Storage Options in the AWS Cloud

Joseph Baron, Amazon Web Services
Robert Schneider, Think88

December 2010

Introduction

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use cloud computing platform that includes a variety of cloud-based data storage options. These alternatives furnish a broad range of options to architects and to developers.

This white paper helps you understand the primary data storage options available with the AWS cloud computing platform. We provide an overview of each storage option, describe ideal usage scenarios, and examine other important cloud-specific characteristics such as elasticity and scalability. We pay particular attention to identifying the level of durability provided by each storage option. This paper concludes with a quick reference table that compares the storage options presented here. Be sure to check out the "References and Further Reading" section for additional resources.

Finally, in a separate companion paper we present several storage use cases that show how to use multiple AWS cloud storage options together. You can employ these use cases as a guide when designing your own storage architecture.

Traditional vs. Cloud-Based Storage Alternatives

Architects of traditional, on-premise IT infrastructure and applications have numerous potential data storage choices, including the following:

- **Memory**—In-memory storage such as file caches, object caches, in-memory databases, and RAM disks provide very rapid access to data.
- **Message Queue**—Temporary durable storage for data sent asynchronously between computer systems or application components.
- **Storage Area Network (SAN)**—Block devices (virtual disk LUNs) on dedicated SANs often provide the highest level of disk performance and durability for both business-critical file data and database storage but they are certainly one of the most expensive systems.
- **Direct Attached Storage (DAS)**— Local hard disk drives or arrays residing in each server provide higher performance than a SAN, but lower durability for temporary and persistent files, database storage, and operating system (OS) boot storage than a SAN.
- **Network Attached Storage (NAS)**—NAS storage provides a file-level interface to storage that can be shared across multiple systems. NAS tends to be slower than either SAN or DAS.
- **Databases**—Structured data is typically maintained and accessed using a relational database such as MySQL, PostgreSQL, Oracle, Microsoft SQL Server, and DB2, or in non-relational database repositories. The database storage volumes typically reside on SAN or DAS devices.
- **Off-line**—Data stored for backup and archival purposes is typically placed on non-disk media such as tape, CDs, DVDs, and so on, which are often stored in remote secure locations for disaster recovery.

Each of these traditional storage options differs in performance, durability, and cost, as well as in their interfaces. Architects consider all these factors when identifying the right storage solution for the task at hand. Notably, most IT

infrastructures and application architectures employ multiple storage technologies in concert, each of which has been selected to satisfy the needs of a particular subclass of data storage. These combinations form a hierarchy of data storage tiers.

As we'll see throughout this paper, AWS offers multiple cloud-based storage options. Each has a unique combination of performance, durability, cost, and interface, and is further enhanced by additional factors such as elasticity, availability, and scalability. These additional factors are critical for web-scale cloud-based solutions. As with traditional on-premise applications, you can use multiple cloud storage options together to form a comprehensive data storage hierarchy.

This paper examines the following AWS cloud storage options:

- **Amazon EC2 Elastic Block Storage (EBS) volumes**
- **Amazon EC2 Local Instance Store (Ephemeral) volumes**
- **Amazon Simple Storage Service (Amazon S3)**
- **Amazon Simple Queue Service (SQS)**
- **Amazon SimpleDB**
- **Amazon EC2 Relational Databases**
- **Amazon Relational Database Service (RDS)**

Let's examine each of the AWS storage options in more detail. For each candidate, we'll present the following information:

- Name and description
- Ideal usage scenario
- Performance
- Durability and availability
- Cost
- Elasticity and scalability
- Interfaces
- Anti-Patterns, situations where other storage options would be a better choice

For additional comparison categories among the AWS storage collection, be sure to review the AWS Storage Quick Reference on page 36.

Amazon Elastic Block Store (EBS) Volumes

Amazon Elastic Block Store (EBS) Volumes provide durable block-level storage for use with Amazon EC2 instances (virtual machines). Amazon EBS volumes are off-instance, network-attached storage that persists independently from the running life of a single Amazon EC2 instance. After an EBS volume has been attached to an Amazon EC2 instance, you are free to interact with it just as you would a physical hard disk drive, typically by formatting it with a file system of your choice. You can use an EBS volume to boot an Amazon EC2 instance (EBS AMIs only), and attach multiple EBS volumes to a single Amazon EC2 instance. Note, however, that any single EBS volume may be attached to only one Amazon EC2 instance at any point in time. An EBS volume cannot be shared with other users, unless you create an EBS snapshot (see the following "Durability and Availability" section). Sizes for EBS volumes range from 1 GB to 1 TB, and are allocated in 1 GB increments.

Ideal Usage Scenario

Amazon EBS is meant for data that changes relatively frequently and requires long-term persistence. EBS provides persistent virtual block mode storage for Amazon EC2 virtual servers, so you can use it just as you would use a hard drive on a physical server. Amazon EBS is particularly well-suited for use as the primary storage for a file system, database or for any applications that require fine granular updates and access to raw, unformatted block-level storage.

Performance

In general, you can expect individual EBS volumes to have performance, mean time to failure (MTTF), and reliability comparable to an externally powered USB drive. Note that while EBS volumes appear as local disk drives, they are actually network-attached to an Amazon EC2 instance. Therefore, other network I/O performed by the instance, as well as the total load on the shared network, can affect individual EBS volume performance.

While each application (and its associated performance) is unique, you are free to design and deploy many traditional disk throughput optimization techniques with EBS volumes. The combination of Amazon EC2 and EBS enables you to use many of the same performance optimization techniques that you use with on-premise servers and storage. For example, you could create several volumes and then attach them all to a single Amazon EC2 instance. With multiple EBS volumes attached you can then partition the total application I/O load by allocating one volume for log data, one volume for the database, and yet another volume for file data. Alternatively, you could stripe your data across multiple EBS volumes using a software RAID 0 device driver, thus aggregating available IOPs, total volume throughput, and total volume size.

Durability and Availability

Each Amazon EBS volume is automatically replicated within the same Availability Zone to prevent data loss due to failure of any single hardware component. Amazon EBS also provides the ability to create point-in-time snapshots of volumes, which are persisted to Amazon S3 (see below). These snapshots can be used as the starting point for new Amazon EBS volumes, and protect data for long-term durability.

