CLOUD: VISION TO REALITY

Prepared by:
Duncan Rutland, Advisory Services Enterprise Architect, Rackspace® Hosting

rackspace® HOSTING

Introduction

There is no longer any question that the cloud computing model will be the prevailing style of delivery for computing over the coming decades; Forrester Research predicts that the global market for cloud computing will grow from $40.7 billion in 2011 to more than $241 billion in 2020[1]. Greenfield application development projects can be designed from the outset to benefit from cloud computing features such as elastic scalability, automated provisioning, infrastructure level APIs, object storage services and other middleware services. However, for existing legacy applications (particularly in the corporate space) the journey to cloud is not quite so straightforward.

Table of Contents

[1] *Sizing The Cloud, Forrester Research, Inc., April 2011*

Considerations

2.1 SECURITY/COMPLIANCE

Concerns around security and/or compliance are often regarded as the primary barrier to entry with regard to public cloud computing. These concerns can largely be attributed to the multi-tenant nature of public cloud computing.

2.1.1 WHY THE FUSS?

Public cloud providers are able to optimize the utilization of physical resources by sharing them between multiple customers using virtualization technologies. This multi-tenant model will always have a degree of associated risk since the underlying hardware and software, being engineered by human-beings, is susceptible to flaws that can be exploited for malicious purposes.

2.1.2 KEY FACTOR: DATA SENSITIVITY

An important factor to consider when evaluating the security requirements of an application is the sensitivity of information that is being stored, processed or transmitted by the application. Areas to be particularly cautious around are:

- Personally Identifiable Information (PII) such as names, national identification numbers, dates of birth, drivers license/passport numbers, etc.
- Personal Health Information (PHI) such as medical history, test results, insurance information
- Financial information such as credit card numbers and account numbers
- Corporate confidential information

2.1.3 MITIGATION STRATEGIES

So your application is handling some sensitive data. What can you do to mitigate the risk of unauthorized access to this data?

It is important to follow standard industry best practices (like those prescribed by the PCI-DSS standard). Some important areas of focus are deploying perimeter and host firewalls with appropriate policies, ensuring software is kept up to date, disabling unnecessary services, enforcing strong permissions and utilizing 2-factor user authentication. However, in the event of an intrusion here are two commonly used practices to help mitigate the risk of a breach of confidentiality:

1. **Encryption.** One option to mitigate the risk of breaching confidentiality/integrity of data is to use public key cryptography at the application layer. This has the benefit of reducing the risk associated with unauthorized access to encrypted data. However, this approach is only as robust as the access control mechanism around the private keys. Failure to adequately protect private encryption keys and passphrases can make the use of encryption ineffective. One promising area of research that may alleviate this risk is Fully Homomorphic Encryption (FHE), which allows operations to be performed on data without decrypting it. This could remove the need for private encryption keys to be stored locally on processing nodes, potentially reducing the impact of a malicious intrusion.
2. **Functional Separation.** Sensitive data is often only being handled by certain areas of your application (for example, an e-commerce application typically only needs to process credit card details during the check-out phase of the transaction). This can be exploited by functionally decomposing the application into separate components, such as using a Service Oriented Architecture, and choosing different hosting models for different components depending on their own security requirements. A hybrid hosting model can be leveraged for an e-commerce application by hosting the back-end database and check-out components in a private cloud environment, and hosting the catalog, shopping cart and other less sensitive components in a public cloud environment.

While future advances in software and hardware could mitigate these concerns, at this time security remains a key consideration when contemplating a transition to cloud computing.

2.2 APPLICATION ARCHITECTURE

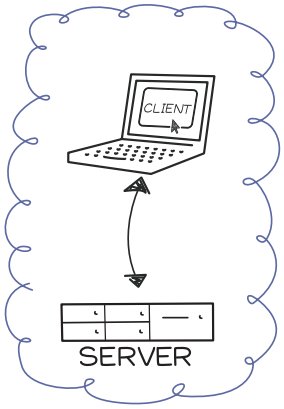
Traditional approaches to application architecture do not always translate effectively to a cloud computing model. Monolithic applications that are tightly coupled, scale vertically and rely on shared physical components for high-availability may require re-engineering to function optimally in the cloud. Here we explore some of these architectural considerations.

