Storage Options in the AWS Cloud

October 2013

Notices

Contents

Introduction

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use cloud computing platform. This whitepaper helps architects and developers understand the primary data storage options available in the AWS cloud. We provide an overview of each storage option, describe ideal usage patterns, performance, durability and availability, cost model, scalability and elasticity, interfaces, and anti-patterns.

In a separate companion whitepaper, 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. You can see this companion whitepaper at [Storage Options in the AWS Cloud: Use Cases](#).[1]

Traditional vs. Cloud-Based Storage Alternatives

Architects of traditional, on-premises IT infrastructures 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 Queues**—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.

- **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 stored in some kind of database, such as a traditional SQL relational database, a NoSQL non-relational

database, or a data warehouse. The underlying database storage typically resides on SAN or DAS devices, or in some cases in memory.

- **Backup and Archive**—Data retained for backup and archival purposes is typically stored on non-disk media such as tapes or optical media, which are usually stored off-site 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, or for the storage of data at a particular point in its lifecycle. These combinations form a hierarchy of data storage tiers.

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

This whitepaper examines the following AWS cloud storage options:

	Amazon S3	Scalable storage in the cloud
	Amazon Glacier	Low-cost archive storage in the cloud
	Amazon EBS	Persistent block storage volumes for Amazon EC2 virtual machines
	Amazon EC2 Instance Storage	Temporary block storage volumes for Amazon EC2 virtual machines
	AWS Import/Export	Large volume data transfer
	AWS Storage Gateway	Integrates on-premises IT environments with cloud storage
	Amazon CloudFront	Global content delivery network (CDN)

	Amazon SQS	Message queue service
	Amazon RDS	Managed relational database server for MySQL, Oracle, and Microsoft SQL Server
	Amazon DynamoDB	Fast, predictable, highly-scalable NoSQL data store
	Amazon ElastiCache	In-memory caching service
	Amazon Redshift	Fast, powerful, full-managed, petabyte-scale data warehouse service
	Databases on Amazon EC2	Self-managed database on an Amazon EC2 instance

For additional comparison categories among the AWS storage collection, see the AWS Storage Quick Reference.

Amazon Simple Storage Service (Amazon S3)

Amazon Simple Storage Service (Amazon S3) is storage for the Internet.[2] It's a simple storage service that offers software developers a highly-scalable, reliable, and low-latency data storage infrastructure at very low costs. 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 Elastic Compute Cloud (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. The number of objects you can store in an Amazon S3 bucket is virtually unlimited. Amazon S3 is also highly secure, supporting encryption at rest, and providing multiple mechanisms to provide fine-grained control of access to Amazon S3 resources. Amazon S3 is also highly scalable, allowing concurrent read or write access to Amazon S3 data by many separate clients or application threads. Finally, Amazon S3 provides data lifecycle management capabilities, allowing users to define rules to automatically archive Amazon S3 data to Amazon Glacier, or to delete data at end of life.

Ideal Usage Patterns

One very common use for Amazon S3 is storage and distribution of static web content and media. This content can be delivered directly from Amazon S3, since each object in Amazon S3 has a unique HTTP URL address, or Amazon S3

can serve as an origin store for 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 no storage provisioning is required, Amazon S3 works well for fast growing websites hosting data intensive, user-generated   content, such as video and photo sharing sites.

Amazon S3 is also frequently used to host entire static websites. Amazon S3 provides a highly-available and highly-scalable solution for websites with only static content, including HTML files, images, videos, and client-side scripts such as JavaScript.

Amazon S3 is also commonly used as a data store for computation and large-scale analytics, such as analyzing financial   transactions, clickstream analytics, 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 often used as a highly durable, scalable, and secure solution for backup and archival of critical data, and to provide 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. Amazon S3's versioning capability is available to protect critical data from inadvertent deletion.

Performance

Access to Amazon S3 from within Amazon EC2 in the same region is fast. Amazon S3 is designed so that server-side latencies are insignificant relative to Internet latencies. Amazon S3 is also built to scale storage, requests, and users to support a virtually unlimited number of web-scale applications. 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 a database, such as Amazon DynamoDB or Amazon RDS. Amazon S3 stores the actual information, and the database serves as the repository for associated

metadata (e.g., object name, size, keywords, and so on). Metadata in the database can easily be indexed and queried, making it very efficient to locate an object's reference via a database query. 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

Multi-Factor Authentication (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 or 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 designed to provide 400 times more durability than a typical disk drive.

