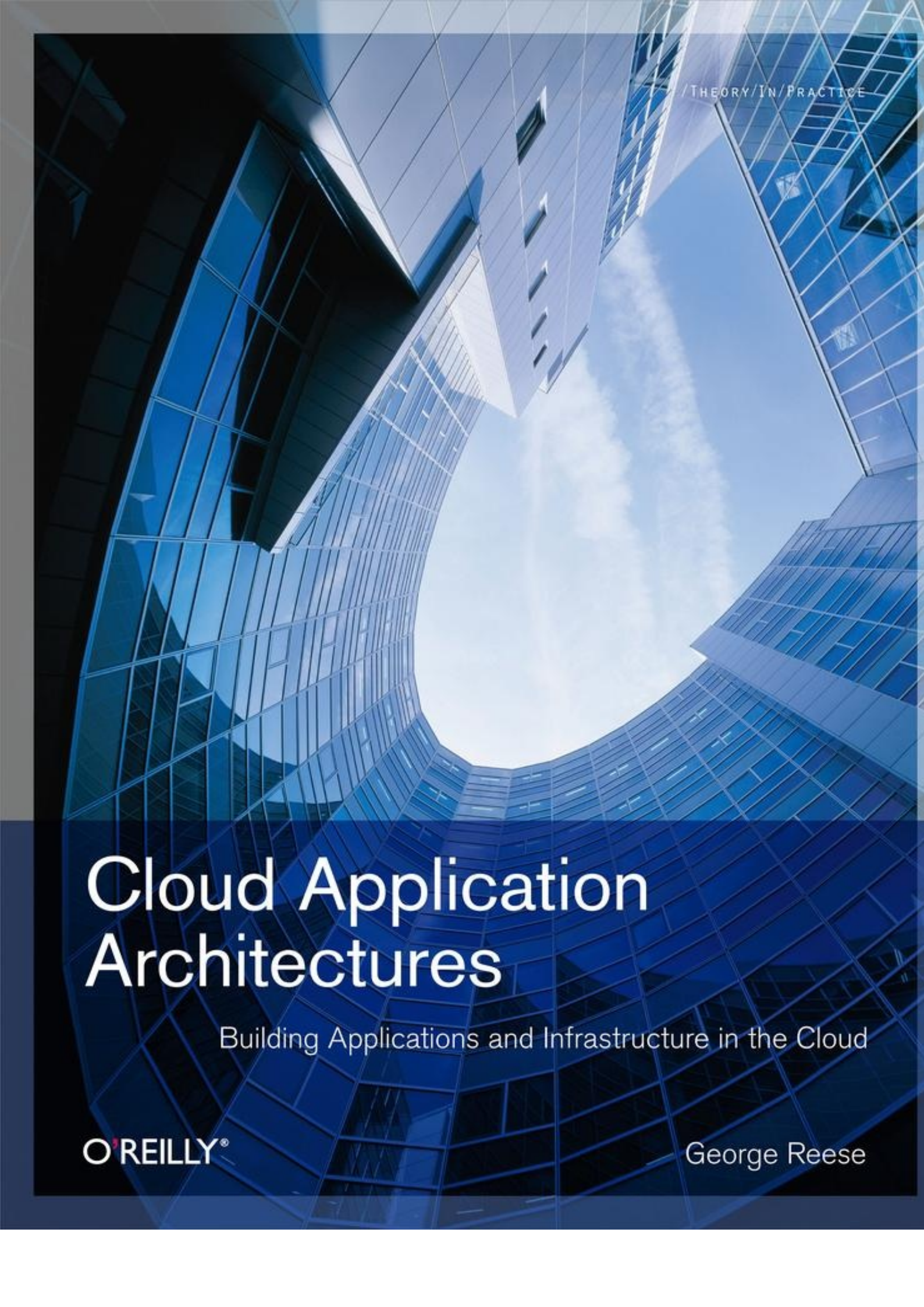


# Cloud Application Architectures

## Building Applications and Infrastructure in the Cloud

O'REILLY®

George Reese

# Cloud Application Architectures

*George Reese*

**Editor**
Andy Oram




O'Reilly Media

# Preface

In 2003, I jumped off the entrepreneurial cliff and started the company Valtira. In a gross oversimplification, Valtira serves the marketing function for companies in much the same way that SalesForce.com serves the sales function. It does online campaign management, customer relationshi management (CRM) integration with marketing programs, personalized web content, and a lot of other marketing things. Valtira's business model differed in one key way from the SalesForce.com business model: the platform required you to build your website on top of the content management system (CMS) at its core.

This CMS requirement made Valtira much more powerful than its competition as a Software as a Service (SaaS) marketing tool. Unfortunately, it also created a huge barrier to entry for Valtira solutions. While many companies end up doing expensive CRM integration services engagements with SalesForce.com, you can get started on their platform without committing to a big integration project. Valtira, on the other hand, demanded a big web development project of each customer.

In 2007, we decided to alter the equation and began making components of the Valtira platform available *on-demand*. In other words, we changed our software so marketers could register via the Valtira website and immediately begin building landing pages or developing personalized widgets to stick on their websites.

Our on-demand application had a different risk profile than the other deployments we managed. Whe a customer built their website on top of the Valtira Online Marketing Platform, they selected the infrastructure to meet their availability needs and paid for that infrastructure. If they had high-availability needs, they paid for a high-availability managed services environment at ipHouse or Rackspace and deployed our software into that infrastructure. If they did not have high-availability needs, we provided them with a shared server infrastructure that they could leverage.

The on-demand profile is different—everyone always expects an on-demand service to be available, regardless of what they are paying for it. I priced out the purchase of a starter high-availability environment for deploying the Valtira platform that consisted of the following components:

- A high-end load balancer
- Two high-RAM application servers
- Two fast-disk database servers
- Assorted firewalls and switches
- An additional half-rack with our ISP

Did I mention that Valtira is entirely self-funded? Bank loans, management contributions, and starter capital from family is all the money we have ever raised. Everything else has come from operational revenues. We have used extra cash to grow the business and avoided any extravagances. We have always managed our cash flow very carefully and were not excited about the prospect of this size of capital expense.