2.2.1 FUNCTIONAL SEPARATION

One of the factors in determining how well a given application will translate into a cloud context is the degree of functional separation within the application. This can typically be expressed as the number of different roles or functions that make up the application. Over the last two decades, software architecture has become increasingly de-composed functionally:

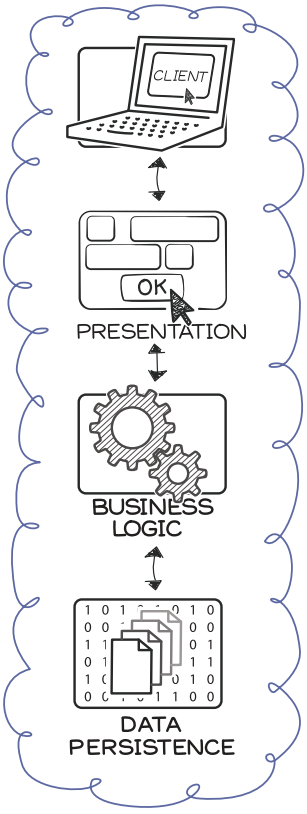
Client/Server 1

presentation/business logic separated from data layer. For example, a typical VB6 application composed of two layers: a fat client binary comprising UI/logic and a separate database tier.

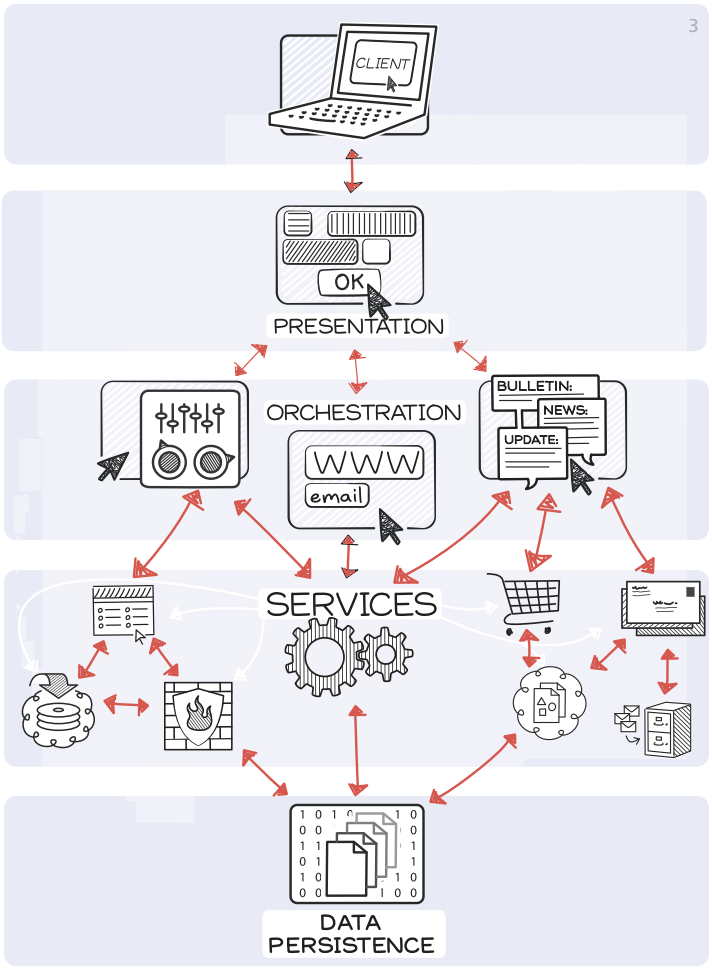


N-tier 2

presentation, business logic and persistence layers entirely separated, often with additional messaging and caching layers.



2



SOA 3

Fine-grained functional separation. For example, an e-commerce application may be de-composed into many hundreds of separate services such as shopping cart, checkout, recommendations, order tracking etc.

Functional separation within an application has other benefits:

1. **Scalability:** By de-composing an application into a set of simpler sub-components, greater scalability can be achieved by allowing these functional units to be spread out onto a larger pool of resources. This pattern of application architecture is referred to as a Service Oriented Architecture (SOA).

