Software Engineering Institute

T-Check in System-of-Systems Technologies: Cloud Computing

Harrison D. Strowd
Grace A. Lewis

September 2010

TECHNICAL NOTE
CMU/SEI-2010-TN-009

Research, Technology, and System Solutions (RTSS) Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

Table of Contents

List of Figures

List of Tables

Acknowledgements

The authors would like to thank Carnegie Mellon University's Master of Science in Information Technology - Software Engineering (MSIT-SE) program for co-sponsoring the independent study that resulted in this report, as well as Carnegie Mellon® Software Engineering Institute colleagues John Klein and Marc Novakouski for their valuable technical reviews.

Abstract

This technical note presents the results of applying the T-Check method in an initial investigation of cloud computing. In this report, three hypotheses are examined: (1) an organization can use its existing infrastructure simultaneously with cloud resources with relative ease; (2) cloud computing environments provide ways to continuously update the amount of resources allocated to an organization; and (3) it is possible to move an application's resources between cloud computing providers, with varying levels of effort required. From the T-Check investigation, the first hypothesis is partially sustained and the last two hypotheses are fully sustained within the context specified for the investigation.

From an engineering perspective, cloud computing is a distributed computing paradigm that focuses on providing a wide range of users with distributed access to virtualized hardware and/or software infrastructure over the internet. From a business perspective, it is the availability of computing resources that are scalable and billed on a usage basis. While scalability is the primary tenet of cloud computing, a host of other advantages are advertised as being inherently obtained through cloud computing.

1 Introduction

Cloud computing is an emerging paradigm that focuses on providing dynamic, on-demand scalability of virtualized hardware and/or software resources to a diverse set of users. While scalability is the primary tenet of cloud computing, a host of other advantages are advertised as being inherently obtained through cloud computing. The purpose of this report is to apply the T-Check[SM] method to examine a set of claims about cloud computing adoption.

A T-Check investigation is a simple and cost-effective way to understand and evaluate the claims made about a technology in a given context [Lewis 2005]. Specifically, this T-Check investigation focuses on finding initial answers to the following questions:

1. How difficult is it for an organization to use existing internal resources simultaneously with cloud resources?
2. What mechanisms are provided for users to update their resource allocations dynamically?
3. How difficult is it to move an application from one cloud provider to another?

The rest of this section will provide a brief introduction to cloud computing and related technologies. Section 2 presents the context for the T-Check investigation and how the above questions were addressed. Section 3 describes the solutions employed to evaluate the proposed hypotheses, and Section 4 presents the results of the evaluation. Section 5 discusses plans for future work in this area and briefly presents the conclusions reached in this experimental setting.[1]

1.1 Cloud Computing

Cloud computing is an emerging technology that has sparked the interest of a wide range of organizations. In general, cloud computing is a distributed computing paradigm that focuses on providing a wide range of users with distributed access to scalable, virtualized hardware and/or software infrastructure over the internet.

Many definitions have been offered for this term. According to Foster and colleagues, cloud computing is

> *a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the internet* [Foster 2008].

McEvoy and Schulze define it as "a style of computing where massively scalable IT-related capabilities are provided as a service across the Internet to multiple external customers" [McEvoy 2008]. Erdogmus provides a concise definition by saying "Cloud computing is an emerging computational model in which applications, data, and IT resources are provided as services to users over the Web" [Erdogmus 2009].

[SM] T-Check is a service mark of Carnegie Mellon University.

[1] A large part of this work was performed as an independent study in Carnegie Mellon University's Master of Science in Information Technology – Software Engineering (MSIT-SE) program.

1.2 Types of Cloud Computing

Cloud computing implementations can be characterized in two orthogonal ways: (1) by the capabilities they provide and (2) by who can access their resources. Based on capabilities, there are three types of cloud computing implementations:

1. **Infrastructure as a Service (IaaS)**

   This is mainly computational infrastructure available over the internet, such as compute cycles and storage, which can be utilized in the same way as internally owned resources. IaaS providers enforce minimal restrictions on their users[2] to allow them maximum control and configuration of the resources. These resources typically provide a variety of interfaces to facilitate interaction, and there are usually additional services provided, such as query services for storage resources.

   From the user's perspective, these resources appear to be identical to resources that are owned, operated, and maintained by the organization. The key difference is that users pay only for the bandwidth, computation, and storage that they use. If at any time the resources are no longer needed, they can potentially be terminated without incurring any additional costs. This removes the large upfront cost associated with acquiring hardware resources, and the scalability of such resources allows users to handle variability in their application's usage, paying for the extra resources only when they are required.

2. **Platform as a Service (PaaS)**

   PaaS refers to application development platforms—hardware and software components—that enable users to leverage the resources of established organizations to create and host applications of a larger scale than an individual or a small organization would be able to handle. Services include, but are not limited to, software installation and configuration, resource scaling, and platform maintenance and upgrading. In order to enable these services, the provider places restrictions on the user by specifying various aspects of the platform, such as the programming languages supported, data storage mechanisms, and resource monitoring capabilities. In this model, user organizations use resources from the cloud and deploy their applications in the cloud as well.

   From the user's perspective, these providers offer significant functionality out-of-the-box. The key requirements for deploying an application into such an environment are to ensure that the selected platform will support the application and that the services offered meet the needs of the user. When these key criteria align, the user is able to leverage a significant amount of functionality with potentially very little effort.

3. **Software as a Service (SaaS)**

   SaaS focuses on providing users with business-specific capabilities—hardware and software applications. In general, SaaS is a model of software deployment in which a provider licenses an application to user organizations for use as a service on demand. However, there is a wide spectrum of what is covered under SaaS, and which parts of SaaS fall under the definition of cloud computing is often debated.

---

[2] The term *user* will be used throughout the report to refer to the organization (or individual) that acquires resources from the cloud to be used as part of its IT infrastructure. It does not refer to the end user of the applications that are hosted in the cloud. If this is the case, the term *end user* will be used. Other terms used by cloud providers and researchers to refer to user organizations include *consumer*, *customer,* and *tenant*.

According to Chong and Carraro, there are multiple levels of SaaS [Chong 2006]:

- Level 1: An application is specifically run for one user organization at an SaaS provider, similar to the traditional ASP (application server provider) model.
- Level 2: The SaaS application is customizable via configuration, and one instance of the application serves only one user organization.
- Level 3: The SaaS application is customizable, and a single instance of the SaaS application serves multiple user organizations.
- Level 4: The SaaS application is developed as a single instance multi-tenant[3] application, and several instances are run in a load-balanced server farm.

From the user organization's perspective, SaaS enables organizations to use out-of-the-box, business-specific capabilities developed by third parties instead of acquiring, hosting, and managing large software packages or developing proprietary solutions.

Table 1 shows some examples of the three cloud computing types. For simplicity, we classified providers based on their primary focus.

Table 1: Examples of Cloud Computing Providers by Type[4]

	Providers	Brief Description
IaaS	Amazon Elastic Compute Cloud (EC2)	Provides users with a special virtual machine (AMI) that can be deployed and run on the EC2 infrastructure [Amazon 2010a]
	Amazon Simple Storage Solution (S3)	Provides users with access to dynamically scalable storage resources [Amazon 2010b]
	GoGrid	Provides users with access to dynamically scalable computing and storage resources, as well as dedicated servers [GoGrid 2010]
	IBM Computing on Demand (CoD)	Provides users with access to highly configurable servers plus value-added services such as data storage [IBM 2010]
	Microsoft Live Mesh	Provides users with access to a distributed file system; targeted at individual use [Microsoft 2010a]
	Rackspace Cloud	Provides users with access to dynamically scalable computing and storage resources, as well as third-party cloud applications and tools [Rackspace 2010]

[3] The term used by SaaS providers to refer to user organizations is *tenant*.

[4] It is difficult to classify providers as purely IaaS, PaaS, or SaaS. For example, the Microsoft Azure Services Platform could be classified as IaaS and PaaS, and Force.com could be classified as PaaS and SaaS. However, for simplicity, the providers are classified based on their primary focus.

Table 1: Examples of Cloud Computing Providers by Type (cont.)

	Providers	Brief Description
PaaS	Akamai EdgePlatform	Provides a large distributed computing platform on which organizations can deploy their web applications; large focus on analysis and monitoring of resources [Akamai 2010]
	Force.com (from salesforce.com, an SaaS provider)	Provides users a platform to build and run applications and components bought from AppExchange or custom applications [Salesforce 2010a]
	Google App Engine (GAE)	Provides users a complete development stack and allows them to run their applications on Google's infrastructure [Google 2010a]
	Microsoft Azure Services Platform	Provides users with on-demand compute and storage services as well as a development platform based on Windows Azure [Microsoft 2010b]
	Yahoo! Open Strategy (Y!OS)	Provides users with a means of developing web applications on top of the existing Yahoo! platform, and in doing so leveraging a significant portion of the Yahoo! Resources [Yahoo 2010]
SaaS	Google Apps	Provides web-based office tools such as e-mail, calendar, and document management tools [Google 2010b]
	Salesforce.com	Provides a full customer relationship management (CRM) application [Salesforce 2010b]
	Zoho	Provides a large suite of web-based applications, mostly for enterprise use [Zoho 2010]

Based on who can access resources, there are two types of cloud computing implementations or deployment models:[5]

1. **Public clouds**

   In public clouds, resources are offered as a service, usually over an internet connection, for a pay-per-usage fee. Users can scale on demand and do not need to purchase hardware. Cloud providers manage the infrastructure and pool resources into capacity required by its users.

2. **Private clouds**

   Private clouds are typically deployed inside a firewall and managed by the user organization. In this case, the user organization owns the software and hardware running in the cloud, manages the cloud, and provides virtualized cloud resources. These resources are typically not shared outside the organization and full control is retained by the organization. Examples of companies that provide resources for organizations to build private clouds include

   - 3tera: Provides developers with tools to build their own cloud computing infrastructures [3tera 2010]
   - Eucalyptus Systems: Provides an open-source application that can be used to implement a cloud computing environment on a datacenter. This organization is also trying to establish a set of open standards for cloud computing [Eucalyptus 2010].

[5] The National Institute of Standards and Technology (NIST) defines two additional types of cloud deployment models: (1) community clouds that are shared by multiple organizations and support specific needs and concerns of a community and (2) hybrid clouds that are the combination of two or more public, private, and community clouds. However, both community and hybrid cloud are specialties of public and private clouds and are therefore not included in the discussion. Additional information is available at http://csrc.nist.gov/groups/SNS/cloud-computing/.

- Ubuntu: Provides server software that can be used to implement scalable, manageable virtual server images [Ubuntu 2010]

1.3 Drivers and Barriers to Cloud Computing Adoption

According to Gartner's Hype Cycle for Emerging Technologies, cloud computing is currently at the "peak of inflated expectations" [Gartner 2009]. One of the primary goals of this study is to better understand the claims being made about cloud computing and whether they accurately depict the technology. In our initial research into cloud computing, we identified a wide range of claims being made about cloud computing adoption. We have classified them as either drivers for or barriers to cloud computing adoption and have documented them in Table 2 and Table 3, respectively. In each table, the entries are listed in alphabetical order.

Some of these claims will be used as input to the study that will be described in the next section.

Table 2: Drivers for Cloud Computing Adoption

Driver	Description
Availability	Users have the ability to access their resources at any time through a standard internet connection.
Collaboration	Users are starting to see the cloud as a way to work simultaneously on common data and information.
Elasticity	The provider transparently manages a user's resource utilization based on dynamically changing needs.
Lower Infrastructure Costs	The pay-per-usage model allows an organization to pay only for the resources it needs, with basically no investment in the physical resources available in the cloud. There are also no infrastructure maintenance or upgrade costs.
Mobility	Users have the ability to access data and applications from around the globe.
Risk Reduction	Organizations can use the cloud to test ideas and concepts before making major investments in technology.
Scalability	Users have access to a large amount of resources that scale based on user demand.
Virtualization	Each user has a single view of the available resources, independently of how they are arranged in terms of physical devices. Therefore, there is potential from a provider perspective to serve a greater number of users with fewer physical resources.

Table 3: Barriers to Cloud Computing Adoption

Barrier	Description
Interoperability	A set of universal standards and/or interfaces has not yet been defined, resulting in a significant risk of vendor lock-in.
Latency	All access to the cloud is done via the internet, introducing latency into every communication between the user and the provider.
Platform or Language Constraints	Some cloud providers support specific platforms and languages only.
Regulations	There are concerns in the cloud computing community over jurisdiction, data protection, fair information practices, and international data transfer that are a concern mainly to organizations that manage sensitive data.
Reliability	Many existing cloud infrastructures leverage commodity hardware that is known to fail unexpectedly.
Resource Control	The amount of control that the user has over the cloud provider and its resources varies greatly between providers.
Security	The main concern is data privacy: users do not have control of or know where their data is being stored.

2 Using the T-Check Method

The T-Check method is a context-dependent technique for evaluating technologies. This method involves (1) formulating hypotheses about the technology and (2) examining these hypotheses against specific criteria through hands-on experimentation. The outcome of this two-stage method is that the hypotheses are either sustained (fully or partially) or refuted. The T-Check method has the advantage of producing very efficient and representative experiments that not only evaluate technologies in the context of their intended use but also generate hands-on competence with the technologies. A claim can be made that the simplicity of the experiments implies that the results do not scale. However, it can be argued that when the experiments disprove a claim "in the small," the results could be similar "in the large" or warrant additional experimentation at a larger scale. A graphical representation of the T-Check process is shown in Figure 1.