The durability of your EBS volume depends on both the size of your volume and the amount of data that has changed since your last snapshot. EBS snapshots are incremental, point-in-time backups, containing only the data blocks changed

since the last snapshot. EBS volumes that operate with 20GB or less of modified data since their most recent snapshot can expect an annual failure rate (AFR) between 0.1% - 0.5%. . In order to maximize both durability and availability of data stored in EBS volumes, users should snapshot their EBS volumes frequently. In the event that your Amazon EBS volume does fail, all snapshots of that volume will remain intact, and will allow you to recreate your volume from the last snapshot point.

Amazon EBS volumes are designed to be highly available. However, because EBS volumes are created in a particular Availability Zone, they will be unavailable if the Availability Zone itself is unavailable. Note that while any single EBS volume is constrained to single Availability Zone, an EBS snapshot of a volume is available across all the Availability Zones within a Region, and you can use an EBS snapshot to create one or more new EBS volumes in any Availability Zone. EBS snapshots can also be shared with other user accounts. This provides an easy-to-use "disk clone" and "disk image" backup and sharing mechanism.

In order to maximize both durability and availability of their EBS data, users should snapshot their EBS volumes frequently.

Cost

As with all Amazon Web Services, with Amazon Elastic Block Store you pay only for what you use, with no minimum fees or long-term contracts. Amazon EBS is priced per GB-month of provisioned storage and per million I/O requests. Volume storage is charged by the amount you allocate until you release it. Amazon EBS Snapshots are priced per GB-month of data stored, as well as per 1,000 PUT requests and per 10,000 GET requests when saving and loading snapshots. For EBS snapshots, you are charged only for storage actually *used (consumed).* Note that EBS snapshots are incremental and compressed, so the storage used in any snapshot is generally much less than the storage consumed on an EBS volume. Full pricing detail is available at http://aws.amazon.com/ec2/pricing/.

Note that there is no charge for transferring information among the various AWS storage offerings (i.e., Amazon EC2 instance with EBS, Amazon S3, Amazon RDS, and so on) as long as they are within the same AWS Region.

Elasticity and Scalability

AWS makes it easy for you to expand your storage space available to your Amazon EC2 Instance. You can create a new EBS volume and attach it to the instance and begin using it together with your existing ones. If you don't want to create and maintain new volumes, here's how to expand the size of a single volume:

1. Quiesce the application or file system.
2. Snapshot your EBS volume's data into Amazon S3 (using "Create Snapshot from Volume").
3. Create a new EBS volume from the snapshot, but specify a larger size than the original volume.
4. Attach the new, larger volume to your Amazon EC2 instance.
5. Detach and delete the original EBS volume.

Interfaces

To create, delete, describe, attach, and detach EBS volumes for your Amazon EC2 instances, Amazon offers control APIs in both SOAP and REST formats. You may use the APIs to create, delete, and describe snapshots from EBS to Amazon S3, as well as their associated attributes. If you prefer to work with graphical tools, the AWS Management Console and the ElasticFox Firefox extension give you all the capabilities of the API in an easy-to-use browser interface. Regardless of how you create your EBS volume, note that all storage is allocated at the time of volume creation, and that you are charged for this storage even if you don't make use of that storage.

EBS presents a block-device interface to the Amazon EC2 instance. That is, to the Amazon EC2 instance, an EBS volume appears just like a local disk drive. To write to and read data from EBS volumes, you therefore use the native file system interfaces of your chosen operating system.

Amazon EBS Anti-Patterns

As described previously, EBS is ideal for information that needs to be persisted beyond the life of a single Amazon EC2 instance. However, in certain situations other AWS storage options may be more appropriate:

- **Temporary storage**—If you are not concerned about your data remaining accessible after your Amazon EC2 instance has been terminated (temporary files, scratch disk, buffers, and so on), consider taking advantage of the storage volume that is automatically provided with most Amazon EC2 instances. These ephemeral volumes are provided at no extra cost beyond the standard cost of the Amazon EC2 instance. For more information, see the next section.

- **Highly-durable storage**—If you need very highly-durable storage, use Amazon S3. Amazon S3 standard storage is designed for 99.999999999% annual durability per object. In contrast, EBS volumes with less than 20GB of modified data since the last snapshot are designed for between 99.5% - 99.9% annual durability; volumes with more modified data can be expected to have proportionally lower durability.

- **Static data or web content**—If your data doesn't change that often, Amazon S3 may represent a more cost-effective and scalable solution for storing this fixed information. Also, web content served out of EBS requires a webserver running on Amazon EC2, while you can deliver web content directly out of Amazon S3.

- **Key-value pair information**—As we'll see in the section "Amazon SimpleDB" on pageAmazon SimpleDB 13, Amazon SimpleDB offers a scalable, elastic means for storing "schema-less" data. If you are attempting to use relational technology (including associated EBS volumes) to maintain key-value information, it's worth exploring SimpleDB as an alternative.

Amazon Elastic Compute Cloud (EC2) Local Instance Store Volumes

Amazon EC2 local instance store volumes (also called ephemeral drives) provide temporary block-level storage for Amazon EC2 instances. When you create an Amazon EC2 instance from an Amazon Machine Image (AMI), in most cases

it comes with a preconfigured block of pre-attached disk storage.[1] Unlike EBS volumes, data on instance store volumes persists only during the life of the associated Amazon EC2 instance. The amount of this disk storage ranges from 160 GB up to 1.7 TB, and varies by Amazon EC2 instance type. Larger Amazon EC2 instances have more and larger instance store volumes. However, even though these amounts appear very generous and can often be useful, this storage is temporary and best used like a scratch volume or RAM disk.

Ideal Usage Scenario

Local instance store volumes are ideal for temporary storage of information that is continually changing, such as buffers, caches, scratch data, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers. Amazon EC2 instance storage is designed for this purpose. It consists of the virtual machine's boot device (for instance store AMIs only), plus one or more additional volumes that are dedicated to the Amazon EC2 instance (for both EBS AMIs and instance store AMIs). This storage is usable only from a single Amazon EC2 instance during its lifetime. Unlike EBS volumes, instance store volumes cannot be detached or attached to another instance.

Performance

Because the Amazon EC2 instance virtual machine and the local instance store volumes are located in the same physical server, interaction with this storage is very fast, particularly for sequential access. To increase I/O operations per second (IOPS) or to improve disk throughput, multiple instance store volumes can be grouped together using RAID 0 (disk striping) software.

