National Institute of Standards and Technology

NIST US Government Cloud Computing Technology Roadmap Volume III

Technical Considerations for USG Cloud Computing Deployment Decisions

---

Document NIST XXX-0XX

First Working Draft

October 31, 2011

DISCLAIMER

This document has been prepared by the National Institute of Standards and Technology (NIST) and describes standards research in support of the NIST Cloud Computing Program.

Certain commercial entities, equipment, or material may be identified in this document in order to describe a concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that these entities, materials, or equipment are necessarily the best available for the purpose.

Table of Contents

1 Executive Summary 4
1.1 Overview 4
1.2 Methodology 5
1.3 Output 7
2 Purpose and Scope 9
2.1 USG Cloud Computing Technology Roadmap Purpose 9
2.2 Document Organization 9
3 Review of the Strategy Decision Framework 12
3.1 Selecting services to move to the cloud 12
3.2 Provisioning cloud services effectively 13
3.3 Managing services rather than assets 13
4 Mapping USG High Priority Requirements 15
4.1 Selecting services to move to the cloud 15
4.2 Provisioning cloud services effectively 16
4.3 Managing services rather than assets 17
5 Generating an Effectively Complete Set of Illustrative Case Example Categories 19
5.1 Applying the Reference Architecture (Service Models/Actors) 19
5.2 Applying the Concept of Roles 20
5.3 Applying the Concept of Tasks 20
5.4 Illustrative Case Example Categories Matrix Analysis including Valid/Not Valid Case Disposition 20
5.5 Redundancy Analysis through Matrix Flattening 22
5.6 Resultant Effectively Complete Set of Illustrative Case Example Categories 26
6 Mapping Case Example Categories, Decision Framework, and Technology Roadmap 33
6.1 Just Reading Application; Info or Docs Lookup probably using Browser (Public) 33
6.2 Agency Info Lookup using Browser (Private) 39
6.3 Transactional Application like Apply for Passport, File Taxes, USPS Email Service, probably using Browser (Public) 46
6.4 Agency Application like Issue Passport, Investigate Taxes, Agency Email, or for Calculations (Private) 57
6.5 Programmable Application like future data.gov using tools, plug-ins or script (Public) 59
6.6 Agency developers making applications (Private) 60
6.7 Agency Developers will use Brokers to access resources from other agencies (Private) 61
6.8 Programming Platform - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility (Public) 62
6.9 Developers will use Brokers to access resources from many agencies (Public) 63
6.10 Agency Research Facility, for Calculations (Private) 64
6.11 Servers and Storage - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility (Public) 66
6.12 Agency Servers and Storage, Classic IaaS (Private) 67
7 Some Infrastructure as a Service Developer Notes 69
7.1 VM Considerations – Units of Measurement, Compatibility 69
7.2 Storage – BLOBs, Blocks/Drives 69
7.3 Load Balancers / Traffic Manager 70
7.4 Virtual Network and Virtual Private Cloud 70
7.5 IP Addresses and Domain Name Service 70
7.6 Content Delivery Network 71
7.7 High Performance / Grid Computing 71
7.8 Compliance 72
8 Some Platform as a Service Developer Notes 73
8.1 Database 73
8.2 Message Queue / Service Bus 73
8.3 Caching 73
8.4 Access Control / Single Sign On 74
8.5 Notification / Email 74
9 Enabling Brokers - Cloud Interoperability 75
10 Acronyms 79
11 Glossary 82

1 Executive Summary

1.1 Overview

Volume III, Release 1.0 (Draft) - Technical Considerations for USG Cloud Computing Deployment, builds on and is released concurrently with the first two volumes of the USG Cloud Computing Technology Roadmap: Volume I: High Priority Requirements to Further USG Agency Cloud Computing Adoption, and Volume II: Useful Information for Cloud Adopters.

Volume III:

IS FOR: USG agency **technical planning and implementation teams** who are charged with responsibility for a Cloud Computing project;

HAS A GOAL TO: inform decision makers regarding questions and decision factors in the context of specific types of **Cloud Computing use cases and Interoperability**; and

DESCRIBES HOW: to **leverage the Federal Cloud Computing Strategy** Decision Framework for Cloud Adoption and the collaborative NIST Cloud Computing Program work completed in the May 2010 through July 2011 timeframe.

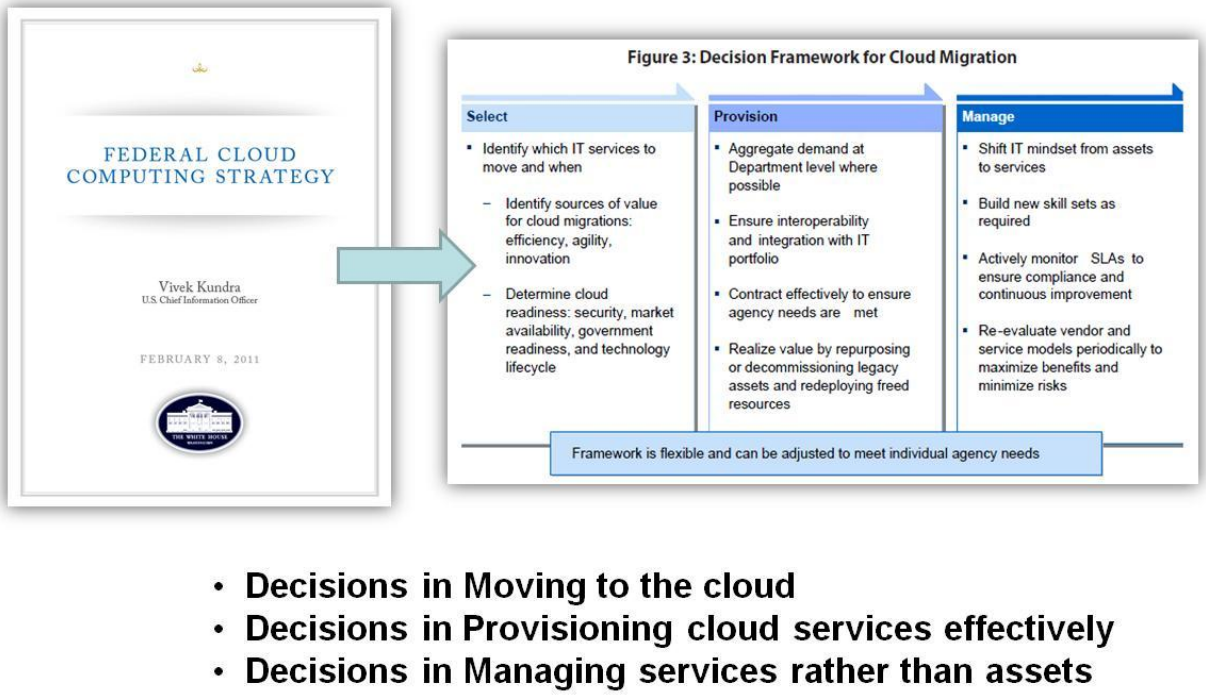


Figure 1: The Decision Framework for Cloud Migration

1.2 Methodology

Volume III accomplishes its goals by choosing "*Case Examples (to) Illustrate Framework* " (as they are called in the Federal Cloud Computing Strategy) and analyzing them against the Decision Framework for Cloud Adoption.

To ensure a comprehensive set of *Illustrative Case Examples* (as we will now call them), we introduce a proposed methodology to characterize cloud applications. The NIST Cloud Computing Reference Architecture (NIST SP 500-292) details "Cloud Provider – Service Models." A Cloud Provider gives a Cloud Consumer the choice of three (3) different kinds of service models:

1. SaaS,
2. PaaS,
3. IaaS,

A Cloud Consumer may also procure the same three service models from a Cloud Broker as well.

We then contrast the models with a set of "*Roles*" and "*Tasks*" which are proposed as part of the proposed methodology. This document introduces the ***simplest*** implementation of this methodology, and so uses very simple *Roles* and *Tasks*.

Roles, in this context, are groupings of users defined by organizational affiliations and, implicitly, by corresponding access rights that users possess while they consume from a cloud. There are three (3) *Roles*

1. US citizens/residents,
2. US government employees/contractors,
3. "Outsiders" [non-US government citizens/residents or employees/contractors (Local, State, or Foreign)].

Tasks are what a *Role* is trying to accomplish by using a cloud. There are five (5) *Tasks*

1. an Application Spectator,
2. an Application Operator,
3. an Application Developer,
4. a Researcher,
5. and a System Administrator.

When we combine these elements in permutations to define unique alternatives (Service Models x Roles x Tasks) the result is a 4 x 3 x 5 matrix, potentially yielding 60 Illustrative Case Examples. After analyzing the alternatives and eliminating cases of redundancy and non-applicability, there are 24 identified Illustrative Case Examples which "make sense."

There is significant commonality between these 24, and so the set is further reduced to 12 Illustrative Case Examples:

1. (Public) Just Reading Application; Info or Docs Lookup using Browser
2. (Private) Agency Info Lookup using Browser
3. (Public) Transactional Application such as Apply for Passport, File Taxes, USPS Email Service, probably using Browser
4. (Private) Transactional Agency Application like Issue Passport, Investigate Taxes, Agency Email, or for Calculations
5. (Public) Programmable Application like future data.gov using tools, plug-ins or script
6. (Private) Agency developers making applications to be (1), (2), (3), (4), or (5)
7. (Private) Agency Developers may use Brokers to access resources from other agencies
8. (Public) Programming Platform - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility
9. (Private) Developers may use Brokers to access resources from many agencies
10. (Private) Agency Research Facility, for Calculations
11. (Public) Servers and Storage - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility
12. (Private) Agency Servers and Storage, Classic IaaS

An initial assumption proposed is that the majority of Cloud Computing use cases can be practically categorized into one of these 12 in order to gain significant insight into the technical factors that need to be considered in cloud deployment decisions. A rules methodology is presented to help categorize the Cloud Implementer's application. If the USG decision maker can identify the Service Model (SaaS, PaaS, etc), Roles (Citizen, Govt Employee, Outsider), and a bit about the detail of the application Tasks (Application User, Virtual Server or Storage Admin, Programmer, etc), the expectation is that he or she can find one of the 12 use case scenarios that fit.

This document proposes and illustrates a methodology for considering the 12 Illustrative Case Examples at a deep-dive level against the Decision Framework for Cloud Adoption. Relevant to the three phases of the Decision Framework for Cloud Adoption[1], this document includes a detailed discussion regarding a number of issues that relate to important technical deployment decisions.

In partiular, this document considers

- the knowledge base of NIST Cloud Computing,
- other Federal documents for further study, and

[1] Office of Management and Budget, U.S. Chief Information Officer, "Federal Cloud Computing Strategy," Feb. 8, 2011. Online: www.cio.gov/documents/Federal-Cloud-Computing-Strategy.pdf.

- USG Cloud Computing Technology Roadmap requirements which have been identified as high priorities to further USG Cloud Computing Technology Adoption',

These are mapped to the phases of the Decision Framework for Cloud Adoption, as illustrated below:

Figure 2: Applying the Decision Framework to the NIST Knowledge Base, Requirements, and Case Examples

1.3 Output

The output from this process is framed in a context of "Smart Questions/Needed Decisions" about "your" (a given) Cloud Computing project. The premise for this approach is the assumption that Better Questions = Better Planning, Better Vendor Discussions, and Better RFP Authoring.

Figure 3: The Decision Framework for Cloud Migration

2 Purpose and Scope

2.1 USG Cloud Computing Technology Roadmap Purpose

Volume I of the USG Cloud Computing Technology Roadmap, High Priority Requirements to Further USG Agency Cloud Computing Adoption, provides an overview of the NIST Cloud Computing program and collaborative initiative to build the roadmap.

Volume II of the USG Cloud Computing Technology Roadmap, Useful Information for Cloud Adopters, describes the analysis that drove the rationale for the requirements introduced in Volume I.

Volume III of the USG Cloud Computing Technology Roadmap document, Technical Considerations for USG Cloud Computing Deployment Decisions focuses on the tactical aspects of applying the technical work completed as part of the effort and presents a strawman methodology to do so. Volume III is designed to serve as a guide for decision makers who are planning and implementing cloud computing solutions, in the form of explaining how the technical work and resources in NISTs' work can be applied, consistent with the Federal Cloud Computing Strategy "Decision Framework for Cloud Migration". Volume III presents the methodology and walks through a representative sample of common cloud computing planning and deployment scenarios. The expectation is that the methodology will be refined in phase 2 of the NIST Cloud Computing program, and applied to the remainder of the 12 exemplar use cases, leveraging public working groups and the Federal Cloud Computing Standards and Technology working group. Volume III also provides several "pointers and tips" to assist the IT decision makers and engineers.

Volume III aims to evolve as a useful technical guidance tool. The intent is to support a consistent logical cookbook-like approach that can be applied by USG agencies and others in the technical journey from concept to deployment. Volume III is intended to assist the IT decision makers in identifying relevant NIST Cloud Computing publications for a given cloud scenario. The goal is to provide a technical reference for those who are trying to define a "to-do list" of questions to be answered, important specifications/standards to be considered and selected, and cloud characteristics to be enumerated, relevant to a given Cloud Computing use-case. It also provides a long term vision for Cloud Interoperability.

2.2 Document Organization

It is assumed that the reader has reviewed volume I of this document before reading Volume III.

Volume III is organized in the form of a "decision tree" journey, using classic Cloud "Use Cases" as starting points. This process is informed by specific NIST documents and sections of documents, and external resources, especially relevant topics, and the context behind them. For each Use Case, this document journeys through the NIST Cloud Computing Reference Architecture, Technical Use Cases (SAJACC), Security, and Standards Roadmap. The journey

highlights specific areas of architecture to study (with implications), technical capabilities of the architecture which need to be specified (essentially component technical use cases), relevant security cases, impediments, and compliance pointers, and important standards areas which need to be considered.

The remainder of Volume III is organized as follows:

Section 3 reviews the Federal Cloud Computing Strategy Decision Framework, issued by the US CIO in February 2011.

Section 4 reviews and maps the USG Cloud Computing Technology Roadmap High Priority Requirements (from Volume 1) to the phases of the *Decision Framework for Cloud Computing*.

Section 5 focuses on the Federal Cloud Computing Strategy "Case Examples to Illustrate (the) Framework" -- the *Illustrative Case Examples* presented in this document. This section presents a decision table based algorithm which is used to generate an *Effectively Complete Set of Illustrative Case Example Categories.* In the Federal Cloud Computing Strategy, three examples were chosen to illustrate the Strategy. It is constructive to choose particular examples, but these are not likely to be the exact cases facing USG agency technical planning and implementation teams. This document seeks to support the strategy by defining a broader coverage set, the "*Effectively Complete Set*" of *Illustrative Case Example Categories*. This section leverages the NIST Cloud Computing Reference Architecture Service Models and Actors concepts to specify fundamental orthogonal characteristics of use cases as part of the decision table algorithm. This approach is consistent with the goal of the Federal Cloud Computing Strategy Case examples and is proposed to cover most of the current common use cases for cloud computing.

Section 6 applies a methodology similar to the Federal Cloud Computing Strategy, *Case Examples to Illustrate Framework*. As a proof of concept, this section illustrates the methodology for a subset of the *Effectively Complete Set of Illustrative Case Example Categories.* For this subset, section 6 covers the three stages as described in the Federal strategy, (Selecting, Provisioning, and Managing) and connects key elements of the NIST cloud computing internal and collaborative efforts. The elements include the reference architecture and taxonomy, the targeted business use cases and SAJACC technical use cases, the standards and gap analysis, and the security requirements and mitigation, and the USG Cloud Computing Technology Roadmap requirements defined in Volumes I and II. Section 6 walks through these elements for this initial proof-of-concept subset drawn from the 12 case examples. For this proof-of-concept set, section 6 delivers a list which systematically highlights key technical questions and requirements which is recommended for consideration by USG IT decision makers to support specification and deployment of cloud services.

Section 7 presents additional considerations for Infrastructure as a Service projects. These are offered as "pointers and tips" for developers to better understand possible issues in moving an application to the Cloud, and questions to ask vendors and technology providers about that area of technology.

Section 8 presents additional considerations for Platform as a Service projects. These are offered as "pointers and tips" for developers to better understand possible issues in moving an

application to the Cloud, and questions to ask vendors and technology providers about that area of technology.

Section 9 expands on the topic of Brokers and Cloud Interoperability, and what it will really take to make an infrastructure of Brokers actually work. Although the "actor" of Broker has been defined by NIST (as described below) and this document details considerations around programmatic access to such a facility, the eco-system for Broker driven Interoperability of Clouds does not exist yet; the NIST view at this time still future oriented. This section discusses the analogy to routed IP NAPs, peering, and exchange points, and also root naming and trust (DNS and certificates), where the global "Internet" is created; a framework of various Cloud technologies and infrastructure (including Brokers) must be created to enable a global Intercloud.

Section 10 lists the acronyms used in the document.

Section 11 is a glossary of terms used in the document.

3 Review of the Strategy Decision Framework

As illustrated above in Figure 1, the Federal Cloud Computing Strategy introduces a Decision Framework for Cloud Adoption. It defines three stages for a cloud project, in terms of moving or creating applications on a cloud:

1. Selecting services to move to the cloud
2. Provisioning cloud services effectively
3. Managing services rather than assets

3.1 Selecting services to move to the cloud

The strategy first calls for the project to ***Identify sources of value* – i.e.,** why is the cloud platform being considered in the first place?