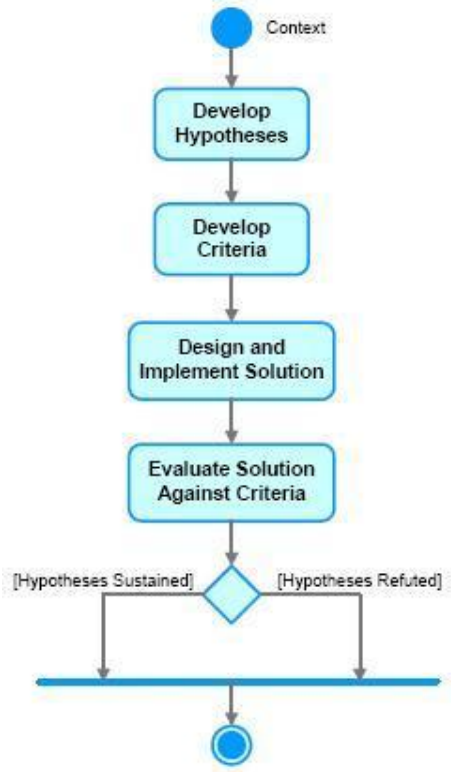

Figure 1: T-Check Process for Technology Evaluation

The T-Check method is part of a larger process for context-based technology evaluation. In this larger process, the context for the T-Check is established and the expectations from the technology are captured within the context in which they are going to be used, including organizational constraints [Lewis 2005].

2.1 T-Check Context

The context for this T-Check investigation is a hypothetical organization that has developed and is maintaining a simple calendar application to advertise events that are of interest to the community. Each entry in the calendar consists of an identification number (issued by the system), a title, a date, and event details. The application has three actions that are available to its end users: creating a new entry, updating an existing entry, and searching the entire set of entries.[6] Because the calendar is primarily used for advertising events that are of interest to many people, the entries in the calendar are created and updated infrequently but are searched frequently. This means that the limiting resource is the compute power required to perform the requested searches.

The organization has been using infrastructure that is owned and operated internally to support its calendar application. Currently, the organization is experiencing rapid growth and its internal resources are no longer sufficient to support its end-user base. At the same time, the organization is constrained by a budget that cannot support the cost required to purchase the machines needed to support peak loads. The organization is highly concerned with stretching its budget and would like to continue to make use of the existing infrastructure as much as possible to reduce overall costs.

While recognizing the need to support the current increase in end users, the organization has plans for growing its applications and expects similar boosts in end-user numbers over the next five years. As a result, the organization's leaders would like to be prepared to handle similar situations more gracefully in the future. Unfortunately, these events will not always be predictable, requiring the ability to modify the resource allocation dynamically. Based on these factors, the organization feels that cloud computing would be an ideal fit. At the same time, the organization realizes that cloud computing is still considered an emerging technology and is concerned about the possibility of vendor lock-in. The ability to change cloud computing providers, if needed, is an important concern for the organization.

In regard to its concerns, the organization would like to answer the following questions about cloud computing:

1. How difficult is it for an organization to run an application using existing internal resources simultaneously with cloud resources?
2. What mechanisms are provided for users to update their resource allocation dynamically? How frequently can users update their resource allocation (once per second, minute, hour, day, etc.)? What size variance, in terms of resource quantities, is supported for users to update their resource allocation (launching/terminating 1, 10s, 100s, or 1,000s of machines)?
3. How difficult is it to move an application from one cloud provider to another?

2.2 Develop Hypotheses

To answer these questions, we have identified the following hypotheses to be verified or refuted in this study:

1. An organization can deliver an application that uses its existing infrastructure simultaneously with cloud resources, with relative ease.
2. Cloud computing environments provide ways to continuously update the amount of resources allocated to an organization.

[6] Delete functionality was left out in order to simplify the implementation.

3. It is possible to move an application's resources from one cloud provider to another, with varying levels of effort required.

2.3 Develop Criteria

Table 4 shows the evaluation criteria used to determine whether a hypothesis has been verified or refuted.

Table 4: Criteria Used to Evaluate the Hypotheses

Hypothesis	Criteria
An organization can deliver an application that uses its existing infrastructure simultaneously with cloud resources, with relative ease.	Assuming the application is **not** currently utilizing cloud computing resources, internal infrastructure can be integrated with cloud resources in less time than was required to develop the data access components[7] of the internal version of the application and by changing only the data access code and less than 10% of the remainder of the application's lines of code.
Cloud computing environments provide ways to continuously update the amount of resources allocated to an organization.	• The cloud environment provides users with access to information about their currently allocated resources and utilization levels. • The cloud environment allows users to modify the resources allocated to them dynamically with no impact to existing capabilities. • The cloud environment is responsible for monitoring the user's utilization and allocating more or less resources as needed.
It is possible to move an application's resources from one cloud provider to another, with varying levels of effort required.	• For cloud providers that support the programming language in which the application was developed, the application can be moved to another cloud provider in half as much time as it took to develop the data access components of the internal version of the application and by changing only the data access code and less than 5% of the remainder of the application's lines of code. • For cloud providers that do not support the language in which the application was developed, this application can be moved to another cloud provider in less than 1.5 times as long as it took to develop the data access components of the internal version of the application and by changing only the data access code and less than 10% of the remainder of the application's lines of code.

[7] The rationale for comparing against development times for data access components is that the solution will switch between data sources at runtime.

3 Designing and Implementing the Solution

3.1 Defining the Initial System Architecture Based on the T-Check Context

The first step in designing the solution was to specify the architecture of the initial system that is entirely supported by the internal resources of the organization, based on the T-Check context. This architecture provided a foundation for the changes that would be required to test each hypothesis. As prescribed by the T-Check method, this architecture represents the simplest solution that can be used to evaluate the hypotheses of the study. In order to achieve the simplest possible solution, we have made the following simplifying assumptions:

- Messages transmitted from the organization's resources to cloud resources, and vice-versa, are not lost.

  Rationale: This assumption will likely not hold all of the time, but the focus of this study is not on reliable message transmission. Other studies have addressed this topic and have highlighted tactics for handling situations when reliability is critical.

- The application is able to determine the utilization of internal resources.

  Rationale: In order to know when requests should be allocated to the cloud resources, the application will need to determine the current utilization of the internal resources. This is a language- and platform-dependent task that, while crucial for an application being deployed, will not affect the results of the T-Check. For this reason we will assume that the application has the ability to determine this information.

After making these simplifying assumptions, we designed the initial application. Figure 2 depicts the static perspective of the architecture for this system in the form of a module view.[8] The element and relationship responsibilities in this view are provided in Table 5 and Table 6, respectively.

PHP was selected as the implementation language because of our experience and also because it is becoming a widely used language for web development [PHP 2010].

[8] In practice, the redirection to the cloud would be handled by a load balancer. However, the *Resource Manager* component is the software equivalent of a load balancer and appropriate for this simple application and T-Check investigation.

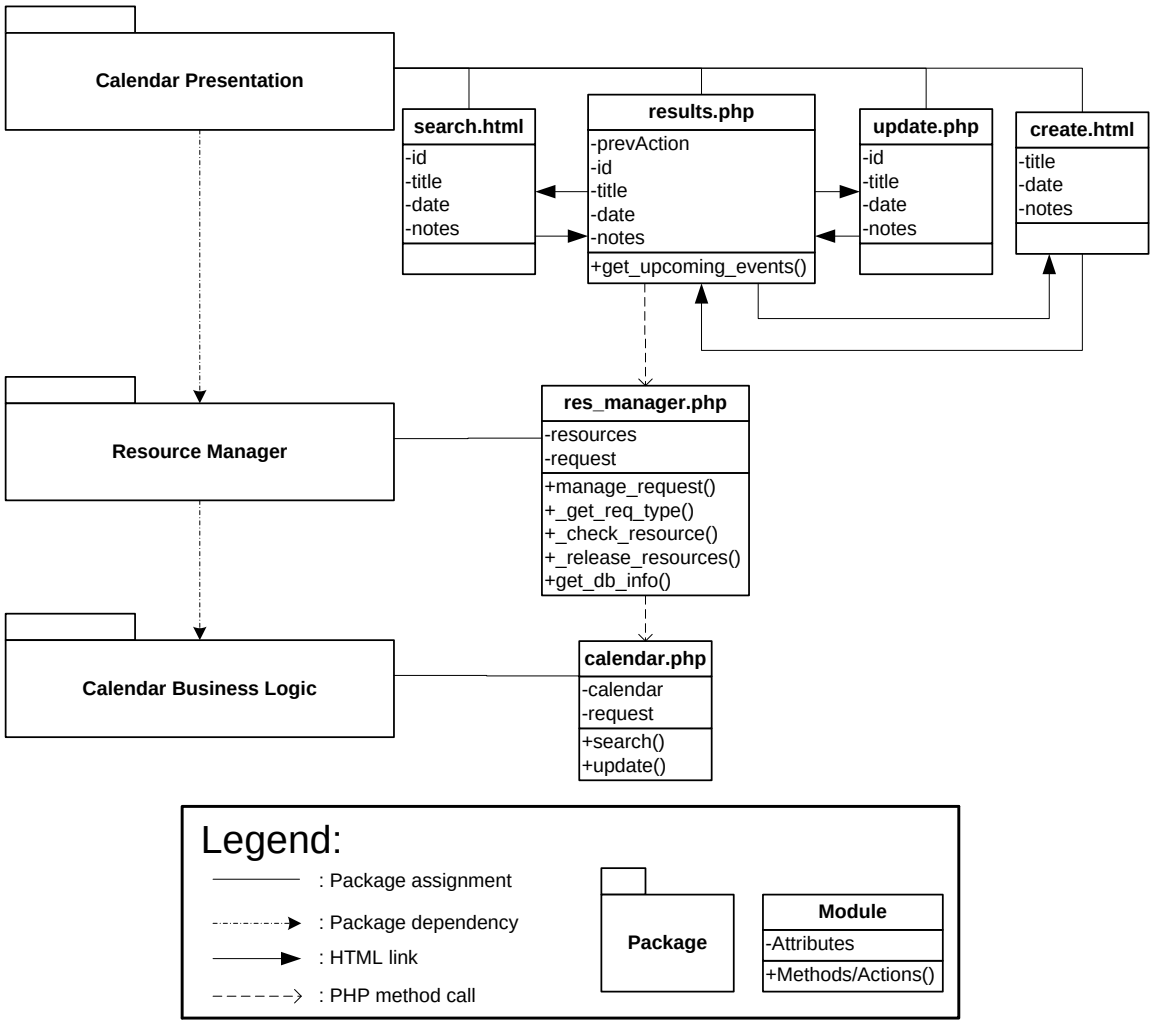

Figure 2: Module View of the Initial System

Table 5: Element Responsibilities for the Module View of the Initial System

Associated Diagram: Figure 2; View: Module	
Element	**Responsibilities**
Calendar Presentation	• Contains all modules that relate to the presentation of entries in the calendar—results.php, search.html, create.html, and update.php—which are served to the end user by an Apache[9] web server (not shown here) • Allows the end user to issue a *create*, *update*, or *search* request to the system • Relies on the *Resource Manager* package to allocate end user requests to the appropriate set of resources
Resource Manager	• Contains the res_manager.php module which is responsible for accepting a request and submitting to a set of resources to be processed • Provides an interface for the *Calendar Presentation* package to submit requests and calls the *Calendar Business Logic* package to process these requests

[9] The Apache web server is an open-source HTTP server maintained by the Apache Software Foundation (http://httpd.apache.org/)

Table 5: Element Responsibilities for the Module View of the Initial System (cont.)