Cost Model

With Amazon S3, you pay only for what you use and there is no minimum fee. 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

5 GB of Amazon S3 storage.[3] Pricing information can be found at [Amazon S3 Pricing](#).[4]

Scalability and Elasticity

Amazon S3 has been designed to offer a very high level of scalability and elasticity automatically. Unlike a typical file system that encounters issues when storing large number of files in a directory, Amazon S3 supports a virtually 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 a virtually unlimited number of bytes. You are able to store any number of objects (files) in a single bucket, and Amazon S3 will automatically manage scaling and distributing redundant copies of your information to other servers in other locations in the same region, all using Amazon's high-performance infrastructure.

Interfaces

Amazon S3 provides standards-based REST and SOAP web services APIs for both management and data operations. 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 or software development kit (SDK) that wraps the underlying REST API. AWS SDKs are available for Java, .NET, PHP, and Ruby. The integrated AWS Command Line Interface (AWS CLI) also provides a set of high-level, Linux-like Amazon S3 file commands for common operations, such as **ls**, **cp**, **mv**, **sync**, etc. The AWS CLI for Amazon S3 allows you to perform recursive uploads and downloads using a single folder-level Amazon S3 command, and supports parallel transfers. The AWS CLI also provides command-line access to the low-level Amazon S3 API. Finally, using the AWS Management Console, you can easily create and manage Amazon S3 buckets, upload and download objects, and browse the contents of your Amazon S3 buckets using a simple web-based user interface.

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. Amazon S3 has the following anti-patterns:

- **File system**—Amazon S3 uses a flat namespace and isn't meant to serve as a standalone, POSIX-compliant file system. However, by using delimiters (commonly either the '/' or '\' character) you are able construct your keys to emulate the hierarchical folder structure of file system 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. Instead, pair Amazon S3 with a database to index and query metadata about Amazon S3 buckets and objects.

- **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 Amazon EBS volumes, Amazon RDS or other relational databases, or Amazon DynamoDB.

- **Backup and archival storage**—Data that requires long-term encrypted archival storage with infrequent read access may be stored more cost-effectively in Amazon Glacier.

- **Dynamic website hosting**—While Amazon S3 is ideal for websites with only static content, dynamic websites that depend on database interaction or use server-side scripting should be hosted on Amazon EC2.

Amazon Glacier

Amazon Glacier is an extremely low-cost storage service that provides highly secure, durable, and flexible storage for data backup and archival.[5] With Amazon Glacier, customers can reliably store their data for as little as $0.01 per gigabyte per month. Amazon Glacier enables customers to offload the administrative burdens of operating and scaling storage to AWS, so that they don't have to worry about capacity planning, hardware provisioning, data replication, hardware failure detection and repair, or time-consuming hardware migrations.

You store data in Amazon Glacier as archives. An archive can represent a single file or you may choose to combine several files to be uploaded as a single archive. Retrieving archives from Amazon Glacier requires the initiation of a job. You organize your archives in vaults. You can control access to your vaults using the AWS Identity and Access Management (IAM) service.

Amazon Glacier is designed for use with other Amazon Web Services. Amazon S3 allows you to seamlessly move data between Amazon S3 and Amazon Glacier using data lifecycle policies. You can also use AWS Import/Export to accelerate moving large amounts of data into Amazon Glacier using portable storage devices for transport.

Ideal Usage Patterns

Organizations are using Amazon Glacier to support a number of use cases. These include archiving offsite enterprise information, media assets, research and scientific data, digital preservation and magnetic tape replacement.

Performance

Amazon Glacier is a low-cost storage service designed to store data that is infrequently accessed and long lived. Amazon Glacier jobs typically complete in 3 to 5 hours.

Durability and Availability

Amazon Glacier is designed to provide average annual durability of 99.999999999% (11 nines) for an archive. The service redundantly stores data in multiple facilities and on multiple devices within each facility. To increase durability, Amazon Glacier synchronously stores your data across multiple facilities before returning SUCCESS on uploading archives. Unlike traditional systems, which can require laborious data verification and manual repair, Amazon Glacier performs regular, systematic data integrity checks and is built to be automatically self-healing.

Cost Model

With Amazon Glacier, you pay only for what you use and there is no minimum fee. In normal use, Amazon Glacier has three pricing components: storage (per GB per month), data transfer out (per GB per month), and requests (per thousand UPLOAD and RETRIEVAL requests per month).