2. **Security:** Often, sensitive data is manipulated by certain parts of the application and not others (for example, the check-out of an e-commerce application that handles credit card data). If these parts of the application can be functionally separated, then the need for stricter security controls and regulatory compliance can potentially be reduced. Using this model, a combination of private dedicated resources (for parts of the application that require more rigorous security measures) and public shared resources (for parts of the application that handle less sensitive data) can be combined within a single solution.

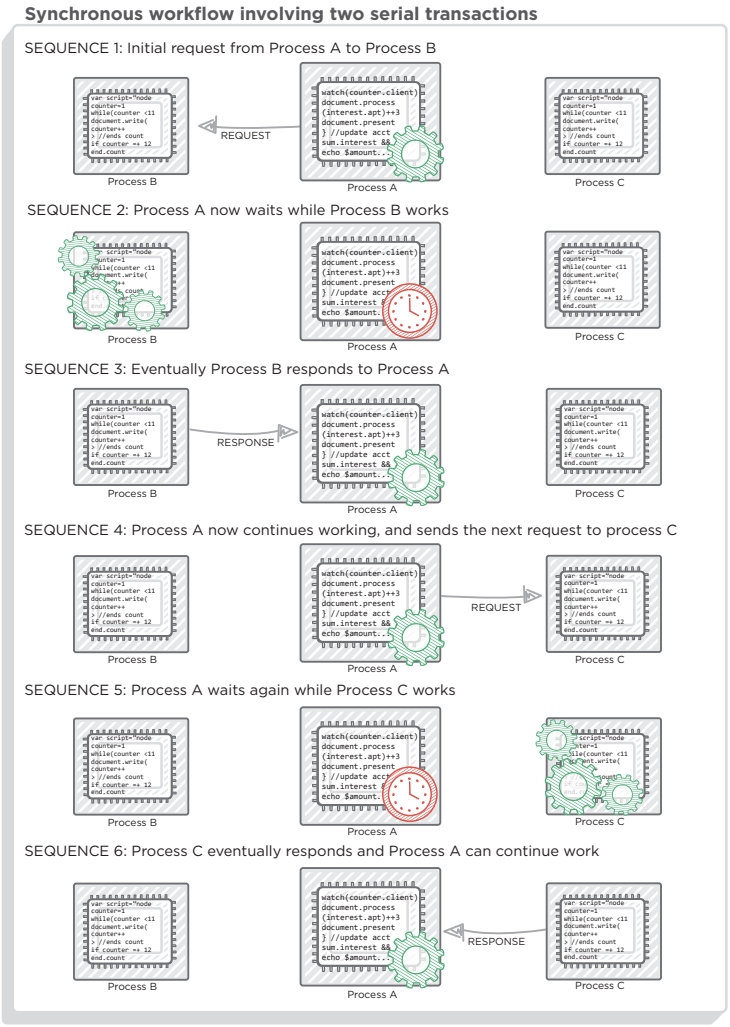
2.2.2 HORIZONTAL SCALABILITY

One of the cost-saving features of public cloud computing is the ability to match the supply of computing resources to the demand, which is made possible by the metered or utility billing model. However, in order to access this feature, the high-level architecture of the application must facilitate the scaling up and down of resources (commonly known as "elastic" scalability).

More explicitly, the application must be able to scale horizontally and preferably in an automated or semi-automated fashion. Furthermore, licensing considerations come into play. Many commercial enterprise applications are licensed per instance, so scaling these instances horizontally rather than vertically may be cost-prohibitive. Open Source Software (OSS) can be utilized whenever elastic scalability is required since there is no cost-impact from running a higher number of nodes.

2.2.3 COUPLING

A key factor in determining how successfully an application will de-compose into layers that can scale elastically is the nature of the coupling between those layers.