Associated Diagram: Figure 2; View: Module	
Element	**Responsibilities**
Calendar Business Logic	• Contains the calendar.php module which has access to all the entries in the calendar and allows these entries to be created, updated, or searched • Provides an interface for the *Resource Manager* package to submit these requests
results.php	• Part of the Calendar Presentation package and is responsible for displaying the results of the end user's request • Contains links to the *search.html*, *create.html*, and *update.php* pages, which are used to issue requests • Does not accept any input from the end user • When an end user submits a request, that information is sent to this module which is then responsible for submitting the request and displaying the results. To do so it passes the request to the *manage_request* method of the *resource_manager.php* module.
search.html	• Part of the *Calendar Presentation* package and is responsible for accepting the appropriate input from the end user to search the calendar for a set of entries • Allows the end user to specify the ID, title, date, and/or notes of the entries they are looking for • Upon receipt of an end user's request, it passes the request to the *results.php* module, which will display the results from the calendar search. • Does not display any calendar information to the end user
create.html	• Part of the *Calendar Presentation* package and is responsible for accepting the appropriate input from the end user to create a new entry in the calendar • Allows the end user to specify the title, date, and/or notes of the entries to create • Upon receipt of an end user's request, it will pass the request to the *results.php* module, which will display the results from creating the entry. • Does not display any calendar information to the end user
update.php	• Part of the *Calendar Presentation* package and is responsible for accepting the appropriate input from the end user to update an entry in the calendar • Receives the information about an entry from the *results.php* module and then populates the text fields with this information to allow the end user to edit the title, date, and/or notes • Upon receipt of an end user's request, it passes the request to the *results.php* module, which will display the results from updating the entry. • Does not display any calendar information to the end user
res_manager.php	• Part of the *Resource Manager* package and is responsible for accepting requests from the *results.php* module, identifying the type of request, identifying the current availability of the internal resources, and assigning the request to the appropriate set of resources (internal or cloud) to process the request • Forwards all requests to the appropriate method of the *calendar.php* module in the selected set of resources
calendar.php	• Part of the *Calendar Business Logic* package and is responsible for servicing the various types of end user requests • Keeps track of all entries in the calendar and allows for them to be searched, created, or updated with the appropriate request • Receives requests from the *res_manager.php* module, processes them, and returns the results back to the *res_manager.php* module

Table 6: Relationship Responsibilities for the Module View of the Initial System

Associated Diagram: Figure 2; View: Module	
Relationship	**Responsibilities**
	• Indicates which modules are assigned to a given package • A package can contain an arbitrary number of modules, but a module may be assigned to only one package.
	• Indicates the dependencies between the packages of the system • If package A is dependent on package B, then in order for package A to function properly, package B must be functioning properly. • In this system, this relationship is realized by PHP method calls between the modules contained in the packages.
	• Indicates that an HTML link exists between the two modules • The direction of the relationship points to the destination module of the link.
	• Indicates that a PHP method call is made between two given modules in the system • The direction of the relationship points to the module that is being called.

3.2 Defining the System Architecture for Testing Hypothesis 1

With the architecture of the initial system as a starting point, we designed a solution to evaluate the criteria corresponding to the first hypothesis. This system focuses on managing the available resources, both internal and cloud resources, transparently to the end user.

3.2.1 Dynamic View of the Solution for Testing Hypothesis 1

In this solution, the end users' data needs to be stored in both the internal resources and in the cloud. The cloud will be utilized only for requests that exceed the capacity of the internal resources, when many end users simultaneously try to access the information in the calendar. Figure 3 shows the dynamic perspective of this solution in the form of a component and connector diagram. The element and relationship responsibilities associated with this diagram are provided in Table 7 and Table 8, respectively.

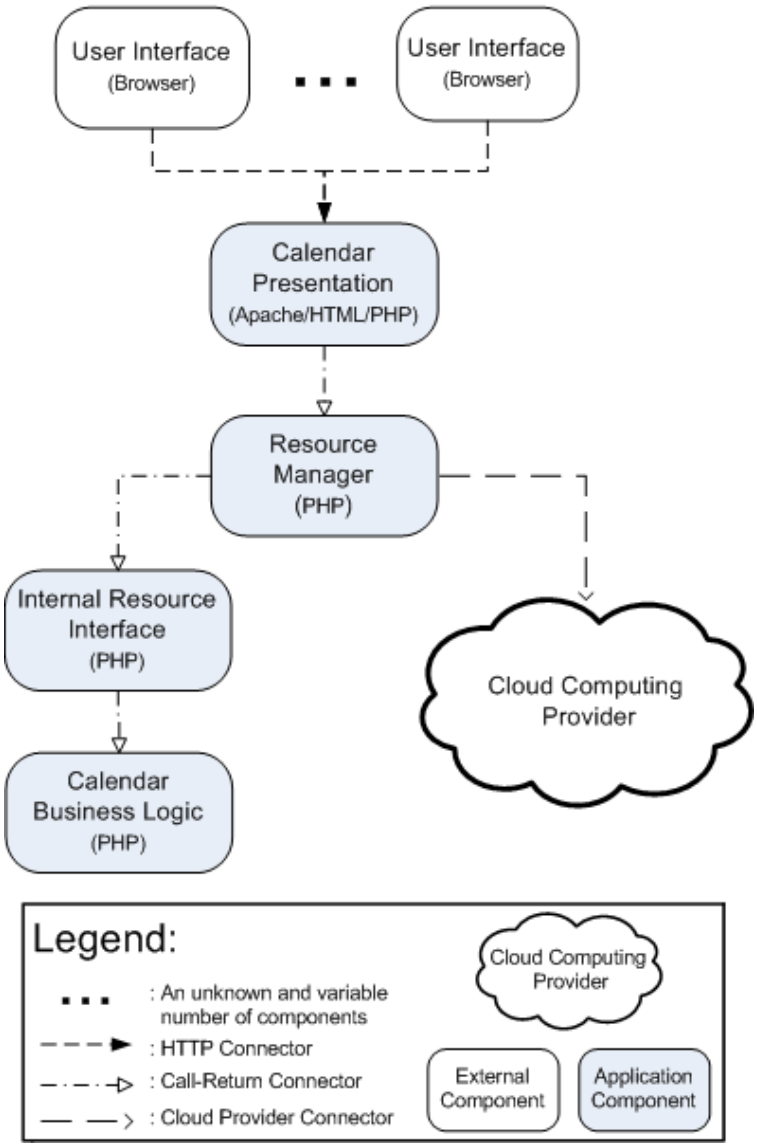

Figure 3: Component and Connector View of the Solution for Testing Hypothesis 1

Table 7: Element Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 1

Associated Diagram: Figure 3; View: Dynamic	
Element	**Responsibilities**
User Interface	• Represents the web browser employed by the end user • Renders any and all information provided by the *Calendar Presentation* component • Allows the end user to enter information and passes it to the *Calendar Presentation* component
Calendar Presentation	• Combination of HTML and PHP that allows the end user to submit requests • Passes the requests to the Resource Manager component to be allocated to the appropriate resources for processing • Receives the resulting information from the Resource Manager component and formats it to be displayed to the end user

Table 7: Element Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 1 (cont.)

Associated Diagram: Figure 3; View: Dynamic	
Element	**Responsibilities**
Resource Manager	• Responsible for allocating end user requests to the appropriate resources for processing • Upon receiving a request to create or update an entry in the calendar, this component will forward the request to both the local infrastructure and the cloud infrastructure to modify both sets of data. • Upon receiving a request to search the entries in the calendar, this component will communicate with the *Internal Resource Interface* component to identify the current utilization of the internal infrastructure. – If the internal resources are sufficient to handle the end-user's request, it will be submitted to the internal resources for processing. – If the internal resources are insufficient to handle the end-user's request, it will be submitted to the cloud resources for processing.
Internal Resource Interface	• Responsible for monitoring the resource utilization of the internal infrastructure available to the application • When requested by the *Resource Manager* component, it will identify the current utilization of the internal resources. • If another request can be supported by the internal resources, it will receive the request from the *Resource Manager* component and assign it to the *Calendar Business Logic* component. • Upon receiving the resulting information from the *Calendar Business Logic* component, it will return this information to the *Resource Manager*.
Calendar Business Logic	• Responsible for processing an end-user's request • Upon receiving a search request, it will search the set of data entries to determine those that match the provided search criteria. • Upon receiving a create request, it will identify the next ID number to be assigned and insert the provided information into the calendar with the identified ID number. • Upon receiving an update request, it will identify the entry to be updated and will make the appropriate modifications to the calendar. • Once the request has been processed, it will return the appropriate information about the results of the operation.
Cloud Computing Provider	• Cloud provider that is responsible for providing the *Resource Manager* component the ability to process end user requests that the internal infrastructure is unable to support • Allows the *Resource Manager* to submit end-user requests to be processed • Contains an instance of the *Calendar Business Logic* component to be used to process these requests • Once a request has been processed, it returns the results to the *Resource Manager* component.

Table 8: Relationship Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 1

Associated Diagram: Figure 3; View: Dynamic	
Relationship	**Responsibilities**
	• Represents an HTTP connection between two components of the system • Utilizes TCP/IP to transmit information between the components • The component to which the connector is pointing is the component receiving the HTTP request

Table 8: Relationship Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 1 (cont.)

Associated Diagram: Figure 3; View: Dynamic	
Relationship	**Responsibilities**
	• Represents a call-return connection between two components of the system • Allows a component to invoke an operation on another component and receive the results. The component to which the connector is pointing is the component being invoked.
	• Represents a connection to a cloud provider • The medium for this connection will depend on the interface available for the cloud provider. • The component to which the connector is pointing is the cloud provider on which an operation is being invoked.

3.2.2 Deployment View of the Solution for Testing Hypothesis 1

From the end-user's perspective, the system should appear no different than if it were using only internal resources. End users should have a single point of access for all requests and receive identical results from both the internal and cloud resources. All of the request and resource management will be done by a single *Resource Manager* component that will be deployed on the organization's internal infrastructure. The deployment view of the system, as shown in Figure 4, identifies how the end user will interact with the system and how the components specified in Figure 3 will be deployed onto the available hardware. The element and relationship responsibilities associated with this diagram are provided in Table 9 and Table 10, respectively.

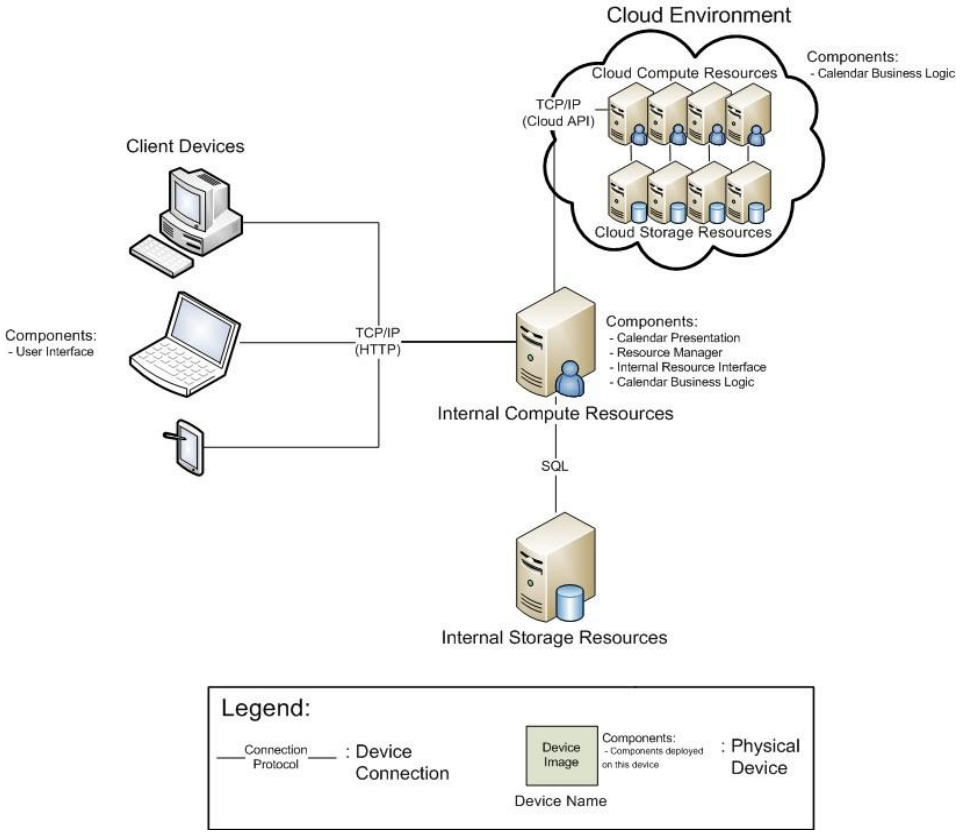

Figure 4: Deployment View of the Solution for Testing Hypothesis 1

Table 9: Element Responsibilities for the Deployment View of the Solution for Testing Hypothesis 1

Associated Diagram: Figure 4; View: Deployment	
Element	**Responsibilities**
Client Devices	• Devices with which the end users interact with the application • Include desktop and laptop computers, PDAs, cellular telephones, and so on. • Interact with the internal infrastructure using TCP/IP to request the web pages that represent the application from the end user's perspective
Internal Compute Resources	• Computation infrastructure owned and operated by the organization • Computation resources are responsible for serving web pages, identifying the current utilization of the internal resources, and forwarding requests to the appropriate set(s) of resources. • All communication with the cloud provider is done via TCP/IP, but many cloud providers have vendor-specific protocols and interfaces built on top of TCP/IP. For this reason, the communication with the *Cloud Environment* will need to be tailored to meet the specific application programming interface (API) of the provider being used.
Internal Storage Resources	• Storage infrastructure owned and operated by the organization • Storage resources hold the entries in the calendar.
Cloud Environment	• Set of resources obtained from a cloud computing provider, including both compute and storage resources • *Internal Compute Resources* forward end-user requests through the API provided by the environment. • Requests are caught and processed by its compute resources, which search, create, and/or update all entries in the calendar stored on its storage resources. • In order to handle user requests, it stores an equivalent of the *Calendar Business Logic* component.