Durability and Availability

Amazon EC2 local instance store volumes are not intended to be used as durable disk storage. Data stored on local instance store volumes persists only for the lifetime of the Amazon EC2 instance. Data on Amazon EC2 local instance store volumes is persistent across orderly instance reboots, but not in situations where the Amazon EC2 instance terminates or goes through a failure/restart cycle.

You should not use local instance store volumes for any data that must persist over time, such as permanent file or database storage. However, although local instance store volumes are not persistent, you can persist your data by periodically copying or backing it up into EBS or Amazon S3.

Cost

The cost of the Amazon EC2 instance includes all local instance store (ephemeral) volumes. While there is no charge for data storage on local instance store volumes, note that data transferred to / from Amazon EC2 instance store volumes from outside of an Amazon EC2 Region will incur Data Transfer charges, and, additional charges will apply for use of any persistent storage, such as Amazon S3, EBS volumes, and EBS snapshots. Detailed information on Amazon EC2, EBS, and Data Transfer pricing can be found on the web at http://aws.amazon.com/ec2/pricing/.

---

[1] Most Amazon EC2 instance types provide local instance storage volumes; however, micro instances, such as type t1.micro, provide EBS storage only. Also, instances that use Amazon EBS for the root device ("boot from EBS") do not expose the ephemeral instance store volumes by default. If desired, you can expose the instance store volumes at instance launch time by specifying a Block Device Mapping. See the Amazon EC2 User Guide for details.

Elasticity and Scalability

Local instance store volumes are fixed in size for a given Amazon EC2 instance type, and tied to a particular instance, so this type of storage is relatively inelastic. However, you can achieve full storage elasticity by including one of the other suitable storage options such as Amazon S3 or Elastic Block Storage (EBS) in your Amazon EC2 storage strategy.

Interfaces

The Amazon EC2 instance works with local instance store volumes as if they are local disk drives. This means that you can interact with data hosted on your local instance storage using native file system mechanisms, such as Windows NTFS or Linux XFS. Note that in some cases, a local instance store volume device will be attached to the Amazon EC2 instance upon launch, but must be formatted with an appropriate file system and mounted before use.

Amazon EC2 Local Instance Store Volume Anti-Patterns

Amazon EC2 local instance store volumes are fast, free (that is, included in the price of the Amazon EC2 instance) "scratch volumes" best suited for storing temporary data that can easily be regenerated. In many situations, however, other AWS storage options may be more appropriate:

- **Persistent storage**—If you need persistent virtual disk storage similar to a physical disk drive for files or other data that must persist longer than the lifetime of a single Amazon EC2 instance, EBS volumes or Amazon S3 are more appropriate.

- **Database storage**—In most cases, databases require storage that persists beyond the lifetime of a single Amazon EC2 instance, making EBS volumes the natural choice.

- **Shared storage**—Instance store volumes are dedicated to a single Amazon EC2 instance, and cannot be shared with other systems or users. If you need storage that can be detached from one instance and attached to a different instance, or if you need the ability to share data easily, Amazon S3 or EBS volumes are the better choice.

- **Snapshots**—If you need the convenience, long-term durability, availability, and shareability of point-in-time disk snapshots, EBS volumes are a better choice.

Amazon Simple Storage Service (Amazon S3)

Amazon S3 is a highly scalable, durable and available distributed object store designed for mission-critical and primary data storage with an easy to use web service interface. In traditional on-premise applications, this type of data would ordinarily be maintained on SAN or NAS. However, a cloud-based mechanism such as Amazon S3 is far more agile, flexible, and geo-redundant. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from within Amazon EC2 or from anywhere on the web. You can write, read, and delete objects containing from 1 byte to 5 terabytes of data each, and the number of objects you can store in an Amazon S3 bucket is unlimited. Amazon S3 is also highly scalable, allowing concurrent read or write access to Amazon S3 data by many separate clients or application threads.

Ideal Usage Scenario

One very common use for Amazon S3 is storage of static web content. This content can be delivered directly out of Amazon S3 via a web server, since each object in Amazon S3 has a unique HTTP URL address, or delivered through a Content Delivery Network (CDN), such as Amazon CloudFront. Because of Amazon S3's elasticity, it works particularly well for hosting web content with extremely 'spiky' bandwidth demands. Also, because there is no storage provisioning, Amazon S3 works well for fast growing websites hosing data intensive, user-generated content such as video and photo sharing.

Amazon S3 is also commonly used as a data store for large-scale computation such as analyzing financial transactions or clickstream data and media transcoding. Because of the horizontal scalability of Amazon S3, you can access your data from multiple computing nodes concurrently without being constrained by a single connection.

Finally, Amazon S3 is very often used as the origin store for mission critical data, highly reliable snapshot storage for Amazon EBS Volumes, backup storage applications and 'hot' disaster recovery solutions for business continuity. Because Amazon S3 stores objects redundantly on multiple devices across multiple facilities, it provides the highly-durable storage infrastructure needed for these scenarios.

Performance

Access to Amazon S3 from within Amazon's Elastic Compute Cloud (Amazon EC2) in the same region is fast. If you access Amazon S3 using multiple threads, multiple applications, or multiple clients concurrently, total Amazon S3 aggregate throughput will typically scale to rates that far exceed what any single server can generate or consume.

To speed access to relevant data, many developers pair Amazon S3 with Amazon SimpleDB. Amazon S3 stores the actual information, and SimpleDB serves as the repository for associated metadata (e.g., object name, size, keywords, and so on). SimpleDB provides automatic indexing, making it very efficient to locate an object's reference via a metadata search. This result can then be used to pinpoint and then retrieve the object itself from Amazon S3.

Durability and Availability

By automatically and synchronously storing your data across both multiple devices and multiple facilities within your selected geographical Region, Amazon S3 storage provides the highest level of data durability and availability in the AWS platform. Error correction is built-in, and there are no single points of failure. Amazon S3 is designed to sustain the concurrent loss of data in two facilities, making it very well-suited to serve as the primary data storage for mission-critical data. In fact, Amazon S3 is designed for 99.999999999% ("11 nines") durability per object and 99.99% availability over a one-year period. In addition to its built-in redundancy, Amazon S3 data can also be protected from application failures and unintended deletions through the use of Amazon S3 versioning. You can also enable Amazon S3 versioning with MFA Delete. With this option enabled on a bucket, two forms of authentication are required to delete a version of an Amazon S3 object: valid AWS account credentials plus a six-digit code (a single-use, time-based password) from a physical token device.