Many legacy applications were intended to be run within a single OS instance or within the same LAN, and the couplings between the different functional layers of the application are often synchronous in nature. Examples of this are:

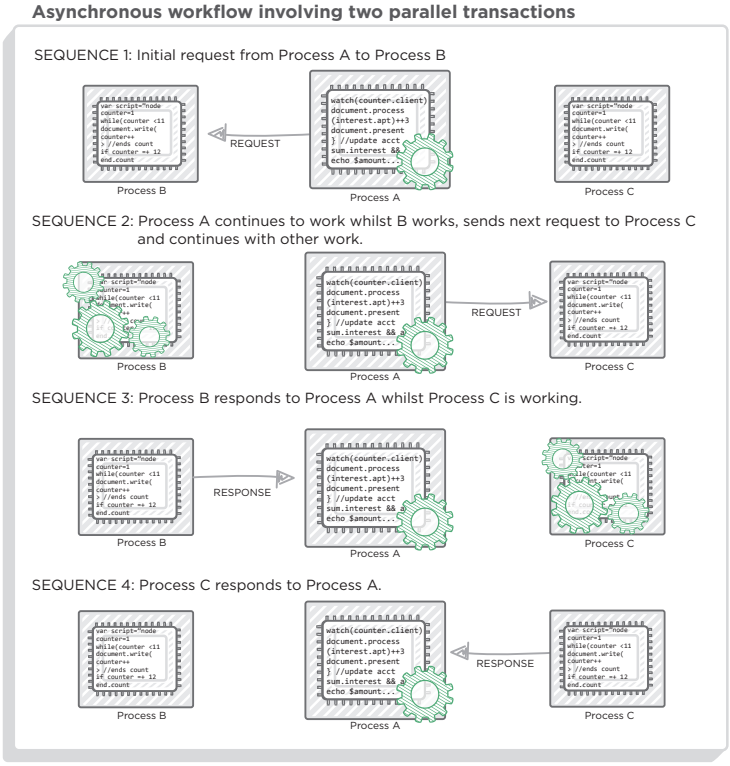
- ⇨ IPC
- ⇨ RPC
- ⇨ CORBA
- ⇨ DCOM

A synchronous call between application components means that after one component sends a request to another, the sender stalls and cannot continue doing useful work until the response is received.

A solution to this problem could be to implement asynchronous couplings between application components. Under this model, after the sender has delivered a request there is no blocking; the flow of execution can continue and the response can be handled when appropriate. This pattern is particularly effective when the latency between components is large and/or variable, and inherently lends itself to applications that need to scale. The most frequent implementation of an asynchronous coupling is the message queue. Some common examples are:

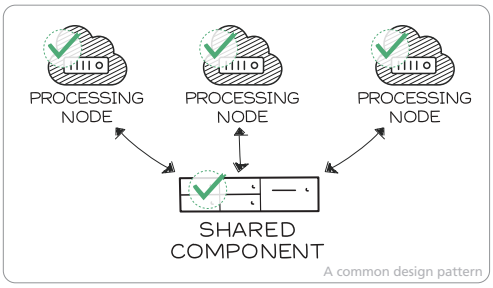
- ⇨ MSMQ
- ⇨ JMS
- ⇨ RabbitMQ

Asynchronous workflow involving two parallel transactions

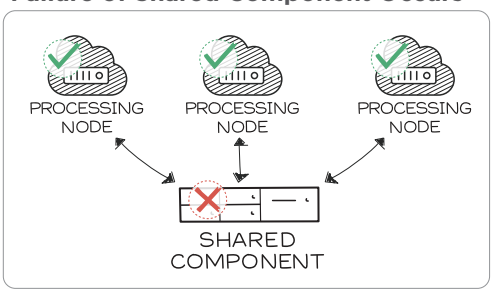


2.2.4 SHARED PHYSICAL COMPONENTS

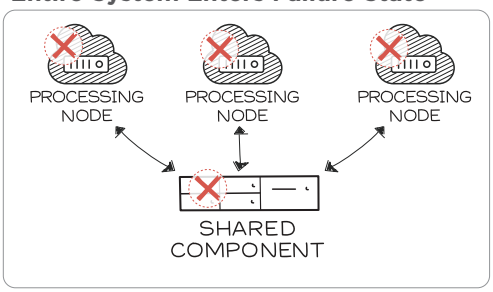
Shared Physical Compenents: Healthy Operation