Table 10: Relationship Responsibilities for the Deployment View of the Solution for Testing Hypothesis 1

Associated Diagram: Figure 4; View: Deployment	
Relationship	**Responsibilities**
TCP/IP (HTTP)	• Represents the connection between the end users of the application and the *Internal Computation Resources* of the organization • HTTP connection that is based on TCP/IP • Allows the end users to request web pages and supply the application with text information
SQL	• Represents the connection between the *Internal Computation Resources* and the *Internal Storage Resources* • The entries of the calendar are stored in an SQL database. • Allows the computation resources to request a subset of the entries based on a provided set of criteria
TCP/IP (Cloud API)	• Represents the connection between the *Internal Computation Resources* and the *Cloud Environment* • Dependent on the interface provided by the *Cloud Environment* • There is a variety of interfaces available, but at their most basic level, they all rely on TCP/IP.

3.2.3 Defining the Resource Manager's Behavior

This solution focuses on computation as the limiting resource. When a search request is received, the *Resource Manager* checks to see whether the internal resources are capable of handling it; if they are, this component passes the request to the internal resources to be processed.[10] If the internal resources are at their maximum capacity, the request is sent to the cloud resources to be processed. The sequence diagrams for servicing a search request when the internal resources are or are not at their maximum usage threshold are shown in Figure 5 and Figure 6, respectively.

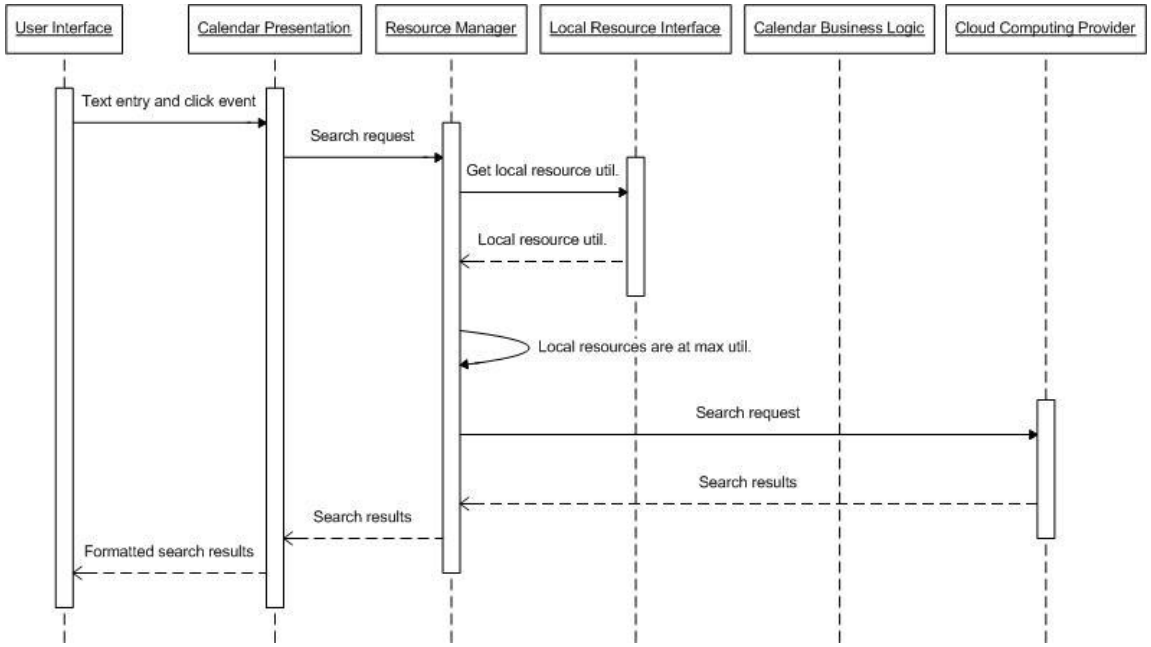

Figure 5: Sequence Diagram for a Search Request when Internal Resources Are at Their Maximum Usage Threshold

[10] In practice, the redirection to the cloud would most probably be handled by a load balancer or proxy. However, the *Resource Manager* component is the software equivalent that is appropriate for this simple application and T-Check investigation.

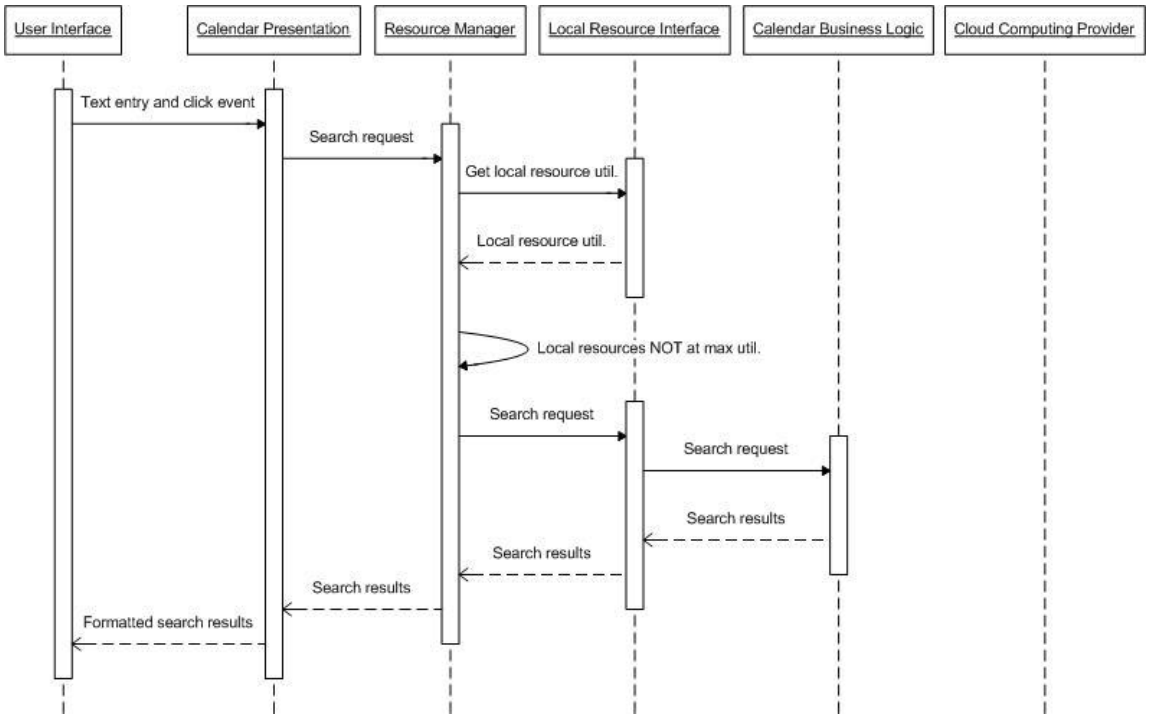

Figure 6: Sequence Diagram for a Search Request when Internal Resources Are Not at Their Maximum Usage Threshold

When a request to create a new entry or to update an existing entry is received, the *Resource Manager* sends the request to both the internal resources and the cloud resources to update both databases appropriately. This ensures consistency between the internal and cloud resources. The sequence diagram for servicing a *create* or an *update* request is provided in Figure *7*.

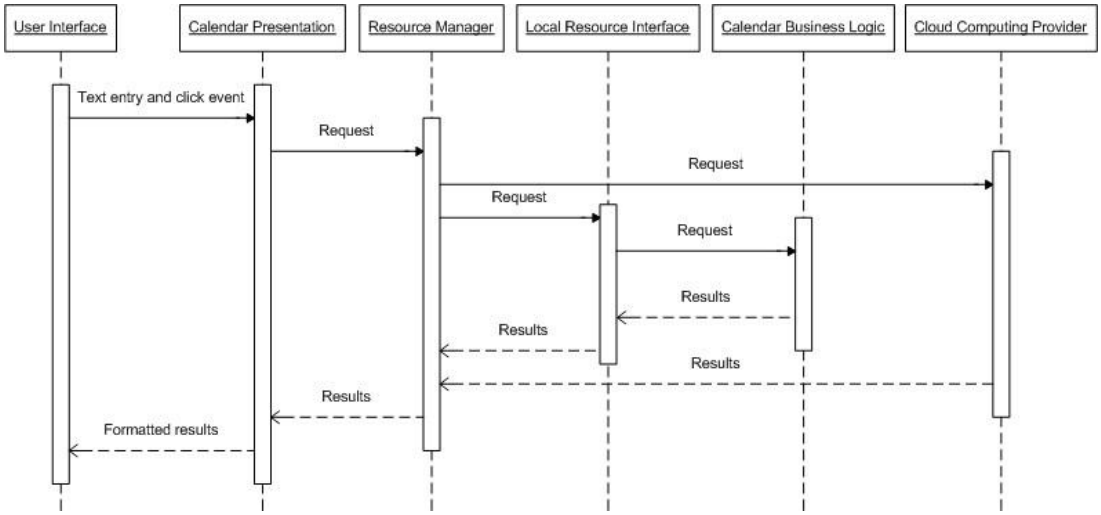

Figure 7: Sequence Diagram for Servicing a Create or Update Request

3.3 Defining the System Architecture for Testing Hypothesis 2

The solution for evaluating hypothesis 2 builds upon the system for testing hypothesis 1. It focuses on the organization's ability to monitor and continuously update the amount of cloud resources at its disposal. With this in mind, we added an administrative console to the system, which could acquire and release cloud resources as necessary.

3.3.1 Dynamic View of the Solution for Testing Hypothesis 2

In this solution, the system administrator is able to access and update the information about the cloud resources available to the system through an administrative console. Figure 8 shows the dynamic view of this solution in the form of a component and connector diagram. The element and relationship responsibilities associated with this diagram are provided in Table 11 and Table 12, respectively.

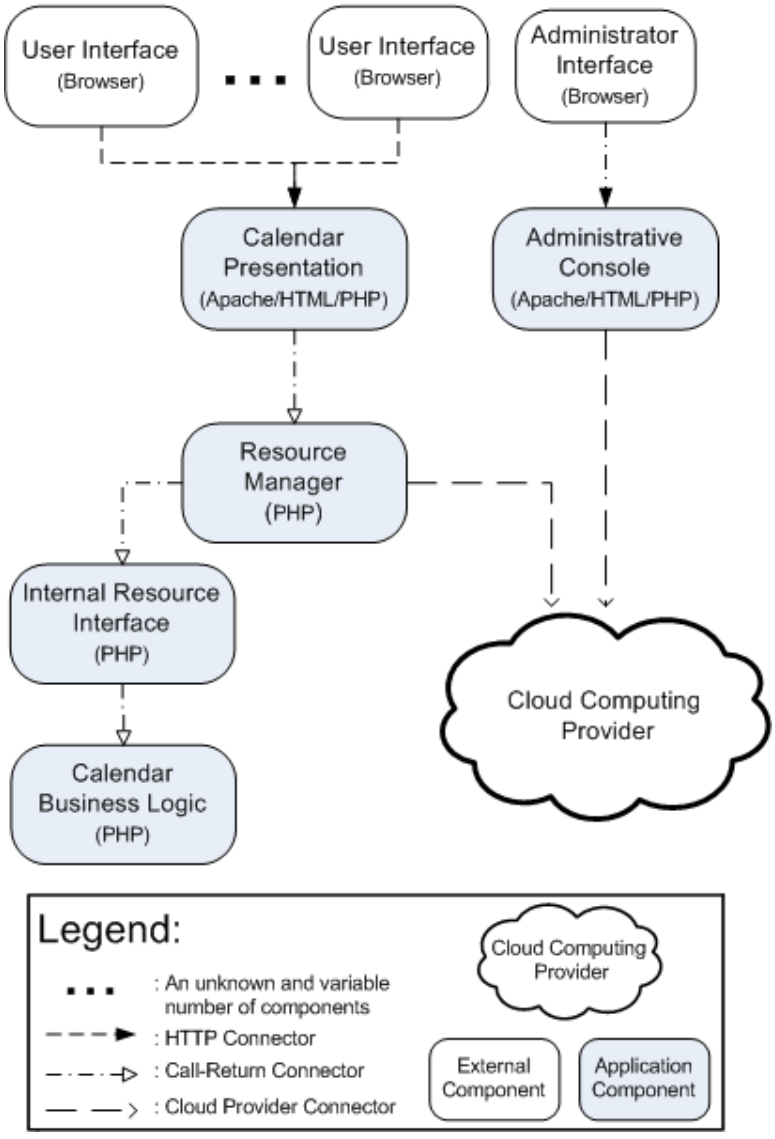

Figure 8: Component and Connector View of the Solution for Testing Hypothesis 2

Table 11: Element Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 2

NOTE: The rows marked in bold correspond to the changes with respect to the solution for testing hypothesis 1.

