Cloud Service Model

Selecting a cloud service model

- Different cloud service models within the enterprise
  - Single cloud provider
  - AWS for IaaS
  - Azure for PaaS
- Force fit all solutions into the cloud service model from the provider
- May need to use multiple models to solve a problem

Considerations to choose a service model

- Study feasibility of each cloud service model based on five categories:

  1. Technical
     - Focus on aspects such as performance, scalability, security, regulation, business continuity, disaster recovery
     - Performance and scalability critical to deciding between PaaS and IaaS service models
       - Platforms abstract the underlying infrastructure from the developer; developer can focus on business requirements while platform handles autoscaling
       - PaaS vendors may enforce limitations on resources to accommodate all clients
     - IaaS and PaaS may offer Database as a Service (DBaaS)
       - Helps automatically manage DB tasks such as replication, autoscaling, monitoring, and backups
       - Client may lack adequate control over the database (DB unavailable)
  2. Financial
     - Focus on total cost of ownership (TCO)
     - Easier to compute TCO for new projects
     - Projects migrating to cloud or constrained by legacy applications add to the complexity of computing TCO
       - Cost to change/integrate the legacy architecture
       - Cost of retrofitting existing architectures to facilitate integration with cloud services
       - Cost to reengineer legacy architectures, employee training, hiring new employees/consultants, acquiring tools/services to help with reengineering, disposal of existing infrastructure
  3. Strategic
     - Importance of speed-to-market for an initiative
     - Leverage cloud services by using PaaS or SaaS to relieve IT from heavy lifting involved with IaaS
     - If control is important, go towards IaaS
     - Business strategies affecting the decision include consolidating data centers, reducing costs, being first-to-market, handling enormous scale, selling product globally 24/7, and integrating with partner supply chains
  4. Organization
     - Assessing the organization
       - Does IT have the skills to build solutions in the cloud?
       - Strong IT skills in the areas of distributed computing, web development, and SOA? If not, lean towards SaaS or PaaS
     - Higher competence for lower hierarchy in the cloud stack
  5. Risk
– Amount of risk acceptable to the company
  * Duration of downtime
  * Security breach
  * Can government seize data in cloud without a warrant
– A factor in deciding between public and private cloud

• Emphasis on technology (social media) or risk (security for medical data)

SaaS

• Most mature of the three cloud models
• SaaS providers keep total control of infrastructure, performance, security, scalability, and privacy
• Two ways to use applications
  1. Web-based user interface accessible on any device that can connect to the internet
     – Includes mobile device applications
  2. Provide APIs to customers to enable consumers to integrate features into existing applications or other SaaS solutions
• Used to outsource all applications, features, and services that are not core competency
• Solution categories
  – Enterprise business applications such as customer relationship management (CRM), enterprise resource planning (ERP), accounting, human resources, and payroll
  – IT infrastructure SaaS solutions dealing with security, monitoring, logging, and testing
  – Data category of business intelligence, DBaaS, data visualization, dashboards, data mining
  – Productivity such as collaboration tools, development tools, surveys, and email campaign tools
• May not provide high flexibility as same tools are needed by many customers
• Factors to take into consideration for TCO before attempting to build own tools
  – Vendor responsible for security updates and patches
  – Vendor manages all infrastructure and data center
  – Vendor provides mobile compatibility for majority of phones and tablets
  – Vendor provides compatibility across all major browsers and versions
  – Vendor frequently updates features and updates are seamless to end user
  – Vendor manages databases, including capacity planning and backup recovery
• Keeping the application current and ahead of emerging competition

PaaS

• Least mature of the three cloud models
  – First generation included solutions like Google, Force.com, and Microsoft Azure
  – Buyers use a specific programming language and run on the service provider’s infrastructure
  – Second generation of solution providers more flexible: Cloud Foundry and OpenShift
  – Open source cloud-based solutions allow control over software and platform
Service consumer can stay on the platform as long as he likes

- **Public PaaS service providers**
  - Manage underlying infrastructure, networks, storage devices, and OS
  - Responsible for periodic security patching, logging, monitoring, scaling, fail over, and other sys admin tasks
  - Developers focused on cloud-ready applications

- **Private PaaS service providers**
  - May not provide abstraction of infrastructure services
  - Capability to deploy software on both private and public cloud
  - Require service consumer to manage the application stack and infrastructure

- **Platform shared by many customers**
  - Limits enforced on developers
  - Known as *throttling*
    - Protect platform from getting overloaded by an individual customer
    - Protect against network collisions and congestion
    - Throttle CPU utilization to reduce heat in data center and conserve power
  - Developers should know limitations and design accordingly
    - May create unacceptable delays in processing time
    - May impact the quality and reliability of application