Failure of Shared Component Occurs



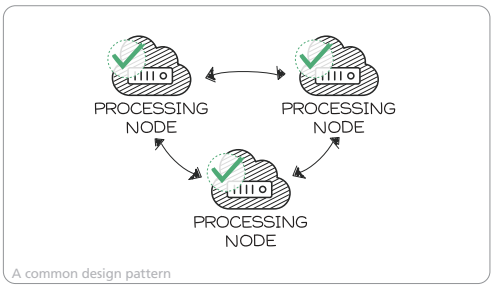
Entire System Enters Failure State



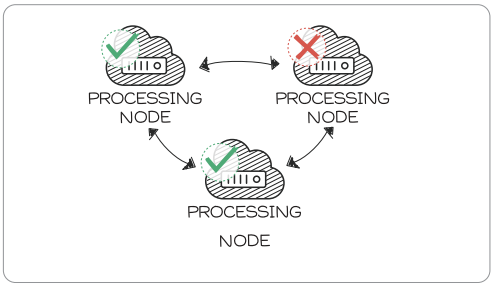
Physical resources (such as storage arrays being shared between multiple database servers) that are shared between two or more processing nodes can become bottlenecks and inhibit scalability. They also effectively bind the instances into a single fault domain, so that a failure of the shared component will result in a failure of the entire sub-system.

A common design pattern to alleviate this risk is the Shared Nothing Architecture (SNA), which relies on replication and other inter-node synchronization methods to achieve the goal of high-availability and fault-tolerance without relying on shared components.

Shared Nothing Architecture: Healthy Operation



Failure Occurs but remaining nodes function



2.2.5 GOOD DESIGN PRACTICES

In conclusion, the following examples of design choices translate particularly well when deploying applications in the cloud:

- Service Oriented Architecture (SOA) – where your application is split into many separate functional components
- Asynchronous Messaging (Loose coupling) – such as message queues
- Shared Nothing Architecture (SNA) – where there are no shared physical components between nodes
- Open Source Software (OSS) – due to the inhibitory impact proprietary licensing can have on scalability

2.3 PERFORMANCE

The utilization of physical resources by an application is a pertinent factor when choosing a hosting strategy, and valid concerns around performance are regarded as one of the barriers to entry when evaluating virtualization (either within a private dedicated environment or with a multi-tenant cloud).

2.3.1 PERFORMANCE OVERHEAD

Acting as an abstraction layer between a guest OS and the physical hardware, the hypervisor must juggle physical resources between multiple competing consumers, whilst maintaining the illusion that they each have sole dominion. This juggling act comes at a price: the brokering of physical resources, handling I/O interrupts, page faults, context switching between VM instances and other activities all consume additional resources. Disk I/O in particular is problematic due to the PCI bus not being virtualization-aware, so many hypervisors are forced to emulate the entire device driver at a significant performance penalty. Fortunately, advances in hardware assisted virtualization (EPT, PCI-SIG SR-IOV) and the use of para-virtualization techniques (altering the OS to become aware of the hypervisor) are reducing this performance overhead. These new features could become ubiquitous within 3-5 years, but for now we should be cognizant of these constraints.

2.3.2 RESOURCE CONTENTION

Many providers of public clouds are able to aggregate and over-provision many competing customer workloads on shared physical hardware under the assumption that consumers will not all want access to the same resources simultaneously. However, due to the unpredictability of demand from competing workloads, observed performance can be variable in many public clouds.

2.3.3 INSTANCE SIZE MISMATCH

Another consideration is around any mismatch between the VM instance size (usually defined as the quantity of CPU, RAM and disk resource assigned to the VM) required by the application, and the instance sizes available from the service provider.

Many scale cloud providers use fixed instance sizes to allow efficient allocation of resources into physical hardware boundaries. There may often be a mismatch between the instance size the application actually requires to function acceptably and the instance size available.