I began looking at alternatives to building out my own infrastructure and priced out a managed services infrastructure with several providers. Although the up-front costs were modest enough to stomach, the ongoing costs were way too high until we reached a certain level of sales. That's when I started playing with Amazon Web Services (AWS).

AWS promised us the ability to get into a relatively high-availability environment that roughly

mirrored our desired configuration with no up-front cash and a monthly expense of under $1,000. I was initially very skeptical about the whole thing. It basically seemed too good to be true. But I started researching....

That's the first thing you should know about the cloud: "But I started researching." If you wanted to see whether your application will work properly behind a high-end load balancer across two application servers, would you ever go buy them just to see if it would work out OK? I am guessing the answer to that question is no. In other words, even if this story ended with me determining that th cloud was not right for Valtira's business needs, the value of the cloud is already immediately apparent in the phrase, "But I started researching."

And I encountered problems. First, I discovered how the Amazon cloud manages IP addresses. Amazon assigns all addresses dynamically, you do not receive any netblocks, and—at that time—there was no option for static IP address assignment. We spent a small amount of time on this challenge and figured we could craft an automated solution to this issue. My team moved on to the next problem.

Our next challenge was Amazon's lack of persistent storage. As with the issue of no static IP addresses, this concern no longer exists. But before Amazon introduced its Elastic Block Storage services, you lost all your data if your EC2 instance went down. If Valtira were a big company with a lot of cash, we would have considered this a deal-breaker and looked elsewhere.

We almost did stop there. After all, the Valtira platform is a database-driven application that cannot afford any data loss. We created a solution that essentially kept our MySQL slave synced with Amazon S3 (which was good enough for this particular use of the Valtira platform) and realized this solution had the virtue of providing automated disaster recovery.

This experimentation continued. We would run into items we felt were potential deal-breakers only t find that we could either develop a workaround or that they actually encouraged us to do things a better way. Eventually, we found that we could make it all work in the Amazon cloud. We also ended up spinning off the tools we built during this process into a separate company, enStratus.

Today, I spend most of my time moving other companies into the cloud on top of the enStratus software. My customers tend to be more concerned with many of the security and privacy aspects of the cloud than your average early-adopter. The purpose of this book is to help you make the transitio and prepare your web applications to succeed in the cloud.

## Audience for This Book

I have written this book for technologists at all career levels. Whether you are a developer who needs to write code for the cloud, or an architect who needs to design a system for the cloud, or an IT manager responsible for the move into the cloud, you should find this book useful as you prepare you journey.

This book does not have a ton of code, but here and there I have provided examples of the way I do things. I program mostly in Java and Python against MySQL and the occasional SQL Server or Oracl database. Instead of providing a bunch of Java code, I wanted to provide best practices that fit any programming language.

If you design, build, or maintain web applications that might be deployed into the cloud, this book is for you.

## Organization of the Material

The first chapter of this book is for a universal audience. It describes what I mean by “the cloud” and why it has value to an organization. I wrote it at such a level that your CFO should be able to read the chapter and understand why the cloud is so useful.

In the second chapter, I take a bit of a diversion and provide a tutorial for the Amazon cloud. The purpose of this book is to provide best practices that are independent of whatever cloud you are using. My experience, however, is mostly with the Amazon cloud, and the Amazon Web Services offerings make up the bulk of the market today. As a result, I thought it was critical to give the reader a way to quickly get started with the Amazon cloud as well as a common ground for discussing terms later in the book.

If you are interested in other clouds, I had help from some friends at Rackspace and GoGrid. Eric “E. J.” Johnson from Rackspace has reviewed the book for issues that might be incompatible with their offering, and Randy Bias from GoGrid has done the same for their cloud infrastructure. Both have provided appendixes that address the specifics of their company offerings.

Chapter 3 prepares you for the cloud. It covers what you need to do and how to analyze the case for th move into the cloud.

Chapters 4 through 7 dive into the details of building web applications for the cloud. Chapter 4 begin the move into the cloud with a look at transactional web application architectures and how they need to change in the cloud. Chapter 5 confronts the security concerns of cloud computing. Chapter 6 show how the cloud helps you better prepare for disaster recovery and how you can leverage the cloud to drive faster recoveries. Finally, in Chapter 7, we address how the cloud changes perspectives on application scaling—including automated scaling of web applications.

## Conventions Used in This Book

The following typographical conventions are used in this book:

*Italic*

Indicates new terms, URLs, filenames, Unix utilities, and command-line options.

`Constant width`

Indicates the contents of files, the output from commands, and generally anything found in programs.

**`Constant width bold`**

Shows commands or other text that should be typed literally by the user, and parts of code or files highlighted for discussion.

***`Constant width italic`***

Shows text that should be replaced with user-supplied values.

## Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example, "*Cloud Application Architectures* by George Reese. Copyright 2009 George Reese, 978-0-596-15636-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

# We’d Like Your Feedback!

We at O’Reilly have tested and verified the information in this book to the best of our ability, but mistakes and oversights do occur. Please let us know about errors you may find, as well as your suggestions for future editions, by writing to:


O’Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (in the U.S. or Canada)

707-829-0515 (international or local)

707-829-0104 (fax)


We have a web page for the book where we list errata, examples, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596156367

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, software, Resource Centers, and the O’Reilly Network, see our website at:

http://www.oreilly.com

# Acknowledgments

This book covers so many disciplines and so many technologies, it would have been impossible for m to write it on my own.

First, I would like to acknowledge the tremendous help I received from Randy Bias at GoGrid and E. J. Johnson at Rackspace. My experience in cloud infrastructure has been entirely with Amazon Web Services, and Randy and E. J. spent a significant amount of time reviewing the book for places where the discussion was specific to AWS. They also wrote the appendixes on the GoGrid and Rackspace offerings.

Next, I would like to thank everyone who read each chapter and provided detailed comments: John Allspaw, Jeff Barr, Christofer Hoff, Theo Schlossnagle, and James Urquhart. They each brought very unique expertise into the technical review of this book, and the book is much better than it otherwise would have been, thanks to their critical eyes.

In addition, a number of people have reviewed and provided feedback on selected parts of the book: David Bagley, Morgan Catlin, Mike Horwath, Monique Reese, Stacey Roelofs, and John Viega.