Note that Amazon Glacier is designed with the expectation that retrievals are infrequent and unusual, and data will be stored for extended periods of time. You can retrieve up to 5% of your average monthly storage (pro-rated daily) for free each month. If you choose to retrieve more than this amount of data in a month, you are charged an additional (per GB) retrieval fee. There is also a pro-rated charge (per GB) for items deleted prior to 90 days.

Pricing information can be found at [Amazon Glacier Pricing](#).[6]

Scalability and Elasticity

Amazon Glacier scales to meet your growing and often unpredictable storage requirements. A single archive is limited to 4 TBs, but there is no limit to the total amount of data you can store in the service. Whether you're storing petabytes or gigabytes, Amazon Glacier automatically scales your storage up or down as needed.

Interfaces

There are two ways to use Amazon Glacier, each with its own set of interfaces. The Amazon Glacier APIs provide both management and data operations.

First, Amazon Glacier provides a native, standards-based REST web services interface, as well as Java and .NET SDKs. The AWS Management Console or the Amazon Glacier APIs can be used to create vaults to organize the archives in Amazon Glacier. You can then use the Amazon Glacier APIs to upload and retrieve archives, monitor the status of your jobs and also configure your vault to send you a notification via [Amazon Simple Notification Service (SNS)](#) when your jobs complete.[7]

Second, Amazon Glacier can be used as a storage class in Amazon S3 by using object lifecycle management to provide automatic, policy-driven archiving from Amazon S3 to Amazon Glacier. You simply set one or more lifecycle rules for an Amazon S3 bucket, defining what objects should be transitioned to Amazon Glacier and when. You can specify an absolute or relative time period (including 0 days) after which the specified Amazon S3 objects should be transitioned to Amazon Glacier. A new RESTORE operation has been added to the Amazon S3 API, and the retrieval process takes the same 3-5 hours. On retrieval, a copy of the retrieved object is placed in Amazon S3 RRS storage for a specified retention period; the original archived object remains stored in Amazon Glacier. For more

information on how to use Amazon Glacier from Amazon S3, see the Object Lifecycle Management section of the *Amazon S3 Developer Guide*.[8]

Note that when using Amazon Glacier as a storage class in Amazon S3, you use the Amazon S3 APIs, and when using "native" Amazon Glacier, you use the Amazon Glacier APIs. Objects archived to Amazon Glacier via Amazon S3 can only be listed and retrieved via the Amazon S3 APIs or the AWS Management Console—they are not visible as archives in an Amazon Glacier vault.

Anti-Patterns

Amazon Glacier has the following anti-patterns:

- **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 Amazon EBS or a database.

- **Real time access**—Data stored in Amazon Glacier is not available in real time. Retrieval jobs typically require 3-5 hours to complete, so if you need immediate access to your data, Amazon S3 is a better choice.

Amazon Elastic Block Store (Amazon EBS) Volumes

Amazon Elastic Block Store (Amazon EBS) volumes provide durable block-level storage for use with Amazon EC2 instances (virtual machines).[9] Amazon EBS volumes are off-instance, network-attached storage (NAS) that persists independently from the running life of a single Amazon EC2 instance. After an Amazon EBS volume is attached to an instance, you can use it like a physical hard drive, typically by formatting it with the file system of your choice and using the file I/O interface provided by the instance operating system. You can use an Amazon EBS volume to boot an Amazon EC2 instance (Amazon EBS-root AMIs only), and you can attach multiple Amazon EBS volumes to a single Amazon EC2 instance. Note, however, that any single Amazon EBS volume may be attached to only one Amazon EC2 instance at any point in time.

Amazon EBS also provides the ability to create point-in-time snapshots of volumes, which are persisted to Amazon S3. These snapshots can be used as the starting point for new Amazon EBS volumes, and to protect data for long-term

durability. The same snapshot can be used to instantiate as many volumes as you wish. These snapshots can be copied across AWS regions, making it easier to leverage multiple AWS regions for geographical expansion, data center migration and disaster recovery. Sizes for Amazon EBS volumes range from 1 GB to 1 TB, and are allocated in 1 GB increments.

Ideal Usage Patterns

Amazon EBS is meant for data that changes relatively frequently and requires long-term persistence. Amazon EBS is particularly well-suited for use as the primary storage for a database or file system, or for any applications that require access to raw block-level storage. Amazon EBS Provisioned IOPS volumes (see next section) are particularly well-suited for use with databases applications that require a high and consistent rate of random disk reads and writes.

Performance

Amazon EBS provides two volume types: standard volumes and Provisioned IOPS volumes. They differ in performance characteristics and pricing model, allowing you to tailor your storage performance and cost to the needs of your applications. You can attach and stripe across multiple volumes of either type to increase the I/O performance available to your Amazon EC2 applications.