Associated Diagram: Figure 8; View: Dynamic	
Element	**Responsibilities**
User Interface	• Represents the web browser employed by the end user • Renders any and all information provided by the *Calendar Presentation* component • Allows the end user to enter information and passes it to the *Calendar Presentation* component
Administrator Interface	• **Represents the web browser employed by the system administrator** • **Allows the system administrator to access information about cloud resources** • **Allows the system administrator to update the cloud resources currently allocated to the system**
Calendar Presentation	• Combination of HTML and PHP that allows the end user to submit requests • Passes the requests to the *Resource Manager* component to be allocated to the appropriate resources for processing • Receives the resulting information from the *Resource Manager* component and formats it to be displayed to the end user
Administrative Console	• **Combination of HTML and PHP that utilizes the API provided by the cloud provider to obtain information about cloud resources and update cloud resources currently allocated to the system** • **Receives the resulting information from the *Cloud Computing Provider* component and formats it to be displayed to the system administrator**
Resource Manager	• Responsible for allocating end user requests to the appropriate resources for processing • Upon receiving a request to create or update an entry in the calendar, this component will forward the request to both the local infrastructure and the cloud infrastructure to modify both sets of data. • Upon receiving a request to search the entries in the calendar, this component will communicate with the *Internal Resource Interface* component to identify the current utilization of the internal infrastructure. – If the internal resources are sufficient to handle the end user's request, it will be submitted to the internal resources for processing. – If the internal resources are insufficient to handle the end user's request, it will be submitted to the cloud resources for processing.
Internal Resource Interface	• Responsible for monitoring the resource utilization of the internal infrastructure available to the application • When requested by the *Resource Manager* component, it will identify the current utilization of the internal resources. • If another request can be supported by the internal resources, it will receive the request from the *Resource Manager* component and assign it to the *Calendar Business Logic* component. • Upon receiving the resulting information from the *Calendar Business Logic* component, it will return this information to the *Resource Manager*.

Table 11: Element Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 2 (cont.)

Associated Diagram: Figure 8; View: Dynamic	
Element	**Responsibilities**
Calendar Business Logic	• Responsible for processing an end user's request • Upon receiving a search request, it will search the set of data entries to determine those that match the provided search criteria. • Upon receiving a create request, it will identify the next ID number to be assigned and insert the provided information into the calendar with the identified ID number. • Upon receiving an update request, it will identify the entry to be updated, and will make the appropriate modifications to the calendar. • Once the request has been processed, it will return the appropriate information about the results of the operation.
Cloud Computing Provider	• Cloud provider that is responsible for providing the *Resource Manager* component the ability to process user requests that the internal infrastructure is unable to support • Allows the *Resource Manager* to submit user requests to be processed • Contains an instance of the *Calendar Business Logic* component to be used to process these requests • Once a request has been processed, it returns the results to the *Resource Manager* component. • Also provides a resource management interface

Table 12: Relationship Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 2

Associated Diagram: Figure 8; View: Dynamic	
Relationship	**Responsibilities**
	• Represents an HTTP connection between two components of the system • Utilizes TCP/IP to transmit information between the components • The component to which the connector is pointing is the component receiving the HTTP request.
	• Represents a call-return connection between two components of the system • Allows a component to invoke an operation on another component and receive the results • The component to which the connector is pointing is the component being invoked.
	• Represents a connection to a cloud provider • The medium for this connection will depend on the interface available for the cloud provider. • The component to which the connector is pointing is the cloud provider on which an operation is being invoked.

3.3.2 Deployment View of the Solution for Testing Hypothesis 2

In this solution, we have added a new user to the system, the system administrator. The system administrator accesses the administrative console via HTTP requests. Figure 9 identifies how the system administrator will interact with the system and how the components specified in Figure 8 will be deployed onto the available hardware. The element and relationship responsibilities catalogs associated with this diagram are provided in Table 13 and Table 14, respectively.

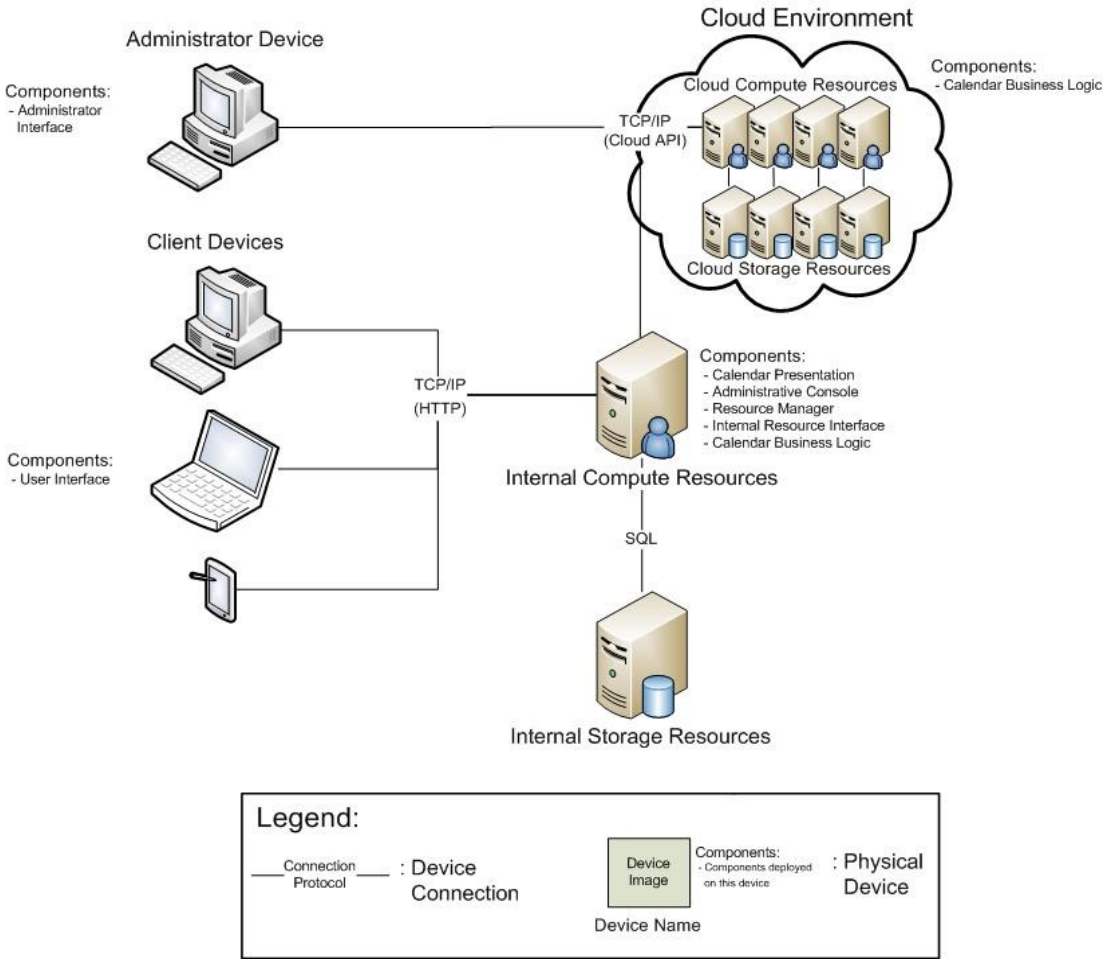

Figure 9: Deployment View for the Solution for Testing Hypothesis 2

Table 13: Element Responsibilities for the Deployment View of the Solution for Testing Hypothesis 2

NOTE: The rows marked in bold correspond to the changes with respect to the solution for testing hypothesis 1.

Associated Diagram: Figure 9; View: Deployment	
Element	**Responsibilities**
Administrator Device	• **Device with which the administrator interacts with the application** • **May be a desktop or laptop computer, PDA, cellular telephone, and so on** **Interacts with the *Cloud Environment* using the protocol specified by the cloud provider to access and update information about the resources currently allocated to the system**
Client Devices	• Devices with which the end users interact with the application • Include desktop and laptop computers, PDAs, cellular telephones, and so on • Interact with the internal infrastructure using TCP/IP to request the web pages that represent the application from the end user's perspective

Table 13: Element Responsibilities for the Deployment View of the Solution for Testing Hypothesis 2 (cont.)

Associated Diagram: Figure 9; View: Deployment	
Element	**Responsibilities**
Internal Compute Resources	• Computation infrastructure owned and operated by the organization • Computation resources are responsible for serving web pages, identifying the current utilization of the internal resources, and forwarding requests to the appropriate set(s) of resources. • All communication with the *Cloud Environment* is done via TCP/IP, but many cloud environments have vendor-specific protocols and interfaces built on top of TCP/IP. For this reason, the communication with the *Cloud Environment* will need to be tailored to meet the specific API of the provider being used.
Internal Storage Resources	• Storage infrastructure owned and operated by the organization • Storage resources hold the entries in the calendar.
Cloud Environment	• Set of resources obtained from a cloud computing provider, including both compute and storage resources • *Internal Compute Resources* forward end-user requests through the API provided by the cloud provider. • Requests are caught and processed by its compute resources, which search, create, and/or update all entries in the calendar stored on its storage resources. • In order to handle end user requests, it stores an equivalent of the *Calendar Business Logic* component. • Also provides a resource management interface

Table 14: Relationship Responsibilities for the Deployment View of the Solution for Testing Hypothesis 2

Associated Diagram: Figure 9; View: Deployment	
Relationship	**Responsibilities**
TCP/IP (HTTP)	• Represents the connection between the end users of the application and the *Internal Computation Resources* of the organization • HTTP connection that is based on TCP/IP • Allows the end users to request web pages and supply the application with text information
SQL	• Represents the connection between the *Internal Computation Resources* and the *Internal Storage Resources* • The entries of the calendar are stored in an SQL database. • Allows the computation resources to request a subset of the entries based on a provided set of criteria
TCP/IP (Cloud API)	• Represents the connection used to access the *Cloud Environment* • Dependent on the interface provided by the cloud provider • There is a variety of interfaces available, but at their most basic level, they all rely on TCP/IP.

3.4 Defining the System Architecture for Testing Hypothesis 3

The solution for evaluating hypothesis 3 also builds on the system for testing hypothesis 1. It focuses on the ability of the organization to move an application from one cloud computing provider to another. This solution modifies an application that has been successfully deployed in a cloud provider to run in a new cloud provider.

3.4.1 Dynamic View of the Solution for Testing Hypothesis 3

For this solution, we incorporated a new cloud computing provider into the system. The assumptions are that the consequences of adding a new cloud provider will be isolated to the interface used by the Resource Manager for forwarding requests and that significant change to the other components will not be required. Figure 10 shows the dynamic view of this solution in the form of a component and connector diagram. The element and relationship responsibilities associated with this diagram are provided in and Table 15 and Table 16, respectively.

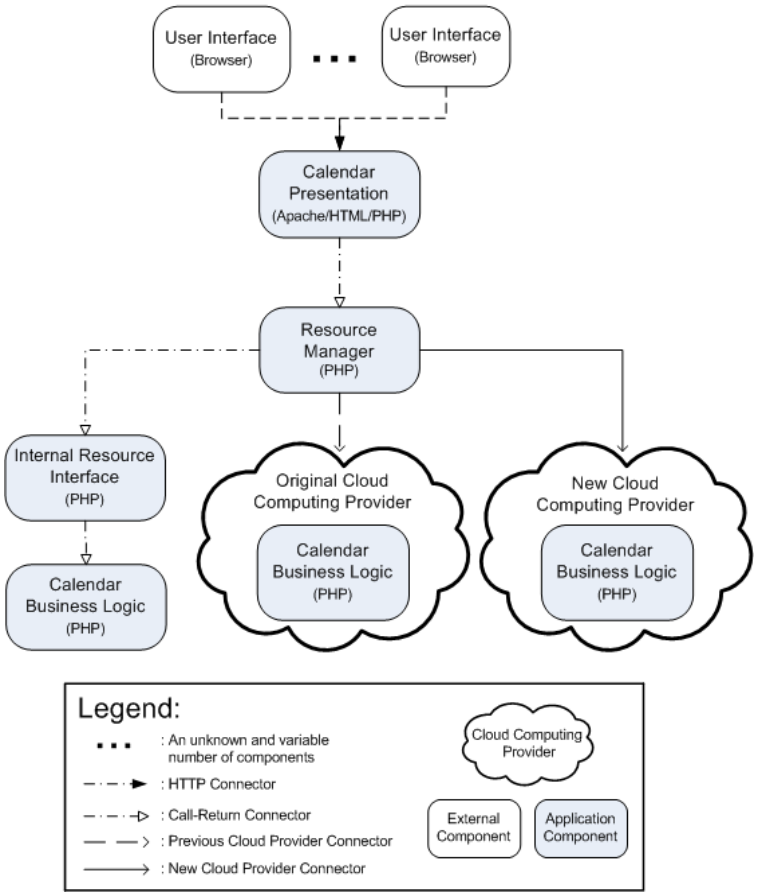

Figure 10: Component and Connector View of the Solution for Testing Hypothesis 3

Table 15: Element Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 3

NOTE: The rows marked in bold correspond to the changes with respect to the solution for testing hypothesis 1.

Associated Diagram: Figure 10; View: Dynamic	
Element	**Responsibilities**
User Interface	• Represents the Web browser employed by the end user • Renders any and all information provided by the *Calendar Presentation* component • Allows the end user to enter information and passes it to the *Calendar Presentation* component

Table 15: Element Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 3 (cont.)