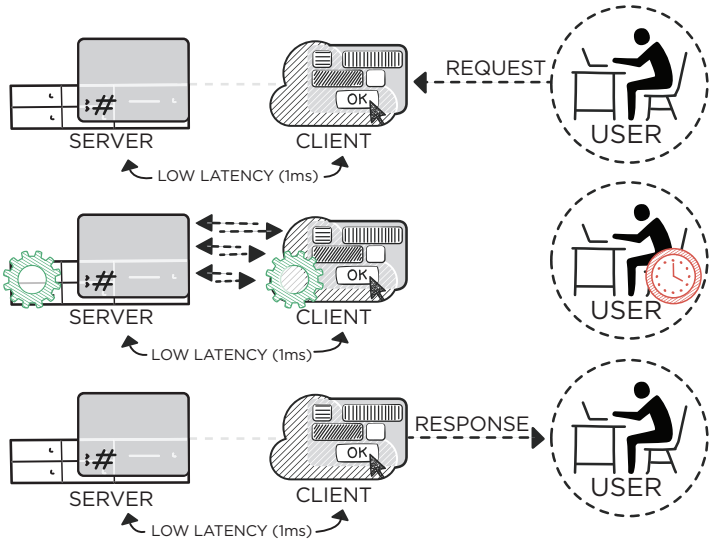
This can lead to wastage (where the consumer pays for resources that are not being used) or under-provisioning in which case it may be necessary to scale the application horizontally across multiple nodes. This is not always possible with monolithic applications that were not designed from the outset to scale horizontally.

2.3.4 LATENCY

Legacy applications that were designed in the Client/Server assumed that the Client and Server components would be located together, with very low latency between them. They often make serveral "round-trips" of dialogue when processing user requests.

Example 1

Client/presentation layer and back-end server are located on the same LAN



Sequence 1: End-user interacts with client/front-end GUI

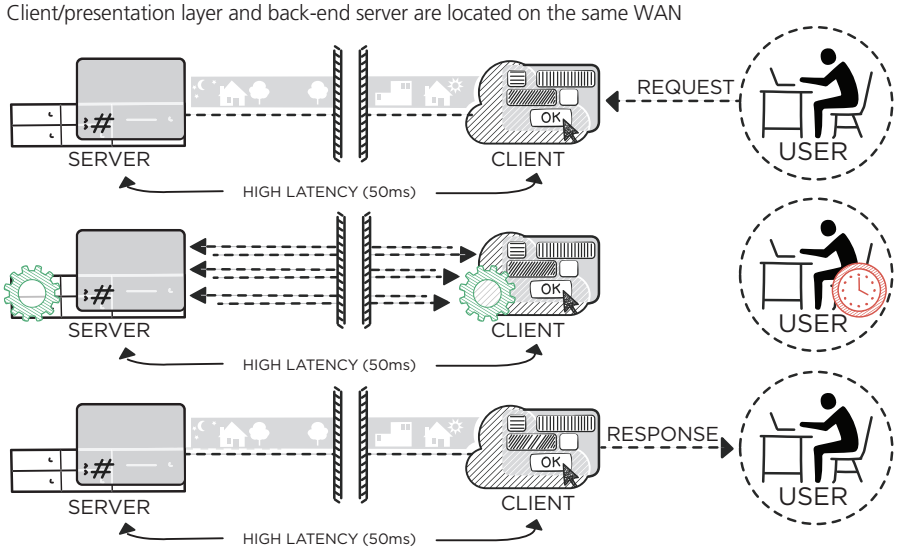
Sequence 2: Client communicates with back-end server (3 round-trips, elapsed time = >6ms)

Sequence 3: Client responds back to end-user.

However, when moving these applications to an external service provider, the client and server component will be separated. For these types of applications, this increase in latency between the client and the server can lead to a catastrophic degradation of performance. Technologies such as remote desktop and application virtualization can remedy this issue by locating the Client and Server together, and presenting a view of the GUI to the remote end-user.

Example 2

Client/presentation layer and back-end server are located on the same WAN



Sequence 1: End-user interacts with client/front-end GUI

Sequence 2: Client communicates with back-end server (3 round-trips, elapsed time = >300ms)

Sequence 3: Client responds back to end-user.

2.4 METHOD OF CONSUMPTION

The way an application/service is consumed by end-users is a critical factor when considering migrating to an external service provider. In order to ascertain the impact of moving, it is essential to understand the consumers of the service in terms of:

- **Geographic Location:** What is the latency between the end-user and the service? At what hours of the day do they consume the service?
- **Method of Connectivity:** How rigid is their connection into the service: do they traverse the internet, use an IPSec VPN tunnel or a private WAN link?
- **Client Type:** Web browser clients are generally designed to perform well over a high-latency WAN connection, whereas binary clients (like VB6) that follow traditional client-server architecture do not typically withstand separation of client and server over even modest latencies.