**IaaS**

- **Leverage IaaS if**
  - Scalability requirements require the developers to manage memory
  - Developers need to configure database servers and applications servers to maximize throughput
  - Developers need to specify distribution of data across disk spindles
  - Developers need to manipulate the OS

- **Cost factor**
  - PaaS reduces cost by reducing the amount of work and number of resources required to build and deploy applications
    - PaaS pay-as-you-go model can become very expensive with large data or CPU/bandwidth demands

- **Mitigating risk of downtime**
  - IaaS customer can architect for failure and build redundant services across multiple physical and virtual data centers

- **Up the stack, SaaS increases speed to market, reduces human resources, and reduces operational costs**

- **Down the stack, IaaS provides more control of infrastructure and a better change to avoid or recover from a vendor outage**

**Common cloud use cases**

- **Startups vs established enterprises**

- **Cloud bursting**
Cloud Service Model

- Leverage cloud to handle peaks in traffic
- Run applications in the data center and send excess capacity out to a cloud provider
- Retailers with seasonal spikes during holidays
- Companies preparing tax returns

- **Archiving/storage**
  - Leverage storage in cloud
  - Eliminate storage media and transportation services
  - Cloud costs are drastically cheaper and process for data retrieval much less complex

- **Data mining and analytics**
  - Analysis of large amount of data – big data, genomics
  - Resources needed only when someone makes a query

- **Test environments**
  - Create virtual machines to test applications under different environments
  - Reduces reliance on system personnel to create test environments
  - Create environment only when needed
  - Simulate large peaks in traffic

---

**Key to the Cloud – RESTful Services**

**RESTful services**

- **Representational State Transfer services**
  - Term originated in 2000
  - An architectural style used for web development for fast performance, reliability, and scalability
  - Based on reusable components that can be managed and updated without affecting the system as a whole while it is running

- **Critical components of any cloud solution**
  - Services in cloud are built on top of IaaS, PaaS, or integrated with one-to-many SaaS components
    * APTs for those services are exposed using RESTful services
    * Clouds connect many services from different companies using different technology stacks
    * Complexities of underlying stacks and protocols need to be abstracted away from business logic
    * APTs encapsulate multiple programming stacks, database technologies, and technologies for integration, caching, queuing, event processing, and so on
  - Many touchpoints for the services
    * Multiple user interfaces – web, mobile, tablet
    * Access same services and are always in sync

- **Virtual and dynamic infrastructure**
  * Resources added and removed dynamically
  * Any point in infrastructure liable to fail
  * Software must be fault-tolerant to take advantage of fault-tolerant infrastructure; should not be tightly coupled with infrastructure
Why REST?

• Internet works collectively with no knowledge of any resource located on any server

• Four constraints

1. Separation of resource from representation
   – Loosely coupled resources and representations
   – Resource – A data store or a chunk of code
   – Representation – XML, JSON, or HTML page
   – Allows scaling of different components separately
     * Resource (photo/video) may be distributed across a content delivery network, replicating data across a high-performance distributed network for speed and reliability; may be hosted on a third party content delivery vendor such as AT&T
     * Representation may be in XML message or HTML page to indicate the resource to retrieve
     * HTML page may be executed on a web server farm across many servers in multiple zones in Amazon’s public cloud

2. Manipulation of resources by representations
   – Representation with attached metadata provides sufficient information to modify/delete resource on server, as long as the client has permission to do so
   – Resource data (a customer row in MySQL table) can be only modified or deleted on the DB server if the client sending the representation (XML message) has enough information (PUT, POST, DELETE) and has permission to do so
   – User credentials become part of the XML message

3. Self-descriptive messages
   – Message provides information to process itself, or contains enough information to describe parsing of data
   – Accept application/xml command tells the parser to expect message format as XML
   – Possible to use technologies such as XML or JSON

4. Hypermedia as the engine of application state (HATEOAS)
   – Client interacts with application only through hypermedia or hyperlinks
   – Representations reflect the current state of hypermedia applications
   – No need to maintain application state on the server
   – Application state represented by a series of links – uniform resource identifiers (URIs) – on client side
     * When a resource (server/connection) fails the resource that resumes working on the services starts with the URI of failed resource

• Cloud treated as a massive network of fault-tolerant independent resources
   – No dependencies on underlying network

• Impact on performance

Challenges of migrating legacy systems to cloud

• Legacy systems reliant on ACID transactions
   – Atomicity, consistency, isolation, durability
   – Ensure that a transaction is complete and consistent
   – Transaction not complete until its is committed and data is up-to-date

• Cloud architectures rely on BASE transactions
Cloud Service Model

- Basically available, Soft state, Eventually consistent
- Resources can fail and data will eventually become consistent
- Volatile environments where nodes may fail or systems need to work whether user is connected to the network or not
  * Important in mobile environments
- Partition tolerance
  * If one instance of a compute resource cannot complete the task, another instance can be called to finish the job