are beyond scope of the discussion presented in this paper.

Also, we have not discussed a number of orthogonal megatrends, which are not (yet) addressed by today's cloud computing platforms but will certainly impact future developments. The most important among those trends is the multicore/multithreading challenge. So far, cloud-computing environments offer virtual machines as basic units of abstraction. These virtual machines may or may not run on multiple physical cores. However, none of the programming environments for cloud computing do explicitly support parallel programming for multicore systems. With the increase of the number of cores in future processor generations (and a decrease in clock speed for every single core) the cloud developer will want to focus on efficient parallel programming models. Microsoft's parallel pattern library or Intel's threading building blocks have outlined how multicore systems can be efficiently used on the desktop and server. Those ideas need to be transported into the cloud.

Given the tremendous amount of experience and know how in system optimization and monitoring present in today's datacenters, another important aspect for successful operation of cloud applications will be the question of how to monitor and manage all layers of the cloud stack. Self-adaptive management mechanisms present in the cloud fabric are only in a very limited way exposed to cloud applications and clients. A consistent monitoring model covering all layers of the cloud stack is badly needed.

As least as important, however, is the question of how to secure data in the cloud. Current levels of access control available in the cloud platforms are very limited. Identity management and federation as well as management of trust among clients and services in the cloud will be most important.

Literature

Richarch Murch, „Autonomic Computing“, IBM Press, 2004.

Pradeep Padala, Kang Shin, Xiayun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, and Kenneth Salem „Adaptive control of virtualized resources in utility computing environments“, SIGOPS Oper. Syst. Rev. 41, 3 (Jun. 2007), 289-302.

Mendel Rosenblum and Tal Garfinkel, „Virtual Machine Monitors: Current Technology and Future Trends“, Computer 38, 5 (May. 2005), pp. 39-47.

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield, “Xen and the art of virtualization“, In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP '03) ACM Press, 2003, pp. 164-177.

Sun Microsystems, Grid Engine Documents, http://gridengine.sunsource.net/documentation.html

Rajkumar Buyya, Chee Shin Yeo, Shrikumar Venugopal, James Broberg, and Ivona Brandic, „Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility“, Future Gener. Comput. Syst. 25, 6 (Jun. 2009), 599-616.

Amazon Web Services LLC., „The Elastic Compute Cloud EC2“, http://aws.amazon.com/ec2/, 2009.

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels, „Dynamo: amazon's highly available key-value store“, SIGOPS Oper. Syst. Rev. 41, 6 (Oct. 2007), 205-220.

Simson L. Garfinkel, „An Evaluation of Amazon’s Grid Computing Services: EC2, S3 and SQS“, Tech. Rep. TR-08-07, Harvard University, August 2007.

VMware Inc., „Cloud Computing with VMware Virtualization“, www.vmware.com/solutions/cloud-computing

VMware Inc., „vCloud API“,http://communities.vmware.com/community/developer/forums/vcloudapi

Microsoft Corp., „Windows Azure“, http://www.microsoft.com/windowsazure/ { windowsazure | sqlazure | dotnetservices }, 2009.

Google Inc., „What is Google App Engine?“, http://code.google.com/intl/de-DE/appengine/docs/whatisgoogleappengine.html, 2009.

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, „The Google file system“, SIGOPS Oper. Syst. Rev. 37, 5 (Dec. 2003), 29-43.

Dean Jacobs (Salesforce.com), „Enterprise Software as Service“, Queue 3, 6 (Jul. 2005), pp. 36-42.

Salesforce.com, Whitepaper, „A Comprehensive Look at the Force.com Cloud Platform“, http://www.developerforce.com/media/Forcedotcom_Whitepaper/WP_Forcedotcom-InDepth_040709_WEB.pdf