Finally, I owe the most thanks on this book to Andy Oram and Isabel Kunkle from O'Reilly. I have said this in other places, but I need to say it here: their editing makes me a better writer.

# Chapter 1. Cloud Computing

The hallmark of any buzzword is its ability to convey the appearance of meaning without conveying actual meaning. To many people, the term *cloud computing* has the feel of a buzzword.

It's used in many discordant contexts, often referencing apparently distinct things. In one conversation, people are talking about Google Gmail; in the next, they are talking about Amazon Elastic Compute Cloud (at least it has "cloud" in its name!).

But cloud computing is not a buzzword any more than the term *the Web* is. Cloud computing is the evolution of a variety of technologies that have come together to alter an organization's approach to building out an IT infrastructure. Like the Web a little over a decade ago, there is nothing fundamentally new in any of the technologies that make up cloud computing. Many of the technologies that made up the Web existed for decades when Netscape came along and made them accessible; similarly, most of the technologies that make up cloud computing have been around for ages. It just took Amazon to make them all accessible to the masses.

The purpose of this book is to empower developers of transactional web applications to leverage clou infrastructure in the deployment of their applications. This book therefore focuses on the cloud as it relates to clouds such as Amazon EC2, more so than Google Gmail. Nevertheless, we should start things off by setting a common framework for the discussion of cloud computing.

## The Cloud

The cloud is not simply the latest fashionable term for the Internet. Though the Internet is a necessar foundation for the cloud, the cloud is something more than the Internet. The cloud is where you go to use technology when you need it, for as long as you need it, and not a minute more. You do not instal anything on your desktop, and you do not pay for the technology when you are not using it.

The cloud can be both software and infrastructure. It can be an application you access through the Web or a server that you provision exactly when you need it. Whether a service is software or hardware, the following is a simple test to determine whether that service is a cloud service:

> *If you can walk into any library or Internet cafe and sit down at any computer without preference for operating system or browse and access a service, that service is cloud-based.*

I have defined three criteria I use in discussions on whether a particular service is a cloud service:

- The service is accessible via a web browser (nonproprietary) or web services API.
- Zero capital expenditure is necessary to get started.
- You pay only for what you use as you use it.

I don't expect those three criteria to end the discussion, but they provide a solid basis for discussion and reflect how I view cloud services in this book.

If you don't like my boiled-down cloud computing definition, James Governor has an excellent blog entry on "15 Ways to Tell It's Not Cloud Computing," at http://www.redmonk.com/jgovernor/2008/03/13/15-ways-to-tell-its-not-cloud-computing.

## Software

As I mentioned earlier, cloud services break down into software services and infrastructure services.

In terms of maturity, software in the cloud is much more evolved than hardware in the cloud.

*Software as a Service* (SaaS) is basically a term that refers to software in the cloud. Although not all SaaS systems are cloud systems, most of them are.

SaaS is a web-based software deployment model that makes the software available entirely through a web browser. As a user of SaaS software, you don't care where the software is hosted, what kind of operating system it uses, or whether it is written in PHP, Java, or .NET. And, above all else, you don' have to install a single piece of software anywhere.

Gmail, for example, is nothing more than an email program you use in a browser. It provides the sam functionality as Apple Mail or Outlook, but without the fat client. Even if your domain does not receive email through Gmail, you can still use Gmail to access your mail.

SalesForce.com is another variant on SaaS. SalesForce.com is an enterprise customer relationship management (CRM) system that enables sales people to track their prospects and leads, see where those individuals sit in the organization's sales process, and manage the workflow of sales from first contact through completion of a sale and beyond. As with Gmail, you don't need any software to access SalesForce.com: point your web browser to the SalesForce.com website, sign up for an accoun and get started.

SaaS systems have a few defining characteristics:

Availability via a web browser

> SaaS software never requires the installation of software on your laptop or desktop. You access it through a web browser using open standards or a ubiquitous browser plug-in. Cloud computing an proprietary desktop software simply don't mix.

On-demand availability

> You should not have to go through a sales process to gain access to SaaS-based software. Once yo have access, you should be able to go back into the software any time, from anywhere.

Payment terms based on usage

> SaaS does not need any infrastructure investment or fancy setup, so you should not have to pay an massive setup fees. You should simply pay for the parts of the service you use as you use them. When you no longer need those services, you simply stop paying.

Minimal IT demands

> If you don't have any servers to buy or any network to build out, why do you need an IT infrastructure? While SaaS systems may require some minimal technical knowledge for their configuration (such as DNS management for Google Apps), this knowledge lays within the realm of the power user and not the seasoned IT administrator.

One feature of some SaaS deployments that I have intentionally omitted is multitenancy. A number o SaaS vendors boast about their multitenancy capabilities—some even imply that multitenancy is a requirement of any SaaS system.

A multitenant application is server-based software that supports the deployment of multiple clients in a single software instance. This capability has obvious advantages for the SaaS vendor that, in some form, trickle down to the end user:

- Support for more clients on fewer hardware components
- Quicker and simpler rollouts of application updates and security patches

- Architecture that is generally more sound

The ultimate benefit to the end user comes indirectly in the form of lower service fees, quicker acces to new functionality, and (sometimes) quicker protection against security holes. However, because a core principle of cloud computing is a lack of concern for the underlying architecture of the applications you are using, the importance of multitenancy is diminished when looking at things fron that perspective.

As we discuss in the next section, virtualization technologies essentially render the architectural advantages of multitenancy moot.

# Hardware

In general, hardware in the cloud is conceptually harder for people to accept than software in the cloud. Hardware is something you can touch: you own it; you don't license it. If your server catches on fire, that disaster matters to you. It's hard for many people to imagine giving up the ability to touc and own their hardware.

With hardware in the cloud, you request a new "server" when you need it. It is ready as quickly as 10 minutes after your request. When you are done with it, you release it and it disappears back into the cloud. You have no idea what physical server your cloud-based server is running, and you probably don't even know its specific geographic location.

