Introduction

This paper illustrates the style of building applications using services available in the Internet cloud.

*Cloud Architectures* are designs of software applications that use Internet-accessible on-demand services. Applications built on Cloud Architectures are such that the underlying computing infrastructure is used only when it is needed (for example to process a user request), draw the necessary resources on-demand (like compute servers or storage), perform a specific job, then relinquish the unneeded resources and often dispose themselves after the job is done. While in operation the application scales up or down **elastically** based on resource needs.

This paper is divided into two sections. In the **first section**, we describe an example of an application that is currently in production using the on-demand infrastructure provided by Amazon Web Services. This application allows a developer to do pattern-matching across millions of web documents. The application brings up hundreds of **virtual servers** on-demand, runs a parallel computation on them using an open source distributed processing framework called *Hadoop*, then shuts down all the virtual servers releasing all its resources back to the cloud—all with low programming effort and at a very reasonable cost for the caller.

In the **second section**, we discuss some best practices for using each Amazon Web Service - Amazon S3, Amazon SQS, Amazon SimpleDB and Amazon EC2 - to build an industrial-strength scalable application.

Why Cloud Architectures?

Cloud Architectures address key difficulties surrounding large-scale data processing. In traditional data processing it is difficult to get as many machines as an application needs. Second, it is difficult to get the machines when one needs them. Third, it is difficult to distribute and co-ordinate a large-scale job on different machines, run processes on them, and provision another machine to recover if one machine fails. Fourth, it is difficult to auto-scale up and down based on dynamic workloads. Fifth, it is difficult to get rid of all those machines when the job is done. Cloud Architectures solve such difficulties.

Applications built on Cloud Architectures run in-the-cloud where the physical location of the infrastructure is determined by the provider. They take advantage of simple APIs of Internet-accessible services that scale on-demand, that are industrial-strength, where the complex reliability and scalability logic of the underlying services remains implemented and hidden inside-the-cloud. The usage of resources in Cloud Architectures is as needed, sometimes ephemeral or seasonal, thereby providing the highest utilization and optimum bang for the buck.

Business Benefits of Cloud Architectures

There are some clear business benefits to building applications using Cloud Architectures. A few of these are listed here:

1. *Almost zero upfront infrastructure investment*: If you have to build a large-scale system it may cost a fortune to invest in real estate, hardware (racks, machines, routers, backup power supplies), hardware management (power management, cooling), and operations personnel. Because of the upfront costs, it would typically need several rounds of management approvals before the project could even get started. Now, with utility-style computing, there is no fixed cost or startup cost.

*2. Just-in-time Infrastructure*: In the past, if you got famous and your systems or your infrastructure did not scale you became a victim of your own success. Conversely, if you invested heavily and did not get famous, you became a victim of your failure. By deploying applications in-the-cloud with dynamic capacity management software architects do not have to worry about pre-procuring capacity for large-scale systems. The solutions are low risk because you scale only as you *grow*. Cloud Architectures can relinquish infrastructure as quickly as you got them in the first place (in minutes).

*3. More efficient resource utilization:* System administrators usually worry about hardware procuring (when they run out of capacity) and better infrastructure utilization (when they have excess and idle capacity). With Cloud Architectures they can manage resources more effectively and efficiently by having the applications request and relinquish resources only what they need (on-demand).

4. *Usage-based costing*: Utility-style pricing allows billing the customer only for the infrastructure that has been used. The customer is not liable for the entire infrastructure that may be in place. This is a subtle difference between desktop applications and web applications. A desktop application or a traditional client-server application runs on customer’s own infrastructure (PC or server), whereas in a Cloud Architectures application, the customer uses a third party infrastructure and gets billed only for the fraction of it that was used.

5. *Potential for shrinking the processing time*: Parallelization is the one of the great ways to speed up processing. If one compute-intensive or data-intensive job that can be run in parallel takes 500 hours to process on one machine, with Cloud Architectures, it would be possible to spawn and launch 500 instances and process the same job in 1 hour. Having available an elastic infrastructure provides the application with the ability to exploit parallelization in a cost-effective manner reducing the total processing time.