Associated Diagram: Figure 10; View: Dynamic	
Element	**Responsibilities**
Calendar Presentation	• Combination of HTML and PHP that allows the end user to submit requests • Passes the requests to the *Resource Manager* component to be allocated to the appropriate resources for processing • Receives the resulting information from the *Resource Manager* component and formats it to be displayed to the end user
Resource Manager	• Responsible for allocating end user requests to the appropriate resources for processing • Upon receiving a request to create or update an entry in the calendar, this component will forward the request to both the local infrastructure and the cloud infrastructure to modify both sets of data. • Upon receiving a request to search the entries in the calendar, this component will communicate with the *Internal Resource Interface* component to identify the current utilization of the internal infrastructure. – If the internal resources are sufficient to handle the end user's request, it will be submitted to the internal resources for processing. – If the internal resources are insufficient to handle the end user's request, it will be submitted to the cloud resources for processing.
Internal Resource Interface	• Responsible for monitoring the resource utilization of the internal infrastructure available to the application • When requested by the *Resource Manager* component, it will identify the current utilization of the internal resources. • If another request can be supported by the internal resources, it will receive the request from the *Resource Manager* component and assign it to the *Calendar Business Logic* component. • Upon receiving the resulting information from the *Calendar Business Logic* component, it will return this information to the *Resource Manager*.
Calendar Business Logic	• Responsible for processing an end user's request • Upon receiving a search request, it will search the set of data entries to determine those that match the provided search criteria. • Upon receiving a create request, it will identify the next ID number to be assigned and insert the provided information into the calendar with the identified ID number. • Upon receiving an update request, it will identify the entry to be updated, and will make the appropriate modifications to the calendar. • Once the request has been processed, it will return the appropriate information about the results of the operation.
Original Cloud Computing Provider	• **Cloud provider that is currently responsible for providing the ability to process user requests that the internal infrastructure is unable to support to the *Resource Manager* component** • **Allows the *Resource Manager* to submit user requests to be processed** • **Contains an instance of the *Calendar Business Logic* component to be used to process these requests** • **Once a request has been processed, it returns the results to the *Resource Manager* component.**
New Cloud Computing Provider	• **Cloud provider that is being transitioned to, that will be responsible for providing the *Resource Manager* component the ability to process user requests that the internal infrastructure is unable to support** • **Will allow the *Resource Manager* to submit user requests to be processed** • **Will contain an instance of the *Calendar Business Logic* component to be used to process these requests** • **Once a request has been processed, it will return the results to the *Resource Manager* component.**

Table 16: Relationship Responsibilities for the Component and Connector View of the Solution for Testing Hypothesis 3

NOTE: The rows marked in bold correspond to the changes with respect to the solution for testing hypothesis 1.

Associated Diagram: Figure 10; View: Dynamic	
Relationship	**Responsibilities**
	• Represents an HTTP connection between two components of the system • Utilizes TCP/IP to transmit information between the components • The component to which the connector is pointing is the component receiving the HTTP request.
	• Represents a call-return connection between two components of the system • Allows a component to invoke an operation on another component and receive the results • The component to which the connector is pointing is the component being invoked.
	• **Represents a connection to the current cloud provider** • **The medium for this connection will depend on the interface available for the current cloud provider.** • **The component to which the connector is pointing is the cloud provider on which an operation is being invoked.**
	• **Represents a connection to the new cloud provider** • **The medium for this connection will depend on the interface available for the new cloud provider.** • **The component to which the connector is pointing is the cloud provider on which an operation will be being invoked.**

3.4.2 Deployment View of the Solution for Testing Hypothesis 3

In this solution, we are replacing the cloud provider available to the internal resources with a new cloud provider. The assumption is that this modification to the system will be completely transparent to the end users of the system. The only changes required should be in the interface used to communicate with the cloud resources. Figure 11 shows how the new cloud provider will be integrated into the system and how the components specified in Figure 10 will be deployed onto the available hardware. The element and relationship responsibilities associated with this diagram are provided in Table 17 and Table 18, respectively.

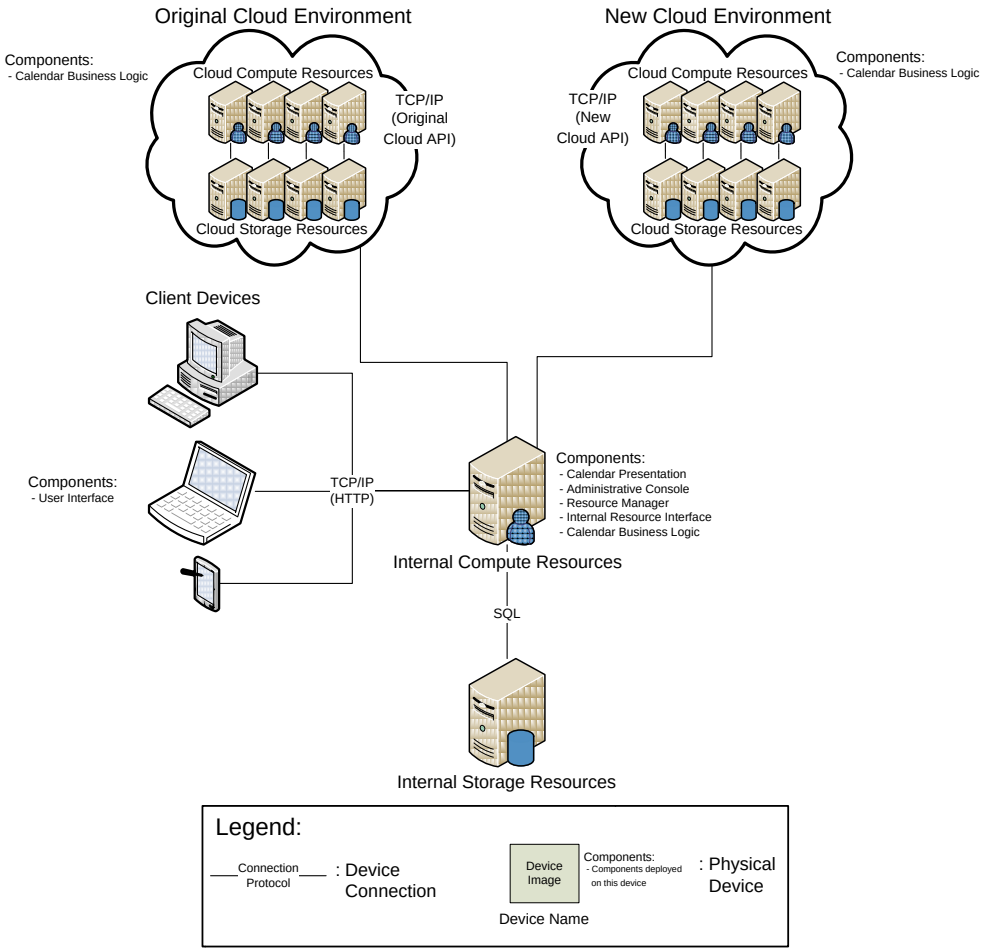

Figure 11: Deployment View for the Solution for Testing Hypothesis 3

Table 17: Element Responsibilities for the Deployment View of the Solution for Testing Hypothesis 3

NOTE: The rows marked in bold correspond to the changes with respect to the solution for testing hypothesis 1.

Associated Diagram: Figure 11; View: Deployment	
Element	**Responsibilities**
Client Devices	• Devices with which the end users interact with the application • Include desktop and laptop computers, PDAs, cellular telephones, etc. • Interact with the internal infrastructure using TCP/IP to request the web pages that represent the application from the end user's perspective
Internal Compute Resources	• Computation infrastructure owned and operated by the organization • Computation resources are responsible for serving web pages, identifying the current utilization of the internal resources, and forwarding requests to the appropriate set(s) of resources. • All communication with the cloud provider is done via TCP/IP, but many cloud providers have vendor-specific protocols and interfaces built on top of TCP/IP. For this reason, the communication with the *Cloud Environment* will need to be tailored to meet the specific API of the provider being used.

Table 17: Element Responsibilities for the Deployment View of the Solution for Testing Hypothesis 3 (cont.)

Associated Diagram: Figure 11; View: Deployment	
Element	**Responsibilities**
Internal Storage Resources	• Storage infrastructure owned and operated by the organization • Storage resources hold the entries in the calendar.
Original Cloud Environment	• **Set of resources obtained from the cloud computing provider, including both compute and storage resources** • ***Internal Compute Resources* forwards user requests through the API provided by this cloud provider.** • **Requests are caught and processed by this cloud provider's compute resources, which search, create, and/or update all entries in calendar stored on this cloud provider's storage resources.** • **In order to handle user requests, a copy of the *Calendar Business Logic* component resides in this cloud provider.**
New Cloud Environment	• **Set of resources obtained from the cloud computing provider to be transitioned to, including both compute and storage resources** • ***Internal Compute Resources* forwards user requests through the API provided by this cloud provider.** • **Requests will be caught and processed by this cloud provider's compute resources, which will search, create, and/or update all entries in calendar stored on this cloud provider's storage resources.** • **In order to handle user requests henceforth, a copy of the *Calendar Business Logic* component will reside in this cloud provider.**

Table 18: Relationship Responsibilities for the Deployment View of the Solution for Testing Hypothesis 3

NOTE: The rows marked in bold correspond to the changes with respect to the solution for testing hypothesis 1.

Associated Diagram: Figure 11; View: Deployment	
Relationship	**Responsibilities**
TCP/IP (HTTP)	• Represents the connection between the end users of the application and the *Internal Compute Resources* of the organization. Connection is based on TCP/IP. • Allows end users to request web pages and supply the application with text information
SQL	• Represents the connection between the *Internal Compute Resources* and the *Internal Storage Resources*. Entries of the calendar are stored in an SQL database. • Allows the computation resources to request a subset of the entries based on a provided set of criteria
TCP/IP (Previous Cloud API)	• **Represents the connection between the *Internal Compute Resources* and the *Previous Cloud Environment*** • **Dependent on the interface provided by the previous cloud provider. There is a variety of interfaces available, but at their most basic level, they all rely on TCP/IP.**
TCP/IP (New Cloud API)	• **Represents the connection between the *Internal Compute Resources* and the *New Cloud Environment*** • **Dependent on the interface provided by the new cloud provider. There is a variety of interfaces available, but at their most basic level, they all rely on TCP/IP.**

3.5 Selecting Cloud Computing Providers

In our initial survey of the cloud computing providers available for the purposes of this study, we identified 10 candidates, as shown in Table 19. We classify them based on type (PaaS, IaaS, or SaaS) and provide a brief description of each offer. The list in Table 19 is in alphabetical order and was created from internet searches for cloud providers, news briefs related to new cloud computing platforms, and web articles comparing cloud providers.

Table 19: Initial Survey of Cloud Computing Providers

Name	Type	URL	Brief Description
Akamai EdgePlatform	PaaS	http://www.akamai.com/html/technology/edgeplatform.html	• Provides a large distributed computing platform on which organizations can deploy web applications • Also provides various content delivery services and consulting services
Amazon Web Services (AWS)	IaaS	http://aws.amazon.com/	• Provides users with access to dynamically scalable compute and/or storage resources • Includes a large suite of independent web services
Force.com	PaaS /SaaS	http://www.salesforce.com/platform/	• Provides enterprise application development with a "no programming" environment • Focuses on ease of use, application customization, and integration with the existing Salesforce platform
GoGrid	IaaS	http://www.gogrid.com/	Provides access to compute and storage resources
Google App Engine (GAE)	PaaS	http://code.google.com/appengine/	• Provides users with a means of uploading web applications and running them on Google's infrastructure • Markets primarily to small-scale/individual users
IBM Computing on Demand (CoD)	IaaS	http://www-03.ibm.com/systems/deepcomputing/cod/	• Provides users with access to enterprise-scale computation facilities • Markets primarily to enterprise users
Microsoft Azure Services Platform	IaaS/ PaaS	http://www.microsoft.com/azure/	• Provides access to compute and storage resources on demand via web services • Offers a software development platform for developing web-based applications to be deployed on Microsoft infrastructure
Rackspace Cloud	IaaS	http://www.rackspacecloud.com/	Provides access to storage and compute resources individually

Table 19: Initial Survey of Cloud Computing Providers (cont.)

Name	Type	URL	Brief Description
Yahoo Open Stack (Y!OS)	PaaS	http://developer.yahoo.com/yos/intro/	• Provides developers with a means of developing web applications on top of the existing Yahoo! Platform • Allows for integration with many of the other Yahoo! Resources • Markets to the large set of Yahoo! users
Zoho	PaaS/ SaaS	http://www.zoho.com/	• Offers a large set of web-based applications that developers can integrate into their own applications • In doing so, developers are able to leverage Zoho's infrastructure to run their applications.

Of the cloud computing providers considered for the study, we selected **Amazon Web Services**, **Google App Engine**, and **Force.com** to test the hypotheses outlined in Section 2.2. The key criteria considered in selecting these providers were the cost of developing and testing the proposed solutions, the purpose or focus of the providers, and the interfaces or languages supported by the providers. Because this effort was simple experimentation, an additional constraint was to find providers that would allow us to initially use their services for free, had promotions that gave us an initial balance for acquiring resources for the study, or offered grants for using the services in an academic environment.

As specified by the T-Check method, we focused on building the simplest solution to test the specified hypotheses. For this reason we worked to identify cloud computing providers that support the language used by the Calendar System described in Section 2. We included one cloud computing provider with support for a different set of languages in an attempt to understand what effect this would have on the hypotheses being evaluated.

The results of evaluating each of the selected cloud computing providers are provided in Table 20.