**THE BARRIER OF OLD EXPECTATIONS**

The hardest part for me as a vendor of cloud-based computing services is answering the question, "Where are our servers?" The real answer is, inevitably, "I don't know—somewhere on the East Coast of the U.S. or Western Europe," which makes some customers very uncomfortable. This lack of knowledge of your servers' location, however, provides an interesting physical security benefit, as it becomes nearly impossible for a motivated attacker to use a physical attack vector to compromise your systems.

## The advantages of a cloud infrastructure

Think about all of the things you have to worry about when you own and operate your own servers:

Running out of capacity?

> Capacity planning is always important. When you own your own hardware, however, you have tw problems that the cloud simplifies for you: what happens when you are wrong (either overoptimistic or pessimistic), and what happens if you don't have the expansion capital when th time comes to buy new hardware. When you manage your own infrastructure, you have to cough up a lot of cash for every new Storage Area Network (SAN) or every new server you buy. You als have a significant lead time from the moment you decide to make a purchase to getting it through the procurement process, to taking delivery, and finally to having the system racked, installed, an tested.

What happens when there is a problem?

> Sure, any good server has redundancies in place to survive typical hardware problems. Even if yo have an extra hard drive on hand when one of the drives in your RAID array fails, someone has to remove the old drive from the server, manage the RMA,[1] and put the new drive into the server. That takes time and skill, and it all needs to happen in a timely fashion to prevent a complete failure of the server.

What happens when there is a disaster?

If an entire server goes down, unless you are in a high-availability infrastructure, you have a disaster on your hands and your team needs to rush to address the situation. Hopefully, you have solid backups in place and a strong disaster recovery plan to get things operational ASAP. This process is almost certainly manual.

Don’t need that server anymore?

Perhaps your capacity needs are not what they used to be, or perhaps the time has come to decommission a fully depreciated server. What do you do with that old server? Even if you give i away, someone has to take the time to do something with that server. And if the server is not fully depreciated, you are incurring company expenses against a machine that is not doing anything for your business.

What about real estate and electricity?

When you run your own infrastructure (or even if you have a rack at an ISP), you may be paying for real estate and electricity that are largely unused. That’s a very ungreen thing, and it is a huge waste of money.

None of these issues are concerns with a proper cloud infrastructure:

- You add capacity into a cloud infrastructure the minute you need it, and not a moment sooner. Yo don’t have any capital expense associated with the allocation, so you don’t have to worry about th timing of capacity needs with budget needs. Finally, you can be up and running with new capacity in minutes, and thus look good even when you get caught with your pants down.
- You don’t worry about any of the underlying hardware, ever. You may never even know if the physical server you have been running on fails completely. And, with the right tools, you can automatically recover from the most significant disasters while your team is asleep.
- When you no longer need the same capacity or you need to move to a different virtual hardware configuration, you simply deprovision your server. You do not need to dispose of the asset or wor about its environmental impact.
- You don’t have to pay for a lot of real estate and electricity you never use. Because you are using fractional portion of a much beefier piece of hardware than you need, you are maximizing the efficiency of the physical space required to support your computing needs. Furthermore, you are not paying for an entire rack of servers with mostly idle CPU cycles consuming electricity.

## Hardware virtualization

Hardware virtualization is the enabling technology behind many of the cloud infrastructure vendors offerings, including Amazon Web Services (AWS).[2] If you own a Mac and run Windows or Linux inside Parallels or Fusion, you are using a similar virtualization technology to those that support clou computing. Through virtualization, an IT admin can partition a single physical server into any numbe of virtual servers running their own operating systems in their allocated memory, CPU, and disk footprints. Some virtualization technologies even enable you to move one running instance of a virtu server from one physical server to another. From the perspective of any user or application on the virtual server, no indication exists to suggest the server is not a real, physical server.

A number of virtualization technologies on the market take different approaches to the problem of virtualization. The Amazon solution is an extension of the popular open source virtualization system called Xen. Xen provides a hypervisor layer on which one or more guest operating systems operate. The hypervisor creates a hardware abstraction that enables the operating systems to share the

resources of the physical server without being able to directly access those resources or their use by another guest operating system.

A common knock against virtualization—especially for those who have experienced it in desktop software—is that virtualized systems take a significant performance penalty. This attack on virtualization generally is not relevant in the cloud world for a few reasons:

- The degraded performance of your cloud vendor's hardware is probably better than the optimal performance of your commodity server.
- Enterprise virtualization technologies such as Xen and VMware use paravirtualization as well as the hardware-assisted virtualization capabilities of a variety of CPU manufacturers to achieve nea native performance.

## Cloud storage

Abstracting your hardware in the cloud is not simply about replacing servers with virtualization. It's also about replacing your physical storage systems.

Cloud storage enables you to "throw" data into the cloud and without worrying about how it is stored or backing it up. When you need it again, you simply reach into the cloud and grab it. You don't kno how it is stored, where it is stored, or what has happened to all the pieces of hardware between the time you put it in the cloud and the time you retrieved it.

As with the other elements of cloud computing, there are a number of approaches to cloud storage on the market. In general, they involve breaking your data into small chunks and storing that data across multiple servers with fancy checksums so that the data can be retrieved rapidly—no matter what has happened in the meantime to the storage devices that comprise the cloud.

I have seen a number of people as they get started with the cloud attempt to leverage cloud storage as if it were some kind of network storage device. Operationally, cloud storage and traditional network storage serve very different purposes. Cloud storage tends to be much slower with a higher degree of structure, which often renders it impractical for runtime storage for an application, regardless of whether that application is running in the cloud or somewhere else.

Cloud storage is not, generally speaking, appropriate for the operational needs of transactional cloud-based software. Later, we discuss in more detail the role of cloud storage in transaction application management. For now, think of cloud storage as a tape backup system in which you never have to manage any tapes.

**NOTE**

Amazon recently introduced a new offering called Amazon CloudFront, which leverages Amazon S3 as a content distribution network. The idea behind Amazon CloudFront is to replicate your cloud content to the edges of the network. While Amazon S3 cloud storage may not be appropriate for the operational needs of many transactional web applications, CloudFront will likely prove to be a critical component to the fast, worldwide distribution of static content.

[1] Return merchandise authorization. When you need to return a defective part, you generally have to go through some vendor process for returning that part and obtaining a replacement.