For noncritical data that can be reproduced easily if needed, such as transcoded media, image thumbnails, you can use the Reduced Redundancy Storage (RRS) option in Amazon S3, which provides a lower level of durability at a lower storage cost. Objects stored using the RRS option have less redundancy than objects stored using standard Amazon S3 storage. In either case, your data is still stored on multiple devices in multiple locations. RRS is designed to provide

99.99% durability per object over a given year. While RRS is less durable than standard Amazon S3, it is still 400 times more durable than a typical disk drive.

Cost

As with all Amazon Web Services, with Amazon S3 you pay only for what you use, with no minimum fees or long-term contracts. Amazon S3 has three pricing components: storage (per GB per month), data transfer in or out (per GB per month), and requests (per n thousand requests per month). For new customers, AWS provides a free usage tier which includes up to 5GB of Amazon S3 storage. Full Amazon S3 pricing information can be found at http://aws.amazon.com/s3/pricing/.

Elasticity and Scalability

Amazon S3 has been designed to offer a very high level of elasticity and scalability automatically. Unlike a typical file system that encounters issues when storing large number of files in a directory, Amazon S3 supports an unlimited number of files in any bucket. Also, unlike a disk drive that has a limit on the total amount of data that can be stored before you must partition the data across drives and/or servers, an Amazon S3 bucket can store an unlimited number of bytes. You are able to store any number of objects, and Amazon S3 will manage scaling and distributing redundant copies of your information onto other servers in other locations in the same Region, all using Amazon's high-performance infrastructure.

Interfaces

Amazon S3 provides both SOAP and RESTful web-services APIs. These APIs allow Amazon S3 objects (files) to be stored in uniquely-named buckets (top-level folders). Each object must have a unique object key (file name) that serves as an identifier for the object within that bucket. While Amazon S3 is a web-based object store rather than a traditional file system, you can easily emulate a file system hierarchy (folder1/folder2/file) in Amazon S3 by creating object key names that correspond to the full path name of each file.

Most developers building applications on Amazon S3 use a higher-level toolkit. AWS and third parties have created Amazon S3 interface toolkits and libraries for several popular software development languages and platforms, such as:

- **Java**, using the AWS SDK for Java (http://aws.amazon.com/sdkforjava/) and AWS Toolkit for Eclipse (http://aws.amazon.com/eclipse/).
- **C#**, using the AWS SDK for .NET (http://aws.amazon.com/sdkfornet/) and Visual Studio templates.
- **PHP**, using the base installation of PHP5, or the AWS SDK for PHP (http://aws.amazon.com/sdkforphp/).
- **Perl**, using the Digest::SHA1, Bundle::LWP, and XML::Simple modules, all of which can be downloaded from the Comprehensive Perl Archive Network at http://www.cpan.org.

While Amazon S3 is designed to be accessed primarily via APIs, the AWS Management Console and third-party tools also provide graphical and command-line interfaces for interacting with your Amazon S3 buckets and objects. Amazon S3 also supports the BitTorrent protocol, which lets consumers concurrently retrieve information from S3 as well as other providers.

To upload or download large volumes of data you can also use the AWS Import/Export service (http://aws.amazon.com/importexport/). AWs Import/Export accelerates moving large amounts of data into and out of AWS using portable storage devices for transport. Amazon will mount your device and copy the data to or from a designated Amazon S3 bucket. This approach is often faster and cheaper than transferring massive amounts of data over the Internet.

Note that RRS uses the same interfaces as standard Amazon S3; the RRS option is specified for an Amazon S3 object or bucket by setting the Storage Class property during a PUT operation, or by selecting the "Use Reduced Redundancy Storage" setting in the AWS Management Console.

Amazon S3 Anti-Patterns

Amazon S3 is optimal for storing numerous classes of information that are relatively static and benefit from its durability, availability, and elasticity features. However, in a number of situations Amazon S3 is not the optimal solution:

- **File system**—Amazon S3 uses a flat namespace and isn't meant to serve as a standalone, POSIX-compliant filesystem. However, by using delimiters (commonly either the '/' or '\' character) you are able construct your keys to emulate the hierarchical folder structure of filesystem within a given bucket.

- **Structured data with query**—Amazon S3 doesn't offer query capabilities: to retrieve a specific object you need to already know the bucket name and key. Thus, you can't use Amazon S3 as a database by itself. A common usage pattern is to place an object in Amazon S3 and then use Amazon SimpleDB to hold the object's metadata. You can also use other database technologies such as Amazon RDS or a database running on an Amazon EC2 instance. You can then search the database to locate the object's bucket name and key, and then retrieve the object itself from Amazon S3 bucket.

- **Rapidly**-changing data—Data that must be updated very frequently might be better served by a storage solution with lower read / write latencies, such as EBS or a database.

Amazon Simple Queue Service

Amazon Simple Queue Service (Amazon SQS) provides a reliable, highly scalable, hosted message queuing service for temporary storage and delivery of short (up to 64 kB) text-based data messages. An Amazon SQS queue is a temporary data repository for messages that are waiting for processing (typically a message produced by one application component and waiting to be consumed by another). Amazon SQS messages can be sent and received by servers or distributed application components within the Amazon EC2 environment or anywhere on the Internet. Amazon SQS supports an unlimited number of queues, and supports unordered, at-least-once delivery of messages.

While Amazon SQS and other message queuing services are usually thought of as an asynchronous communication protocols, Amazon SQS can also be viewed as a store for providing temporary but durable data storage for many classes of applications. Use of Amazon SQS as temporary storage can minimize the use of other storage mechanisms, such as temporary disk files.

Ideal Usage Scenario

Amazon SQS is ideally suited to any scenario where multiple application components must communicate and coordinate their work in a loosely coupled manner. This occurs particularly in producer-consumer scenarios where some components may work faster or slower than others, or where the number of interacting components changes with time or load. Amazon SQS can serve as the "software glue" that enables components to communicate reliably without being tightly coupled or highly dependent upon synchronous operation, or on a fixed number of components.