Examples of Cloud Architectures

There are plenty of examples of applications that could utilize the power of Cloud Architectures. These range from back-office bulk processing systems to web applications. Some are listed below:

 Processing Pipelines

 Document processing pipelines – convert hundreds of thousands of documents from Microsoft Word to PDF, OCR millions of pages/images into raw searchable text

 Image processing pipelines – create thumbnails or low resolution variants of an image, resize millions of images

 Video transcoding pipelines – transcode AVI to MPEG movies

 Indexing – create an index of web crawl data

 Data mining – perform search over millions of records

 Batch Processing Systems

 Back-office applications (in financial, insurance or retail sectors)

 Log analysis – analyze and generate daily/weekly reports

 Nightly builds – perform nightly automated builds of source code repository every night in parallel

 Automated Unit Testing and Deployment Testing – Test and deploy and perform automated unit testing (functional, load, quality) on different deployment configurations every night

 Websites

 Websites that ―sleep‖ at night and auto-scale during the day

 Instant Websites – websites for conferences or events (Super Bowl, sports tournaments)

 Promotion websites

 ―Seasonal Websites‖ - websites that only run during the tax season or the holiday season (―Black Friday‖ or Christmas)

In this paper, we discuss one application example in detail – code-named as “GrepTheWeb”

**Introduction**

For several years, software architects have discovered and implemented several concepts and best practices to build **highly scalable applications**. In today’s "era of tera", these concepts are even more applicable because of ever-growing datasets, unpredictable traffic patterns, and the demand for faster response times. This paper will reinforce and reiterate some of these traditional concepts and discuss how they may evolve in the context of cloud computing. It will also discuss some unprecedented concepts such as **elasticity** that have emerged due to the dynamic nature of the cloud.

This paper is targeted towards ***cloud architects***who are gearing up to move an enterprise-class application from a fixed physical environment to a **virtualized** cloud environment. The focus of this paper is to highlight concepts, principles and best practices in **creating** new *cloud applications* or **migrating** existing applications to the cloud.

**Background**

As a cloud architect, it is important to understand the benefits of cloud computing. In this section, you will learn some of the business and technical benefits of cloud computing and different **AWS services** available today.

**Business Benefits of Cloud Computing**

There are some clear business benefits to building applications in the cloud. A few of these are listed here:

***Almost zero upfront infrastructure investment***: If you have to build a large-scale system it may cost a fortune to invest in real estate, physical security, hardware (racks, servers, routers, backup power supplies), hardware management (power management, cooling), and operations personnel. Because of the high upfront costs, the project would typically require several rounds of management approvals before the project could even get started. Now, with utility-style cloud computing, there is no fixed cost or startup cost.

***Just-in-time Infrastructure***: In the past, if your application became popular and your systems or your infrastructure did not scale you became a victim of your own success. Conversely, if you invested heavily and did not get popular, you became a victim of your failure. By deploying applications in-the-cloud with just-in-time self-provisioning, you do not have to worry about pre-procuring capacity for large-scale systems. This increases agility, lowers risk and lowers operational cost because you scale only as you *grow* and only pay for what you use.

***More efficient resource utilization****:* System administrators usually worry about procuring hardware (when they run out of capacity) and higher infrastructure utilization (when they have excess and idle capacity). With the cloud, they can manage resources more effectively and efficiently by having the applications request and relinquish resources on-demand.

***Usage-based costing***: With utility-style pricing, you are billed only for the infrastructure that has been used. You are not paying for allocated but unused infrastructure. This adds a new dimension to cost savings. You can see immediate cost savings (sometimes as early as your next month’s bill) when you deploy an optimization patch to update your cloud application. For example, if a caching layer can reduce your data requests by 70%, the savings begin to accrue immediately and you see the reward right in the next bill. Moreover, if you are building platforms on the top of the cloud, you can pass on the same flexible, variable usage-based cost structure to your own customers.

***Reduced time to market***: Parallelization is the one of the great ways to speed up processing. If *one compute-intensive or data-intensive job* that can be run in parallel takes 500 hours to process on one machine, with cloud architectures [6], it would be possible to spawn and launch 500 instances and process the same job in 1 hour. Having available an elastic infrastructure provides the application with the ability to exploit parallelization in a *cost-effective manner* reducing time to market.