[2] Other approaches to cloud infrastructure exist, including physical hardware on-demand through companies such as AppNexus a NewClouds. In addition, providers such as GoGrid (summarized in Appendix B) offer hybrid solutions.

# Cloud Application Architectures

We could spend a lot of precious paper discussing Software as a Service or virtualization technologie (did you know that you can mix and match at least five kinds of virtualization?), but the focus of this book is how you write an application so that it can best take advantage of the cloud.

## Grid Computing

Grid computing is the easiest application architecture to migrate into the cloud. A grid computing application is processor-intensive software that breaks up its processing into small chunks that can then be processed in isolation.

If you have used SETI@home, you have participated in grid computing. SETI (the Search for Extra-Terrestrial Intelligence) has radio telescopes that are constantly listening to activity in space. They collect volumes of data that subsequently need to be processed to search for a nonnatural signal that might represent attempts at communication by another civilization. It would take so long for one computer to process all of that data that we might as well wait until we can travel to the stars. But many computers using only their spare CPU cycles can tackle the problem extraordinarily quickly.

These computers running SETI@home—perhaps including your desktop—form the grid. When they have extra cycles, they query the SETI servers for data sets. They process the data sets and submit th results back to SETI. Your results are double-checked against processing by other participants, and interesting results are further checked.[3]

Back in 1999, SETI elected to use the spare cycles of regular consumers' desktop computers for its data processing. Commercial and government systems used to network a number of supercomputers together to perform the same calculations. More recently, *server farms* were created for grid computing tasks such as video rendering. Both supercomputers and server farms are very expensive, capital-intensive approaches to the problem of grid computing.

The cloud makes it cheap and easy to build a grid computing application. When you have data that needs to be processed, you simply bring up a server to process that data. Afterward, that server can either shut down or pull another data set to process.

Figure 1-1 illustrates the process flow of a grid computing application. First, a server or server cluste receives data that requires processing. It then submits that job to a message queue (1). Other servers—often called workers (or, in the case of SETI@home, other desktops)—watch the message queue (2) and wait for new data sets to appear. When a data set appears, the first computer to see it processes it and then sends the results back into the message queue (3). The two components can operate independently of each other, and one can even be running when no computer is running the other.

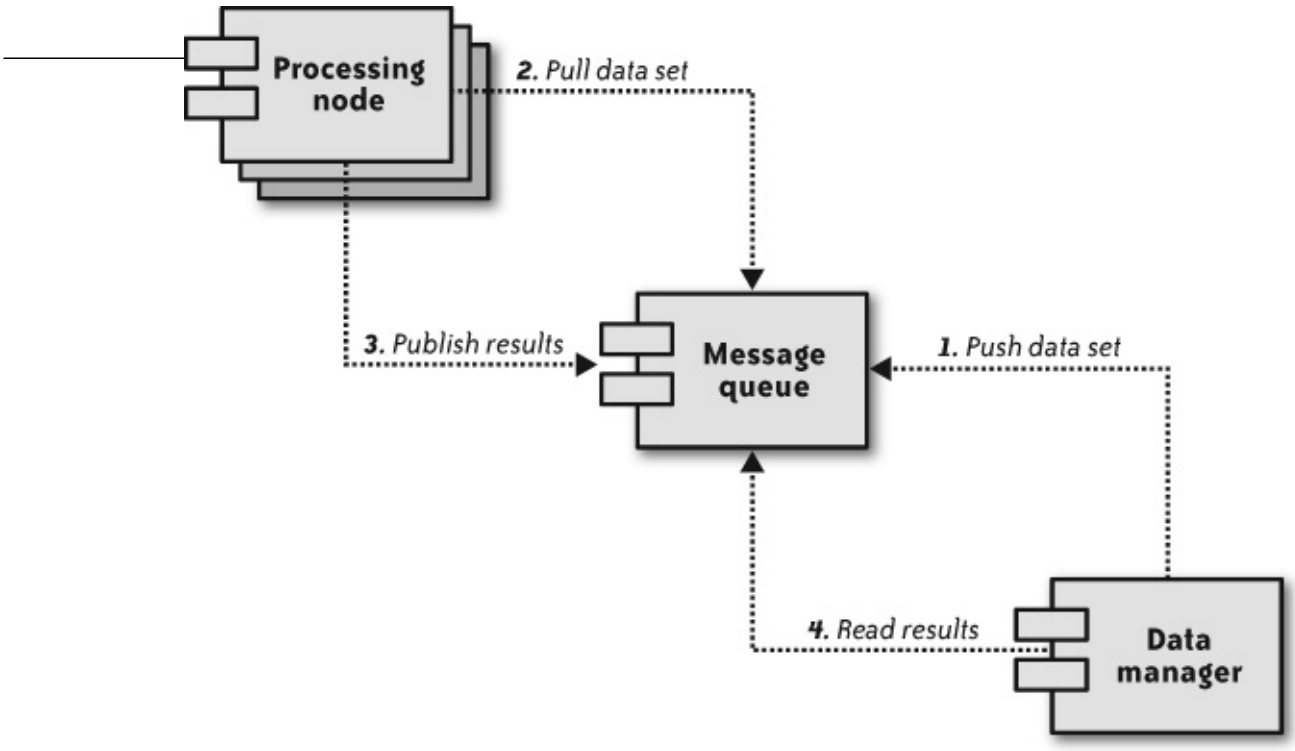


*Figure 1-1. The grid application architecture separates the core application from its data processing nodes*

Cloud computing comes to the rescue here because you do not need to own any servers when you hav no data to process. You can then scale the number of servers to support the number of data sets that are coming into your application. In other words, instead of having idle computers process data as it comes in, you have servers turn themselves on as the rate of incoming data increases, and turn themselves off as the data rate decreases.

Because grid computing is currently limited to a small market (scientific, financial, and other large-scale data crunchers), this book doesn't focus on its particular needs. However, many of the principle in this book are still applicable.

## Transactional Computing

Transactional computing makes up the bulk of business software and is the focus of this book. A *transaction system* is one in which one or more pieces of incoming data are processed together as a single transaction and establish relationships with other data already in the system. The core of a transactional system is generally a relational database that manages the relations among all of the da that make up the system.