A classic use of Amazon SQS is to coordinate a multi-step processing pipeline, where each message is associated with a task that must be processed. Each task is described by an Amazon SQS message indicating the task to be done and a pointer to the task data in Amazon S3.

To illustrate, suppose you have a number of image files to encode. In an SNS worker queue, you create an SNS message for each file specifying the command (jpeg-encode) and the location of the file in Amazon S3. A pool of Amazon EC2 instances running the needed image processing software does the following:

1. Asynchronously pull the task messages from the queue
2. Retrieve the named file
3. Process the conversion
4. Write the image back to Amazon S3
5. Write a "task complete" message to another queue
6. Delete the original task message
7. Check for more messages in the worker queue

Use of the Amazon SQS queue enables the number of worker instances to scale up or down, and also enable the processing power of each single worker instance to scale up or down, to suit the total workload, without any application changes.

Performance

Amazon SQS is a distributed queuing system that is optimized for horizontal scalability, not for single-threaded sending or receiving speeds. A single client can send or receive Amazon SQS messages at a rate of about 5 to 50 messages per second. Higher receive performance can be achieved by requesting multiple messages (up to 10) in a single call. It may take several seconds before a message that has been to a queue is available to be received.

Durability and Availability

By design, messages in Amazon SQS are highly durable but temporary. To prevent messages from being lost or becoming unavailable, all messages are stored redundantly across multiple servers and data centers. Message retention time is

configurable on a per-queue basis, from a minimum of one hour to a maximum of 14 days. Messages are retained in a queue until they are explicitly deleted, or until they are automatically deleted upon expiration of the retention time.

Cost

As with all Amazon Web Services, with Amazon SQS you pay only for what you use, with no minimum fees or long-term contracts. In order to get started and to support simple applications, Amazon SQS provides a free tier of service which provides 100,000 requests per month at no charge. Beyond the free tier, Amazon SQS pricing is based on number of requests (priced per 10,000 requests) and the amount of data transferred in and out (priced per GB per month). Full Amazon SQS pricing details can be found at http://aws.amazon.com/sqs/pricing/.

Elasticity and Scalability

Amazon SQS is both highly elastic and massively scalable. It is designed to enable an unlimited number of computers to read and write an unlimited number of messages at any time. It supports an unlimited number of queues and an unlimited number of messages per queue for any user.

Interfaces

Amazon SQS can be accessed through SOAP and Query (HTTP) interfaces. Five APIs make it easy for developers to get started with SQS: CreateQueue, SendMessage, ReceiveMessage, ChangeMessageVisibility, and DeleteMessage. Additional APIs provide advanced functionality. In all cases, the SOAP and query APIs can be used with Java, C#, Perl, and PHP.

Amazon SQS Anti-Patterns

- **Binary or large data**—Amazon SQS messages must be text, and can be a maximum of 64 KB in length. If the data you need to store in a queue exceeds this length, or is binary, it is best to use Amazon S3 or RDS to store the large or binary data, and store a pointer to the data in Amazon SQS.

- **Long-term storage**—If message data must be stored for longer than 14 days, Amazon S3 or some other storage mechanism is more appropriate.

Amazon SimpleDB

Amazon SimpleDB represents a fresh approach to storing and managing structured data in the cloud, which differs from the traditional technique of deploying relational database servers. SimpleDB is a highly available, scalable, and flexible non-relational data store that offloads much of the work of database administration and associated systems management. With Amazon SimpleDB, you have a “schema-less” data model, and you can store data items comprised of a flexible number of name/value pairs.

Ideal Usage Scenario

This storage alternative is well-suited for situations where you have structured, fine-grained data that you need to persist and then query (primarily read-oriented interactions), with high availability and durability, but for which you don't want or need to incur the overhead of administering a full-blown relational database.

Many organizations use SimpleDB as part of a larger cloud-based storage architecture. A common pattern is to use SimpleDB to keep track of metadata about information that is stored in other AWS offerings. For example, suppose that you are building a cloud-based solution that will store millions of multi-megabyte images. In this situation, you can use either Amazon Elastic Block Storage (EBS) or Amazon Simple Storage Service (Amazon S3) to hold the images themselves, with SimpleDB serving as the repository for details about each image. You may then issue queries to SimpleDB to locate a particular image, and use this information to retrieve your results from the other Amazon storage offerings.

Performance

By leveraging Amazon's proven, high-speed infrastructure, the SimpleDB approach creates a fast, highly available database that is very scalable. For highest throughput and lowest latency, Amazon SimpleDB by default provides eventually consistent reads. For applications needing "read my last write", a consistent read is also provided.

Durability and Availability

SimpleDB provides very high data durability through fully automatic, geo-redundant replication. Multiple replicas of every SimpleDB data item are stored in different locations within your selected geographic Region.

Cost

As with all Amazon Web Services, with Amazon SimpleDB you pay only for what you use, with no up-front fees or long-term contracts. Costs for SimpleDB include structured data storage (per GB-month), data transfer (per GB-month), and machine hours (per month) associated with PUT and GET operations. The first 1 GB-month of structured data storage and the first 25 machine hours in a month (roughly 2M GET or SELECT API requests) are free. By using this free tier, many applications will never incur any SimpleDB charges.

Full information on Amazon SimpleDB pricing can be found at http://aws.amazon.com/simpledb/pricing/.

Elasticity and Scalability

When you create a SimpleDB domain, AWS allocates a maximum of 10 GB of storage per domain. Because you pay only for compute and storage resources as you utilize them, you are able to transparently scale down to zero as well as up. Storage and indexing requirements are handled by Amazon SimpleDB, which stores your automatically-indexed data redundantly across multiple locations within the Region you selected when you created your SimpleDB domain. This removes database administration and allows you to simply PUT and GET data. For horizontal scalability to increase throughput or data set size beyond the limitations of a single domain, you can create multiple SimpleDB domains, and execute multiple operations in parallel. For datasets greater than 10GB, you can consider using partitioning your data across multiple SimpleDB domains.