**Technical Benefits of Cloud Computing**

Some of the technical benefits of cloud computing includes:

***Automation – “Scriptable infrastructure”:***You can create repeatable build and deployment systems by *leveraging* programmable (API-driven) infrastructure.

***Auto-scaling:***You can scale your applications up and down to match your unexpected demand without any human intervention. Auto-scaling encourages automation and drives more efficiency.

***Proactive Scaling****:* Scale your application up and down to meet your *anticipated demand* with proper planning understanding of your traffic patterns so that you keep your costs low while scaling.

***More Efficient Development lifecycle****:* Production systems may be easily cloned for use as development and test environments. *Staging environments* may be easily promoted to production.

***Improved Testability****:* Never run out of hardware for testing. Inject and *automate testing* at every stage during the development process. You can spawn up an “instant test lab” with pre-configured environments only for the duration of testing phase.

***Disaster Recovery and Business Continuity:***The cloud provides a *lower cost option* for maintaining a fleet of DR servers and data storage. With the cloud, you can take advantage of geo-distribution and replicate the environment in other location within minutes.

***“Overflow” the traffic to the cloud:***With a few clicks and effective load balancing tactics, you can create a *complete overflow-proof application* by routing excess traffic to the cloud.

**Understanding the Amazon Web Services Cloud**



The Amazon Web Services (AWS) cloud provides a highly reliable and scalable infrastructure for deploying **web-scale solutions**, with minimal support and administration costs, and more flexibility than you’ve come to expect from your own infrastructure, either on-premise or at a datacenter facility.

AWS offers variety of infrastructure services today. The diagram below will introduce you the AWS terminology and help you understand how your application can interact with different Amazon Web Services and how different services interact with each other.

**Amazon Elastic Compute Cloud (Amazon EC2)**1 is a web service that provides resizable compute capacity in the cloud. You can bundle the operating system, application software and **associated configuration settings** into an *Amazon Machine Image* (AMI). You can then use these AMIs to provision **multiple virtualized *instances***as well as decommission them using simple **web service calls** to scale capacity up and down quickly, as your capacity requirement changes. You can purchase *On-Demand Instances* in which you pay for the instances by the hour or *Reserved Instances* in which you pay a low, one-time payment and receive a lower usage rate to run the instance than with an On-Demand Instance or *Spot Instances* where you can bid for unused capacity and further reduce your cost. Instances can be launched in one or more geographical ***regions***. Each region has multiple ***Availability Zones***. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.

***Elastic IP* addresses** allow you to allocate a static IP address and programmatically assign it to an instance. You can enable monitoring on an Amazon EC2 instance using **Amazon CloudWatch2** in order to gain visibility into resource utilization, operational performance, and overall demand patterns (including metrics such as CPU utilization, disk reads and writes, and network traffic). You can create ***Auto-scaling Group***using the Auto-scaling feature3 to automatically scale your capacity on certain conditions based on metric that **Amazon CloudWatch** collects. You can also distribute incoming traffic by creating an *elastic load balancer* using the **Elastic Load Balancing4** service. **Amazon Elastic Block Storage (EBS)***5 volumes* provide **network-attached persistent storage** to Amazon EC2 instances. **Point-in-time consistent *snapshots***of EBS volumes can be created and stored on **Amazon Simple Storage Service (Amazon S3)**6.

Amazon S3 is **highly durable and distributed data store**. With a simple web services interface, you can store and retrieve large amounts of data as *object*s in *buckets* (containers) at any time, from anywhere on the web using **standard HTTP verbs**. Copies of objects can be distributed and cached at **14 *edge locations***around the world by creating a *distribution* using **Amazon CloudFront7 service** – a web service for content delivery (static or streaming content). **Amazon SimpleDB*8*** is a web service that provides the core functionality of a database, real-time lookup and simple querying of structured data - without the operational complexity. You can organize the dataset into ***domains***and can run queries across all of the data stored in a particular domain. Domains are collections of *items* that are described by ***attribute-value pairs***.