2.5 IMPACT ON AVAILABILITY

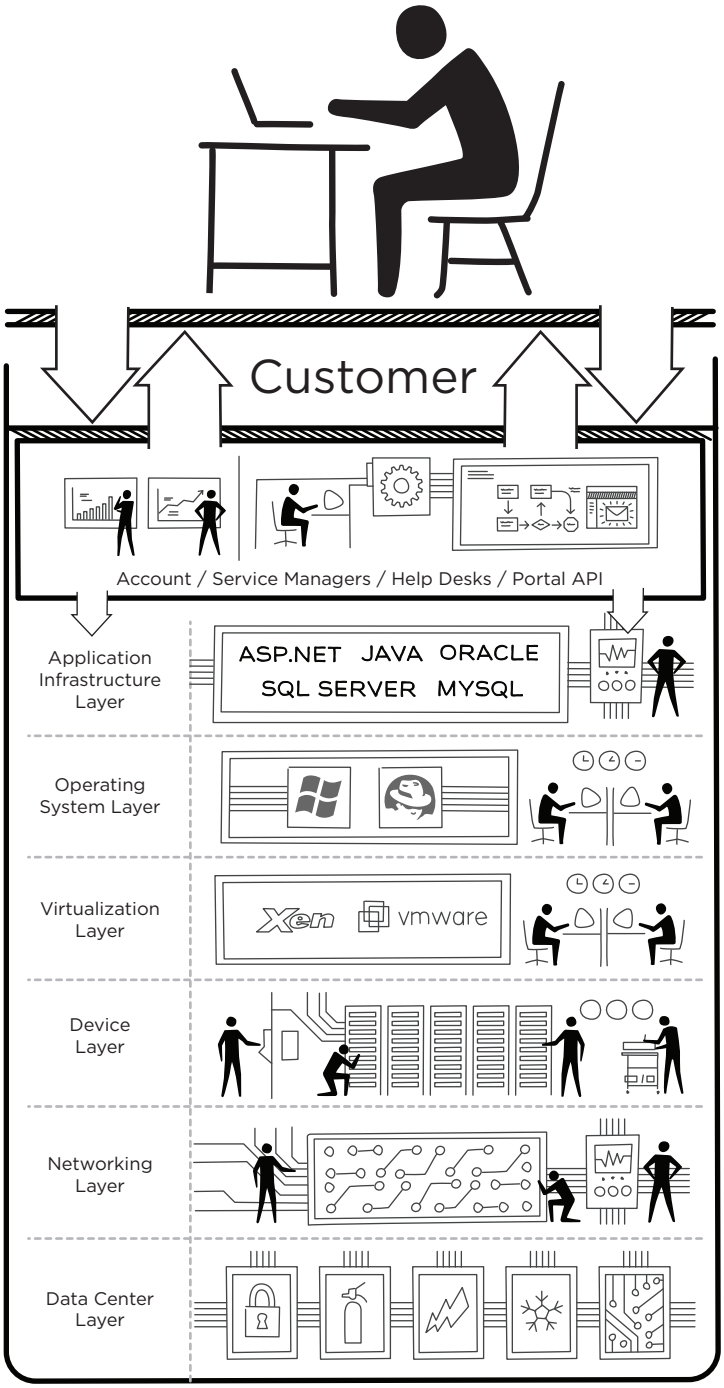
If your application is bound by an availability SLA or RPO/RTO metric this should be re-visited when considering moving to an external service provider. Here are some common factors that you should consider:

1. **Network Latency –** Any change in network latency between primary and DR locations can potentially affect RPO due to the impact on data replication. Additionally, application performance can be impacted when synchronous replication is being utilized.
2. **Network Redundancy –** Imagine the scenario: you have moved a business critical service from an in-house hosting facility to a remote service provider. Your internal users must now traverse your primary internet connection to access the application. Bottom line: The uptime of your application is now dependent on the uptime of your internet connection. For business critical applications, this link should be protected.
3. **Operational Latency –** Your IT operations may be used to having direct control over the technology platform behind an application. When moving an application to a remote service provider, they will no longer have the same degree of control. Previously, executing a failover to a Disaster Recovery solution could be performed entirely by your internal IT operations team. However, after migrating to an external service provider, many actions must be performed by an external party. This adds additional latency into the process as requests are raised, authorized and tasks synchronized between application owner and service provider. The result? Your RTO will increase.