Interfaces

You have a number of different techniques at your disposal for programmatic interaction with SimpleDB. Amazon offers a web service API, available in both SOAP and RESTful styles. The API currently offers nine operations, enabling domain management (creating, listing, deleting, and obtaining metadata) and working with attributes (getting, writing—individually or in batch, and deleting). While standard SQL isn't available for SimpleDB, you may use the SimpleDB Select operation to create SQL-like queries that retrieve a set of attributes based on criteria that you provide.

If you prefer to use a software development kit, toolkits and libraries for popular software development languages and platforms are available, such as:

- **Java**, using the AWS SDK for Java (http://aws.amazon.com/sdkforjava/) and AWS Toolkit for Eclipse (http://aws.amazon.com/eclipse/).
- **C#**, using the AWS SDK for .NET (http://aws.amazon.com/sdkfornet/) and Visual Studio templates
- **PHP**, using the base installation of PHP5, or the AWS SDK for PHP (http://aws.amazon.com/sdkforphp/).
- **Perl**, using the Digest::SHA1, Bundle::LWP, and XML::Simple modules, all of which can be downloaded from Comprehensive Perl Archive Network at http://www.cpan.org.

Amazon SimpleDB Anti-Patterns

It's easy to determine which usage scenarios are not appropriate for SimpleDB. If any of the following criteria apply, consider using one of the other AWS storage options:

- **Prewritten application tied to a traditional relational database**—If you are attempting to port an existing application to the AWS cloud, and need to continue using a relational database, you may elect to use either Amazon RDS (if your database is MySQL), or one of the many preconfigured Amazon EC2 database AMIs. You are also free to create your own Amazon EC2 instance, and install a database engine on it.
- **Joins and/or complex transactions**—While many solutions are able to leverage SimpleDB to support their users, it's possible that your application may require joins, complex transactions, and other relational infrastructure provided by traditional database platforms. If this is the case, you may want to explore Amazon RDS or Amazon EC2 with an installed database.
- **BLOb (Binary Large Objects) data**—If you plan on storing binary (e.g., video, pictures, or music), you'll want to consider either Amazon Elastic Block Storage (EBS) or Amazon Simple Storage Service (Amazon S3). However, SimpleDB still has a role to play in this scenario, for keeping track of metadata (e.g., item name, size, date created, owner, location, and so on) about your binary objects.
- **Typed (numeric) data**—SimpleDB stores all data as text strings, so if you need to manipulate typed or numeric data, RDS or a traditional database on Amazon EC2 may be a better solution.
- **Large amounts of data**—As stated earlier, SimpleDB offers a compelling pricing and storage model for many applications. However, if you are faced with maintaining very large data sets, you should also consider storage

alternatives such as Amazon S3 or EBS. Once again, you are still able to employ SimpleDB as an efficient mechanism for keeping track of metadata about your large data sets.

Amazon Relational Database Service (Amazon RDS)

Amazon Relational Database Service (RDS) is a fully functional, MySQL relational database provided as a managed, cloud-based service. If your application requires relational storage, but you want to reduce the time you spend on database management, Amazon RDS automates common administrative tasks to reduce the complexity and total cost of ownership. Amazon RDS automatically backs up your database and maintains your database software, allowing you to spend more time on application development. It's optimal for traditional structured data that requires more sophisticated querying and joining capabilities than that provided by Amazon's other database offering, SimpleDB.

Ideal Usage Scenario

Amazon RDS is a great choice for any application that relies on MySQL as its information repository and you want to take advantage of a highly scalable, low-maintenance, cost-effective, cloud-based database without needing to make any code changes.

Performance

Amazon RDS delivers high performance through a combination of configurable instances running on Amazon's proven, world-class infrastructure with fully automated maintenance and backup operations. Available database configurations range from a small instance (64-bit platform with 1.7 GB of RAM and 1 Elastic Computing Unit (ECU)) up to a quadruple extra-large instance (64-bit platform with 68 GB of RAM and 26 ECUs).

To achieve optimal performance, it's incumbent upon database designers and administrators to select the proper instance profile (including RAM and storage) for their particular computing need. If tuning is necessary, they are free to use the API provided by Amazon to adjust their database settings. Amazon CloudWatch provides metrics that can help determine when performance tuning and/or additional elasticity is needed. Finally, administrators may opt to move to a larger database configuration should demand outstrip their current instance's resources.

Elasticity and Scalability

RDS resources can be scaled elastically in several dimensions: database storage size, database instance compute capacity, and the number of Read Replicas.

To achieve additional compute elasticity, you can configure additional Amazon RDS instances and leverage partitioning to spread the workload.

To scale the RDS database storage elastically, you can use the command-line tools, API, or AWS Management Console to request additional storage. Depending on your needs, this added storage can be added either immediately or during the next maintenance window.

Scaling computational resources up or down is easily performed with a single API or AWS Management Console command. For example, you might need additional compute power to create invoices at the end of each month. It's

simple to temporarily scale up to a quadruple extra-large instance, run the computationally intensive workloads, and then return to a smaller, more cost-effective configuration for the remainder of the month.

To scale RDS database resources in or out (horizontal scaling using multiple RDS instances), administrators can create one or more RDS read replicas. Read replicas use MySQL's built-in asynchronous replication capability to allow you elastically scale RDS resources to support read heavy workloads.

Finally, administrators may configure additional Amazon RDS instances and leverage database partitioning or sharding to spread the workload, achieving even greater database elasticity and scalability.

Durability and Availability

. For enhanced durability, RDS offers two types of database backups that are replicated across multiple Availability Zones: automated DB Instance backups and user-initiated database snapshots. If you enable automated DB instance backups, RDS will automatically perform a full daily backup of your data during the specified backup window, and will also capture DB transaction logs. These automated backups are provided at no additional charge, can be retained for up to eight days, and can be used to do a point-in-time restore to any point from the start of the retention period to about the last five minutes from current time. DB Snapshots are user-initiated, can be created at any time, and are kept until explicitly deleted. DB Snapshots allow you to restore your database to a known state.

The RDS Multi-AZ deployment feature enhances both the durability and the availability of your database by synchronously replicating your data between a primary RDS DB Instance and a standby instance in another Availability Zone. In the unlikely event of a DB component failure or an Availability Zone failure, RDS will automatically failover to the standby (which typically takes about three minutes) and the database transactions can be resumed as soon as the standby is promoted. The synchronous replication ensures that there is no loss of data.