> **Efficiency**: Efficiency gains can come in many forms, including higher computer resource utilization due to the employment of contemporary virtualization technologies, tools that extend the reach of the system administrator, and hosting customer workloads using possibly lower cost physical resources.
>
> **Agility:** Many cloud computing efforts support rapid automated provisioning of computing and storage resources.
>
> **Innovation**: Agencies can compare their current services to contemporary marketplace offerings, or look at their customer satisfaction scores, overall usage trends, and functionality to identify the need for potential improvements through innovation.

The strategy next calls for the project to ***Determine cloud readiness* -** Agencies are directed to make risk-based decisions which carefully consider the readiness of commercial or government providers to fulfill their Federal needs.

> **Security Requirements**: Federal Government IT programs must satisfy a range of security requirements. Federal Information Security Management Act (FISMA) requirements include but are not limited to: compliance with Federal Information Processing Standards agency specific policies; Authorization to Operate requirements; and vulnerability and security event monitoring, logging, and reporting. Depending on the nature of the data being processed or stored, additional requirements may apply, such as the Sarbanes-Oxley Act (SOX), the Payment Card Industry Data Security Standard (PCI DSS), or the Health Information Protection and Accountability Act (HIPAA). It is essential that the decision to apply a specific cloud computing model to support mission capability considers such requirements.
>
> **Service characteristics**: Service characteristics can include service interoperability, availability, performance, performance measurement approaches, reliability, scalability, portability, vendor reliability, and architectural compatibility.

Market Characteristics: Agencies are advised to consider the cloud market competitive landscape and maturity, including commercial and government-provided cloud services. Considerations include the degree to which cloud markets are sufficiently competitive and not dominated by a small number of players.

Network infrastructure, application and data readiness: Before migrating to the cloud agencies must ensure that the network infrastructure can support the demand for higher bandwidth and that there is sufficient redundancy for mission critical applications.

Government readiness: In addition, agencies are advised to consider whether the organization is pragmatically ready to migrate mission data and operations to the cloud model.

Technology lifecycle: Agencies are directed to consider the status of technology services (and the underlying computing assets) with respect to lifecycle maturity.

3.2 Provisioning cloud services effectively

After appropriate applications are selected, as they are provisioned, agencies are advised to ***rethink their processes as provisioning services*** - rather than simply contracted assets:

Aggregate demand: When considering "commodity" and common IT services, agencies should consider the advantages of pooling purchasing power by aggregating demand to the greatest extent possible before migrating services to the cloud.

Integrate services: Agencies are advised to ensure that the cloud computing IT services are effectively integrated into their wider application portfolio. Agencies are advised to evaluate architectural compatibility of the planned cloud services and other critical applications.

Contract effectively: Agencies are advised to take measures to ensure that contracted services ensure portability. Service level agreements (SLAs) need to address security, continuity of operations, and service quality.

Realize value: Steps need to be taken during migration to fully realize the expected value of cloud services.

3.3 Managing services rather than assets

When applications are running on the cloud, they must be managed differently than in other models. Cloud computing requires a service-based orientation rather than an asset-based focus when completing the fundamental management decisions that are common to all IT services:

Shift mindset: Organizations need to consciously re-orient the focus of all parties involved – providers and consumers – to services. Effective management of services places a focus on customer-focused output metrics (e.g., SLAs) as opposed to IT-focused input metrics (e.g., number of servers).

Actively monitor: SLAs need to be actively tracked and vendors accountable for failures. Cloud Computing does not eliminate the need to evolve an effective security posture in consideration of emerging security threats.

Re-evaluate periodically: Agencies need to periodically re-evaluate services and providers to maximize efficiency, agility, and innovation.

4 Mapping USG High Priority Requirements

The USG Cloud Computing Technology Roadmap Volume I, released as a draft special publication concurrently with this working document, lists the High Priority Requirements to Further USG Agency Cloud Computing Adoption as follows:

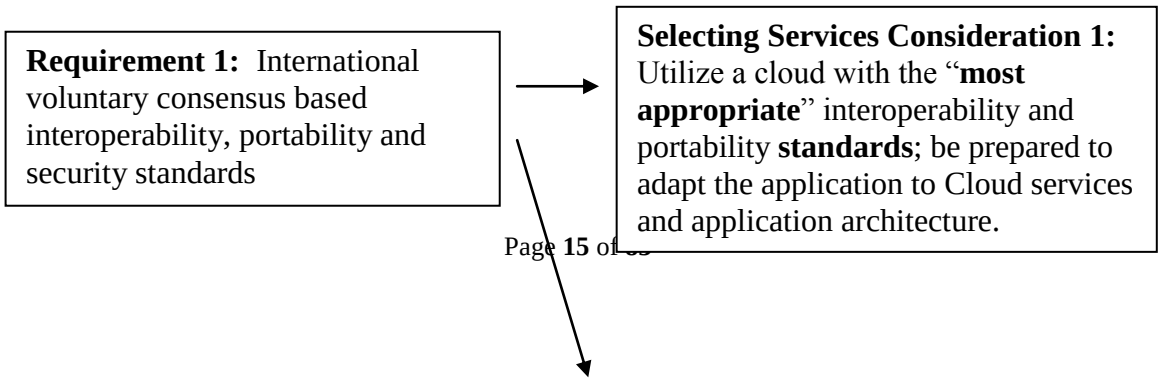
Requirement 1: International voluntary consensus based interoperability, portability and security standards
Requirement 2: Solutions for high priority Security Requirements
Requirement 3: Technical specifications to enable development of consistent, high quality Service Level Agreements
Requirement 4: Clearly and consistently categorized cloud services
Requirement 5: Frameworks to support seamless implementation of federated community cloud environments
Requirement 6: Technical security solutions which are de-coupled from organizational policy decisions
Requirement 7: Defined unique government regulatory requirements, technology gaps, and solutions
Requirement 8: Collaborative parallel strategic "future cloud" development initiatives
Requirement 9: Defined and implemented reliability design goals
Requirement 10: Defined and implemented cloud service metrics

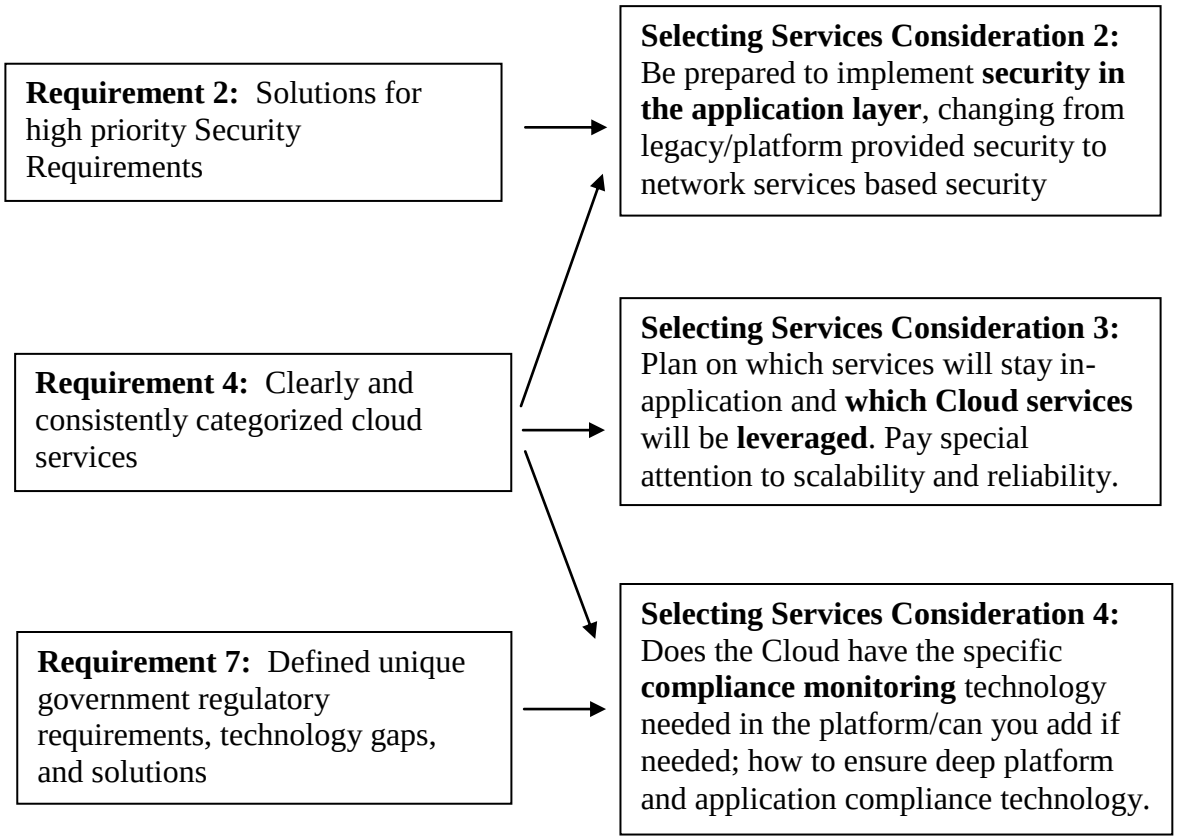
These requirements relate to elements of the Decision Framework for Cloud Adoption. The requirements are presented here as **"platform consideration issues"** in the context of the three phases in the Decision Framework for Cloud Adoption.

It is important to note that there are strategic and tactical objectives that apply to each of the requirements above. The requirements are presented in the context that they need to be met to accelerate USG adoption of cloud computing; the requirements are not presented as prerequisite to any specific cloud adoption or deployment. In the context of the decision framework, USG agencies are advised to take assess the extent to which the requirements would facilitate a specific proposed cloud computing deployment under consideration.

4.1 Selecting services to move to the cloud

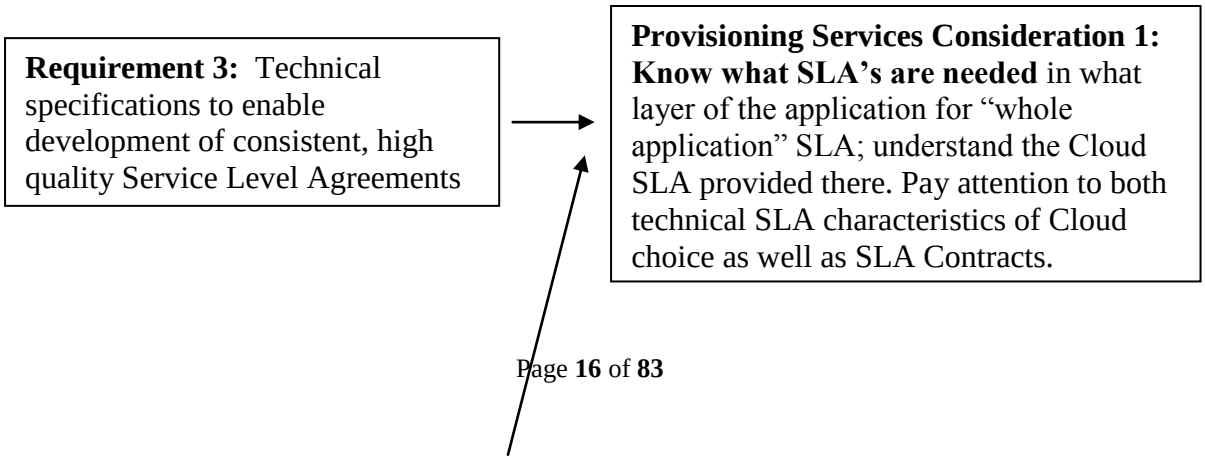
Requirements 1, 2, 4, and 7 map to the following "Decision Framework Considerations" for Selecting Services to move to the Cloud:

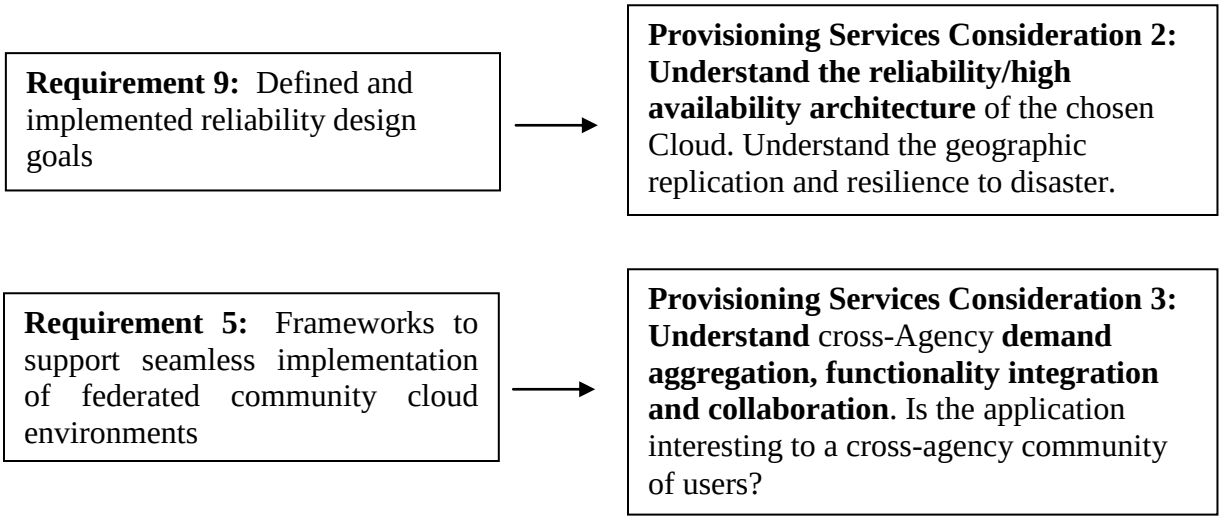




4.2 Provisioning cloud services effectively

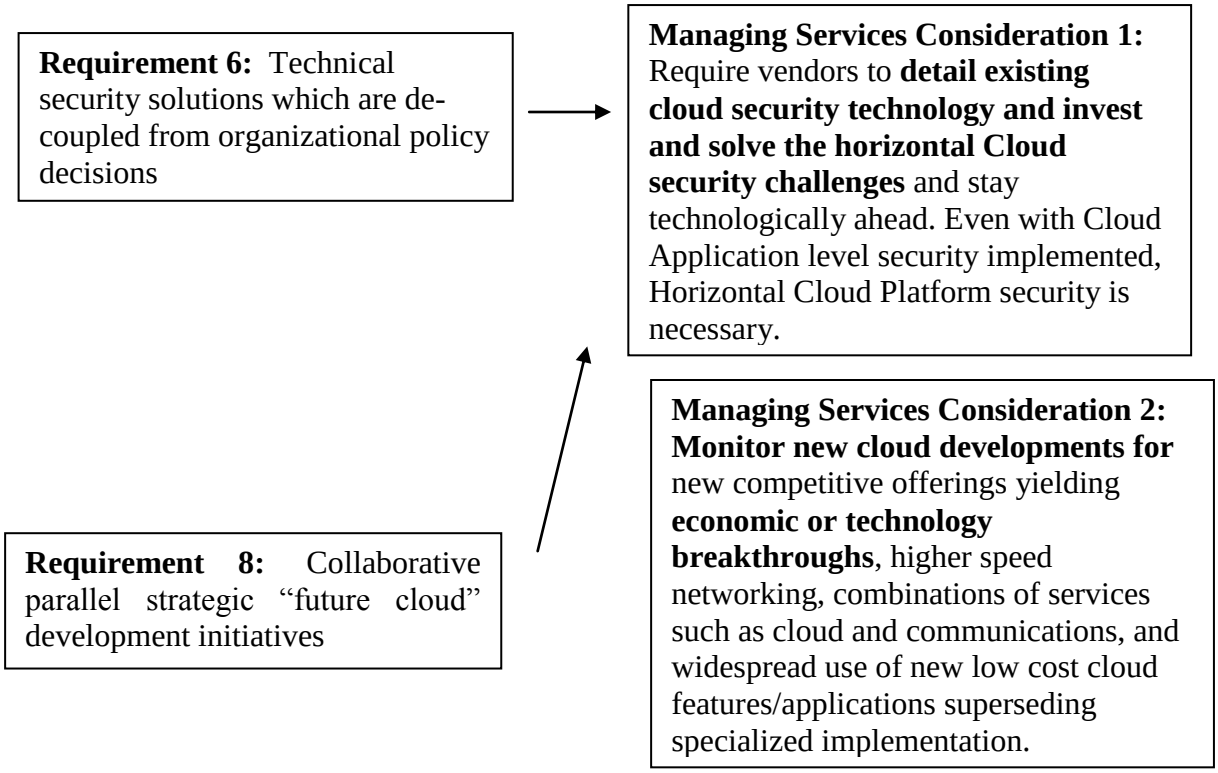
Requirements 3, 5 and 9 map into the following "Decision Framework Considerations" for Provisioning cloud services effectively.