Figure 1-2 shows the logical layout of a high-availability transactional system. Under this kind of architecture, an application server typically models the data stored in the database and presents it through a web-based user interface that enables a person to interact with the data. Most of the websit and web applications that you use every day are some form of transactional system. For high availability, all of these components may form a cluster, and the presentation/business logic tier can hide behind a load balancer.

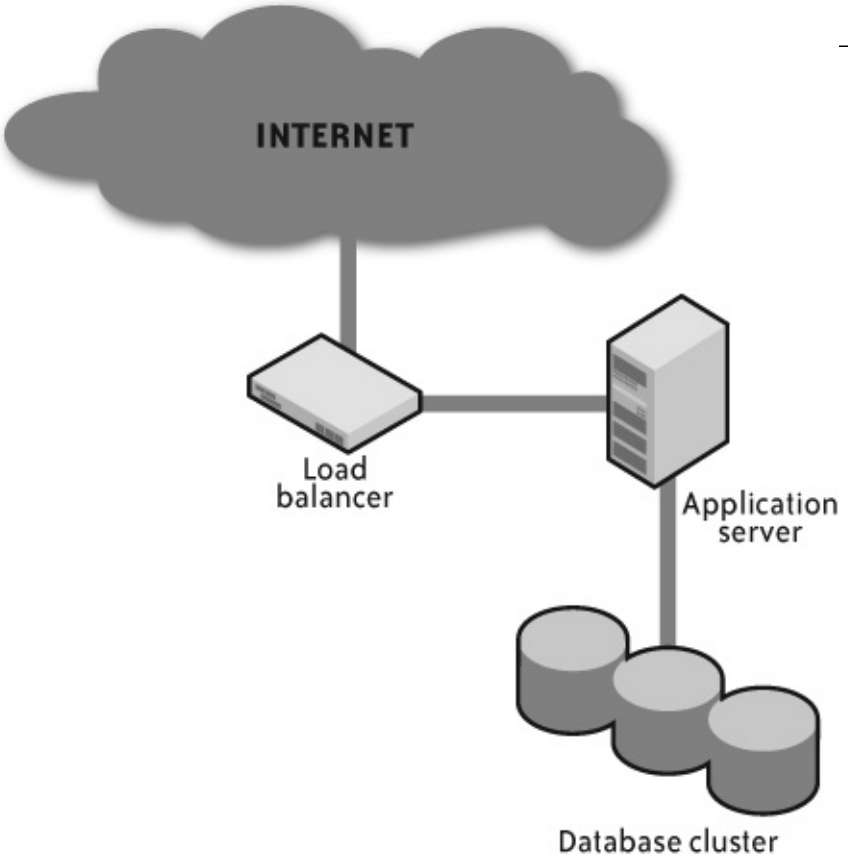


*Figure 1-2. A transactional application separates an application into presentation, business logic, and data storage*

Deploying a transactional system in the cloud is a little more complex and less obvious than deployi a grid system. Whereas nodes in a grid system are designed to be short-lived, nodes in a transactiona system must be long-lived.

A key challenge for any system requiring long-lived nodes in a cloud infrastructure is the basic fact that the mean time between failures (MTBF) of a virtual server is necessarily less than that for the underlying hardware. An admittedly gross oversimplification of the problem shows that if you have two physical servers with a three-year MTBF, you will be less likely to experience an outage across the entire system than you would be with a single physical server running two virtual nodes. The number of physical nodes basically governs the MTBF, and since there are fewer physical nodes, the is a higher MTBF for any given node in your cloud-based transactional system.

The cloud, however, provides a number of avenues that not only help mitigate the lower failure rate o individual nodes, but also potentially increase the overall MTBF for your transactional system. In thi book, we cover the tricks that will enable you to achieve levels of availability that otherwise might n be possible under your budget while still maintaining transactional integrity of your cloud applications.

---

[3] For more information on SETI@home and the SETI project, pick up a copy of O'Reilly's *Beyond Contact* (http://oreilly.com/catalog/9780596000370).

# The Value of Cloud Computing

How far can you take all of this?

If you can deploy all of your custom-built software systems on cloud hardware and leverage SaaS systems for your packaged software, you might be able to achieve an all-cloud IT infrastructure. Table 1-1 lists the components of the typical small- or medium-sized business.

*Table 1-1. The old IT infrastructure versus the cloud*

| Traditional | Cloud |
|---|---|
| File server | Google Docs |
| MS Outlook, Apple Mail | Gmail, Yahoo!, MSN |
| SAP CRM/Oracle CRM/Siebel | SalesForce.com |
| Quicken/Oracle Financials | Intacct/NetSuite |
| Microsoft Office/Lotus Notes | Google Apps |
| Stellent | Valtira |
| Off-site backup | Amazon S3 |
| Server, racks, and firewall | Amazon EC2, GoGrid, Mosso |

The potential impact of the cloud is significant. For some organizations—particularly small- to medium-sized businesses—it makes it possible to never again purchase a server or own any software licenses. In other words, all of these worries diminish greatly or disappear altogether:

- Am I current on all my software licenses? SaaS systems and software with cloud-friendly licensin simply charge your credit card for what you use.
- When do I schedule my next software upgrade? SaaS vendors perform the upgrades for you; you rarely even know what version you are using.
- What do I do when a piece of hardware fails at 3 a.m.? Cloud infrastructure management tools are capable of automating even the most traumatic disaster recovery policies.
- How do I manage my technology assets? When you are in the cloud, you have fewer technology assets (computers, printers, etc.) to manage and track.
- What do I do with my old hardware? You don’t own the hardware, so you don’t have to dispose of it.
- How do I manage the depreciation of my IT assets? Your costs are based on usage and thus don’t involve depreciable expenses.
- When can I afford to add capacity to my infrastructure? In the cloud, you can add capacity discretely as the business needs it.

SaaS vendors (whom I’ve included as part of cloud computing) can run all their services in a hardwa cloud provided by another vendor, and therefore offer a robust cloud infrastructure to their customers without owning their own hardware. In fact, my own business runs that way.

# Options for an IT Infrastructure

The cloud competes against two approaches to IT:

- Internal IT infrastructure and support
- Outsourcing to managed services

If you own the boxes, you have an internally managed IT infrastructure—even if they are sitting in a rack in someone else's data center. For you, the key potential benefit of cloud computing (certainly financially) is the lack of capital investment required to leverage it.

Internal IT infrastructure and support is one in which you own the boxes and pay people—whether staff or contract employees—to maintain those boxes. When a box fails, you incur that cost, and you have no replacement absent a cold spare that you own.

Managed services outsourcing has similar benefits to the cloud in that you pay a fixed fee for someone else to own your servers and make sure they stay up. If a server goes down, it is the managed services company who has to worry about replacing it immediately (or within whatever terms have been defined in your service-level agreement). They provide the expertise to make sure the servers are fixed with the proper operating system patches and manage the network infrastructure in which the servers operate.

Table 1-2 provides a comparison between internal IT, managed services, and cloud-based IT with respect to various facets of IT infrastructure development.

*Table 1-2. A comparison of IT infrastructure options*

| | Internal IT | Managed services | The cloud |
|---|---|---|---|
| **Capital investment** | Significant | Moderate | Negligible |
| | How much cash do you have to cough up in order to set up your infrastructure or make changes to it? With internal IT, you have to pay for your hardware before you need it (financing is not important in this equation).[a] Under managed services, you are typically required to pay a moderate setup fee. In the cloud, you generally have no up-front costs and no commitment. | | |
| **Ongoing costs** | Moderate | Significant | Based on usage |
| | Your ongoing costs for internal IT are based on the cost of staff and/or contractors to manage the infrastructure, as well as space at your hosting provider and/or real estate and utilities costs. You can see significant variances in the ongoing costs—especially with contract resources—as emergencies occur and other issues arise. Although managed services are often quite pricey, you generally know exactly what you are going to pay each month and it rarely varies. The cloud, on the other hand, can be either pricey or cheap, depending on your needs. Its key advantage is that you pay for exactly what you use and nothing more. Your staff costs are greater than with a managed services provider, but less than with internal IT. | | |
| **Provisioning time** | Significant | Moderate | None |
| | How long does it take to add a new component into your infrastructure? Under both the internal IT and managed services models, you need to plan ahead of time, place an order, wait for the component to arrive, and then set it up in the data center. The wait is typically significantly shorter with a managed services provider, since they make purchases ahead of time in bulk. Under the cloud, however, you can have a new "server" operational within minutes of deciding you want it. | | |
| **Flexibility** | Limited | Moderate | Flexible |

| | | | |
|---|---|---|---|
| | How easily can your infrastructure adapt to unexpected peaks in resource demands? For example, do you have a limit on disk space? What happens if you suddenly approach that limit? Internal IT has a very fixed capacity and can meet increased resource demands only through further capital investment. A managed services provider, on the other hand, usually can offer temporary capacity relief by uncapping your bandwidth, giving you short-term access to alternative storage options, and so on. The cloud, however, can be set up to automatically add capacity into your infrastructure as needed, and to let go of that capacity when it is no longer required. | | |
| **Staff expertise requirements** | Significant | Limited | Moderate |
| | How much expertise do you need in-house to support your environments? With internal IT, you obviously need staff or contractors who know the ins and outs of your infrastructure, from opening the boxes up and fiddling with the hardware to making sure the operating systems are up-to-date with the latest patches. The advantage here goes to the managed services infrastructure, which enables you to be largely ignorant of all things IT. Finally, the cloud may require a lot of skill or very little skill, depending on how you are using it. You can often find a cloud infrastructure manager (enStratus or RightScale, for example) to manage the environment, but you still must have the skills to set up your machine images. | | |
| **Reliability** | Varies | High | Moderate to high |
| | How certain are you that your services will stay up 24/7? The ability to create a high-availability infrastructure with an internal IT staff is a function of the skill level of your staff and the amount of cash you invest in the infrastructure. A managed services provider is the safest, most proven alternative, but this option can lack the locational redundancy of the cloud. A cloud infrastructure, finally, has significant locational redundancies but lacks a proven track record of stability. | | |

[a] From a financial perspective, the difference between coughing up cash today and borrowing it from a bank is inconsequential. Either way, spending $40K costs you money. If you borrow it, you pay interest. If you take it out of your bank account, you lose the opportunity to do something else with it (cost of capital).

The one obvious fact that should jump out of this chart is that building an IT infrastructure from scratch no longer makes any sense. The only companies that should have an internal IT are organizations with a significant preexisting investment in internal IT or with regulatory requirements that prevent data storage in third-party environments.

Everyone else should be using a managed services provider or the cloud.

# The Economics

Perhaps the biggest benefit of cloud computing over building out your own IT infrastructure has nothing to do with technology—it's financial. The "pay for what you use" model of cloud computing is significantly cheaper for a company than the "pay for everything up front" model of internal IT.

## Capital costs

The primary financial problem with an internally based IT infrastructure is the *capital cost*. A capital cost is cash you pay for assets prior to their entering into operations. If you buy a server, that purchas is a capital cost because you pay for it all up front, and then you realize its benefits (in other words, you use it) over the course of 2–3 years.

Let's look at the example of a $5,000 computer that costs $2,000 to set up. The $5,000 is a capital co and the $2,000 is a one-time expense. From an accounting perspective, the $5,000 cost is just a "funn money" transaction, in that $5,000 is moved from one asset account (your bank account) into another asset account (your fixed assets account). The $2,000, on the other hand, is a real expense that offsets your profits.

The server is what is called a *depreciable asset*. As it is used, the server is depreciated in accordance with how much it has been used. In other words, the server's value to the company is reduced each month it is in use until it is worth nothing and removed from service. Each reduction in value is considered an expense that offsets the company's profits.

Finance managers hate capital costs for a variety of reasons. In fact, they hate any expenses that are not tied directly to the current operation of the company. The core rationale for this dislike is that yo are losing cash today for a benefit that you will receive slowly over time (technically, over the course of the depreciation of the server). Any business owner or executive wants to focus the organization's cash on things that benefit them today. This concern is most acute with the small- and medium-sized business that may not have an easy time walking into the bank and asking for a loan.

The key problem with this delayed realization of value is that money costs money. A company will often fund their operational costs through revenues and pay for capital expenses through loans. If you can grow the company faster than the cost of money, you win. If you cannot grow that rapidly or—worse—you cannot get access to credit, the capital expenses become a significant drain on the organization.

## Cost comparison

Managed services infrastructures and the cloud are so attractive to companies because they largely eliminate capital investment and other up-front costs. The cloud has the added advantage of tying yo costs to exactly what you are using, meaning that you can often connect IT costs to revenue instead o treating them as overhead.

Table 1-3 compares the costs of setting up an infrastructure to support a single "moderately high availability" transactional web application with a load balancer, two application servers, and two database servers. I took typical costs at the time of writing, October 2008.

*Table 1-3. Comparing the cost of different IT infrastructures*

| | Internal IT | Managed services | The cloud |
|---|---|---|---|
| **Capital investment** | $40,000 | $0 | $0 |
| **Setup costs** | $10,000 | $5,000 | $1,000 |
| **Monthly service fees** | $0 | $4,000 | $2,400 |
| **Monthly staff costs** | $3,200 | $0 | $1,000 |
| **Net cost over three years** | $149,000 | $129,000 | $106,000 |

Table 1-3 makes the following assumptions:

- The use of fairly standard 1u server systems, such as a Dell 2950 and the high-end Amazon instances.
- The use of a hardware load balancer in the internal IT and managed services configuration and a software load balancer in the cloud.
- No significant data storage or bandwidth needs (different bandwidth or storage needs can have a significant impact on this calculation).
- The low end of the cost spectrum for each of the options (in particular, some managed services

providers will charge up to three times the costs listed in the table for the same infrastructure).

- Net costs denominated in today's dollars (in other words, don't worry about inflation).
- A cost of capital of 10% (cost of capital is what you could have done with all of the up-front cash instead of sinking it into a server and setup fees—basically the money's interest rate plus opportunity costs).
- The use of third-party cloud management tools such as enStratus or RightScale, incorporated into the cloud costs.
- Staff costs representing a fraction of an individual (this isolated infrastructure does not demand a full-time employee under any model).

Perhaps the most controversial element of this analysis is what might appear to be an "apples versus oranges" comparison on the load balancer costs. The reality is that this architecture doesn't really require a hardware load balancer except for extremely high-volume websites. So you likely could get away with a software load balancer in all three options.

A software load balancer, however, is very problematic in both the internal IT and managed services infrastructures for a couple of reasons:

- A normal server is much more likely to fail than a hardware load balancer. Because it is much harder to replace a server in the internal IT and managed services scenarios, the loss of that software load balancer is simply unacceptable in those two scenarios, whereas it would go unnoticed in the cloud scenario.
- If you are investing in actual hardware, you may want a load balancer that will grow with your IT needs. A hardware load balancer is much more capable of doing that than a software load balancer. In the cloud, however, you can cheaply add dedicated software load balancers, so it becomes a nonissue.

In addition, some cloud providers (GoGrid, for example) include free hardware load balancing, which makes the entire software versus hardware discussion moot. Furthermore, Amazon is scheduled to offer its own load-balancing solution at some point in 2009. Nevertheless, if you don't buy into my rationale for comparing the hardware load balancers against the software load balancers, here is the comparison using all software load balancers: $134K for internal IT, $92K for managed services, and $106K for a cloud environment.

## The bottom line

If we exclude sunk costs, the right managed services option and cloud computing are always financially more attractive than managing your own IT. Across all financial metrics—capital requirements, total cost of ownership, complexity of costs—internal IT is always the odd man out.

As your infrastructure becomes more complex, determining whether a managed services infrastructure, a mixed infrastructure, or a cloud infrastructure makes more economic sense becomes significantly more complex.

If you have an application that you know has to be available 24/7/365, and even 1 minute of downtime in a year is entirely unacceptable, you almost certainly want to opt for a managed services environment and not concern yourself too much with the cost differences (they may even favor the managed services provider in that scenario).

On the other hand, if you want to get high-availability on the cheap, and 99.995% is good enough, you can't beat the cloud.

## URQUHART ON BARRIERS TO EXIT

*In November 2008, James Urquhart and I engaged in a Twitter discussion[4] relating to the total cost of ownership of cloud computing (James is a market manager for the Data Center 3.0 strategy at Cisco Systems and member of the CNET blog network). What we realized is that I was looking at the problem from the perspective of starting with a clean slate; James was looking at the problem from the reality of massive existing investments in IT. What follows is a summary of our discussion that James has kindly put together for this book.*

While it is easy to get enthusiastic about the economics of the cloud in "green-field" comparisons, most modern medium-to-large enterprises have made a significant investment in IT infrastructure that must be factored into the cost of moving to the cloud.

These organizations already own the racks, cooling, and power infrastructure to support new applications, and will not incur those capital costs anew. Therefore, the cost of installing and operating additional servers will be significantly less than in the examples.

In this case, these investments often tip the balance, and it becomes much cheaper to use existing infrastructure (though with some automation) to deliver relatively stable capacity loads. This existing investment in infrastructure therefore acts almost as a "barrier-to-exit" for such enterprises considering a move to the cloud.

Of course, there are certain classes of applications that even a large enterprise will find more cost effective to run in the cloud. These include:

- Applications with widely varying loads, for which peak capacity is hard to predict, and for which purchasing for peak load would be inefficient most of the time anyway.
- Applications with occasional or periodic load spikes, such as tax processing applications or retail systems hit hard prior to the holidays. The cloud can provide excess capacity in this case, through a technique called "cloudbursting."
- New applications of the type described in this book that would require additional data center space or infrastructure investment, such as new cooling or power systems.

It seems to me highly ironic—and perhaps somewhat unique—that certain aspects of the cloud computing market will be blazed not by organizations with multiple data centers and thousands upon thousands of servers, but by the small business that used to own a few servers in a server hotel somewhere that finally shut them down and turned to Amazon. How cool is that?

---

[4] http://blog.jamesurquhart.com/2008/12/enterprise-barrier-to-exit-to-cloud.html