Note that the synchronous replication provided by RDS Multi-AZ deployment feature is complementary to the built-in asynchronous replication provided by RDS read replicas. You can use either feature alone, or both in combination.

Cost

As with all Amazon Web Services, with Amazon RDS you pay only for what you use, with no minimum fees or long-term contracts. Amazon RDS offers a tiered pricing structure, based on the size of the database instance, the deployment type (Single-AZ/Multi-AZ), and the AWS Region. Pricing for Amazon RDS is based on several factors: the DB instance hours (per hour), the amount of provisioned database storage (per GB-month and per million IO requests), additional backup storage (per GB-month), and Data Transfer in / out (per GB per month). Full information on RDS pricing can be found at http://aws.amazon.com/rds/pricing/.

Interfaces

To help you get started, Amazon offers a series of easy-to-understand command line scripts for creating, maintaining, monitoring, and halting your database instance. Alternatively, you may use Amazon's web service–based operations to accomplish the same series of tasks.

Once your instance has been created, you're free to set up your schema and data using any tool that can work with a relational database. Depending on the amount and location of your existing onsite data, you may instead opt to extract

your local data using the mysqldump utility and pipe it directly to the MySQL executable for insertion into Amazon RDS. For larger data sets, it might be more advantageous to first construct your schema in Amazon RDS, extract your data locally into a flat file, and then use the mysqlimport utility to upload it into Amazon RDS. The topic of migrating to Amazon RDS will be discussed in more detail in an upcoming white paper.

After your schema and data are in place, you interact with your information via standard MySQL SQL, as well as JDBC and other popular APIs, as well as any graphical tools that can work with relational data. There are no code changes to be made to let your application interact with RDS: you simply replace your database server's address (e.g. dbserver.yourcompany.com) with the public DNS endpoint (e.g. myinstance.c0cafggtpzd2.us-east-1.rds.amazonaws.com) provided by AWS when you create the instance. This DNS endpoint will remain the same for the lifetime of your instance. Aside from configuring the endpoint, everything else about your MySQL-based application is unchanged.

Amazon RDS Anti-Patterns

Amazon RDS is a great solution for cloud-based MySQL data, but in a number of scenarios it is not the right choice:

- **Index-and-query focused-data**—Many cloud-based solutions don't require advanced features found in a relational database, such as joins and complex transactions. If your application is more oriented toward indexing and querying data, you may find Amazon SimpleDB to be more appropriate for your needs.

- **Numerous binary large objects (BLObs)**—While MySQL (and thus Amazon RDS) is equipped to store BLObs, if your application makes heavy use of them (audio files, videos, images, and so on), you may find Amazon Simple Storage Service (Amazon S3) to be a better choice.

- **Automatic elasticity**—As stated previously, it's up to administrators to configure their Amazon RDS environment to achieve elasticity. If you want automated elasticity (and your data structures are a good fit), you may opt for another storage choice such as Amazon SimpleDB or Amazon S3.

- **Other database platforms**—At this time, Amazon RDS provides a MySQL database. If you need another database platform (such as IBM DB2 or Informix, Microsoft SQLServer, Oracle, PostgreSQL, or Sybase) you need to deploy a specialized Amazon EC2 relational database AMI, or create one of your own.

Relational Database on Amazon EC2 / Relational Database AMIs

Amazon EC2, together with EBS volumes, provides an ideal platform for you to operate your own relational database in the cloud. Many leading database solutions are available as prebuilt, ready-to-use Amazon EC2 AMIs, including IBM DB2 and Informix, Oracle Database, MySQL, Microsoft SQL Server, PostgreSQL, Sybase, EnterpriseDB, and Vertica.

Ideal Usage Scenario

Running a relational database on Amazon EC2 and EBS is the ideal scenario for users whose application requires a specific traditional relational database, or for those users who require a maximum level of administrative control and configurability.

Performance

The performance of a relational database instance on Amazon EC2 depends on many factors, including the Amazon EC2 instance type, the number and configuration of EBS volumes, the database software and its configuration, and the application workload. In general, you can expect database performance on Amazon EC2 to be similar to the performance of the same database installed on similarly configured on-premise equipment. We encourage you to benchmark your actual application on several Amazon EC2 instance types using several storage configurations in order to select the best configuration.

To increase database performance you can scale up memory and compute resources by choosing a larger Amazon EC2 instance size. To scale up I/O performance, you can change the number of EBS volumes, or use software RAID 0 (disk striping) across multiple EBS volumes, which will increase total IOPS and bandwidth. In many cases, you can also scale the total database system performance by scaling horizontally with database clustering, replication, and multiple read slaves. In general, you have the same database performance tuning options in the Amazon EC2 environment that you have in a physical server environment.

Durability and Availability

Relational databases on Amazon EC2 provide persistent storage for structured data using EBS volumes as the data store, so all the notes about the durability and availability of EBS data apply here as well. And, again, the basic durability and availability of relational data stored on EBS volumes can be further enhanced by using EBS snapshots, or by using third-party database backup utilities (such as Oracle's RMAN) to store database backups in Amazon S3.

Cost

By running a database on Amazon EC2, you pay only for what you use, with no minimum fees or long-term contracts. The cost of running your own database on Amazon EC2 depends on the size and number of Amazon EC2 instances used to run your database, the size of the EBS volumes used for database storage, the amount of data transferred in and out of Amazon EC2, and, in many cases, the license cost of the third-party database software. Many open-source database packages use a no-cost license model; some commercial software vendors use the Amazon DevPay model; many others provide a bring-your-own-license model. Contact your database software vendor or Amazon Web Services to understand the license cost pricing model that applies.

Detailed information on Amazon EC2, EBS, and Data Transfer pricing can be found on the web at http://aws.amazon.com/ec2/pricing/.

Elasticity and Scalability

In many cases, users of traditional relational database solutions on Amazon EC2 can take advantage of the elasticity and scalability of the underlying AWS platform. For example, after you configure an Amazon EC2 instance with your database solution, you can bundle the instance into a custom AMI, either by using the Bundle commands for instance store AMIs, or by using the Create Image command for EBS AMIs. You can then create multiple new instances of your database configuration in a few moments.