4.3 Managing services rather than assets

Requirements 6, 8, and 10 map into the following "Decision Framework Considerations" for Managing services rather than assets:



Requirement 10: Defined and implemented cloud service metrics

→

Managing Services Consideration 3: Require **vendors to compete for business based on standard service metrics**. Push for constant comparisons. Monitor the needs of the Application and understand its costs in units of standard metrics

5 Generating an Effectively Complete Set of Illustrative Case Example Categories

5.1 Applying the Reference Architecture (Service Models/Actors)

As detailed in Volume II of the NIST SP 500-293 document, the NIST cloud computing reference architecture is a logical extension to the NIST cloud computing definition. It is a generic high-level conceptual model that is a powerful tool for discussing the requirements, structures, and operations of cloud computing. The model is not tied to specific vendor products, services or reference implementations, nor does it define prescriptive solutions that inhibit innovation. The reference architecture serves as a starting point to frame the discussion of cloud computing for CIOs, IT program managers and procurement officials.

This document introduces a methodology to characterize cloud applications in order to define a comprehensive set of *Illustrative Case Examples.*

Figure 4 depicts the NIST Reference Architecture service models and Broker as described in SP 500-292.

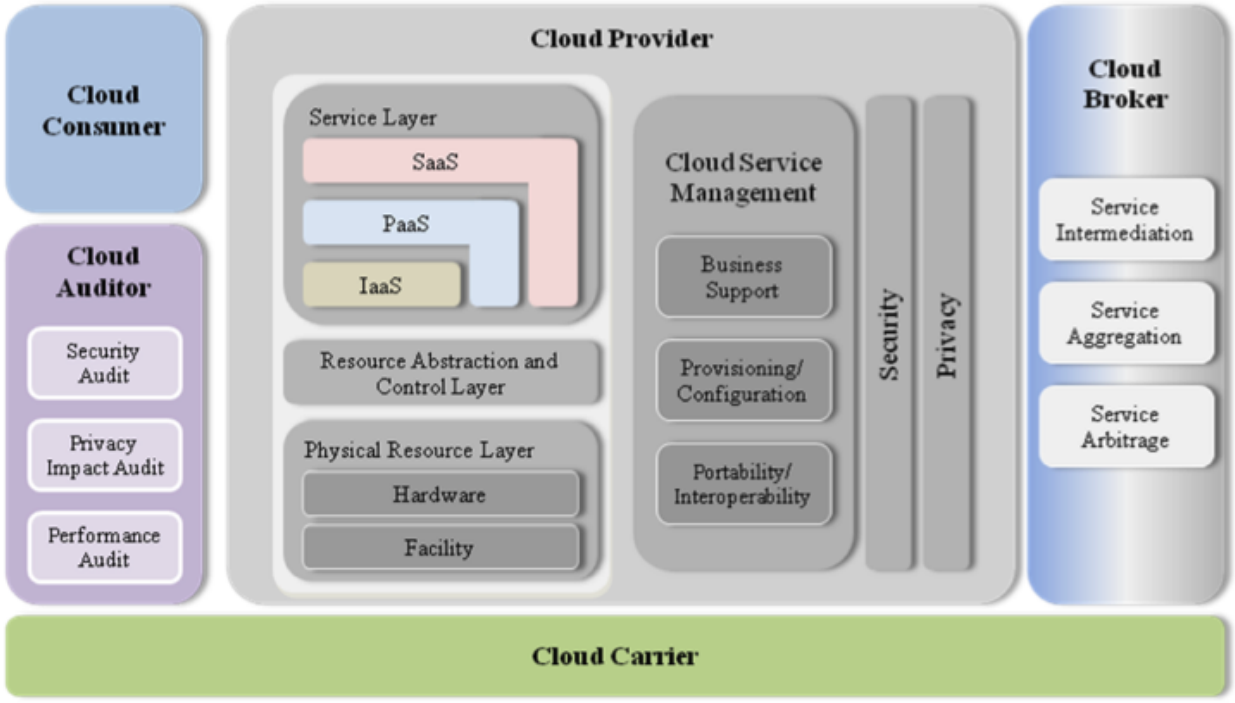


Figure 4: The Reference Architecture from SP500-292

5.2 Applying the Concept of Roles

A set of "*Roles*" and "*Tasks*" are introduced as part of the methodology presented here. This document introduces the ***simplest*** implementation of this methodology, and so uses very simple *Roles* and *Tasks*.

Roles, in this context, are groupings of users defined by organizational affiliations and, implicitly, by corresponding access rights that users possess while they consume from a cloud. There are three (3) *Roles:*

1. US citizens/residents,
2. US government employees/contractors,
3. "Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].

5.3 Applying the Concept of Tasks

Tasks are what a *Role* is trying to accomplish by using the cloud. There are five (5) *Tasks*

1. an Application Spectator,
2. an Application Operator,
3. an Application Developer,
4. a Researcher,
5. a System Administrator.

5.4 Illustrative Case Example Categories Matrix Analysis including Valid/Not Valid Case Disposition

In considering all possible combinations of roles, tasks and service models, there are 60 (sixty) potential categories:

Roles	Tasks	Service Models			
		SaaS	*PaaS*	*IaaS*	*Broker*
US citizens/residents	Application Spectator	(1) (Public) Just Reading App; Info or Docs Lookup probably using Browser	n/a	n/a	n/a
US government employees/contractors	Application Spectator	(2) (Private) Agency Info Lookup using Browser	n/a	n/a	n/a
"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	Application Spectator	Same as (1)	n/a	n/a	n/a

US citizens/residents	Application Operator	(3) (Public) Transactional App like Apply for Passport, File Taxes, USPS Email Service, probably using Browser	n/a	n/a	n/a
US government employees/contractors	Application Operator	(4) (Private) Agency App like Issue Passport, Investigate Taxes, Agency Email, or for Calculations	n/a	n/a	n/a
"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	Application Operator	Same as (3)	n/a	n/a	n/a

US citizens/residents	Application Developer	n/a	(5) (Public) Programmable App like future data.gov using tools, plug-ins or script	n/a	n/a
US government employees/contractors	Application Developer	n/a	(6) (Private) Agency developers making applications to be (1), (2), (3), (4), or (5)	If they are re-factoring the application, using any Cloud APIs, same as (6); if they are redeploying using just virtual servers/storage it's same as (12)	(7) (Private) Agency Developers will use Brokers to access resources from other agencies [implicit, explicit, open issues]
"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	Application Developer	n/a	Same as (5)	n/a	n/a

US citizens/residents	Researcher	n/a	(8) (Public) Programming Platform - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility	If they are re-factoring the application, using any Cloud APIs, same as (8); if they are redeploying using just virtual servers/storage it's same as (11)	(9) (Private) Developers will use Brokers to access resources from many agencies [implicit, explicit, open issues]
US government employees/contractors	Researcher	n/a	(10) (Private) Agency Research Facility, for Calculations	If they are re-factoring the application, using any Cloud APIs, same as (6); if they are redeploying using just virtual servers/storage it's same as (12)	Same as (7)
"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	Researcher	n/a	Same as (8)	n/a	Same as (9)

US citizens/residents	System Administrator	n/a	n/a	(11) (Public) Servers and Storage - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility	n/a
US government employees/contractors	System Administrator	n/a	n/a	(12) (Private) Agency Servers and Storage, Classic IaaS	Same as (7)
"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	System Administrator	n/a	n/a	Same as (11)	Same as (9)

Figure 5: Preliminary Case Examples Category Matrix

The Case Examples Category Matrix above uses color coding to present the results of an analysis of each candidate category. Candidates which are not viable (based on an informal analysis) are colored light red. Candidates which represent a viable Case Example Category are colored light green.

Each viable Case Example Category is numbered.

The chart also indicates whether a Category primarily reflects a Public Cloud or Private Cloud deployment. In the context of the NIST Cloud Computing Reference Architecture, Public Cloud as represented here includes all the physical aspects of Hybrid Cloud, Community Cloud, and Public Cloud deployments which are "outsourced"; Private Cloud in the context of the analysis includes the physical aspects of Hybrid Cloud, Community Cloud, and Private Cloud deployments which are On-Site.

A summary statement or label is presented for each Category.

In the case where a Category is functionally equivalent to another Category in the Case Examples Category Matrix, a notation indicates the reference number of the redundant Category.

Although the categorization of the table cells is based on careful reasoning, it should be viewed as preliminary.

5.5 Redundancy Analysis through Matrix Flattening

The next step re-organizes the Case Examples Category Matrix into a flattened list, grouping equivalent Categories together.

This provides a mechanism to review common features within:

(1) (Public) Just Reading Application; Info or Docs Lookup probably using Browser

SaaS	Application Spectator	US citizens/residents
SaaS	Application Spectator	"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].

(2) (Private) Agency Info Lookup using Browser

SaaS	Application Spectator	US government employees/contractors

(3) (Public) Transactional Application like Apply for Passport, File Taxes, USPS Email Service, probably using Browser

SaaS	Application Operator	US citizens/residents
SaaS	Application Operator	"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].

(4) (Private) Agency Application like Issue Passport, Investigate Taxes, Agency Email, or for Calculations

SaaS	Application Operator	US government employees/contractors

(5) (Public) Programmable Application like future data.gov using tools, plug-ins or script

PaaS	Application Developer	US citizens/residents
PaaS	Application Developer	"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].

(6) (Private) Agency developers making applications to be (1), (2), (3), (4), or (5)

PaaS	Application Developer	US government employees/contractors	
IaaS	Application Developer	US government employees/contractors	If they are re-factoring the application, using any Cloud APIs, same as (6); if they are redeploying using just virtual servers/storage - same as (12)
IaaS	Researcher	US government employees/contractors	If they are re-factoring the application, using any Cloud APIs, same as (6); if they are redeploying using just virtual servers/storage - same as (12)

(7) (Private) Agency Developers will use Brokers to access resources from other agencies [implicit, explicit, open issues]

Broker	Application Developer	US government employees/contractors
Broker	Researcher	US government employees/contractors
Broker	System Administrator	US government employees/contractors

(8) (Public) Programming Platform - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility

PaaS	Researcher	US citizens/residents	
IaaS	Researcher	US citizens/residents	If they are re-factoring the application, using any Cloud APIs, same as (8); if they are redeploying using just virtual servers/storage - same as (11)
PaaS	Researcher	"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	

(9) (Public) Developers will use Brokers to access resources from many agencies [implicit, explicit, open issues]			
Broker	Researcher	US citizens/residents	
Broker	Researcher	"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	
Broker	System Administrator	"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	
(10) (Private) Agency Research Facility, for Calculations			
PaaS	Researcher	US government employees/contractors	
(11) (Public) Servers and Storage - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility			
IaaS	System Administrator	US citizens/residents	
IaaS	System Administrator	"Outsiders" [non-US government citizens/residents or employees/contractors (Local/Municipal, State, or Foreign)].	
IaaS	Researcher	US citizens/residents	If they are re-factoring the application, using any Cloud APIs, same as (8); if they are redeploying using just virtual servers/storage - same as (11)
(12) (Private) Agency Servers and Storage, Classic IaaS			
IaaS	System Administrator	US government employees/contractors	
IaaS	Application Developer	US government employees/contractors	If they are re-factoring the application, using any Cloud APIs, same as (6); if they are redeploying using just virtual servers/storage it's same as (12)
IaaS	Researcher	US government employees/contractors	If they are re-factoring the application, using any Cloud APIs, same as (6); if they are redeploying using just virtual servers/storage - same as (12)

Figure 6: Case Examples Category Matrix as Flattened List

5.6 Resultant Effectively Complete Set of Illustrative Case Example Categories

The results from this approach are 12 Illustrative Case Example Categories, proposed here as an Effectively Complete Set.

The following are more detailed descriptions of each Illustrative Case Example Category.

5.6.1 (#1) (Public) Just Reading Application; Info or Docs Lookup probably using Browser

Applications in this category are usable by anyone. They are read-only presentations (browser) of data. The concept is one of "a classic web page" presenting information on the pages, which may be static, looked up in a catalog or database, or dynamic based on user input. The discerning factor is there is no storing of data of any kind; there are no transactions as in, e.g., a "transfer balance" operation of an on-line banking application.

There may be a login or user registration system to display the data differently for users, for example, language or accessibility preferences, or other preferences. The application could also store history or favorites. For the purpose of this category that kind of storage is not considered to be "transactional", as in the "transfer balance" example. Generally, there are no places in the application for data entry and for "save" or "commit".

The user interface is most likely a Browser, but may also be a "thick" (must be downloaded) or specialized (not browser) user interface as well. For example Google Earth falls largely into this category for a PC. Specialized smart-phone "apps" will often be downloaded clients as well.

As anyone can access this application and the data it has access to, there are few security subsystems requirements (as far as access control is concerned). Security for service availability (combating Denial of Service), Intrusion detection, and Auditing (depending on the needs of the application) are required.

5.6.2 (#2) (Private) Agency Info Lookup using Browser

This application category is very much like #1 except the use of the application is limited to US government employees/contractors.

It is a read-only presentation (browser) of data. The look and feel is that of "a classic web page" presenting information which may be static in the pages, looked up in a catalog or database, or dynamic based on user input. As above, a discerning factor is there is no storing of data excepting authentication data and user preferences; there are no transactions.

However, in this case there *must* be a login or user registration system, along with operation over a secure connection – the content served up must be limited to US government employees/contractors.

Specific care must be put into application design to ensure that output is not implicitly (invisibly) stored on the client machine. For example use of cookies is likely not practical.

Again, the user interface is most likely a Browser, but can be a "thick" (must be downloaded) or specialized (not browser) user interface, or a specialized smart-phone "app".

5.6.3 (#3) (Public) Transactional Application like Apply for Passport, File Taxes, Free Email System, probably using Browser

This application category is usable by anyone. It is a bi-directional and interactive real application. Bi-directional means, there is a bi-directional flow of information (data) into and out of the application (and thus the cloud), which is handled "transactionally" by the cloud – data is considered to be persistent and the application is designed to work with the cloud to treat it as such. Interactive means, that all application functionality, including the invocation and context of the application, are on the request of the user. It is not considered a "background process" or "daemon"; overall the application components are associated with a user context and a user interface. The user expects the application (really, the cloud) to have flow, persistence, and transaction support in it, in a model which they understand.

For example, there may be the notion of "save" or "commit changes" in the application, and, the user expects that to be a transaction. There may be an intermediate storage of application state, of which most are familiar with the "Shopping Cart" metaphor.

The application allows for sophisticated form filling, queries and search, methods of organization such as folders, and so on. Applications such as "Apply for Passport", "File Taxes", or "Email" are all good examples. In the US Government case, "Apply for Passport" and "File Taxes" are realistic examples. Free email systems are reasonable examples.

The user interface is most likely a Browser, but can be a "thick" (must be downloaded) or specialized (not browser) user interface as well. Consider Email for example. There are Webmail versions (work with a browser), specialized User interfaces, and highly tuned smartphone applications. There are even API-level accesses to email supporting the underlying semantics.

Applications in this case are considered "SaaS" (Software as a Service) applications. While they might be a result of an set of modules deployed into an "IaaS" (Infrastructure as Service) or a "PaaS" (Platform as a Service) substrate, the applications do not expose the IaaS or PaaS system itself.

This category will have extensive support for the transactional data and the other data organizational features (searching, reports, etc) built into the substructure of the application. The application in this example, while potentially deep in functionality, is usually considered a *front-end* to the complete functionality and as such will likely have lots of "glue" middleware connecting it to the *processing* systems or *systems of record* (back-ends) for the application in question.

In the examples used above, when the front-end of "Apply for Passport" and "File Taxes" finish, an individual must complete the process. Likewise when an "Email" is sent, a *back-end* engine, which might be a mix of software and security hardware, actually sends the email.

5.6.4 (#4) (Private) Agency Transactional Application like Issue Passport, Investigate Taxes, Agency Email, or for Calculations

This application category is very much like #3 except it is limited to US government employees/contractors. These applications may be identical to #3, if one considers a call-center case, where the "Apply for Passport" or "File Taxes" capability is actually typed in by the US government employee/contractor during a phone contact. The applications may be the *back-end* part of an application where the individuals complete the process, using a User Interface.

The application may be operated by a person with a User Interface and be "interactive" as defined above.

Or the applications may be operated with or without a User Interface, and be "background" or "daemon" as defined above. In that latter case, the application may be as simple as a computational engine without a user interface; readying and writing files or databases configured by the application designer. This is in contrast to #3 where a user-interface is required ("SaaS"). In this case the applications are still SaaS, but, there may not be much of a User Interface – for example, just command line application through SSH. The point, with this distinction, is that because this category of example is facing inwards to the US government employees/contractors, the user interface may be subject to more variability than those used by the public. The user interface may be a specialized tool for a government scientist.

In this case, there *must* be a login or user registration system, along with operation over a secure connection – the content served up must be limited to US government employees/contractors.

Specific care must be put into application design to ensure that output is not implicitly (invisibly) stored on the client machine. For example use of cookies is likely not practical.

5.6.5 (#5) (Public) Programmable Application like future data.gov using tools, plug-ins or script

This application category is a departure from a fixed functionality application. In this case there is a sophisticated platform essentially for developing new applications using datasets, templates, tools, scripting, and a variety of interesting functionality including search, RSS feeds, and more, allowing users to create extremely sophisticated applications. Data.gov allows for application development with the following tools:

- **Interactive Datasets** - With interactive datasets, you can search, filter, and explore on the fly, through a web browser. You can also create charts and maps. APIs are available for developers.
- **Raw Data** - The Raw Data catalog provides an instant download of machine readable, platform-independent datasets.
- **Apps** - These are apps which include widgets, gadgets, tools, and RSS feeds.