**Amazon Relational Database Service9 (Amazon RDS)** provides an easy way to setup, operate and scale a relational database in the cloud. You can launch a *DB Instance* and get access to a **full-featured MySQL database** and not worry about common database administration tasks like backups, patch management etc.

**Amazon Simple Queue Service (Amazon SQS)10** is a reliable, highly scalable, hosted distributed queue for storing *messages* as they travel between computers and application components.

**Amazon Simple Notifications Service (Amazon SNS) 11** provides a simple way to notify applications or people from the cloud by creating *Topics* and using a **publish-subscribe protocol.**

**Amazon Elastic MapReduce12** provides a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3) and allows you to create customized ***JobFlows***. JobFlow is a sequence of **MapReduce *steps****.*

**Amazon Virtual Private Cloud (Amazon VPC)13** allows you to extend your corporate network into a private cloud contained within AWS. Amazon VPC uses **IPSec tunnel mode** that enables you to create a secure connection between a gateway in your data center and a gateway in AWS.

Amazon Route53 is a **highly scalable DNS service** that allows you manage your DNS records by creating a *HostedZone* for every domain you would like to manage.

**AWS Identity and Access Management (IAM)14** enable you to create **multiple Users** with **unique security credentials** and manage the permissions for each of these Users within your AWS Account. IAM is natively integrated into AWS Services. No **service APIs** have changed to support IAM, and **exiting applications and tools** built on top of the AWS service APIs will continue to work when using IAM.

AWS also offers various payment and billing services15 that leverages Amazon’s payment infrastructure.

All AWS infrastructure services offer **utility-style pricing** that require no long-term commitments or contracts. For example, you pay by the hour **for Amazon EC2 instance usage** and pay by the **gigabyte for storage** and **data transfer** in the case of Amazon S3. More information about each of these services and their **pay-as-you-go pricing** is available on the AWS Website.

Note that using the AWS cloud doesn’t require sacrificing the flexibility and control you’ve grown accustomed to:

* You are free to use the programming model, language, or operating system (Windows, OpenSolaris or any flavor of Linux) of your choice.
* You are free to pick and choose the AWS products that best satisfy your requirements—you can use any of the services individually or in any combination.
* Because AWS provides resizable (storage, bandwidth and computing) resources, you are free to consume as much or as little and only pay for what you consume.
* You are free to use the **system management tools** you’ve used in the past and extend your datacenter into the cloud.

**Cloud Concepts**

The cloud reinforces some old concepts of building highly scalable Internet architectures [13] and introduces some new concepts that entirely change the way applications are built and deployed. Hence, when you progress from concept to implementation, you might get the feeling that “Everything’s changed, yet nothing’s different.” The cloud changes several processes, patterns, practices, philosophies and reinforces some traditional **service-oriented architectural principles** that you have learnt as they are even more important than before. In this section, you will see some of those **new cloud concepts** and **reiterated SOA concepts**.

Traditional applications were built with some **pre-conceived mindsets** that made **economic and architectural-sense** at the time they were developed. The cloud brings some new philosophies that you need to understand and are discussed below:

**Building Scalable Architectures**

It is critical to build a scalable architecture in order to take advantage of a scalable infrastructure.

The cloud is designed to provide conceptually infinite scalability. However, you cannot leverage all that scalability in infrastructure if your architecture is not scalable. Both have to work together. You will have to identify the **monolithic components** and bottlenecks in your architecture, identify the areas where you cannot leverage the **on-demand provisioning capabilities** in your architecture and work to *refactor* your application in order to leverage the scalable infrastructure and take advantage of the cloud.

Characteristics of a truly scalable application:

 Increasing resources results in a **proportional increase** in performance

 A scalable service is capable of handling heterogeneity

 A scalable service is **operationally efficient**

 A scalable service is resilient

 A scalable service should become **more cost effective** when it grows (Cost per unit reduces as the number of units increases)

These are things that should become an inherent part of your application and if you design your architecture with the above characteristics in mind, then both your **architecture** and **infrastructure** will work together to give you the scalability you are looking for.