Relational Database on Amazon EC2 Anti-Patterns

Running your own relational database on Amazon EC2 is a great solution for many users, but a number of scenarios exist where other solutions might be the better choice:

- **Index-and-query focused data**—Many cloud-based solutions don't require advanced features found in a relational database, such as joins or complex transactions. If your application is more oriented toward indexing and querying data, you may find Amazon SimpleDB to be more appropriate for your needs, and significantly easier to manage.

- **Numerous binary large objects (BLObs)**—Many relational databases support BLObs (audio files, videos, images, and so on). If your application makes heavy use of them, you may find Amazon S3 to be a better choice, using a relational database or SimpleDB for managing metadata.

- **Automatic elasticity**—As stated previously, users of relational databases on AWS can in many cases leverage the elasticity and scalability of the underlying AWS platform, but this requires system administrators or DBAs to perform a manual or scripted task. Assuming that you need or want fully automated elasticity (and that your data structures are a good fit), you may opt for another storage choice such as Amazon SimpleDB or Amazon S3.

Cloud Storage Use Cases

For illustrations of real-world usage of AWS storage options, please see the companion whitepaper: Storage Options in the AWS Cloud: Use Cases at: http://media.amazonwebservices.com/AWS_Storage_Use_Cases.pdf.

References and Further Reading

AWS Storage Services

Amazon Elastic Block Store (EBS) – http://aws.amazon.com/ebs
Amazon EC2 Instance Store Volumes – http://docs.amazonwebservices.com/AWSAmazon EC2/latest/UserGuide
(see sections on Instance Types, Instance Storage, and Block Device Mapping)
Amazon Simple Storage Service (Amazon S3) – http://aws.amazon.com/s3
Amazon Simple Queue Service (Amazon SQS) – http://aws.amazon.com/sqs
Amazon SimpleDB – http://aws.amazon.com/simpledb
Running Databases on AWS – http://aws.amazon.com/running_databases
Amazon Relational Database Service (Amazon RDS) – http://aws.amazon.com/rds

AWS Storage Articles

Werner Vogel's article, "Choosing Consistency" –
http://www.allthingsdistributed.com/2010/02/strong_consistency_simpledb.html

Amazon Web Services Blog article, "Amazon SimpleDB Consistency Enhancements" – http://aws.typepad.com/aws/2010/02/amazon-simpledb-consistency-enhancements.html
Amazon Web Services Developer Center article, "Amazon SimpleDB Consistency Enhancements" – http://developer.amazonwebservices.com/connect/entry.jspa?externalID=3572&categoryID=152
Amazon Web Services Simple Monthly Calculator – http://aws.amazon.com/calculator
Oracle in the Cloud FAQ – http://www.oracle.com/technetwork/topics/cloud/faq-098970.html
Amazon Web Services Developer Center article, "Running Databases on AWS" – http://aws.amazon.com/running_databases/
Oracle Case Study: Oracle Database in the Cloud – http://www.oracle.com/technetwork/database/features/availability/311356-129177.pdf

Other AWS Information

AWS Free Usage Tier – http://aws.amazon.com/free
Public Data Sets on AWS – http://aws.amazon.com/publicdatasets
AWS Import/Export – http://aws.amazon.com/importexport
Amazon CloudFront – http://aws.amazon.com/cloudfront
AWS Case Studies – http://aws.amazon.com/solutions/case-studies

Appendix: AWS Storage Quick Reference

	Unstructured Data			Structured Data		
	Amazon EC2 Instance Storage	**Amazon EBS Volumes**	**Amazon S3**	**Amazon SimpleDB**	**Other Relational DB (on EC2 and EBS)**	**Amazon RDS**
Performance	High	High	Moderate (single thread) to Very High (multiple threads)	Moderate to High (batched Puts / Gets)	High	High
Durability	Low	Moderate	High	High	High	Moderate
Cost	Included in EC2 cost	Provisioned per GB/Month	Stored per GB/Month	Provisioned First GB free, then per GB/Month	Provisioned (same as EBS)	Provisioned per GB/Month (5 GB minimum)
Availability	Low	Moderate to High (using EBS snapshots)	High	High	Moderate to High	High
Elasticity / Scalability	No	Manual (adding more volumes)	Automatic	Automatic	Manual	Manual (one command to modify DB Instance)
Size Limits	160 GB to 1.6 TB (larger instances have both larger volumes and more volumes)	1 GB to 1 TB per volume (can use multiple volumes or striping for larger capacities)	Effectively Unlimited (5 TB per object, unlimited objects per bucket)	10 GB/domain 100 domains (more domains available upon request)	(same as EBS)	5 GB to 1 TB per DB Instance
Persistence Across Instantiations	No	Yes	Yes	Yes	Yes	Yes
Interfaces	Block Device, access via OS / file system on EC2	N/A, access through EC2 OS / file system	HTTP, REST or SOAP	REST or SOAP	MySQL or JDBC libraries	MySQL or JDBC libraries
Security (encryption at-rest)	Run Encrypted FS	Run Encrypted FS	Encrypt using 256-bit AES	Encrypt using 256-bit AES		
Security (encryption in-transit)	N/A	N/A	SSL (HTTPS)	SSL (HTTPS)	SSL (HTTPS)	SSL (HTTPS)
RDBMS Platforms Supported	MySQL, SQL Server, Oracle, DB2, etc.	MySQL, SQL Server, Oracle, DB2, etc.	N/A	N/A	MySQL, SQL Server, Oracle, DB2 etc.	MySQL 5.1
Model (relational or otherwise)	Block	Block	Object	Non-relational, flexible schema, entity store	Relational	Relational
Degree of Automation	None	Auto-mirroring	Auto-replication, Versioning	Indexing, replication, provisioning, patching	Depends on DB	Automated backups, software

	Unstructured Data			Structured Data		
	Amazon EC2 Instance Storage	**Amazon EBS Volumes**	**Amazon S3**	**Amazon SimpleDB**	**Other Relational DB (on EC2 and EBS)**	**Amazon RDS**
Degree of Redundancy	Not redundant	Redundant within an Availability Zone	Highly redundant across multiple data centers	Maintain multiple, geographically diverse copies of all user data	None (asynchronous replication available)	Offer both single DB Instance (one AZ) and Multi-AZ options
Cross-Instance Access (i.e., shareability)	No	No	Yes	Yes	Yes	Yes
Management and Administration	Manual	Manual	Auto	Auto	Manual	Auto