When the public is accessing this type of platform there are presumably areas to store application metadata, queries, and data set identifiers. There is also a specialized application/UI Builder that allows access to assembling applications which have script andRSS feeds.

User login, community tools like Wiki, application repositories like Git – these allow submission of completed applications and then allow other users to access them. **This is a sophisticated category of an application that might be characterized as a domain-specific PaaS system**.

5.6.6 (#6) (Private) Agency developed applications

This application category is easy to envision. It is a "put this on the Cloud" type of assignment. The project may be one which states a broad objective such as "Make this (referring to an existing server-based application") into a cloud application" or "Put this (ibid.) on a Cloud". **There are two scenarios in moving applications to the Cloud model. These can be characterized as**:

1. No changes to applications' architecture; deploy server or client/server or Internet server modules on virtual servers in the cloud; and no changes (if any) to existing applications code. The goal is to take the application as it is and "move" it to cloud
2. Alternatively, given that the cloud is a new execution environment, choose to change some of the application to take advantage of the cloud platform in terms of scalability, reliability, efficiency, or other attributes.

The application category refers to the ability and desire to change the application - even if just to refactor it to be easier to group or to install the software on a cloud. This application category includes only those applications which are specifically engineered to the take advantage of the cloud in some way.

From the application developer side of the equation, this category represents a developer who has an account with a cloud provider, and is charged for the resources used in the cloud. The developer may access IaaS features like VM's, and Storage, or PaaS features like coding to message queues, coding to web server or load balancers, and running code in managed code containers.

The developer develops the application and deploys it in the cloud. The developer may use other tools to help run and manage that application.

The main point of interest in this category is that this is not a "free access public cloud platform", in that, the operator of the cloud, which might be a US government agency, or a public Operator, can assert billing, policies, limits, and enforce and manage certain behaviors or performance levels (perhaps based on costs associated with different levels). The risk of attack from the developer set of users is a factor to be evaluated.

Developers deploy their applications, which may "look like" a Cloud application described above in Categories #1, #2, #3, #4, or #5.

5.6.7 (#7) (Private) Agency Developers use Brokers to access resources from other agencies

This category is an extension or special case from Category #6 which introduces the idea of the Cloud Broker. In the context of the NIST Cloud Computing Reference Architecture, the Cloud Broker is an Actor with a unique set of activities and behaviors including the ability to engage in Provider to Provider interactions.

Where the integration of cloud services is too complex for cloud consumers to manage directly, a cloud consumer may request cloud services from a cloud broker instead of contacting a cloud provider directly. A cloud broker is an entity that manages the use, performance and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers.

In general, a cloud broker can provide value in three categories, as defined by the NIST Cloud Computing Reference Architecture:

1. Service Intermediation: A cloud broker enhances a given service by improving some specific capability and providing value-added services to cloud consumers. The improvement can be managing access to cloud services, identity management, performance reporting, enhanced security, etc.
2. Service Aggregation: A cloud broker combines and integrates multiple services into one or more new services. The broker provides data integration and ensures the secure data movement between the cloud consumer and multiple cloud providers.
3. Service Arbitrage: Service arbitrage is similar to service aggregation except that the services being aggregated are not fixed. Service arbitrage means a broker has the flexibility to choose services from multiple agencies. The cloud broker, for example, can use a credit-scoring service to measure and select an provider with the best score.

This category includes an application which calls a Cloud Broker, through an API for example, and is asking for resources outside of the local execution environment. This makes the application very different. Suddenly, the resources – including the data – that the application uses are dynamically provisioned *outside* of the domain of the application, to/from a cloud in potentially an entirely different location. This is beyond a "mash-up" of services, where the developer has wired together a "composite Application." Depending on the depth of functionality enabled, the Cloud Broker may serve up resources or data which the developer does not expect – going beyond an agency to other clouds and the Internet.

Security is a concern with a broker as different subject areas and levels of sensitivity can be seen by the broker and then served up to the application: a procedure and technical controls will need to be used to establish an appropriate level of trust in brokers that process sensitive information. Because the users in this case scenario are US government employees/contractors, the broker will be able to serve up resources, including infrastructure, or data/content, to the cloud user, through the broker methods. This broker will require careful compliance and security design. This will allow the developer to aggregate data from multiple places in the organization; for example, coordinate tax records, health records, personnel records, and others.

The application as described, running on a Government cloud, which could access broker services operating within the US government context might be used to develop breakthrough applications.

5.6.8 (#8) (Public) Programming Platform - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility

Applications in this category are largely similar to #6, except anyone can access the platform and build programs. An IaaS model is possible, as well as PaaS. This category however includes applications which have taken some steps to use a "native" cloud facility.

Although applications in this category are open to the public, there is a need for secure user accounts, for billing and also to impose limits, to store images and configurations and code, and so on. In other words, a full authentication, authorization, and accounting system should be put in place.

It is worth mentioning that in the implementation of an IaaS or PaaS platform, there are a variety of "styles" and "standards" to choose among. In a cloud implementation, the key design elements of a platform are tied together at many layers and build upon each other. For example, security, elasticity, networking, and storage techniques, approaches, and patterns, which are in the lowest levels of the IaaS part of a system, show through in higher layers such as a PaaS. It will not be possible to have a "neutral" or totally "ecumenical" cloud for public use; there will be several "flavors" of them in the absence of a strong and sustained standardization effort.

This document will not address the issues with the US Government being a provider of free and limitless computing resources, but the category is a valid one and there are various Public "Programming Platforms" available from parts of the Government for use in science, education, business grants, non-profits, and other categories of users.

5.6.9 (#9) (Public) Developers will use Brokers to access resources from many agencies

Applications in this category are very much like #7, but here the public is accessing the cloud and by utilizing a broker, which will then serve up further resources and data from other agencies, other outside domestic clouds, and even outside clouds in distant countries. These applications are much "larger" than themselves. Developers are encouraged to use PaaS techniques so that their applications have the "look and feel" of modern interfaces. They are also encouraged to use other APIs. Eventually the hope for utilizing API's to access federated services, eg, the Broker API.

The most important point to recognize in this category is that the use of brokers is an indirect federation, access, integration, or pointer to resources and capabilities which are outside of the local cloud and with whom the user has made a business arrangement encompassing authentication, authorization and accounting mechanism. Using the same brokers which serve up resources and data for US Government workers is clearly something that has to be done carefully as brokers so that there are no security compromises.

5.6.10 (#10) (Private) Agency Research Facility, for Calculations

This application category, on the surface, is very similar to category #6. In category #6, the applications are all developed by US government employees/contractors destined to become usable for a wide variety of people through a UI. In contrast, in this scenario, the US government employees/contractors develop applications for themselves.

As a result of the intended audience (the developer himself), the UI is usually not a time effective development project. Therefore, the types of applications developed under this scenario are usually computational in nature, with little or no UI.

5.6.11 (#11) (Public) Servers and Storage - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility

This application category, like category #8, allows for the straight IaaS model to be used for applications, while the US Government puts up the infrastructure to run them. Although applications in this category are in some ways open to the public, there is a need for secure user accounts, for billing but also to impose limits, to store images and configurations and code, etc. In other words a full authentication, authorization, and accounting system should be put in place.

Because it is so easy to use this model, and it aims to potentially replace many physical servers, it should be a concern that this might get to be a very large infrastructure.

5.6.12 (#12) (Private) Agency Servers and Storage, Classic IaaS

Applications in this category provide “classic IaaS” services as a “private cloud” to the agencies.

6 Mapping Case Example Categories, Decision Framework, and Technology Roadmap

This section presents sample implementation techniques with a discussion of issues to illustrate architectural characteristics and the use of the decision framework; it is not a complete list of all possible implementations nor do these implementations serve as recommended implementation approaches or choices; however in the discussion of issues, recommendations are presented as decision processes the implementer will find useful to follow..

6.1 Just Reading Application; Info or Docs Lookup probably using Browser (Public)

This application is most like a classic web application using a web server and a static content repository. The application may require a database for it's content, but that will be functioning in read-only mode.

6.1.1 Case Example Issues in Moving to the cloud

Efficiency

Prior to migrating to a cloud, it is likely the machines the application is running on are highly underutilized. For this application category, efficiency gains are immense, particularly as the scale-out of the application architecture was likely very challenging if this was a popular web site; usual scale approaches are inelastic and wasteful during times of low use.

As this is a public-facing application, it has the potential to be heavily loaded; as such one must pay special attention to two areas of this applications category (a) the load balancing needs of the application and (b) how much of the content which is being served is coming from a local repository (files, database) or how much is integrated in via outside HTTP references, SOA integration techniques, remote files, etc.

One approach is to take the whole OS, Web Server, and Database stack and put it on a machine image, and then put into the Cloud on a VM using IaaS. One would them manually include and configure into the machine image the content integration used. If load balancing or other application networking services were needed, the preferred method would be to use the cloud provider's built-in load balancing mechanism, whether that is their own hardware, or a software facility inherent to the Cloud. The load balancing scripts would need to be re-programmed to fit the Cloud system. Alternatively, one could use a software load balancer deployed manually, as a virtual appliance, in front of several replicates of the web server stack. Along with the pool of load balancers, one could deploy an elastic pool of front ends, sometimes called Servlets, sometimes called Web Workers. These modules would scale up and down in concert with the load balancers and all call the statically sized back end pool.

Overall, this is a classic IaaS approach. Without much (if any) refactoring, this application category can be very densely deployed onto a pool of VM's in an IaaS cloud and realize tremendous efficiency gains.

While this approach may be the most straightforward, there will be limits to the overall scalability of the application. In this configuration, the underlying application architecture has not changed, and there has been no "elasticity" built into that architecture. With some effort, and depending on the particular application server used, the back end pool size can be made dynamic as well, by adding elasticity in the form of dynamic cluster re-sizing. This can be an adequate technique without radically changing the application architecture. One would utilize a tool to monitor the back end pool saturation and dynamically add instances to the cluster (using scripts dependent on the application server). This would run ultimately into database connection pooling limits but will go quite far.

Another approach likely to yield even more efficiency gains is to use a PaaS platform which contains the web server stack, load balancer, applications server itself, and other elements integrated within the platform. Deploying as a PaaS application, however, may require a significant rewrite of the servicer-side logic to integrate with platform services. If one uses this approach, the inherent load balancing, the built-in elasticity at the HTTP server level, all will add significantly to the efficiency gains.

To the point regarding integration of external ("mash up") type information into the site, there are two concerns. One as mentioned previously is that since the application is now capable of scaling up significantly, are the data feeds integrated via HTTP references, SOA integration techniques, remote files, also able to handle more traffic? Can they traverse the firewall and security architecture of the applications running on the Cloud? And finally, a review of their "spoofability" is likely in order, to ensure that the application is as robust as possible. It might be a useful time to switch to SHTTP, e.g., or a more secure SOA integration (if the mash up target supports it).

Agility

For this application category, putting up new web sites without the need to provision physical hardware is a big win. It is a key aspect of cloud computing. Furthermore, if the cloud contains "instant web site" kinds of capabilities, and systems to manage DNS, WYSIWYG site editors with templates, then there is even more agility for a static content web application.

Look for *simple* web tools.

Innovation

For this application category, we look to the Cloud system to provide capabilities that enhance our static web site application. For example, can the cloud provide us a "search" capability easily? Can we get visitor statistics and even analytics to gain insight on what content readers are interested in? Can we mash-up into the static web site external content that would make this rather basic application category more alive and interesting?

Look for easy to incorporate new features for the web site, like discussion forums, search engines, feedback or review capabilities, favorites, photo tools and galleries, etc.

Security Requirements

For this applications category, the main concerns are high performance availability of the content and integrity of the data. Therefore, we are counting on the cloud platform the handle DDOS attacks, web site exploits and vulnerabilities, and all the "horizontal" security aspects which should be handled by the platform.

Service characteristics

For this application category, the primary capabilities needed are high availability and responsiveness.

Look for cloud platforms that contain attributes to help web sites have great performance and availability, like built in load balancers or CDN capability.

Market Characteristics

For this application category, there should be plenty of choices for platforms or service providers. Also there should be nice elasticity management tools which can help with larger or more complicated deployments. For this application, almost any well-connected cloud platform will work; there is no specific low latency or particular geographic location requirement.

Look for a cloud which is focused on cloud based site hosting; this will bring more web site oriented capabilities to the table, rather than straight VM's and storage.

Network infrastructure

For this application category, there is no complicated requirement for network infrastructure, other than the well-connectedness of the underlying cloud, and the multi-site availability requirements which one might have for site availability.

If the site is extremely popular, or there is a very strict uptime requirement, look for a cloud system with multiple sites or zones which can be controlled by a load balancer across multiple geographies.

Government readiness

For this application category, if the site is particularly complicated, or changes often, the new and remote development environment of the cloud may be a challenge to the existing organization.

Look for the ability to run the same content and site management tools that one runs in the current environment.

Technology lifecycle

For this application category, there are technology discontinuities to consider. Web sites keep using new technologies, even if they are static content sites.

For example, many sites have gone to technologies like HTML5, in order to present a more dynamic site. Also, in the re-programming to a new technology, the site can be made to work well on mobile devices automatically.

If this is of interest to the organization, shifting to the cloud may present an opportunity to adopt a new Web Server framework, and organizations can look specifically to those cloud systems which offer that as a built in system, as mentioned above, most likely a PaaS system.

Most Appropriate Standards

For this type of application, often the web site is built with a "stack" from a particular vendor, or using a particular family of open source or language, from top to bottom including the Operating System. A number of proprietary and Open Source software stacks exist; an analysis is needed to judge the appropriateness for the application. Stacks that support application functionality directly, without significant layers of translation, are appropriate.

Look for the "more native" cloud for the stack you are choosing.

Security in the Application Layer

For this application category, it is possible that the servers or the network infrastructure, that the Web Server originally existed within, offered significant security capability. It is possible that the cloud platform will offer less than that. For example, access logs may be something that was handled by the former server or network infrastructure, or access blocking from blacklisted addresses. These precise features may not be built into the cloud platform.

Understand what security your application needs, and if the cloud platform does not provide it for you, be prepared to implement it as part of your application.

Which Cloud Services Leveraged

For this application, it is likely that many clouds offer services to help make an efficient and high performance Web Site. Many of the services (like page caching) which used to be done in your web server may now be available as a cloud service.

Look specifically at all the features the cloud offers, and decide which ones to leverage. The rule of thumb is, leverage everything you can, even if the cloud service is much simpler or rudimentary that your equivalent service, try to use it if it meets application requirements.

The rationale for this approach is that the cloud services will evolve and improve, and they are not what is core to your web site. Monitor in the future what new services come out and try to remove functionality for your application and use the cloud.

Compliance Monitoring

For this application category, there is typically little compliance monitoring. For the transactional SaaS application case considered later, where e-commerce is occurring, or interactive medical care related activities, we will address this technology area there.

6.1.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

This application category is suitable for aggregated demand with a larger cloud based web hosting environment across Government agencies. A special infrastructure custom built for this category of application is generlly not needed, there are many service providers or even a shared infrastructure with another agency which has capacity to share.

Look for chances to aggregate capacity, for this simplest of application categories, this is the easiest to share infrastructure with others.

Integrate services

For this application, it should be analyzed how completely static, or whether there is any dynamic content served up, and where that comes from. The mechanism by which content is updated should be analyzed for best ways to accomplish the same thing in a cloud environment.

For example, a manual update of a database might have been done in the original web site. In the cloud, there would be an opportunity to automate this through some of the connectivity features which the cloud offers, such as message queues or SOA integration capabilities.

Contract effectively

For this application, there is a storage pool, which is largely fixed, and a variable pool of web servers, depending on load, and there is traffic, also depending on load. Look carefully at how the cloud provider charges for these aspects and make comparisons. Some PaaS providers will have very complicated charging rules for their advanced facilities such as message queues and load balancers.

Use modeling tools and the emerging cloud comparison services to look hard at the widely varying business models of clouds. Be prepared with an understanding of the size of the system as it exists and what load that can handle.

For this application, there is an opportunity to simplify the billing to not focus on number of VM's or servers but just amount of HTTP traffic. This will push the efficiency burden to the cloud provider.

Realize value

Once the cloud system is running and has been shown to meet application requirements, don't forget to decommission the old web site. Make plans to move groups of sites from same datacenters to cloud in order to shut down or decommission/un-contract from whole sections of datacenter.

Realize value by really making the jump.

Know what SLA's are needed

Understand the "pinch point" of the application. Is it the database? Is it the dynamic generation of pages by the web server? Is it the raw bandwidth you have to push through to serve the community? Know the SLA's you need and explore these with the cloud provider prior to migration.

Understand the reliability/high availability architecture

For this application, look at each elemental part, and ask the redundancy/replication/DR question. For example, for the storage, how many replicates are kept? Are the replicates all in the same zone/datacenter or are they geographically separated. Where are the servers? Pose the same question about geography. Does the load balancing involve more than one geography? For the network, understand the cloud connectivity, how many carrier networks can your content flow to directly from the cloud.

Understand demand aggregation, functionality integration and collaboration

For this application, it may very well be on a multi-agency platform. This provides an opportunity to aggregate content from other sites/agencies to better display meaningful information.

It is easier to mash-up web sites together once they are on the same infrastructure.

6.1.3 Case Example Issues in Managing services rather than assets

Shift mindset

For this application, the most important metric for understanding it's health, it's utility, and everything about it, is no longer about the web server itself, it's about how much HTTP this application serves to customers, and the content.

The promise of the cloud is to make uptime and capacity (storage/compute) less of a concern than in traditional deployments. What one needs to worry about is how fast the overall service to the end user is: how much service in the totality is being delivered - is it growing or shrinking.

Actively monitor

For this application, there are a rich set of web site statistics and analysis tools. Many of these will be built into the platform itself, especially if one is using the PaaS type of architecture.

Logging, statistics, analytics, all take up storage space and processing capacity. This might have been a limiting factor on the previous server architecture web site; a busy site generates large logs. Now there is huge capacity in the cloud and off the shelf applications which, especially for this application category, can provide huge insight into the site.

Monitor the SLA as far as site performance in the users eyes.

Re-evaluate periodically

Monitor expense, resources used, reports, and understand how this is all working on the cloud. Compare with your old server environment. Is it what you expected? Is the cost more or less than you imagined?

Detail cloud security technology; push for the horizontal cloud security solutions

For this application category, how much security did you need to build into the application? In which areas? Speak to the cloud platform people, and insist they begin to take on those areas to push this into the platform.

Look for roadmaps of improved "horizontal" security in the cloud platform.

Monitor new cloud developments for economic or technology breakthroughs

Take these performance characteristics to other cloud vendors or options and see if there is a faster/better/lower cost alternative.

This application should be easy to move if there was a good reason to do so.

Vendors to compete for business based on standard service metrics

As standards emerge, see if cloud vendors begin to offer pricing based on them. For example, it is highly possible that for this application category, a benchmark of HTTP traffic with a certain performance characteristic might be a simple enough metric that a cloud provider will sign up to that and worry about everything else underneath.

For this application, that is the only important metric, shop around for that.

6.2 Agency Info Lookup using Browser (Private)

Like the public web site example above, the transition to the Cloud is a matter of moving a classic web application using a web server and a static content repository or database to the Cloud. However, unlike the public site challenges that need to be solved in the previous example, depending on the agency size the scale challenge is likely not as large. However, there may very well be security aspects of the information, such as auditing, or access levels, which need to be looked after.

6.2.1 Case Example Issues in Moving to the cloud

Efficiency

On the first subject, it is highly likely that the classic IaaS approach will be quite successful. The simplest approach is to take the whole OS, Web Server, and Database stack and put it on a machine image, and then put into the Cloud on a VM using IaaS. One would them manually include and configure into the machine image the content integration used with any outside sources, as described above. Again, if the application utilized any load balancer or application networking services, the preferred method would be to use the cloud provider's built-in load balancing mechanism, whether that be their own hardware, or a software facility inherent to the Cloud. The load balancing scripts would need to be re-programmed to fit the Cloud system. Alternatively, one could use a software load balancer deployed manually, as a virtual appliance, in front of several replicates of the web server stack. Along with the pool of load balancers, one would deploy an elastic pool of front ends, sometimes called Servlets, sometimes called Web Workers. These modules scale up and down in concert with the load balancers and all call the statically sized back end pool.

The existing application specific authentication mechanisms would move with the application in this IaaS scenario. This might be an additional VM stack deployment for the authentication and audit/logging system or it may be built into the application. Depending on the architecture of this system, one would study where to place the auditing and logfile output. Cloud would provide a unique opportunity to expand the comprehensiveness of auditing by allowing for much longer retention of records (due to the "never full" aspect of Cloud Storage), however it is like that mechanism would have to be added to the system to change storage files or drives rather that manage a fixed file/drive size.

It is likely that many applications, when moved to the Cloud, will carry a requirement to integrate with a Single Sign On (SSO) system. It is quite typical that SSO systems are implemented at the same time that a movement of several applications to a Cloud, if they haven't already been integrated as part of an agency Single Sign On initiative. Depending on how the SSO system is architected as far as integration goes, one must be careful to account for the dynamic nature of the network in a Cloud. Often in a physical datacenter SSO systems have "known" IP addresses to reduce DHCP and DNS spoofing risks. However, any integrations based on static IP addresses will fail in Cloud; special review of DHCP, DNS, as well as NAT should be considered in SSO integration.

If the application/agency was large, all of the scale considerations in the previous example apply here, that is either to build "elasticity" into that architecture, or to utilize a PaaS platform which contains the web server stack, load balancer, applications server itself, and other elements integrated within the platform. Depending on how the original application was written, this may or may not require much re-write of the "web site". If one uses this approach, the inherent load balancing, the built-in elasticity at the HTTP server level, all will add significantly to the efficiency gains.

Agility

Similar to the previous example, for this application category, putting up new web sites without the need to provision physical hardware is a big win. It is the "essence" of cloud computing. Even more so, if the cloud contains "instant web site" kinds of capabilities, and systems to manage DNS, WYSIWYG site editors with templates, then there is even more agility for a static content web application.

Look for *simple* web tools. Look for simple connectivity to SSO. This is the way to get a new internally facing application up quickly.

Innovation

As with the public facing application example before, there are now opportunities to the Cloud system to provide capabilities on top of our static web site application. For example, can the cloud provide us a "search" capability easily? Can we get visitor statistics and even analytics to gain insight on what content readers are interested in? Can we mash-up into the static web site external content that would make this rather basic application category more alive and interesting?

Again, one should look for easy to incorporate new features to the web site, like discussion forums, search engines, feedback or review capabilities, favorites, photo tools and galleries, etc.

In this private cloud application, there is another option as well, especially if SSO has been implemented. That is the ability to cross link applications together with some simple user interface and HTTP extensions to other applications and integrate multiple applications. This should be investigated while the applications are moved to Cloud.

Security Requirements

For this applications category, the main concern is simple high performance availability of the content. Therefore, we are counting on the cloud platform the handle DDOS attacks, web site exploits and vulnerabilities, and all the "horizontal" security aspects which should be handled by the platform.

As mentioned previously, SSO and auditing are additional characteristics to consider in this application example.

Service characteristics

For this application category, the primary capability needed is great web site performance.

Look for cloud platforms that contain attributes to help web sites have great performance and availability, like built in load balancers or CDN capability.

Market Characteristics

As above, for this application category, there should be plenty of choices for platforms or service providers. Also there should be nice elasticity management tools which can help with larger or more complicated deployments. For this application, almost any well connected cloud platform will work fine; there is no exotic low latency or particular geographic location requirement.

Look for a cloud which is focused on cloud based site hosting, who will bring more web site oriented capabilities to the table, rather than straight VM's and storage.

Network infrastructure

For this application category, there is no complicated requirement for network infrastructure, other than the well-connectedness of the underlying cloud, and the multi-site availability requirements which one might have for site availability.

Depending on the agency being served, and the sensitivity of information, the precise location and characteristics of the Cloud datacenter might be of concern, whether a service provider is utilized, or a Cloud is constructed.

Additionally, the connectivity of the Cloud to internal networks might be worth considering. If the agency is hosting it's own Cloud for this application, then one would want the Cloud on network in the address space of the LAN of the agency (interior to the firewall perimeter). If the Cloud is on a public Cloud, telecommunications carriers can provide secure (encrypted) tunnels

(via IPSEC or MPLS VPN) from a public Cloud infrastructure to the agency network. In this configuration, often the LAN for that portion of public Cloud is further isolated by equipment in the Cloud provider, and appears to be in the address space of the LAN of the agency (as if it was hosted locally in the prior example). This is called VPC or Virtual Private Cloud and is offered as a commercial package by several Cloud vendors.

Government readiness

As before, for this application category, if the site is particularly complicated, or changes often, the new and remote development environment of the cloud may be a challenge to the existing organization.

Look for the ability to run the same content and site management tools which one runs in the current environment.

Technology lifecycle

For this application category, there are some technology discontinuities to consider. Web sites keep using new technologies, even if they are static content sites.

For example, many sites have gone to technologies like HTML5, in order to present a more dynamic site. Also, in the re-programming to a new technology, the site can be made to work well on mobile devices automatically.

If this is of interest to the application, that is to use the opportunity of shifting to the cloud to adopt a new Web Server framework, then look specifically to those cloud systems which offer that as a built in system, as mentioned above, most likely a PaaS system.

Most Appropriate Standards

For this type of application, often the web site is built with a "stack" from a particular vendor, or using a particular family of open source or language, from top to bottom including the Operating System. A number of proprietary and Open Source software stacks exist; an analysis is needed to judge the appropriateness for the application. Stacks that support application functionality directly, without significant layers of translation, are appropriate.

Look for the "more native" cloud for the stack you are choosing.

Security in the Application Layer

For this application category, it is possible that the servers or the network infrastructure that the Web Server sat within, offered a lot of security capability. It is likely that the cloud platform may offer less than that. For example, access logs may be something that was handled by the former server or network infrastructure, or access blocking from blacklisted addresses. These precise features may not be built into the cloud platform.

Again, it worth integrating with the prevalent SSO system, this will handle much of the applications specific security (at least access) issues.

Understand what security your application needs, and if the cloud platform does not provide it for you, be prepared to implement it as part of your application.

Which Cloud Services Leveraged

For this application, it is likely that many clouds offer services to help make an efficient and high performance Web Site. Many of the services (like page caching) which used to be done in your web server may now be available as a cloud service.

Look specifically at all the features the cloud offers, and decide which ones to leverage. The rule of thumb is, leverage everything you can, even if the cloud service is much simpler or rudimentary that your equivalent service, try to use it.

The cloud services will evolve and improve. These are not what is core to your web site. Monitor in the future what new services come out and try to remove functionality for your application and use the cloud.

Compliance Monitoring

Compliance with access control, to control and stay in accordance with access to information, is likely an important component of this application. In analyzing the existing web site, it is likely that much monitoring and recording of access, and all kinds of instrumentation, have been disabled. It is very typical to disable many of the capabilities in a production system because of the volume of data which is generated and/or the extra processing required generating it. In the Cloud environment, neither of these is as much of a concern. The details of compliance should be studied and as much granularity as needed for reporting which user saw what information should be configured into the application as logfile data. Furthermore, there are tools which can be obtained to process this data in conjunction with third party compliance products (some tools include so-called “big data” processing capabilities to cope with the volume of data generated).

These techniques can augment the web application to an extent far beyond the capability allowed on a physical server environment.

6.2.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

This application category is perfect for aggregated demand with a larger cloud based web hosting environment across Government agencies. In no case should a special infrastructure be built for this category of application, there are many service providers or even a shared infrastructure with another agency which has capacity to share.

Look for chances to aggregate capacity, for this simplest of application categories, this is the easiest to share infrastructure with others.

The caveat with the private application is to ensure that if the security strategy around network isolation is utilizing a separate cloud, or a VPC as described above, ensure that the host Cloud can accommodate that capability.

Integrate services

For this application, it should be analyzed how completely static, or whether there is any dynamic content served up, and where that comes from. The mechanism by which content is updated should be analyzed for best ways to accomplish the same thing in a cloud environment.

For example, a manual update of a database might have been done in the original web site. In the cloud, there would be an opportunity to automate this through some of the connectivity features which the cloud offers, such as message queues or SOA integration capabilities.

Contract effectively

For this application, there is a storage pool which is largely fixed, and a variable pool of web servers, depending on load, and there is traffic, also depending on load. Look carefully at how the cloud provider charges for these aspects and make comparisons. Some PaaS providers will have very complicated charging rules for their advanced facilities such as message queues and load balancers.

Use modeling tools and the emerging cloud comparison services to look hard at the widely varying business models of clouds. Be prepared with an understanding of the size of the system as it exists and what load that can handle.

For this application, there is an opportunity to simplify the billing to not focus on number of VM's or servers but just amount of HTTP traffic. This will push the efficiency burden to the cloud provider.

Realize value

Once the cloud system is running, don't forget to decommission the old web site. Make plans to move groups of sites from same datacenters to cloud in order to shut down or decommission/un-contract from whole sections of datacenter.

Realize value by really making the jump.

Know what SLA's are needed

Understand the "pinch point" of the application, is it the database? Is it the dynamic generation of pages by the web server? Is it the raw bandwidth you have to push through to serve the community? Know the SLA's you need and explore these with the cloud provider.

Understand the reliability/high availability architecture

For this application, look at each elemental part, and ask the redundancy/replication/DR question. For example, for the storage, how many replicates are kept? Are the replicates all in the same zone/datacenter or are they geographically separated. Where are the servers? Same question about geography. Does the load balancing involve more than one geography? For the network, understand the cloud connectivity, how connections can your content flow to directly from the cloud.

Just because it is a private cloud application, do not change the reliability/high availability strategy from that of a Public Cloud application. It is likely there are constraints around the network connectivity of the Cloud to the LAN of the agency. Analyze the application as deployed on cloud to ensure replication of elements, and multiple datacenters, with redundant, even multi-carrier VPC connections. The more important the information served up by the application the more scrutiny this part should receive.

Understand demand aggregation, functionality integration and collaboration

For this application, it may very well be on a multi-agency platform. This provides for an interesting opportunity to aggregate content from other sites/agencies to better display meaningful information.

It is easy to mash-up web sites together once they are on the same infrastructure.

The above comments regarding network access may mitigate the ability to utilize shared infrastructure with other agencies, depending on how clever the networking configurations can be made.

6.2.3 Case Example Issues in Managing services rather than assets

Shift mindset

For this application, the most important metric for understanding it's health, it's utility, and everything about it, is no longer about the web server itself, it's about how much HTTP does this application serve to customers, containing what content.

Presumably, uptime will be something you no longer have as much to worry about. Capacity will likely be something that will be no worry. What one needs to worry about is how fast the overall service to the end user is: how much service in the totality is being delivered - is it growing or shrinking.

Actively monitor

For this application, there are a rich set of web site statistics and analysis tools. Many of these will be built into the platform itself, especially if one is using the PaaS type of architecture.

Logging, statistics, analytics, all take up storage space and processing capacity. This might have been a limiting factor on the previous server architecture web site, a busy site generates gigantic logs. Now there is huge capacity in the cloud and off the shelf applications which, especially for this application category, can provide huge insight into the site.

Monitor the SLA as far as site performance in the users eyes.

Re-evaluate periodically

Monitor expense, resources used, reports, and understand how this is all working on the cloud. Compare with your old server environment. Is it what you expected? Is the cost more or less than you imagined?

Detail cloud security technology; push for the horizontal cloud security solutions

For this application, there are two key explicit security techniques being used, that is, (1) to have the Cloud in the LAN of the agency (either by private cloud hosting, or by VPC technique) and (2) to closely control authentication with logfiles and auditing (either with SSO and application specific instrumentation, or otherwise).

The rest of the security picture, from DDOS protection, to perimeter firewall and IDS/IPS, to even regulatory issues such as keeping data at rest encrypted, these should be handled by the host Cloud. One should ensure that these built-in or so-called horizontal security mechanisms are detailed, understood, configured as needed, and monitored, so that the capabilities they provide do not need to be replicated by the application itself.

Monitor new cloud developments for economic or technology breakthroughs

Take these performance characteristics to other cloud vendors or options and see if there is a faster/better/lower cost alternative.

This application should be easy to move if there was a good reason to do so.

Vendors to compete for business based on standard service metrics

As standards emerge, see if cloud vendors begin to offer pricing based on them. For example, it is highly possible that for this application category, a benchmark of HTTP traffic with a certain performance characteristic might be a simple enough metric that a cloud provider will sign up to that and worry about everything else underneath.

For this application, that is the only important metric, shop around for that.

6.3 Transactional Application like Apply for Passport, File Taxes, USPS Email Service, probably using Browser (Public)

This example is likely the most prevalent and important example. It is a full functioning, web based application that has business logic and transactional data. It is a true "web application" which is going to be deployed on the Cloud and become SaaS. These applications are most often multi-tier application server based in architecture.

For illustrative purposes only, because the architectural issues will need to be discussed in detail, we will consider the architecture to follow a modern, multi-tier application server blueprint such as that of J2EE or .NET.[2] The following terminology will be used in the discussion, beginning from the ingress of web traffic from a Browser to an J2EE or .NET based application.

First the Application Networking Services modules, more commonly referred to as Load Balancers, are at the top of the architecture. These are highly programmable devices which can

[2] The use of J2EE and .NET in this document is not intended as an endorsement. Many competing technologies may be suitable alternatives; J2EE and .NET are described here because a detailed discussion is needed for clarity of exposition.

recognize, route, or transform application messages. They can know about source domains, IP addresses, content of HTTP traffic, time of day, patterns of potentially dangerous traffic, and so on. They can also be informed of instrumentation from lower levels of the stack such as how busy the web servers or the database is. All of this information is used by the programming in the Load Balancer to change or re-direct traffic to the appropriate next stage.

The next stage of the architecture typically handles the HTTP traffic and the generation of pages; in J2EE this tier is called the "Servlet Tier"; in .NET it is called "Web Tier". In J2EE the Servlet Tier may have simple web servers, servlets, or Java Server Pages servers. In .NET the Web Tier usually contains Windows IIS and ASP.NET. There is usually a pool of servers in the Servlet/Web Tier and the Load Balancers direct traffic across the pool based on a traditional scheduling algorithm (round robin, randomization, ..) or based on some other pre-programmed or reactive behavior.