Table 20: Results of Initial Cloud Computing Providers Evaluation

	Amazon Web Services (AWS)	Force.com	Google App Engine (GAE)
Provider Type	IaaS	PaaS/SaaS	PaaS
Experiment Cost	Acquired a student research grant that provided credit for the AWS services necessary for the project	Free developer environment	Free for limited use (up to 500 MB storage)
Provider Focus	• Provides IaaS to any size user • Offers a complete suite of both compute and storage resources	Primarily supports enterprise application development that is designed to integrate with the Salesforce CRM	Supports web application development

Table 20: Results of Initial Cloud Computing Providers Evaluation (cont.)

	Amazon Web Services (AWS)	Force.com	Google App Engine (GAE)
Interface / Language	• Provides complete control over the resources and therefore can be tailored to support any language • AWS has a set of SOAP and REST APIs with libraries for utilizing them in a variety of languages, including .Net, Java, PHP, and Ruby.	• Requires the application to be written in their custom Apex language • Provides a SOAP API with libraries available for Perl, PHP, Python, Java, .Net, and Ruby	Provides Java and Python web application support
Rationale for Selection	• Amazon is perceived as the leader in cloud computing [Singh 2009]. • It offers IaaS that may provide a very different perspective than the two other PaaS providers being evaluated.	• Markets primarily to enterprise organizations, which is consistent with the T-Check context for this study • Has a unique marketing strategy that focuses on integration with the Salesforce CRM • Currently offers access to a free environment for developers	Focuses on users developing web applications, which aligns well with the solutions designed
Concerns		Use of the Apex language may significantly affect the results because of learning curve.	Supports a limited set of programming languages

3.6 Implementing the T-Check Solutions

We evaluated each of the cloud computing providers selected against each hypothesis using the corresponding solutions described in Sections 3.2, 3.3, and 3.4. In this section, we will describe the steps taken to implement each of the solutions.

3.6.1 Implementing the Solution for Testing Hypothesis 1

Evaluating hypothesis 1 consisted of integrating each cloud computing provider with the existing internal application. This solution was designed to show not only that it is feasible to use internal resources and cloud resources simultaneously, but also that it could be done in a cost-effective manner.

Starting from the internal version of the calendar application, we modified the application to redirect the excess search requests to each of the cloud computing providers' resources. The term used to describe the situation in which resources from the cloud are used temporarily to offload a system or deal with spike demands is called *cloudbursting*.

We created three instances of the calendar application, one for each cloud provider. For each instance of the calendar application, we tailored the *Resource Manager* component to interact with the cloud resources using the interface specified by the cloud computing provider. For the AWS and GAE applications, we were able to redirect the end-user's browser to the cloud web server, which performed the desired operation and redirected the end user back to the internal web server. For the Force.com application, we used the provided PHP library to invoke operations through the SOAP interface defined.

In testing this hypothesis, we tracked both the modifications made to the code and the amount of time required to complete the modifications specified in the solution. Code modifications were tracked based on the number of lines of code changed, added, or removed, and time was tracked in five-minute increments. As defined in the criteria for evaluating this hypothesis, these metrics were used to determine whether the solution had verified or refuted the hypothesis.

3.6.2 Implementing the Solution for Testing Hypothesis 2

For hypothesis 2, we extended the solution for testing hypothesis 1 to evaluate the mechanisms offered by the cloud computing providers for identifying the current resource usage of the calendar and updating the resources allocated to the calendar. The implementation of this solution was highly dependent on the cloud computing provider. The IaaS cloud computing provider offered both an API and graphical user interface (GUI) for accessing and updating this information, while the PaaS providers handled all resource allocation concerns on behalf of the user. For the IaaS provider (AWE) we were able to effectively evaluate the hypothesis by implementing an administrative console that interacts with the cloud environment through the provided API. For the PaaS providers (Force.com and GAE), we were forced to consult the documentation to evaluate the hypothesis because no APIs were provided to access or modify the resources allocated.

The criteria for evaluating this hypothesis are Boolean properties that must be satisfied by the cloud provider in order for the hypothesis to be sustained. For this reason, no additional metrics will be tracked related to this hypothesis.

3.6.3 Implementing the Solution for Testing Hypothesis 3

To evaluate hypothesis 3, the solution for testing hypothesis 1 was modified to utilize resources from another cloud computing provider. At this point we had already implemented the application for each of the three cloud computing providers selected and decided that porting the application between these providers would not accurately represent the cost of porting the application to a completely unknown provider. For this reason, we identified a fourth provider, GoGrid, to which we ported the solution for testing hypothesis 1.

We only ported the application from one of the cloud computing providers because the results of porting from the other two providers to the same destination provider would likely be skewed on account of our increased understanding of the new cloud provider. Also, if we ported the other two providers to different destination providers, the results would not be comparable because the cost of moving a set of resources from one cloud computing provider to another is highly dependent on the difference and constraints between source and destination providers.

We selected AWS as the cloud provider to be used in this solution. The most influential factor in this decision was that both AWS and GoGrid are IaaS providers and provide similar services.[11] As with the first hypothesis, we tracked both the code modifications and the time spent in order to evaluate the level of ease associated with this solution.

---

[11] In theory, we would expect an organization to move resources between providers of the same type (IaaS or PaaS) because their services would be comparable. In practice however, because cloud computing is an emerging technology, the organization could have made a wrong decision and needed to change provider types. It was our experience that if an organization were to move an application from an IaaS provider to a PaaS provider, it would be far more restricted in the configuration of the platform onto which the application is deployed, which could potentially mean additional development cost for the organization.

4 Evaluation and Experiences with Cloud Computing

By definition, a T-Check is the simplest experiment possible to verify a set of claims with respect to the use of a technology in a specific context. A summary of the assumptions and conditions under which these results are valid includes

1. The initial system is only 752 lines of code (LOC). Although appropriate for the T-Check experiments, the results may certainly be different for a larger system.
2. The initial system was architected "from scratch" and had clear separation between business logic and data access code. This might not be the case for all systems to be migrated to the cloud.
3. Even though the selections of open source components for infrastructure and development and of PHP as the programming language represent a generic, common configuration, the results are dependent on this selected configuration.

4.1 Results for Hypothesis 1

Hypothesis 1: **An organization can use its existing infrastructure simultaneously with cloud resources, with relative ease.**

This hypothesis is partially sustained. For each cloud computing provider, we were able to utilize the existing internal infrastructure simultaneously with the cloud resources. Whether this was accomplished with relative ease is not entirely conclusive.

Based on the criteria defined for evaluating this hypothesis, it must take less time than the development of the data access components of the initial application (the *Resource Manager*, *Internal Resource Interface*, and *Calendar Business Logic* components) and require changes to only the data access code and less than 10% of the remainder of the application. The initial calendar application consisted of 752 LOC, of which 479 LOC reside in the data access components. This means that a maximum of 506 lines of code could be modified, according to the established criteria (479 + (0.1*(752-479)) = 506)

As shown in Table 21, the time to develop the data access components in the initial calendar application was 2.26 hours. Table 21 also shows that the AWS application was the only one that was completed in less time than the original development time. Table 22 shows that none of the applications required even half of the maximum 506 lines of code to be modified. However, the experience showed that while the number of modified LOC was acceptable, these limited modifications were very tedious, error prone, and time consuming.

Table 21: Time to Develop the Solution for Testing Hypothesis 1

Task	Time (Hours)	Time Relative to Development of the Data Access Components of the Initial Calendar Application (%)
Developing the data access components of the initial calendar application	2.26	
Implementing the solution to use GAE	10.83	477.94%
Implementing the solution to use Force.com	8.17	360.29%
Implementing the solution to use AWS	1.50	66.18%

Table 22: Code Modifications Required for the Solution for Testing Hypothesis 1

Application	Lines Changed (LOC)	Lines Added (LOC)	Total Modifications (LOC)
GAE version of the Calendar	47	164	211
Force.com version of the Calendar	105	118	223
AWS version of the Calendar	47	49	96

To avoid the effects of the learning curve, the numbers provided in Table 21 exclude the time spent researching each cloud computing provider. In developing this solution, we have differentiated between researching about the cloud computing provider and implementing the specified modifications. The boundary between these two activities is not always completely clear. To ensure consistency across the providers, our research about the cloud provider consists of all the activities performed to implement a simple "Hello World" application using the cloud resources that are remotely accessible. All activities performed after this point are considered to be part of implementing the specified modifications. This level of research provides an understanding of the constructs used and the interfaces available for interacting with the cloud provider, but does not get into the application-specific concerns or issues.

4.1.1 Effects of Domain Experience

An interesting observation is that we first implemented the modifications for GAE, then those for Force.com provider, and finally for AWS. As shown in Table 21, the time required to implement the modifications appears to decrease as the experience in the domain increases. After analyzing the specific issues that arose in implementing each solution, we identified a few that were similar in regard to the interfaces for interacting with the providers, but these instances were relatively infrequent. The most time-consuming issues were unique to the provider and the requirements imposed for developing each application, which means that the decreasing numbers are simply a coincidence in our experiments.

4.1.2 IaaS vs. PaaS

Implementing this solution on both IaaS and PaaS providers offered an interesting look into the key differences between these two types of cloud computing providers.

PaaS providers place more constraints on applications deployed in the cloud because these have to fit into the environment (similar to deploying an application on an internal server). In the solution that used the IaaS provider, there was much more flexibility. For example, in the IaaS solution we were able to install and configure the machines to our liking and therefore had full control over the database and data access. In contrast, for the PaaS solution we had to modify the implementation of the *Calendar Business Logic* component and its means for storing and retrieving data to fit the conventions of the provider's platform.

From the numbers presented in Table 21 and Table 22, it would appear that IaaS providers have a distinct advantage over PaaS providers, requiring less time and code modification to incorporate their resources into an application. What these tables do not account for is the services provided out of the box by PaaS providers that must be developed by the user of IaaS providers. The environment updates, continuous scaling, and security that are standard services of many PaaS providers, for example, can be very costly for the user to develop. This simple experiment did not use any of these PaaS ser-

vices. What this means is that an additional criterion for selecting a cloud computing provider is to consider out-of-the-box services offered by the provider that could be included as part of the solution and therefore reduce implementation costs. The tradeoff, however, is the reduced control of the environment when using these out-of-the box services.

4.2 Results for Hypothesis 2

Hypothesis 2: **Cloud computing environments provide ways to continuously update the amount of resources allocated to an organization.**

This hypothesis is sustained. Although no two cloud computing providers offered the same mechanisms for scaling the resources allocated to the application, each provider offered some mechanism for the user's resources to be scaled.

With scalability as a key tenet of cloud computing, the fact that each provider offered this functionality came as no surprise. However, the mechanisms to do this with each cloud computing provider were completely different. The ability to scale an application to meet the constantly changing demand of its users is a crucial advantage of using cloud resources. Because each provider uses different mechanisms for managing the scalability of its resources, the user faces a significant risk of vendor lock-in, or at least a great deal of unnecessary hardship when it wants to move resources between providers.

4.2.1 Scaling Resources with Google App Engine

As a PaaS provider, GAE completely controls the scalability of resources for the applications using its environment. According to the documentation, a free account is allocated up to 500 MB of storage and enough bandwidth to serve approximately 5 million pages per month. After this, users must enable billing for their accounts and will be charged for any storage or bandwidth used beyond these free quotas. For billable accounts, GAE claims to scale sufficiently to handle up to approximately 500 requests per second. If they need more resources, users can request them by completing a form.

In this way, GAE fulfills the criteria for this hypothesis by monitoring the user's utilization and scaling its resources as needed. For accounts with billing enabled, two types of quotas are used to manage the resources available to an application:

1. **Billable Quota**: Set by the user, this quota ensures that the resources used by its application do not exceed its budget. The user sets its daily maximum for the amount of resources, and the environment controls the application's usage to not exceed this maximum.
2. **Fixed Quota**: Set by GAE, this quota that specifies the maximum amount of resources an application can use without inhibiting the performance of other applications in the environment. In order to ensure that all applications in the environment attain the desired performance, GAE restricts the amount of resources available to an application in the environment to reserve sufficient resources for the remaining applications.

Through the billable quota, the user is able only to restrict the amount of resources available to its application; therefore, the user has only partial control over scalability. On the other hand, GAE provides a wealth of information about an application's utilization for the current day. A few of the metrics provided include

- CPU (central processing unit) hours used
- outgoing and incoming bandwidth used

- amount of stored data
- requests/Second (average and peak)
- CPU Seconds Used/Second

The Python SDK[12] provided by GAE includes an API for performance profiling that allows the user to see the current amount of CPU resources used to process the current request. This operation is the only one provided in this API, and none of the APIs allow the user to increase or decrease its application's billable quota programmatically.

4.2.2 Scaling Resources with Force.com

Similarly to GAE, Force.com is a PaaS provider and does not allow the user to manage the resources allocated to its applications. Instead, Force.com handles the scalability concerns of all applications using its environment. While this service can save the user a significant amount of time and effort, it comes at a price. In order to develop an application on the Force.com platform, the application must be written in Apex, Force.com's custom programming language. Enforcing this requirement allows all applications in the environment to operate on a common platform and facilitates the multitenant architecture that Force.com uses to provide on-demand scalability [Wainewright 2008]. By monitoring the user's utilization and scaling of resources on its behalf, Force.com fulfills the corresponding criteria for this hypothesis.