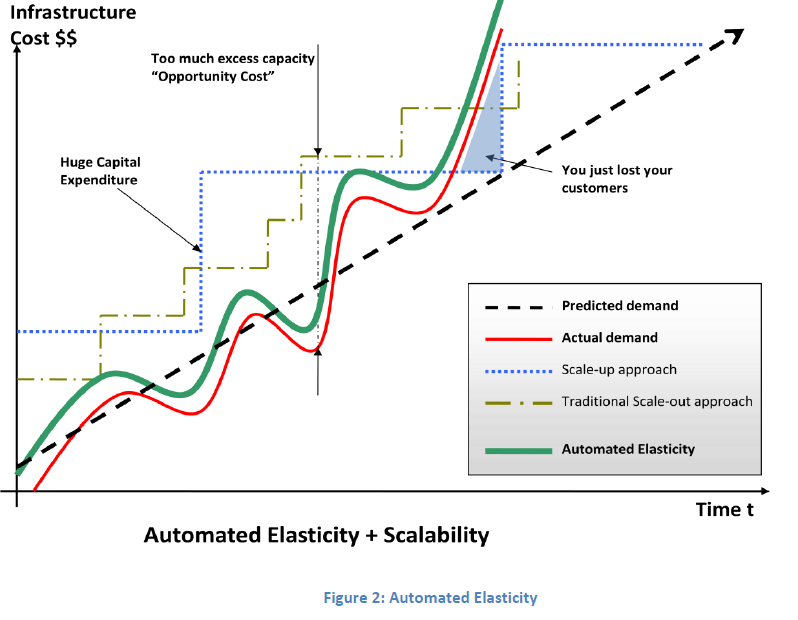
**Understanding Elasticity**

The graph below illustrates the different approaches a cloud architect can take to scale their applications to meet the demand.

***Scale-up approach****:* not worrying about the **scalable application architecture** and investing heavily in larger and more powerful computers (vertical scaling) to accommodate the demand. This approach usually works to a point, but could either cost a fortune (See “Huge capital expenditure” in diagram) or the demand could **out-grow capacity** before the new “big iron” is deployed (See “You just lost your customers” in diagram).

***The traditional scale-out approach:***creating an architecture that scales horizontally and investing in infrastructure in small chunks. Most of the businesses and **large-scale web applications** follow this pattern by distributing their application components, federating their datasets and employing a service-oriented design. This approach is often more effective than a scale up approach. However, this still requires predicting the demand at **regular intervals** and then deploying infrastructure in chunks to meet the demand. This often leads to **excess capacity** (“burning cash”) and **constant manual monitoring** (“burning human cycles”). Moreover, it usually does not work if the application is a victim of a viral fire (often referred to as the Slashdot Effect16).

**Note**: Both approaches have initial start-up costs and both approaches are reactive in nature.



Traditional infrastructure generally necessitates predicting the amount of **computing resources** your application will use over a period of several years. If you under-estimate, your applications will not have the horsepower to handle unexpected traffic, potentially resulting in customer dissatisfaction. If you over-estimate, you’re wasting money with superfluous resources.

The **on-demand and elastic nature** of *the cloud approach* (Automated Elasticity), however, enables the infrastructure to be closely aligned (as it expands and contracts) with the actual demand, thereby increasing **overall utilization** and reducing cost.

*Elasticity is one of the fundamental properties of the cloud.* Elasticity is the power to scale computing resources up and down easily and with minimal friction. It is important to understand that elasticity will ultimately drive most of the benefits of the cloud. As a cloud architect, you need to internalize this concept and work it into your **application architecture** in order to take maximum benefit of the cloud.

Traditionally, applications have been built for fixed, rigid and pre-provisioned infrastructure. Companies never had the need to provision and install servers on **daily basis**. As a result, most software architectures do not address the **rapid deployment** or **reduction of hardware**. Since the **provisioning time** and **upfront investment** for acquiring new resources was too high, software architects never invested time and resources in optimizing for **hardware utilization**. It was acceptable if the hardware on which the application is running was under-utilized. The notion of “elasticity” within an architecture was overlooked because the idea of having new resources **in minutes** was not possible.