The next stage of the architecture typically handles the business logic. In J2EE this is called the EJB Tier; in .NET it is called the Worker Tier. In J2EE the EJB Tier is comprised of Java modules programmed to one of several techniques such as Session Bean or Entity Bean, it is important to realize that this Tier usually holds considerable state data because of the business logic and so can be a barrier to scalability. Likewise, for .NET the Worker Tier contains modules programmed in one of many .NET languages (C# or Java or other) run in a .NET server. There is usually a pool of servers in the EJB/Worker Tier, and the Servlet/Web Tiers has multiple connections to the EJB/Worker Tier.

Sometimes, all of the EJB/Worker Tier machines connect to a back-end database directly. As long as there are not too many machines in the EJB/Worker Tier and they all have sufficient memory to process business services and handle queries and transaction management, a separate database interface tier is not needed. However, for a Public web site, this may not the case.

Therefore, a database tier is usually next in the stack. In J2EE this is an Entity Bean representing the state of the database and it runs special J2EE code called JTA (Java Transactions Architecture) and JCA (Java Connection Architecture) to connect to the database. In .NET, this tier is fulfilled by ADO.NET (stands for Applications Data Objects). As before, there is usually a pool of servers in the Database Tier and the EJB/Worker Tier has multiple connections to the Database Tier.

Finally there is the Database, which usually runs in a cluster itself. It connects to storage, usually a disk partition but sometimes a file system (i.e., an abstraction of a disk partition).

The machines within one tier are usually connected on an L2 switch segment using a VLAN, and the different tiers are also on an L2 switch segment, but each with their own VLAN. The tiers all together, are called an Application Server Cluster. Typically, there is a system directory which knows how many servers are in each Tier, and knows what services are running on each server in each Tier and knows how the Tiers are connected together. Both J2EE and .NET support dynamic Tier (and thus cluster) expansion although their configuration files to process and inform the cluster of newly added servers and the services running on them. Also, the network may need to be programmed to add servers to a VLAN, for example. One can see that applying

Cloud Elasticity to today's multi-tier architectures requires careful consideration for issues of scale, resource allocation, dynamic configuration, and failure recovery. These are some of the challenges in moving this Application to the Cloud while also taking advantage of the cloud architecture.

The integrity of the system will depend on reliable interactions between the tiers and a preservation of the transaction semantics between the tiers. For example, the EJB/Worker Tier is going to expect an ACID behavior from the Database Tier, which in turn is going to expect that same behavior from the database itself, which will have that behavior from the underlying Cloud storage system.

However, a typical cloud storage system does not in every case offer ACID properties. The exception cases must be worked around in implementation (see below).

6.3.1 Case Example Issues in Moving to the cloud

Efficiency

In the case of a Public facing application, for an application structured as we have outlined, it is certain that each tier will have been provisioned to allow for plenty of headroom on top of the peak demand case. This means, that on normal demand periods the cluster is likely to be hardly utilized and that means a lot of wasted machines. To take advantage of efficiency, one has two choices: (1) use the same enterprise architecture but try not to change too much of the tested tiers and cluster configuration, or (2) take maximal advantage of the cloud programming environment and re-write a significant part of the application to the cloud API's.

For the first alternative, this means one needs to essentially map the multiple tiers to a more cloud friendly networking platform and ensure that, depending on scale needs, the dynamic ability to add to the tiers and also expand the cluster is there. Since this is such an important example case, it will be considered in detail here.

On the Load Balancer Tier, the typical approach for an application is to utilize a physical load balancer. This is not desirable with the Cloud as the efficiencies are gained by using as many software based services as possible. The considerations of what load balancer to use might be as follows: first, consider the load balancer capability which comes with the cloud. If the cost is not prohibitive, use this one. It is one more part of your architecture you no longer need to worry about. If the cloud provider does not provide a load balancer, or your application makes extensive use of scripts and configuration of a particular brand of load balancer, then find a software implementation of the favorite load balancer. If the application is not specific about the brand of load balancer needed, choose one based on simplicity, cost, and experience with Cloud implementations. Some analysis should be done on how the Servlet/Web Tier connects to the Load Balancers. Hard coded IP addresses and clever use of DNS tricks, are all likely to need revisions to run properly. Understand how to deploy and undeploy load balancers, and how to configure it into the script or algorithm. Consider revising the algorithm to take direction from instrumentation placed in the servlet/Web tier, either explicitly in the Web Tier, or by exposing CPU busy statistics in the underlying VM's on the servlet/Web tier. There may be a separate "watch dog" type of tool for this too. In other words, seek to implement elasticity in the load

balancer tier, it will save operational expense on the lighter load side drastically and on the upper scalability side will keep the middle tiers of the application from falling over.

For the servlet/Web tier, there are two considerations. One is to instrument it, as described above. This instrumentation can be fed to the tool which can then provision more servlets/Web workers in the tier. It can also configure the Cloud network as needed. Finally the pool of servers in the tier needs to be managed in its own clustering system when a server is added or subtracted.

For the EJB/Worker tier, the exact same considerations apply – instrumentation and scalability automation. It is likely to be somewhat complicated to dynamically expand the Back End of this Tier but it is the last line of defense for creating a large application without resorting to a total re-write.

For the database tier, it is likely that the number of connections to the EJB/Worker tier will be flexibly configurable. It is not likely that the number of database tier machines is easily changeable because they connect to the database and the connection pooling configuration of a database is rarely configurable. Elasticity (from the Cloud) in the database tier is typically a challenge.

Finally, if the application is truly database limited, there is little the cloud can improve with respect to scale. If all the upper tiers have been maximized and can grow/shrink, a huge amount of efficiency has been gained. The central database machine will have to scale based on classic database scaling techniques, which can get expensive. For the largest social networks, auction sites, and so on, their main scalability worry is in the database: all the other tiers are cranking at full efficiency and load. This application, if it becomes exteemely large, will end up in the same situation.

It is possible that the application is not using an application server; it may manage a cluster with multiple components itself. Mechanisms might exist in that stack where certain modules can be replicated to help scale out. Some modules may be constrained to a single server. If this is the case, one needs to analyze the architecture, and apply similar principles of horizontal elasticity of layers, and of the possible assumptions of topology and addressing which may be built into the structure of the application cluster.

To consider the second alternative, it might be in order to consider a re-write into a more cloud-native (PAAS) architecture. A PaaS platform typically contains the web server stack, load balancer, applications server itself, and other elements integrated within the platform. Depending on how the original application was written, this may or may not require much re-write of the "web site". If one uses this approach, the inherent load balancing, the built-in elasticity at the HTTP server level, all will add significantly to the efficiency gains.

To the point regarding integration of external ("mash up") type information into the site, there are two concerns. One as mentioned previously is that since the application is now capable of scaling up significantly, are the data feeds integrated via HTTP references, SOA integration techniques, remote files, also able to handle more traffic? Can they traverse the firewall and security architecture of the applications running on the Cloud?

Agility

For this application category, the same tools, which were used to create the application in the first place, should continue to be used. It is quite complicated to connect the user interface to the business processes to the business services to the database schema; the tools for the J2EE or .NET or other selected stack should continue to be used in the Cloud deployment.

One should be able to shorten the cycle, significantly, if one adopts the development and deployment philosophy called "DEVOP" – meaning, for any given application, the team which develops it, also deploys and operates it. Many argue that there is no other successful way to do software engineering on a Cloud based application. Changes made in an application can be deployed immediately. Usually a test deployment is done first, but the point is, that the people who write the application also launch and care for it while it is live. This new philosophy should be studied carefully in order to realize the acceleration advantages of Cloud.

Innovation

As with the public facing application example before, there are now opportunities to the Cloud system to provide capabilities on top of our static web site application. For example, can the cloud provide us a "search" capability easily? Can we get visitor statistics and even analytics to gain insight on what content readers are interested in? Can we mash-up into the static web site external content that would make this rather basic application category more alive and interesting?

Again, one should look for easy to incorporate new features to the web site, like discussion forums, search engines, feedback or review capabilities, favorites, photo tools and galleries, etc.

Security Requirements

For this applications category, the main concerns are simple high performance availability of the content and data integrity. Therefore, we are counting on the cloud platform to handle DDOS attacks, web site exploits and vulnerabilities, and all the "horizontal" security aspects, which should be handled by the platform.

Depending on the type of application, compliance may be a crucial issue as well. On the one hand, the application itself should implement it's domain specific compliance capabilities, in a similar way in which it did before the application was moved to the cloud. On the other hand, it must be made sure that nothing about a Cloud deployment can compromise that. For example, one might want to make sure nothing is installed into the virtual environment which introduces accidental memory space overlap, or access to customer data from a system manager without proper procedures. One should look specifically to the Cloud vendor for compliance statements to various regulatory standards, and also one might look to third party, Cloud specific software products which can monitor this type of issue.

Service characteristics

For this application category, although a primary capability needed is high web site performance and availability, because the application structure is significantly more complicated than serving

up static content, Clouds which facilitate automated deployment of related components, might be of positive utility. These kinds of capabilities have tools which allow the grouping of the tiers in the cluster, as described above, and the deployment, monitoring, and management of those. There are clouds with this sort of "blueprinted deployment" as a feature of the Cloud platform; there are also tools from software vendors which accomplish this. One should not underestimate the difficulty in simply getting the deployment of a large multi-tier application to the Cloud to work.

Speaking now to the challenges of using Cloud storage, consider whether the cloud storage offers ACID properties, and where it doesn't. There are also considerations of how to work around that.

Many cloud storage systems guarantee atomicity for a single key-value pair update. Guaranteeing atomicity for updates across several key-value pairs or even several buckets is left to the software interfacing, which in this case is the database. In the database, and the transaction management capabilities in the database tier (JTA or ADO.NET) this task is fulfilled by applying transactions as well as appropriate recovery mechanisms in order to handle transaction aborts. Therefore, atomicity is covered.

For consistency, many cloud storage systems provide at most read-after-write consistency for put operations of new data. Updates are treated with eventual consistency guarantees, especially if the reads are from another part of the Cloud, where in fact the read is being fulfilled by a different replicate than to which the first write occurred. Thus, if a key-value pair is written and read immediately thereafter, it is possible that an older version is retrieved. In practice, the database will cache a read after write, and this will not happen. However, if one takes a multiple instance high availability or geographical distribution strategy, where one replicates the entire cluster in two geographies, and has two database servers pointing to the same storage, then there is a possibility where the second database will read an older key-value pair than it should. This is an unusual application architecture but, on a primarily "read" mode application, might be done for latency purposes across geographies. In this scenario, even when "write" mode is used, usually users are not operating on the same accounts from two different geographies, inconsistency will rarely happen. However, reservations of a resource, like a campground spot, could yield a double booking with this scenario. Application logic which reserves the available spot, then checks again on it's availability before finally booking it, will cure the problem. One must check the application functionality to verify this.

Isolation gives the user the impression to work alone with the system in a multi-user environment. It is necessary to keep isolation in order to guarantee consistent data. Many cloud storage systems by default offer no synchronization techniques in order to provide isolation. If several users update a certain key value pair, the last write operation wins and updates of the others are lost. For several applications, like social networks or the shopping cart scenario this is sufficient, since all users work on disjoint data sets. However, in cooperative design applications (for example) sophisticated synchronization techniques must be added to the application.

Finally, guaranteeing durability means storing successfully performed updates persistently. Many cloud storage systems themselves guarantee persistent storage of data without data loss.

The conclusion of this subject is two-fold (1) if the application involves cooperative, simultaneous updates of a data set like a cooperative design application involving many simultaneous users, the move to Cloud storage will likely break the application; application level synchronization must be added; (2) for most applications, if one instance of a database is used the database and thus the application will not be affected by the move to Cloud storage. If there are two instances of the database and especially if they are separated geographically, that has the small but statistically possible ability to cause problems. It should be noted, that the cost of using a distributed/synchronized database or other storage system is considerably more, being that it would include a special network provisioning. The use case or data set at hand may make the statistics so small this is completely acceptable given how cost effective an alternative it is.

Know how the storage system works in the application and find out the ACID properties of the cloud storage system – make sure the behavior of the overall system with the two put together will be as one expects.

Market Characteristics

Because this application structure is considerably more complicated, looking for a Cloud platform that has expertise in deploying that structure of application will be well worth the time.

The above example covered .NET and J2EE style application container for illustration purposes. Actually, there are also Cloud service providers and Cloud platforms which offer specialized capabilities in other kinds of application server stacks as well – Ruby on Rails, or Spring and Hibernate, to name a couple of examples.

Look for a Cloud or a service provider who has expertise in hosting exactly the type of application infrastructure which the application consists of.

Network infrastructure

As far as internal networking options, the multi-tier application can provide some significant challenges if one wants to deploy it to Cloud with little or no changes. One important area is the networking infrastructure. Many multi-tier applications rely on VLAN technology, so that all the systems in the cluster can reside on the same L2 switch, but the switch can be configured to group the tiers into separate VLANS. This way, the networks in each tier are separated as to their Ethernet switching protocols, enhancing performance, and broadcast traffic across the tier stays in the tier for which it is intended. At L3, sometimes the machines know each other by IP address, not by DNS name, and this is configured into the cluster manager.

When moving an application to the Cloud, these considerations become paramount. Some will decide to look for a Cloud which can support the VLAN configuration, for example, but one should understand that this approach will eliminate all but a few smaller Cloud service providers. Therefore it is likely that one will need to do some re-engineering around how the network is used in the Cloud. Each Cloud service provider or platform will have different limitations, and some will have limitations that interact. For example, if one uses Cloud load balancing, there are limits in the networking configurations which can be used. There are also limits to the total numbers of VM's in an IP subnet. All of this should be looked at very carefully.

Another internal network consideration is bandwidth and latency. Applications are used to being connected to lightly loaded networking equipment, with both logically and physically short network segments. In a Cloud, it is possible to see very variable networking performance, as servers may or may not end up near each other in both the logical or physical sense, with many elements of very busy networking hardware and software between them. System characteristics which rely on high speed networks for shared memory models, or low latency RPC, might exhibit poor or failing behaviors on the Cloud because the networking performance is non-optimal. In particular, shared memory models will fail miserably on the Cloud unless explicit memory caches are used or HPC configurations of Cloud are used. See the following section on HPC for more discussion of this.

As far as connectivity to the outside world is concerned, there is no complicated requirement for network infrastructure, other than the well-connectedness of the underlying cloud, and the multi-site availability requirements which one might have for site availability.

Depending on the agency being served, and the sensitivity of information, the precise location and characteristics of the Cloud datacenter might be of concern, whether a service provider is utilized, or a Cloud is constructed.

Government readiness

As before, for this application category, if the site is particularly complicated, or changes often, the new and remote development environment of the cloud may be a challenge to the existing organization.

Look for the ability to run the same content and site management tools which one runs in the current environment.

Technology lifecycle

Large applications with this type of architecture are major investments. However it can be another major investment to repurpose them for the Cloud. After some analysis, one should come to some conclusion of the size of this effort. It is highly dependent on the details of the application architecture and if a service provider or a Cloud platform can be found which is particularly hospitable to that architecture. If that cannot be found, one may consider a larger engineering project to switch to a more Cloud-friendly architecture. Especially if there are other major features which the application needs, such as HTML5 or Mobile support, or a switch to a new database, then a revision to a PaaS implementation might be considered.

Most Appropriate Standards

For this type of application, often the web site is built with a "stack" from a particular vendor, or using a particular family of open source or language, from top to bottom including the Operating System. A number of proprietary and Open Source software stacks exist; an analysis is needed to judge the appropriateness for the application. Stacks that support application functionality directly, without significant layers of translation, are appropriate.

Look for the "more native" cloud for the stack you are choosing.

Security in the Application Layer

For this application category, it is possible that the servers or the network infrastructure that the application server sat within, offered a lot of security capability. It is likely that the cloud platform may offer less than that. For example, access log management may be something that was handled by the former server or network infrastructure, or access blocking from blacklisted addresses. These precise features may not be built into the cloud platform.

Which Cloud Services Leveraged

For this application, it is likely that many clouds offer services to help make an efficient and high performance application. If the platform includes built-in load balancers and built-in Servlet-tier or Web-tier, pre-integrated, this is something to consider for the application, depending on the level of integration and customization in the application architecture with this tier.

Look specifically at all the features the cloud offers, and decide which ones to leverage. The rule of thumb is, leverage everything you can, even if the cloud service is much simpler or rudimentary that your equivalent service, try to use it.

The cloud services will evolve and improve. These are not what is core to your web site. Monitor in the future what new services come out and try to remove functionality for your application and use the cloud.

Compliance Monitoring

Expanding on the discussion above, compliance and access control, to control and stay in accordance with access to information, is likely an important component of this application. In analyzing the existing web site, it is likely that much monitoring and recording of access, and all kinds of instrumentation, have been disabled. It is very typical to disable many of the capabilities in a production, public facing system because of the volume of data which is generated and/or the extra processing required generating it. In the Cloud environment, neither of these is as much of a concern. The details of compliance should be studied and as much granularity as needed for reporting which user saw what information should be configured into the application as logfile data. Furthermore, there are tools which can be obtained to process this data in conjunction with third party compliance products (some tools include so-called "big data" processing capabilities to cope with the volume of data generated).