Because the target market of Force.com is enterprise applications, its pricing model is not based on the amount of physical resources consumed by an application. Instead, higher priced accounts allow the user to develop more applications, allow more end users to access those applications, provide better integration with other Salesforce.com data and applications, and provide mobile access to the application.

4.2.3 Scaling Resources with Amazon Web Services

With the large investment Amazon has made in cloud computing, arguably being the industry leader, it is reasonable to expect that AWS would set the standard for how users manage the scalability of their resources. The majority of our experience is with the EC2 service, and thus we will focus primarily on the scalability mechanisms offered for this service. The user has two primary means for scaling EC2 resources: manually or programmatically. Through the AWS Management Console, a browser-based GUI for managing resources, users can

- launch or terminate a variety of Amazon Machine Instances (AMIs)
- obtain information about a running AMI
- manage the AMIs they have created
- assign IP addresses or security groups to an AMI
- provide an AMI with access to Amazon's Elastic Block Store volumes
- carry out a variety of other necessary resource management tasks

Similarly, the EC2 API allows users to scale their resources programmatically and offers the same set of operations using either the SOAP or REST protocol.

---

[12] SDK is software development kit or software developer's kit.

The primary advantages of the Management Console are the increased usability of the point-and-click, menu-based interface, and the small learning curve required to get a minimal set of resources up and running. Programmatically, the APIs allow software developers to dynamically integrate scalability into the runtime behavior of the application. This allows the user to scale the application on-demand and ensure that the application never runs out of resources.

Looking back at the criteria for evaluating this hypothesis, we see that through either the manual or programmatic interface, the user can both identify and update the current set of resources allocated to its application. In this way, AWS has fulfilled this hypothesis.

We did not specifically evaluate the third criteria for this hypothesis because we considered all the above information to be sufficient to sustain the hypothesis.

4.3 Results for Hypothesis 3

Hypothesis 3: **It is possible to move an application's resources between cloud computing providers, with varying levels of effort required.**

This hypothesis is sustained. For this hypothesis, we focused on moving the existing AWS application to GoGrid, another IaaS provider. As stated earlier, the fact that these are both IaaS providers significantly influenced our decision to focus on this transition. To move the application from Force.com to GoGrid, or even another PaaS provider, would likely have required us to remove significant parts of the system developed for Force.com and essentially start over. It is highly unlikely that the Apex modules developed for the Force.com platform would be helpful in porting the application to another cloud computing provider. Similarly, the structure of GAE applications is highly prescriptive and the benefit to be gained from the existing system would be minimal, at best. In this way, we expected that the results of using either of these providers would be resemble the results obtained in testing Hypothesis 1, in which we initially moved the application to the different cloud providers.

By focusing on an application deployed on an IaaS provider and moving it to another IaaS provider, we hoped to see the maximum amount of reuse that can be expected from the existing application. The intent is to gain a better understanding of the similarity of the standards employed by IaaS providers. As shown in Table 23 and Table 24, the commonality across cloud computing providers was positive.

In reflecting on the criteria defined to evaluate this hypothesis, we were unable to meet the criteria of it taking less than half the amount of time required to develop the data access components of the initial calendar application. However, when compared to the results obtained for hypothesis 1, the time required was less. The average time required for hypothesis 1 was nearly 3.8 times longer than expected, while the time required for this hypothesis was only 1.5 times longer than expected.

Table 23: Time Requirements to Develop the Solution for Testing Hypothesis 3

Task	Time (Hours)	Time Relative to Development of the Data Access Components of the Initial Calendar Application (%)
Developing the data access components of the initial Calendar application	2.26	
Modifying the AWS solution to use GoGrid	1.67	73.53%

Table 24: Code Modifications Required for Modifying the AWS Calendar to Use GoGrid Resources

Application	Lines Changed	Lines Added	Total Modifications
GoGrid Version of the Calendar	2	5	7

Based on these figures, the hypothesis has been sustained. But this is not to say that issues did not arise in implementing this solution. The majority of the issues encountered were related to the configuration, activation, and authorization of machines on the GoGrid infrastructure and not the application itself, allowing for a high degree of code reuse across providers.

One issue that we had was that the types of machines available in GoGrid were extremely different from those supported by AWS. In AWS, we had been using an Ubuntu[13] AMI, but GoGrid did not offer the Ubuntu operating system (OS) on any of its machines. Instead, we were forced to use CentOS[14] machines, which do not support the same installation steps that we had used with Ubuntu. Other issues that we had were due to differences in the way users interact with AWS and GoGrid. GoGrid employs a different set of steps to allow the user to initiate and connect to its machines. This sequence of actions did not align with the experience of working with AWS. In turn, this difference initially led to confusion in working with GoGrid that required consulting its documentation. While these issues were not catastrophic, they did highlight the variety of perceptions of what cloud computing is and the role users play in the environment.

[13] Ubuntu is an operating system based on the Debian GNU/Linux distribution (http://www.ubuntu.com).

[14] CentOS (Community ENTerprise Operating System) is an enterprise-class operating system based on Red Hat Enterprise Linux.

5 Conclusions and Open Questions

Cloud computing is in essence an economic model—a different way to acquire and manage IT resources. Many organizations are using cloud computing resources for activities such as

- making data available to large communities
- performing computation-intensive activities
- engaging in one-time usage
- supporting short-term projects
- making small applications available to mobile users

While this T-Check investigation did not look at the economic aspects of cloud computing, the experiments showed that it can be a practical option, from a technical perspective, for organizations with tight budget constraints or continuously changing resource needs. We found that the initial move to the cloud is time consuming, but that many of the difficulties encountered were the result of not adequately understanding the cloud provider into which the application was being deployed. With this in mind, an organization maintaining a large number of applications may find that moving applications to the cloud is significantly easier as it gains experience with the particular cloud provider. This small study also highlights the importance of

1. upfront consideration of the level of control the organization requires over the platform onto which its application will be deployed
2. the ability to move an application from one cloud provider to another

As with other T-Check investigations that we have done, we encountered a number of open questions that we would like to answer in future studies, such as

- Are the results obtained in this study consistent with other cloud computing providers?
- At what point does it become cost effective for an organization to manage its own datacenter (private cloud) as opposed to using external/public cloud resources?
- How much variability in load/traffic is necessary to make the cloud computing pay-per-use cost model cost effective?
- What level of functionality is included in various cloud computing providers in addition to access to resources (e.g., user authorization or resource locking)?
- How much effort is required by users to manage their cloud resources?
- Do cloud computing providers offer any mechanisms for users to work intermittently without internet connection?
- How much, if any, performance cost is paid for using cloud resources as opposed to local resources, remote resources, or an organization-owned infrastructure?
- Does the loss of control over the environment associated with cloud computing significantly complicate software development and maintenance?
- How easy is it to determine the current usage of a given application in order to see if more resources are necessary or beneficial?
- What, if any, standards are being used across cloud computing providers?

- How much does the programming language used to develop an application constrain the choice of cloud computing provider?
- What level of security is offered by various cloud computing providers?
- To what degree are the environmental concerns of developing and maintaining large datacenters being addressed by cloud computing providers?

The team in the System of Systems Practice (SoSP) Initiative at the Carnegie Mellon® Software Engineering Institute (SEI) that is investigating cloud computing and other technologies using the T-Check method is interested in feedback from and collaboration with the communities that are considering these technologies for SoS implementation. Write to the SoSP team at sosp-sei@sei.cmu.edu.

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

References

URLs are valid as of the publication date of this document.

[3tera 2010]
3tera, Inc. *Cloud Computing without Compromise*. http://www.3tera.com (2010).

[Akamai 2010]
Akamai Technologies. *Akamai Edge Platform: Application Acceleration that Delivers Content and Applications Quickly*. http://www.akamai.com/html/technology/edgeplatform.html (2010).

[Amazon 2009]
Amazon Web Services. *Amazon Elastic Compute Cloud Developer Guide* (API Version 2009-04-04). http://docs.amazonwebservices.com/AWSEC2/2009-04-04/DeveloperGuide/ (2009).

[Amazon 2010a]
Amazon Web Services. *Amazon Elastic Compute Cloud (Amazon EC2)*. http://aws.amazon.com/ec2/ (2010).

[Amazon 2010b]
Amazon Web Services. *Amazon Simple Storage Service (Amazon S3)*. http://aws.amazon.com/s3/ (2010).

[Chong 2006]
Chong, F. & Carraro, G. *Building Distributed Applications Architecture Strategies for Catching the Long Tail*. MSDN Architecture Center. http://msdn2.microsoft.com/en-us/library/aa479069.aspx (2006).

[Erdogmus 2009]
Erdogmus, H. *Cloud Computing: Does Nirvana Hide Behind the Nebula? IEEE Software 26*, 2 (March/April 2009): 4-6.

[Eucalyptus 2010]
Eucalyptus Systems Inc. *Eucalyptus Systems Inc.* http://www.eucalyptus.com/ (2010).

[Foster 2008]
Foster , I., Yong Zhau, Raicu, I., & Lu, S. "Cloud Computing and Grid Computing 360-Degree Compared," 1-10. *Proceedings of the Grid Computing Environments Workshop (GCE '08)*. Austin, TX (USA), November 12–16, 2008. DOI 10.1109/GCE.2008.4738445.

[Gartner 2009]
Gartner. *Hype Cycle for Emerging Technologies.* http://www.gartner.com/it/page.jsp?id=1124212 (2009).

[GoGrid 2010]
GoGrid. *CloudHosting, Hybrid Hosting, Cloud Servers from GoGrid*. http://www.gogrid.com/ (2010).

[Google 2010a]
Google. *Google App Engine*. http://code.google.com/appengine/ (2010).

[Google 2010b]
Google. *Google Apps for Business | Official Website*. http://www.google.com/apps/intl/en/business/index.html (2010).

[IBM 2010]
IBM. *IBM Computing on Demand*. http://www-03.ibm.com/systems/deepcomputing/cod/index.html (2010).

[Lewis 2005]
Lewis, Grace A. & Wrage, Lutz. *A Process for Context-Based Technology Evaluation* (CMU/SEI-2005-TN-025, ADA441251). Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/library/abstracts/reports/05tn025.cfm

[McEvoy 2008]
McEvoy, G. & Schulze, B. "Using Clouds to address Grid Limitations." *Proceedings of the 6th International Workshop on Middleware for Grid Computing (MGC 2008)*. Leuven, Belgium, December 1–5, 2008. ACM, 2008. http://portal.acm.org/citation.cfm?id=1462704.1462715

[Microsoft 2010a]
Microsoft Corporation. *Live Mesh Beta*. http://www.mesh.com/ (2010).

[Microsoft 2010b]
Microsoft Corporation. *Windows Azure Platform.* http://www.microsoft.com/azure/ (2010).

[PHP 2010]
The PHP Group. *PHP: Hypertext Preprocessor*. http://php.net/index.php (2010).

[Rackspace 2010]
Rackspace. *Cloud Computing, Cloud Hosting & Online Storage by Rackspace Hosting*. http://www.rackspacecloud.com/ (2010).

[Salesforce 2010a]
Salesforce.com. *Cloud Computing – salesforce.com.* http://www.salesforce.com/platform/ (2010).

[Salesforce 2010b]
Salesforce.com. *salesforce.com*. http://www.salesforce.com/crm/products.jsp (2010).

[Singh 2009]
Singh, B. *Top 10 Cloud Computing Service Providers of 2009*. http://www.techno-pulse.com/2009/12/top-cloud-computing-service-providers.html (2009).

[Ubuntu 2010]
Ubuntu.com. *Cloud | Ubuntu*. http://www.ubuntu.com/cloud (2010).

[Wainewright 2008]
Wainewright, P. *Many Degrees of Multi-Tenancy*. ZDNet. http://www.zdnet.com/blog/saas/many-degrees-of-multi-tenancy/533 (2010).

[Yahoo 2010]
Yahoo. *Introducing the Yahoo! Open Strategy*. http://developer.yahoo.com/yos/intro/ (2010).

[Zoho 2010]
Zoho Corporation. *Email, Hosting, CRM, Project Management, Office Suite, Document Management, Remote Support*. http://www.zoho.com/ (2010).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2010	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
T-Check in System-of-Systems Technologies: Cloud Computing	FA8721-05-C-0003

6. AUTHOR(S)
Harrison D. Strowd, Grace A. Lewis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2010-TN-009

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note presents the results of applying the T-Check method in an initial investigation of cloud computing. In this report, three hypotheses are examined: (1) an organization can use its existing infrastructure simultaneously with cloud resources with relative ease; (2) cloud computing environments provide ways to continuously update the amount of resources allocated to an organization; and (3) it is possible to move an application's resources between cloud computing providers, with varying levels of effort required. From the T-Check investigation, the first hypothesis is partially sustained and the last two hypotheses are fully sustained within the context specified for the investigation.

From an engineering perspective, cloud computing is a distributed computing paradigm that focuses on providing a wide range of users with distributed access to virtualized hardware and/or software infrastructure over the internet. From a business perspective, it is the availability of computing resources that are scalable and billed on a usage basis. While scalability is the primary tenet of cloud computing, a host of other advantages are advertised as being inherently obtained through cloud computing.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Cloud computing, T-check, IaaS, PaaS, SaaS	56

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL