RIGHTSCALE™

Cloud Computing. Delivered.

# Application Architecture for Cloud Computing

## Introduction

CONTRIBUTORS

*Jeff Barr, Evangelist*

*Thorsten von Eicken,*

*CTO and Founder*

*Erik Troan, CTO and Founder*

Cloud computing is a relatively new way of referring to the use of shared computing resources, and it is an alternative to having local servers handle applications. Cloud computing groups together large numbers of compute servers and other resources and typically offers their combined capacity on an on-demand, pay-per-cycle basis. The end users of a cloud computing network usually have no idea where the servers are physically located – they just spin up their application and start working.

Cloud computing is fully enabled by virtualization technology (hypervisors) and virtual appliances. A virtual appliance is an application that is bundled with all the components that it needs to run, along with a streamlined operating system. In a cloud computing environment, a virtual appliance can be instantly provisioned and decommissioned as needed, without complex configuration of the operating environment.

This flexibility is the key advantage to cloud computing, and what distinguishes it from other forms of grid or utility computing and software as a service (SaaS). The ability to launch new instances of an application with minimal labor and expense allows application providers to:

- Scale up and down rapidly
- Recover from a failure
- Bring up development or test instances
- Roll out new versions to the customer base
- Efficiently load test an application

Once you've decided to offer your application in a cloud computing environment, it is important to avoid the "success disaster". This is when your application becomes popular overnight, so much so that it crashes under the unanticipated load. In this paper, we'll discuss how you can design your application for the cloud from the outset to take maximum advantage of the cloud environment.

If you haven't yet decided to offer your application in the cloud, now is the time. Cloud computing is not yesterday's fad; it is today's reality. Your competition is likely already working on their cloud application, allowing them to deploy faster, better and cheaper than you.

## The Economics of the Cloud

Before we delve into how to architect an application for a cloud computing environment, we should explain why it is financially advantageous to do so.

The first clear advantage of using an existing cloud infrastructure is that you don't have to make the capital investment yourself. Rather than expending the time and cost to build a data center, you use someone else's investment and turn what would have been a massive capital outlay into a manageable variable cost.

In the pay-per-cycle model of cloud computing, you can start small and requisition more computer time as your business grows. This makes starting up inexpensive, and gives you time to build your on-demand business before investing in additional capacity. Instead of investing ahead of demand, you simply use and pay for exactly what you need when you need it.

Development time can also be a significant cost in creating an on-demand application environment. If you adopt a SaaS model, your entire application must be re-architected to support multi-tenancy. With cloud computing, the cost of a machine year in the Amazon EC2 cloud (~$880 annually) is much less than the cost of a fully-loaded developer (anywhere from $400-$1000 per day). This makes it a lot less expensive to scale up more virtual servers in the cloud than it is to spend even one day on development.

Finally, you can save money by designing your application with a simpler architecture ideal for cloud computing, which we'll spend the rest of this paper discussing. A simpler architecture speeds time to market because it is easier to test, and you can eliminate some of the equipment and processes required to migrate an application from development into production. All the activities involved with development, test, QA and production can exist side-by-side in separate instances running in the cloud.

## Architectural Considerations

Designing an application to run as a virtual appliance in a cloud computing environment is very different than designing it for an on-premise or SaaS deployment. We discuss the following considerations. To be successful in the cloud, your application must be designed to scale easily, tolerate failures and include management tools.

Scale

Cloud computing offers the potential for nearly unlimited scalability, as long as the application is designed to scale from the outset. The best way to ensure this is to follow some basic application design guidelines:

*Start simple:* Avoid complex design and performance enhancements or optimizations in favor of simplicity. It's a good idea to start with the simplest application and rely on the scalability of the cloud to provide enough servers to ensure good application performance. Once you've gotten some traction and demand has grown, then you can focus on improving the efficiency of your application, which allows you to serve more users with the same number of servers or to reduce the number of servers while maintaining performance. Some common design techniques to improve performance include caching, server affinity, multi-threading and tight sharing of data, but they all make it more difficult to distribute your application across many servers. This is the reason you don't want to introduce them at the outset and only consider them when you need to and can ensure that you are not breaking horizontal scalability.

*Split application functions and couple loosely:* Use separate systems for different pieces of application functionality and avoid synchronous connections between them. Again, as demand grows, you can scale each one independently instead of having to scale the entire application when you hit a bottleneck. The separation and reusability of functions inherent in SOA make it an ideal architecture for the cloud.

*Network communication:* Design the application to use network based interfaces and not interprocess communication or file based communication paradigms. This allows you to effectively scale in the cloud because each piece of the application can be separated into distinct systems.

*Consider the cluster:* Rather than scale a single system up to serve all users, consider splitting your system into multiple smaller clusters, each serving a fraction of the application load. This is often called “sharding” and many web services can be split up along one dimension, often users or account. Requests can then be directed to the appropriate cluster based on some request attribute or users can be redirected to a specific cluster at login. To deploy a clustered system, determine the right collection of servers that yield efficient application performance, taking any needed functional redundancy into account; for example, 2 web, 4 application and 2 database servers. You can then scale the application by replicating the ideal cluster size and splitting the system load across the servers in the clusters.

You’ll soon realize the advantages of cloud computing when it comes to scalability, such as:

- Inexpensive testing - testing can be done against a test cluster without risking the performance or integrity of the production system. You can also test the upper limits of the ideal cluster’s performance by using “robot users” in the cloud to generate load.
- Reduced risk - bring up a test instance of the cluster to prove a new code base, and roll out a new version one cluster at a time. Fall back to an older version if the new version doesn’t work, without disrupting current users.
- Ability to segment the customer base – use clusters to separate customers with varying demands, such as a large customer who wants a private instance of the application, or one who requires extensive customizations.
- Auto-scaling based on application load – with the ready availability of resources, applications can be built to recognize when they are reaching the limits of their current configuration and automatically bring up new resources.

## Fail

Inevitably, an application will fail, no matter what its environment. When you design an on-premise or SaaS application, you typically consider several “doomsday” scenarios. The same must be true for designing an application that runs in the cloud.

*Build-in resiliency and fault tolerance:* To tolerate failure, applications must operate as a part of a group, while not being too tightly coupled to their peers. Each piece of the application should be able to continue to execute despite the loss of other functions. Asynchronous interfaces are an ideal mechanism to help application components tolerate failures or momentary unavailability of other components.

*Distribute the impact of failure:* With a distributed cloud application, a failure in any one application cluster affects only a portion of the application and not the whole application. By spreading the load across multiple clusters in the cloud, you can isolate the individual clusters against failure in another cluster.

*Get back up quickly:* Automate the launching of new application clusters in order to recover quickly. Application components must be able to come up in an automated fashion, configure themselves and join the application cluster. Cloud computing provides the ideal environment for this fast startup and recovery process.

*Data considerations:* When an application fails, data persistence and system state cannot be taken for granted. To ensure data preservation, put all data on persistent storage and make sure it is replicated and distributed. If system state is stored and then used in the recovery process, treat it like data so the system can be restarted from the point of failure.

*Test your "doomsday" scenario:* Cloud computing makes it easy to bring up an instance of your application to test various failure scenarios. Because of the flexible nature of cloud computing, it is possible to simulate many different failure scenarios at a very reasonable cost. Single instances of a system can be taken off-line to see how the rest of the application will respond. Likewise, multiple recovery scenarios can be planned and executed ahead of any real production failure.

*Be aware of the real cost of failure:* Of course the ideal situation is avoiding any application failure, but what is the cost to provide that assurance? A large internet company once said that they could tolerate failure as long as the impact was small enough as to not be noticeable to the overall customer base. This assertion came from an analysis of what it would cost to ensure seven nines of application uptime versus the impact of a failure on a portion of the customer base.

## Manage

Deploying cloud applications as virtual appliances makes management significantly easier. The appliances should bring with them all of the software they need for their entire lifecycle in the cloud. More important, they should be built in a systematic way, akin to an assembly line production effort as opposed to a hand crafted approach. The reason for this systematic approach is the consistency of creating and re-creating images. We have shown how effectively scaling and failure recovery can be handled by rapid provisioning of new systems, but these benefits cannot be achieved if the images to be provisioned are not consistent and repeatable.

When building appliances, it is obvious that they should contain the operating system and any middleware components they need. Less obvious are the software packages that allow them to automatically configure themselves, monitor and report their state back to a management system, and update themselves in an automated fashion. Automating the appliance configuration and updates means that as the application grows in the cloud, the management overhead does not grow in proportion. In this way appliances can live inside the cloud for any length of time with minimal management overhead.

When appliances are instantiated in the cloud, they should also plug into a monitoring and management system. This system will allow you to track application instances running in the cloud, migrate or shutdown instances as needed, and gather logs and other system information necessary for troubleshooting or auditing. Without a management system to handle the virtual appliances, it is likely that the application will slowly sprawl across the cloud, wasting resources and money.

Your management system also plays an important role in the testing and deployment process. We've already highlighted how the cloud can be used for everything from general testing to load testing to testing for specific failure scenarios. Including your testing in your management system allows you to bring up a test cluster, conduct any testing that is required and then migrate the tested application into production. The uniform resources that underlie the cloud mean that you can achieve a rapid release to production process, allowing you to deliver updated features and functions to your customers faster.

Finally, by automating the creation and management of these appliances, you are tackling one of the most difficult and expensive problems in software today: variability. By producing a consistent appliance image and managing it effectively, you are removing variability from the release management and deployment process. Reducing the variability reduces the chances of mistakes – mistakes that can cost you money.

The advantages of designing your application for management in the cloud include:

- Reducing the cost and overhead of preparing the application for the cloud
- Reducing the overhead of bringing up new instances of the application
- Eliminating application sprawl
- Reducing the chance for mistakes as the application is scaled out, failed over, upgraded, etc.

## Summary

Cloud computing represents an exciting opportunity to bring on-demand applications to customers in an environment of reduced risk and enhanced reliability. However, it is important to understand that existing applications cannot just be unleashed on the cloud as is. Careful attention to design will help ensure a successful deployment.

In particular, cloud-based applications should be deployed as virtual appliances so they contain all the components needed to operate, update and manage them. Simple design will aid with scalability as demand for the application increases. And, planning for failure will ensure that the worst doesn't happen when the inevitable occurs.

Don't be afraid to have your head in the clouds when it comes to application deployment. Your customers will reap the benefits, and you'll gain the competitive advantage of a flexible, scalable application solution.

amazon web services™

1.206.266.1100
aws.amazon.com

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Just as Amazon Simple Storage Service (Amazon S3) enables storage in the cloud, Amazon EC2 enables "compute" in the cloud. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use.

RightScale™

1.866.720.0208
info@rightscale.com

RightScale's software as a service platform and expertise enable companies to create scalable web solutions on cloud computing services that are reliable, easy to manage, and cost less. RightScale provides a software platform on top of Amazon Web Services that lets software businesses harness the power of the on-demand, pay-as-you-go storage and computing services through automated management tools and packaged application solutions.

rPath

1.866.508.6200
info@rpath.com

For software companies that want to accelerate license growth, expand into new markets, and reduce support and development costs, rPath's rBuilder transforms applications into virtual appliances. A virtual appliance is an application combined with just enough operating system (JeOS) for it to run optimally in any virtualized environment. Virtual appliances eliminate the hassles of installing, configuring and maintaining complex application environments. Only rPath's technology produces appliances in multiple virtual machine formats, simplifies application distribution, and lowers the customer service costs of maintenance and management.

© Copyright 2008, rPath, Inc. rPath, rBuilder and the rPath logo are registered trademarks of rPath, Inc. All other trademarks are property of their respective owners.