These techniques can augment the web application to an extent far beyond the capability allowed on a physical server environment.

6.3.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

This application category is perfect for aggregated demand with a larger cloud based web hosting environment across Government agencies. There are many service providers or even a shared infrastructure with another agency which has capacity to share.

The caveat with the transactional, full featured application is it's potential need for more sophisticated networking, higher performance storage, or otherwise more demands on the Cloud platform.

Integrate services

As with the other Cloud projects, the opportunity to add new capability or content into the application while doing the technical project of moving it to Cloud, exists and should be analyzed. In many cases, access to other information sources, or capabilities from other Government agencies, could be adjacent and easy to integrate with, especially of a shared infrastructure was being used.

Contract effectively

This application will potentially utilize a wider set of features of the Cloud platform. For example, load balancers, more complex storage models, advanced networking capabilities supporting the multiple tiers, the bandwidth used by the application, the application blueprinting and deployment facilities - are all priced "a la carte". It is quite difficult to predict how to contract for these applications.

There are some modeling tools out there, which are worth studying. A proof of concept with the vendor, to more accurately forecast, is also worth a quick project.

Monitor and keep track of what is costing in the application. Have discussions with the team or the vendor running the Cloud to understand what is contributing to the expense. For example overall network expense might warrant placing the application stack in multiple Cloud locations on either coast for example.

Realize value

Once the application is running, decommission and re-use or retire the assets on which the application originally ran. In order to realize value one must make the jump to Cloud.

Know what SLA's are needed

Understand the "pinch point" of the application, is it the database? Is it the dynamic generation of pages by the web server? Is it the raw bandwidth you have to push through to serve the community? Know the SLA's you need and explore these with the cloud provider.

Understand the reliability/high availability architecture

For this application, look at each elemental part, and ask the redundancy/replication/DR question. For example, for the storage, how many replicates are kept? Are the replicates all in the same zone/datacenter or are they geographically separated. Where are the servers? Same question about geography. Does the load balancing involve more than one geography? For the

network, understand the cloud connectivity, how connections can your content flow to directly from the cloud.

Understand demand aggregation, functionality integration and collaboration

For this application, it may very well be on a multi-agency platform. This provides for an interesting opportunity to aggregate content from other sites/agencies to better display meaningful information.

It is easier to mash-up web sites together once they are on the same infrastructure.

The above comments regarding network access may mitigate the ability to utilize shared infrastructure with other agencies, depending on how clever the networking configurations can be made.

6.3.3 Case Example Issues in Managing services rather than assets

Shift mindset

There are two extremely important shifts one must adopt in moving the application to the cloud: (1) The *users* of the application, because it is a *service* now, not a web site, not an "application", expect it to behave differently, and (2) the care and feeding of the application, or *service* as it should be called, must be handled in the DEVOP philosophy, as described before, in order for it to be scalable, reliable, and responsive to users' needs as they develop.

On the behavior mode, a successful application is more like a utility – easy to use, evolves to be simpler, not more complicated, and caters to the casual user rather than the trained user. The service will be accessed by people on the run, via a Pad or a laptop, and shouldn't require any training or instructions. In the way enterprise email has evolved to webmail, in the way geo-spatial databases have evolved to online maps, in the way smart phones are evolving, the user community for this service will expect it to become easier to use, more visually pleasing, and more powerful as speech, search, location, prediction, video, and other computationally intensive features become built in to the application behavior. A successful application will need to evolve into a service utility, wherever and however the users want to access it.

On the development and operation of the service, as mentioned before, the traditional software development and deployment "waterfall methodology" will no longer work. The scalability requirements, migration to leverage of the platform, and all the other attributes of Cloud, require that the team who builds the service, *runs* the service. Embrace the DEVOPs philosophy for Cloud.

Actively monitor

For this application, there are a rich set of web site as well as application server instrumentation and statistics and analysis tools. Many of these will be built into the platform itself, especially if one is using the PaaS type of architecture.

There are specific tools for the application server architecture which MUST be used, as well. Behavior on the Cloud will be different than the server architecture; monitoring is essential to understand this. This is a new area and all applications are different.

Logging, statistics, analytics, all take up storage space and processing capacity. This might have been a limiting factor on the previous server architecture web site, a busy site generates gigantic logs. Now there is huge capacity in the cloud and off the shelf applications which, especially for this application category, can provide huge insight into the site.

Monitor the SLA as far as site performance in the users eyes.

Re-evaluate periodically

Monitor expense, resources used, reports, and understand how this is all working on the cloud. Compare with your old server environment. Is it what you expected? Is the cost more or less than you imagined?

Detail cloud security technology; push for the horizontal cloud security solutions

Understand how well the application based security is working. Make sure that the entitlements mechanisms within the application are working at scale. Often times these systems never intended to support larger user communities.

The rest of the security picture, from DDOS protection, to perimeter firewall and IDS/IPS, to even regulatory issues such as keeping data at rest encrypted, these should be handled by the host Cloud. One should ensure that these built-in or so-called horizontal security mechanisms are detailed, understood, configured as needed, and monitored, so that the capabilities they provide do not need to be replicated by the application itself.

Monitor new cloud developments for economic or technology breakthroughs

Get roadmap presentations from the Cloud platform vendor or service provider. Make sure they are the right home for the application architecture.

Vendors to compete for business based on standard service metrics

Once it is known what the major cost bases are for the application, and what the key needed Cloud architectural aspects are, periodically check for better vendors emerging.

6.4 Agency Application like Issue Passport, Investigate Taxes, Agency Email, or for Calculations (Private)

6.4.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.4.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.4.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.5 Programmable Application like future data.gov using tools, plug-ins or script (Public)

6.5.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.5.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.5.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.6 Agency developers making applications (Private)

6.6.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.6.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.6.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.7 Agency Developers will use Brokers to access resources from other agencies (Private)

6.7.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.7.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.7.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.8 Programming Platform - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility (Public)

6.8.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.8.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.8.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.9 Developers will use Brokers to access resources from many agencies (Public)

6.9.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.9.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.9.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.10 Agency Research Facility, for Calculations (Private)

6.10.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.10.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.10.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.11 Servers and Storage - Free access, Educational, Grant or Fee Based, or otherwise Public Research Facility (Public)

6.11.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.11.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.11.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

6.12 Agency Servers and Storage, Classic IaaS (Private)

6.12.1 Case Example Issues in Moving to the cloud

Efficiency

Agility

Innovation

Security Requirements

Service characteristics

Market Characteristics

Network infrastructure

Government readiness

Technology lifecycle

Most Appropriate Standards

Security in the Application Layer

Which Cloud Services Leveraged

Compliance Monitoring

6.12.2 Case Example Issues in Provisioning cloud services effectively

Aggregate demand

Integrate services

Contract effectively

Realize value

Know what SLA's are needed

Understand the reliability/high availability architecture

Understand demand aggregation, functionality integration and collaboration

6.12.3 Case Example Issues in Managing services rather than assets

Shift mindset

Actively monitor

Re-evaluate periodically

Detail cloud security technology; push for the horizontal cloud security solutions

Monitor new cloud developments for economic or technology breakthroughs

Vendors to compete for business based on standard service metrics

7 Some Infrastructure as a Service Developer Notes

This section presents additional considerations for Infrastructure as a Service projects. These are offered as "pointers and tips" for developers to better understand possible issues in moving an application to the Cloud, and questions to ask vendors and technology providers about that area of technology.

7.1 VM Considerations – Units of Measurement, Compatibility

The notion of "compute unit" which is supplied by the IaaS system is not any type of standard. For example (as of November 1, 2011):

- In Amazon AWS, a "small" instance consists of 1 equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor, 1.7 GB of memory, 160 GB of local instance storage, 32-bit platform, nothing specified for I/O performance.
- In Windows Azure, a "small" instance consists of 1 equivalent CPU capacity of a 1.6 GHz (CPU generation not specified), 1.75 GB of memory, 225 GB of local instance storage, 64-bit platform, "moderate" I/O performance.

There are many types and different pricing models. Make sure it is understood what the application will get, and that the important SLA's are specified for the application needs. It is possible that the Cloud service provider will decline to provide SLA's in certain areas such as throughput to storage or network throughput, even with a statistical dimensionality of characterization. This is something that might have to be lived with.

Ensure that the key software you are moving to the Cloud, is supported on the hypervisor technology out of which that Cloud is constructed, and that the license is "portable" as well.

7.2 Storage – BLOBs, Blocks/Drives

There are several storage models. Make sure the semantics and limitations are understood, as discussed above, not all storage modes contain the behavior that local storage even by the same name does.

For example, suppose the BLOB storage model fits the application. Something to consider is the difference between different vendors' BLOB storage services. For example, Windows Azure BLOB Service requires that all BLOBS larger than 64MB use a block-type interface in which a BLOB is composed of a set of smaller blocks and a listing of the ordering of the blocks. Thus, for large BLOBs the connector for Azure must break the data into smaller pieces and create a manifest and then upload them and the ordering to the Azure BLOB Service in two distinct operations whereas Amazon supports a single-piece BLOB using a single get/put operation.

Depending on the relative mix of the size of data transfers and how the application connector is implemented, the same operation, for instance a "putBLOB" call, could have dramatically different performance. This might seem like a low level of detail for the Cloud engineering team to know, but it might be very important to the application performance.

One can use a Block (drive) model for storage, if all these low level specifics about BLOB storage are unfamiliar. Alternatively, one can use "raw" interfaces on top of which the user formats a file system of their choice.

7.3 Load Balancers / Traffic Manager

Many applications make extensive programming use of the Load Balancer, using for example iRules in F5 BigIP, or Policies and Expressions in Citrix Netscaler, or any of a number of other load balancers. If this is an important part of the overall application cluster one might consider getting the virtual appliance version of that load balancer and deploying with the applications stack. It really isn't a good design pattern to try to integrate physical load balancer equipment with applications in the Cloud the way it might have been done for a co-located or hosted application.

Load balancers come with cloud platforms, and can be configured (not to the extent of programmability that the physical load balancers have evolved to). They can cover multiple geographies as well as a collection of different traffic management algorithms. They cost extra to use in some cases. This are should receive detailed scrutiny in a deployment.

Load balancers in virtual appliance forms are becoming available from a variety of vendors.

7.4 Virtual Network and Virtual Private Cloud

The technology which enables you to connect an on-premise network to a portion of a Cloud, is usually called Virtual Private Cloud but it is also confusingly called Virtual Network. The idea is that systems running in the Cloud can still access systems running in the datacenter. This is NOT cloudbursting which is a term with various vendor specific definitions. This capability can make administration and access of assets placed into the Cloud much easier in addition to facilitating integration of local and Cloud systems.

Over the top solutions utilize software to create encrypted tunnels. Carriers can implement this as part of the access network using MPLS VPN.

7.5 IP Addresses and Domain Name Service

There are at least two main areas of consideration for the developer. First is inside the cloud. Second is having a longer lasting external IP address for the application.

Externally visible IP addresses are in short supply and will likely cost to obtain. Cloud providers also change them based on their own Carrier arrangements.

Internally IP addressing will be handled by the Cloud using facilities like DHCP. VMs are configured to manage L3 traffic in a range of basic firewall-like capabilities to group VM's from one customer into an address space and block others. When deploying in the cloud, the various VMs that make up the application will come and go, changing IP addresses in the process. Utilizing DNS as part of the application cluster is thusly very important.

Any part of an application including management software which references hard-coded IP addresses is likely to break; the assumption is IP addresses CAN be changed and DNS is there to handle the issue, which is a crucial part of any network. Most clouds include many DNS servers as DNS is referenced often in a Cloud context.

The develop may wish to understand how per-server firewalls, NAT, and per-server IP Tables (or other per server firewall technique) are configured, to ensure proper address and port capabilities for the application.

7.6 Content Delivery Network

CDN is an important addition to an application which contains rich media. Clouds contain CDN capabilities in two forms: (1) CDN API's, and (2) the actual Network, in other words CDN servers scattered around a region or several regions.

The Cloud service providers see a synergy in CDN because they can use their Cloud datacenters around a region or the world as a location to place CDN servers. They already pay for network service in order to connect their datacenters together so this is a value added service.

CDN is a pricey service and so the density and breadth of geography covered should have a good overlap with the users of the Cloud application. Because of the integration with the Cloud, it is very worthwhile to investigate switching/using to the Cloud native CDN for the new cloud based system.

7.7 High Performance / Grid Computing

Cloud Computing, like Internet/Web style computing, is a loosely-coupled model. Servers speak to each other over networks where latencies can be in at worst in the hundreds of milliseconds and traditionally the overall bandwidth utilized, compared to bandwidth to storage or bandwidth to memory, and was much smaller. Hardware platforms are optimized for cost, and contain industry standards CPUs, memory controllers, and network interfaces.

High Performance and Grid computing supports tightly-coupled models, that is, where servers speak to one another in terms of microseconds and the overall bandwidth utilized was compared to bandwidth to storage or bandwidth to memory. In fact shared memory architectures are ideal for HPC / Grid programming models.

HPC and Grid also utilized large memory footprints, and might even utilize a high performance floating point or vector processing processor for specialized calculations. And typically, the interconnect between servers was an exotic for of networking in the highest speeds obtainable.

The programming model emerged in HPC / Grid to utilize these architectures and several common platforms emerged to serve this marketplace.

Cloud and HPC/Grid seem like they are very different - which is quite true. This means, a straightforward adaptation of an HPC / Grid application is unlikely to realize any benefit from

being re-hosted to a Cloud platform, in fact, doing so might not really be possible given the lack of support in Cloud for the HPC / Grid model.

But the economics of Cloud are so compelling, the communities have been experimenting with some new Cloud architecture more suitable for HPC / Grid problems, and also utilizing the extremely fast networks we now have to map onto from the HPC/ Grid world.

If the application in question has an HPC / Grid part to it, speak to the platform provider or service provider, they might have a trial of cloud with HPC / Grid interfaces and capabilities sooner than you think! Right now, for example, several Cloud vendors have indeed launched a dedicated portion of their Cloud in a HPC configuration, to learn more about the space.

Make sure that if the project is really an HPC / Grid project, to do homework on the state of these initiatives as they are changing quickly, It is likely that adding these capabilities to the proper "Web" design cloud would not be practical for either the Cloud provider or consumer. But a special purpose part of a cloud is a good answer.

Also, the open source community has produced some HPC style API's (Message Passing Interface, Globus Toolkit) but as implemented on AWS. This is worth a look.

7.8 Compliance

Cloud changes many of the traditional measures of compliance. Measures which accommodate compliance in the Cloud environment are not yet mature.

For example, many compliance profiles require that in an IT infrastructure certain activities or departments and be separated from each other. The implementation is performed by placing the different activities or departments on different network segments, defined as different physical switches separated by other gear such as a network router. In the cloud, the different activities or departments may be scheduled to run on the same machine or even the same CPU (in different VM's). Separation is accomplished through software networks and hypervisors which replace physical separation with logical separation.

These Compliance measures apply both the underlying cloud itself, but also of course the applications which run on top of that. So even if the cloud is arranged just so, and the management guidelines are followed, the Cloud itself can be considered secure. Many Cloud vendors claim they have reached this state and are pending audit to announce as such.

This is very worthwhile to research and understand. But it is not the whole picture. The same criteria must be given to the application on the cloud. So, some kind of monitoring system that understands all the virtual structures, as well as the newer way Clouds are administered, needs to be installed and watch over the Cloud application for compliance. Such monitoring systems are becoming available and should be installed into the Cloud.

8 Some Platform as a Service Developer Notes

This section presents additional considerations for Platform as a Service projects. These are offered as "pointers and tips" for developers to better understand possible issues in moving an application to the Cloud, and questions to ask vendors and technology providers about that area of technology.

8.1 Database

The safest approach to bringing an existing application to the Cloud is to bring the whole stack along first, as in, including the database, with the existing application which has proven some large degree of scalability.

If you don't want to bring you own database along, license comparisons will be necessary for the different databases and data stores available from cloud vendors. At some point, the database encounters a scale problem. The database can be "sharded" at that point but now that is a high maintenance approach to software architecture.

Many intermediate steps can be taken to use products which claim they have SQL compatible interfaces, but can scale much larger than the traditional SQL databases.

If the application is getting so large it is out-growing even the largest of databases, because of the size of data, you likely have a "Big Data" problem, and you need to think differently how to architect the application. Ultimately you will have to substitute out "Query" capabilities and begin to make sense of how to display "Search" in the application instead. This is major application reconstruction.

8.2 Message Queue / Service Bus

A popular way to allow applications, especially ones that are geographically disparate, is to connect them with a persistent Message Queue.

The Message Queue can help one integrate disparate sources into the Cloud as well. There are many tools for using SOA interfaces of such a Message Queue or Service Bus.

8.3 Caching

Adding caching to an application provides applications with high-speed access and scale to application data. This is not a shared memory model for HPC, but it is an accelerant to the individual application.

Many platforms support "memcached", a popular implementation.

If caching helped the large application in the on-premise configuration, this should be one of the first places investigated for the transition to the Cloud.