2.6 MATURITY OF IT OPERATIONS

When assessing the feasibility of moving services from an in-house to an externally hosted context, the maturity of an organization's IT Service Management (ITSM) capability is of concern.

Traditionally, many internal IT operations follow a Plan -> Build -> Operate cycle and may have varying degrees of governance in place around how the service is delivered. Even when there is no formal governance in place, the business is usually able to scrutinize and govern the delivery of the service due to it being facilitated internally.



When outsourcing technology to an external provider, this transparency is lost and the customer must rely on externally exposed interfaces such as:

- ⇨ help desks
- ⇨ account/service managers
- ⇨ online portals and APIs

in order to gain visibility over the delivery of the service.

So what are some ways a customer can try to ensure that the vendor is meeting contracted SLAs for availability and quality?

Adopting a formal ITSM framework such as ITIL or ISO20000 provides an organization with the governance structure necessary to manage the delivery of IT services both internally and from external suppliers. Thus, your IT organization should naturally move away from traditional operations towards a service management function.

Summary

While the task of assessing an installed base of legacy applications may seem daunting at first glance, the problem is not insurmountable. Here are some "Keys to Success" when performing such an assessment:

- First, effective decisions to determine what hosting model your application is best suited to cannot be made without supporting data:
  - Install a monitoring system to collect inventory, performance and capacity data from your infrastructure.
  - Gather all documentation relative to the application and supporting infrastructure into a central location, and address any gaps.
- Second, define and prioritize what business goal(s) you need to satisfy, such as cost reduction, risk mitigation, or increased focus & agility.
- Next, it is important to determine the scope of your assessment and prioritize applications in terms of business criticality, size/complexity and known migration feasibility. Focus first on some "quick wins" that are not business critical and can serve as proof-of-concept cases.
- Finally, assess the feasibility of migrating your application in terms of:
  - Security & Performance – How amenable is your application to being virtualized and running in a public or private cloud context?
  - Application Architecture – Can your application function optimally in a cloud context without re-engineering?
  - Consumption – How do your end-users consume your application/service and what are the implications of migrating to the cloud?
  - Operations/Governance – Is your IT operations organization ready to transition to an IT Service Management function?
  - Integration – How tightly coupled is your application into its present environment?
  - Supply/Demand – Does sufficient variation in demand and supply-side flexibility exist to permit the metered-billing benefits of multi-tenant cloud computing to be leveraged in order to reduce costs?

If you need help in assessing the feasibility of migrating your legacy applications to a cloud hosting model, Rackspace can help. E-mail us directly at advisory_services@rackspace.com or visit http://www.rackspace.com/enterprise_hosting/advisory_services/ for more details.

DISCLAIMER

This Whitepaper is for informational purposes only and is provided "AS IS." The information set forth in this document is intended as a guide and not as a step-by-step process, and does not represent an assessment of any specific compliance with laws or regulations or constitute advice. We strongly recommend that you engage additional expertise in order to further evaluate applicable requirements for your specific environment. RACKSPACE MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, AS TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS DOCUMENT AND RESERVES THE RIGHT TO MAKE CHANGES TO SPECIFICATIONS AND PRODUCT/SERVICES DESCRIPTION AT ANY TIME WITHOUT NOTICE. RACKSPACE RESERVES THE RIGHT TO DISCONTINUE OR MAKE CHANGES TO ITS SERVICES OFFERINGS AT ANY TIME WITHOUT NOTICE. USERS MUST TAKE FULL RESPONSIBILITY FOR APPLICATION OF ANY SERVICES AND/OR PROCESSES MENTIONED HEREIN. EXCEPT AS SET FORTH IN RACKSPACE GENERAL TERMS AND CONDITIONS, CLOUD TERMS OF SERVICE AND/OR OTHER AGREEMENT YOU SIGN WITH RACKSPACE, RACKSPACE ASSUMES NO LIABILITY WHATSOEVER, AND DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO ITS SERVICES INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT.

Except as expressly provided in any written license agreement from Rackspace, the furnishing of this document does not give you any license to patents, trademarks, copyrights, or other intellectual property.