With the cloud, this mindset needs to change. Cloud computing streamlines the process of acquiring the necessary resources; there is no longer any need to place orders ahead of time and to hold unused hardware captive. Instead, cloud architects can request what they need **mere minutes before** they need it or automate the procurement process, taking advantage of the vast scale and rapid response time of the cloud. The same is applicable to releasing the unneeded or under-utilized resources when you don’t need them.

If you cannot embrace the change and implement elasticity in your application architecture, you might not be able to take the full advantage of the cloud. As a cloud architect, you should think creatively and think about ways you can implement elasticity in your application. For example, infrastructure that used to run **daily nightly builds** and perform regression and unit tests every night at 2:00 AM for two hours (often termed as the “QA/Build box”) was sitting idle for rest of the day. Now, with elastic infrastructure, one can run nightly builds on boxes that are “alive” and being paid for only for 2 hours in the night. Likewise, an internal trouble ticketing web application that always used to run on **peak capacity** (5 servers 24x7x365) to meet the demand during the day can now be provisioned to run on-demand (5 servers from 9AM to 5 PM and 2 servers for 5 PM to 9 AM) based on the traffic pattern.

Designing **intelligent elastic cloud architectures**, so that infrastructure runs only when you need it, is an **art** in itself. Elasticity should be one of the architectural design requirements or a system property. Question that you need to ask: What components or layers in my application architecture can become elastic? What will it take to make that component *elastic*? What will be the impact of implementing elasticity to my overall system architecture?

In the next section, you will see specific techniques to implement elasticity in your applications. **To effectively leverage the cloud benefits,** it is important to architect with this mindset.

**Not fearing constraints**

When you decide to move your applications to the cloud and try to map your system specifications to those available in the cloud, you will notice that cloud might not have the exact specification of the resource that you have on-premise. For example, “Cloud does not provide X amount of RAM in a server” or “My Database needs to have more IOPS than what I can get in a single instance”.

You should understand that cloud provides ***abstract resources***and they become powerful when you combine them with the **on-demand provisioning model**. You should not be afraid and constrained when using cloud resources because it is important to understand that even if you might not get an **exact replica** of your hardware in the cloud environment, you have the ability to get more of those resources in the cloud to **compensate that need**.

For example, if the cloud does not provide you with exact or greater amount of RAM in a server, try using a **distributed cache** like *memcached*17 or partitioning your data across multiple servers. If your databases need more IOPS and it **does not directly map** to that of the cloud, there are several recommendations that you can choose from depending on your type of data and use case. If it is a read-heavy application, you can distribute the read load across a **fleet of synchronized slaves**. Alternatively, you can use a *sharding* [10] algorithm that routes the data **where it needs to be** or you can use various database clustering solutions.

In retrospect, when you combine the on-demand provisioning capabilities with the flexibility, you will realize that apparent constraints can actually be broken in ways that will actually improve the scalability and overall performance of the system.

**Virtual Administration**

The advent of cloud has changed the role of System Administrator to a “Virtual System Administrator”. This simply means that **daily tasks** performed by these administrators have now become even more interesting as they learn more about applications and decide what’s best for the business as a whole. The System Administrator no longer has a need to provision servers and install software and wire up **network devices** since all of that grunt work is replaced by few clicks and command line calls. The cloud encourages automation because the infrastructure is programmable. System administrators need to move up the **technology stack** and learn how to manage **abstract cloud resources** using scripts.

Likewise, the role of Database Administrator is changed into a “Virtual Database Administrator” in which he/she manages resources through a **web-based console**, executes scripts that add new capacity programmatically in case the database hardware runs out of capacity and automates the day-to-day processes. The virtual DBA has to now learn **new deployment methods** (virtual machine images), embrace new models (query parallelization, geo-redundancy and asynchronous replication [11]), rethink the architectural approach for data (sharding [9], horizontal partitioning [13], federating [14]) and leverage **different storage options** available in the cloud for different types of datasets.

In the traditional enterprise company, application developers may not work closely with the network administrators and network administrators may not have a clue about the application. As a result, several possible optimizations in the network layer and application architecture layer are overlooked. With the cloud, the two roles have merged into one to some extent. When architecting future applications, companies need to encourage more cross-pollination of knowledge between the two roles and understand that they are merging.