8.4 Access Control / Single Sign On

On the subject of Identity and Access Management (Access Control, Single Sign On), when implementing a Private cloud application, or more likely a set of applications, this is an incredibly powerful capability to take advantage of. This will allow a user to have access to all of their Cloud applications under one password.

8.5 Notification / Email

The application one is moving to cloud, may have some kind of notification or email built into it. Most likely it required a user ID on the company email server to send. Once the application is now living in the Cloud why go back to the enterprise for sending mails. Maybe email is not the best way to make a notification, perhaps it is desired to notify another program via a programmatic interface with reliable delivery. This is where the email and notification services come in. It seems highly efficient to utilize the Cloud based facility in the application.

9 Enabling Brokers - Cloud Interoperability

This section expands on the topic of Brokers, and what it will really take to make an infrastructure of Brokers work. Although the "actor" of Broker has been defined by NIST (as described below) and this document details considerations around programmatic access to such a facility, the eco-system for a Broker driven Interoperability of Clouds does not exist yet; the NIST view is at this time still future oriented. This section discusses the analogy to routed IP NAPs, peering, and exchange points, and also root naming and trust (DNS and certificates), where the global "Internet" is created; a roadmap of various Cloud technologies and infrastructure (including Brokers) must be created to enable a global Intercloud.

This discussion of the Intercloud should be considered as one interesting set of ideas as to how to implement broker functionality on a wide area, multi-cloud basis. The Intercloud ideas presented here are based on existing standards and not existing products; the Intercloud concepts outlined to not imply any specific company products. The Intercloud concept is presented here to inform decision makers regarding questions and decision factors in the context of Cloud Computing interoperability. The specific examples of Intercloud implementation and topology are not intended as an endorsement of a particular architecture. Many competing technologies may be suitable alternatives; the specific examples and topology are described here because a detailed discussion is needed for clarity of exposition.

Cloud instances must be able to dialog with each other. One cloud must be able to find one or more other clouds, which for a particular interoperability scenario is ready, willing, and able to accept an interoperability transaction with and furthermore, exchanging whatever subscription or usage related information which might have been needed as a pre-cursor to the transaction. Thus, an Intercloud Protocol for presence and messaging needs to exist which can support the 1-to-1, 1-to-many, and many-to-many use cases. The discussion between clouds needs to encompass a variety of content, storage and computing resources.

The vision is an analogy with the Internet itself: in a world of TCP/IP and the WWW, data is ubiquitous and interoperable in a network of networks known as the "Internet"; in a world of Cloud Computing, content, storage and computing is ubiquitous and interoperable in a network of Clouds known as the "Intercloud". The elements and topology for the Intercloud has been expressed by many researchers recently, where a reference Intercloud network topology and elements has been developed[3]:

[3] **Ranjan, R., and Buyya, R.**, *Distributed Overlay for Federation of Enterprise Clouds* (2008)
Mei, L., W.K. Chan, and Tse, T.H.: *A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues.*, APSCC pp.464-469, 2008 IEEE Asia-Pacific Services Computing Conference (2008)
Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., and Morrow, M., *Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability*. In Proceedings of ICIW '09, the Fourth International Conference on Internet and Web Applications and Services, pp. 328-336 (2009)
Buyya, R., Pandey, S., and Vecchiola, C.: *Cloudbus toolkit for market-oriented cloud computing*. In Proceedings of the 1st International Conference on Cloud Computing (CloudCom) (2009)
Bernstein, D.: *The Intercloud: Cloud Interoperability at Internet Scale*. In Proceedings of the 2009 NPC, pp. xiii (Keynote 2), Sixth IFIP International Conference on Network and Parallel Computing (2009).;

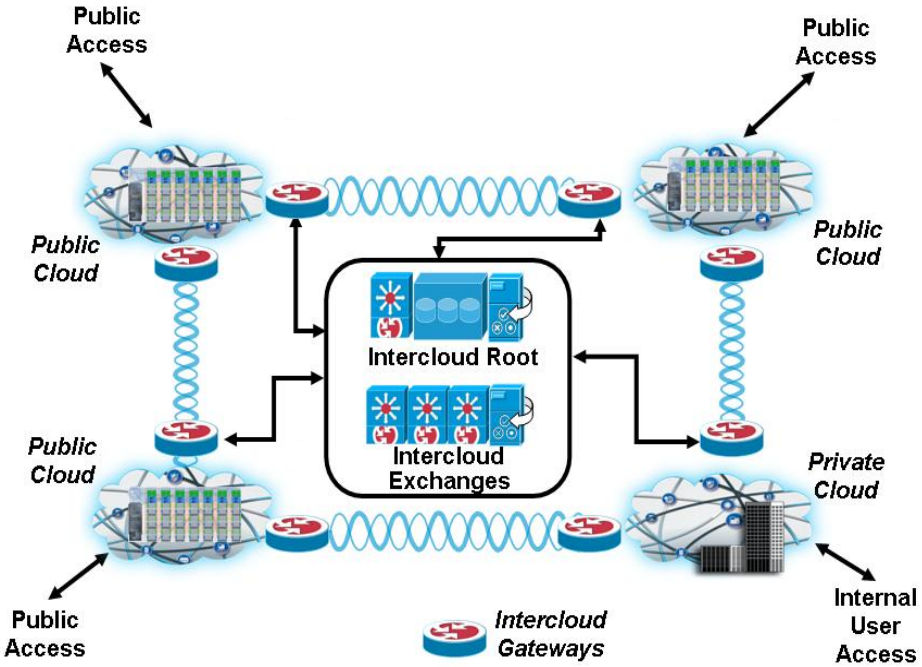


Figure 7: Reference Intercloud network topology and elements, from *Bernstein, "The Intercloud"(2009)*

As shown, it is modeled after the public Internet infrastructure. Again, using the generally accepted terminology, there are Public Clouds, which are analogous to ISP's. There are Private Clouds which is simply a Cloud which an organization builds to serve itself. There are Intercloud Exchanges (analogous to Internet Exchanges and Peering Points – called Brokers in the NIST Reference Architecture) where clouds can interoperate, and there is an Intercloud Root, containing services such as Naming Authority, Trust Authority, Directory Services, and other "root" capabilities. It is envisioned that the Intercloud root is of course physically not a single entity, a global replicating and hierarchical system similar to DNS would be utilized.

All elements in the Intercloud topology contain some gateway capability analogous to an Internet Router, implementing Intercloud protocols in order to participate in Intercloud interoperability. These would be called Intercloud Gateways.

Bernstein, D., Vij, D., Diamond, S.,: *An Intercloud Cloud Computing Economy - Technology, Governance, and Market Blueprints*, IEEE Service Research and Innovation Global Conference – SRII 2011, (2011)
Bernstein, D., Vij, D, *Intercloud Exchanges and Roots Topology and Trust Blueprint*, ICOMP'11 - The 2011 International Conference on Internet Computing, (2011)

The Intercloud Gateways would provide mechanism for supporting the entire profile of Intercloud protocols and standards. The Intercloud Root and Intercloud Exchanges would facilitate and mediate the initial Intercloud negotiating process among Clouds.

Once the initial negotiating process is completed, each of these Cloud instance would collaborate directly with each other via a protocol and transport appropriate for the interoperability action at hand; for example, a reliable protocol might be needed for transaction integrity, or a high speed streaming protocol might be needed optimized for data movement over a particular link.

As described earlier that various providers will emerge in the enablement of the Intercloud. One can first envision a community governed set of Intercloud Root providers who will act as brokers and host the Cloud Computing Resource Catalogs for the Intercloud computing resources. They would be governed in a similar way in which DNS and Top Level Domains by an organization such as ISOC or ICANN. There would also be a responsible for mediating the trust based federated security among disparate clouds by acting as Security Trust Service providers using standards such as SASL and SAML. This might be the IGTF.

As part of the proposed topology, the Intercloud Root providers would be federated in nature. Each of these federated nodded in the overall Intercloud topology will independently manage the "root" capabilities such as Cloud Resources Directory Services, Trust Authority, Presence Information etc. Additionally, each Intercloud Root instance will be associated with its affiliated Exchanges by defining the affiliation relationship as part of the Intercloud "root" instance.

In order for the Intercloud capable Cloud instances to federate or otherwise interoperate resources, a Cloud Computing Resources Catalog system is necessary infrastructure. This catalog is the holistic and abstracted view of the computing resources across disparate cloud environments. Individual clouds will, in turn, will utilize this catalog in order to identify matching cloud resources by applying certain Preferences and Constraints to the resources in the computing resources catalog.

The technologies to use for this are based on the Semantic Web which provides for a way to add "meaning and relatedness" to objects on the Web. To accomplish this, one defines a system for normalizing meaning across terminology, or Properties. This normalization is called Ontology. Cloud Computing resources can be described, cataloged, and mediated using Semantic Web Ontologies, implemented using RDF techniques.

Due to the sheer size of global resources ontology information, a centralized approach for hosting the repository is not a viable solution due to the fact that one single entity can not be solely responsible and burdened with this humongous and globally dispersed task. Instead, Intercloud Roots will host the globally dispersed computing resources catalog in a federated manner.

Intercloud Exchanges, in turn, will leverage the globally dispersed resources catalog information hosted by federated Intercloud Roots in order to match cloud resources by applying certain Preferences and Constraints to the resources. From overall topology perspectives, Intercloud Exchanges will provide processing nodes in a peer-to-peer manner on the lines of DHT overlay based approach in order to facilitate optimized resources match-making queries. Ontology

information would be replicated to the Intercloud Exchanges (DHT overlay nodes) from their affiliated Intercloud Roots using a "Hash" function.

Nodes within the DHT overlay system are uniformly distributed across key space and maintain list of neighbors in the routing table. Each peer in the DHT overlay system is responsible for some part of the overall key space and maintains additional routing information to forward queries to neighboring peers. As the number of machines taking part in the network and the amount of shared information evolve, peers opportunistically organize their routing tables according to a dynamic and distributed binary search tree.

From Intercloud topology perspectives, Intercloud Roots will provide PKI CA root like functionality. According to the current PKI based trust model, once the CA authorizes the certificate for an entity, the entity is either trusted or non-trusted. However, in the cloud computing environment, especially in the Intercloud environment, this model needs to be extended to have "Trust Zone" to go along with the existing PKI based trust model. Intercloud exchanges will be responsible for the "Trust Zone" based trust model layered on top of the PKI certificate based trust model.

Exchanges are the custodians/brokers of "Domain based Trust" systems environment for their affiliated cloud providers. Cloud providers rely on the Intercloud exchanges to manage trust. As part of the identification process for matching desired cloud resources, individual consumer cloud provider will signify the required "Trust Zone" value such as "Local Intercloud Exchange" domain or "Foreign Intercloud Exchange". Depending on the desired "Trust Zone" value, for example, one Intercloud provider might trust another provider to use its storage resources but not to execute programs using these resources. Intercloud Exchanges, in turn, will utilize the desired "Trust Zone" value as part of the matching Preferences and Constraints in order to identify matching cloud resources.

This overall design of decentralized, scalable, self-organizing federated "Intercloud" topology fits with the overall NIST Cloud Computing Reference Architecture, in the various aspects of Intercloud topology components such as Intercloud Resources Directory, Intercloud Collaboration, and Intercloud Security. This guideline will help solidify what might be behind the Broker that is referenced to federate to "outside" Cloud resources.

10 Acronyms

ACID – Atomicity, Consistency, Isolation, Durability, for a Transaction

ADO.NET – ActiveX Data Object for .NET

ASP.NET – Active Server Pages for .NET

AWS – Amazon Web Services

BLOB – Binary Large Object

CA – Certificate Authority

CDN – Content Distribution Network

CPU – Central Processing Unit

DDOS – Distributed Denial of Service

DEVOP – Develop and Operate, referencing the "you build it – you run it" Cloud philosophy

DHCP – Dynamic Host Control Protocol

DHT – Distributed Hash Table

DNS – Domain Name System

DR – Disaster Recovery

EBS – Elastic Block Storage

EJB – Enterprise Java Bean

HPC – High Performance Computing

HTTP – Hyper Text Transfer Protocol

IAAS – Infrastructure As A Service

ICANN - Internet Corporation for Assigned Names and Numbers

IDS – Intrusion Detection System

IGTF – The International Grid Trust Federation

IIS – Internet Information Server

IPS – Intrusion Prevention System

ISP – Internet Service Provider

IPSEC – Internet Protocol Security

ISOC – The Internet Society

J2EE – Java 2 Platform, Enterprise Edition

JCA – Java Connector Architecture

JTA – Java Transactions Architecture

L2 – Layer Two Network

L3 – Layer Three Network

LAMP – Linux, Apache, MySQL, and PHP/Perl/Python

LAN – Local Area Network

MPLS – Multiprotocol Label Switching

.NET - Software framework from Microsoft that runs primarily on Microsoft Windows

NAP – Network Access Point

NAT – Network Address Translation

NTFS – Windows NT Filesystem

PAAS – Platform As A Service

PKI – Public Key Infrastructure

RDF – Resource Description Framework

RPC – Remote Procedure Call

S3 – Simple Storage Service from Amazon

SAAS – Software As A Service

SAJACC – Standards Acceleration to Jumpstart the Adoption of Cloud Computing

SAML – Security Assertion Markup Language

SASL – Simple Authentication and Security Layer

SimpleDB – Simple Database

SHTTP – Secure HTTP

SOA – Service Oriented Architecture

SQL – Structured Query Language

SSO – Single Sign On

VHD – Virtual Hard Disk

VLAN – Virtual LAN

VM – Virtual Machine

VPC – Virtual Private Cloud

VPN – Virtual Private Network

WYSIWYG – What You See Is What You Get

11 Glossary

Actor – Each actor is an entity (a person or an organization) that participates in a transaction or process and/or performs tasks in cloud computing

App - That software which a person or an organization operates, to perform a task. An App runs on a Mobile Platform

Application, or Cloud Application – That software which a person or an organization operates, to perform a task. An Application runs on a Platform (Computing Platform or Cloud Platform)

Application Developer - A type of Task, where a person or an organization is creating, with programming, a cloud based application which will be used by them, as well as other persons or organizations

Application Operator – A type of Task, where a person or an organization is using a cloud based application in a mode with a data entry mode; persistent data is entered by person or an organization and the application reliably stores this data for use at any later time

Application Spectator – A type of Task, where a person or an organization is using a cloud based application in a mode with no data entry mode; no persistent data is entered by person or an organization - the application is presenting requested data to the person or organization and they are "a spectator" of that data

Browser - Software for retrieving, presenting, and traversing information resources on the World Wide Web

Case Examples, Illustrative Case Examples – Cloud powered application scenarios chosen to illustrate an a particular way that an application interacts with users, and uses software features of a one or more cloud(s)

Cloud Implementer – A specialized person or organization within a Cloud Provider who constructs the physical cloud (servers, storage, and network) and/or the Cloud Operating System

Cloud Interoperability - The capability to communicate between or among multiple clouds with the capability that clouds can provide services to each other

Cloud Operating System – That software which runs on the physical cloud (servers, storage, and network) and by running, implements one or more Cloud Service Models. Typically installed by the Cloud Implementer and operated by the Cloud System Administrator

Cloud Migration – The process of taking an application from a traditional physical server, storage, and network platform to a cloud platform. The process may include taking the application as well as supporting operating system, middleware, storage, and networking software

Cloud Portability – The ability to move data or applications across multiple cloud environments

Cloud Provider – A person, an organization; it is the entity responsible for making a service available to interested parties

Cloud Services – An Application, running on a Cloud

Daemon - A computer program that runs as a background process, rather than being under the direct control of an interactive user

Decision Framework for Cloud Adoption- As presented in the Federal Cloud Computing Strategy, defines three stages for a cloud project, in terms of moving or creating applications on a cloud: (1) Selecting services to move to the cloud, (2) Provisioning cloud services effectively, and (3) Managing services rather than assets

Intercloud – A form of Cloud Interoperability with an architectural analogy to the Internet

Mobile Platform – A computing system which can be easily carried by a human and can connect by wire or wireless to the Internet

Platform, Computing Platform, or Cloud Platform - Supporting software (middleware and Operating System) and/or physical computers, storage, and networking

Private Cloud - Gives a single Cloud Consumer's organization the exclusive access to and usage of the infrastructure and computational resources. It may be managed either by the Cloud Consumer organization or by a third party, and may be hosted on the organization's premises (i.e. on-site private clouds) or outsourced to a hosting company (i.e. outsourced private clouds).

Public Cloud - The cloud infrastructure and computing resources are made available to the general public over a public network. A public cloud is owned by an organization selling cloud services, and serves a diverse pool of clients.

Researcher - A type of Task, where a person or an organization is using a cloud based application as computing tool; like an Application Developer, except the resultant application is primarily intended to be used by that Researcher; it may be a calculational tool without a user interface

Role – Groupings of users defined by organizational affiliations and, implicitly, by corresponding access rights that users possess while they consume from a cloud

Service Models - Three service models identified by the NIST Cloud Computing Definition, i.e. SaaS, PaaS, and IaaS

System Administrator - A type of Task, where a person or an organization is operating, reconfiguring, repairing, and maintaining, a cloud or a set of cloud applications for others to use

Tasks - What a Role is trying to accomplish by using a cloud

Transaction - An event that generates or modifies data that is eventually stored in an information system. To be considered a transaction processing system the computer must pass the ACID test