Standard volumes offer cost effective storage for applications with moderate or bursty I/O requirements. Standard volumes are designed to deliver approximately 100 input/output operations per second (IOPS) on average with a best effort ability to burst to hundreds of IOPS. Standard volumes are also well suited for use as boot volumes, where the burst capability provides fast instance start-up times.

Provisioned IOPS volumes are designed to deliver predictable, high performance for I/O intensive workloads such as databases. With Provisioned IOPS, you specify an IOPS rate when creating a volume, and then Amazon EBS provisions that rate for the lifetime of the volume. Amazon EBS currently supports up to 2,000 IOPS per Provisioned IOPS volume. You can stripe multiple volumes together to deliver thousands of IOPS per Amazon EC2 instance to your application.

Because Amazon EBS volumes are network-attached devices, other network I/O performed by the instance, as well as the total load on the shared network, can

affect individual Amazon EBS volume performance. To enable your Amazon EC2 instances to fully utilize the Provisioned IOPS on an Amazon EBS volume, you can launch selected Amazon EC2 instance types as Amazon EBS-optimized instances. Amazon EBS-optimized instances deliver dedicated throughput between Amazon EC2 and Amazon EBS, with options between 500 Mbps and 1,000 Mbps depending on the instance type used. When attached to Amazon EBS-optimized instances, Provisioned IOPS volumes are designed to deliver within 10% of the Provisioned IOPS performance 99.9% of the time.

The combination of Amazon EC2 and Amazon EBS enables you to use many of the same disk performance optimization techniques that you would use with on-premises servers and storage. For example, by attaching multiple Amazon EBS volumes to a single Amazon EC2 instance, you can partition the total application I/O load by allocating one volume for database log data, one or more volumes for database file storage, and other volumes for file system data. Each separate Amazon EBS volume can be configured as Amazon EBS standard or Amazon EBS Provisioned IOPS as needed.

Alternatively, you could stripe your data across multiple similarly-provisioned Amazon EBS volumes using RAID 0 or logical volume manager software, thus aggregating available IOPs, total volume throughput, and total volume size.

Durability and Availability

Amazon EBS volumes are designed to be highly available and reliable. Amazon EBS volume data is replicated across multiple servers in a single Availability Zone to prevent the loss of data from the failure of any single component. The durability of your Amazon EBS volume depends on both the size of your volume and the amount of data that has changed since your last snapshot. Amazon EBS snapshots are incremental, point-in-time backups, containing only the data blocks changed since the last snapshot. Amazon EBS volumes that operate with 20 GB or less of modified data since their most recent snapshot can expect an annual failure rate (AFR) between 0.1% and 0.5%. Amazon EBS volumes with more than 20 GB of unmodified data since the last snapshot should expect higher failure rates that are roughly proportional to the increase in modified data.

To maximize both durability and availability of their Amazon EBS data, you should create snapshots of your Amazon EBS volumes frequently. (For data

consistency, it is a best practice to briefly pause any writes to the volume, or unmount the volume, while the snapshot command is issued. You can then safely continue to use the volume while the snapshot is pending completion.) 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. Because an Amazon EBS volume is created in a particular Availability Zone, the volume will be unavailable if the Availability Zone itself is unavailable. An   Amazon EBS snapshot of a volume, however, is available across all the Availability Zones within a region, and you can use an Amazon EBS snapshot to create one or more new Amazon EBS volumes in any Availability Zone in the region. Amazon EBS snapshots can also be copied from one region to another, and can easily be shared with other user accounts. Thus, Amazon EBS snapshots provides an easy-to-use disk clone or disk image mechanism for backup, sharing, and disaster recovery.

Cost Model

With Amazon EBS, you pay only for what you use. Amazon EBS pricing has three components: provisioned storage, I/O requests, and snapshot storage. Amazon EBS standard volumes are charged per GB-month of provisioned storage and per million I/O requests. Amazon EBS Provisioned IOPS volumes are charged per GB-month of provisioned storage and   per Provisioned IOPS-month. For both volume types, Amazon EBS snapshots are charged per GB-month of data stored. Amazon EBS snapshot copy is charged for the data transferred between regions, and for the standard Amazon EBS snapshot charges in the destination region. It's important to remember that for Amazon EBS volumes, you are charged for *provisioned (allocated)* storage, whether or not you actually use it. For Amazon EBS snapshots, you are charged only for storage actually *used (consumed)*. Note that Amazon EBS snapshots are incremental and compressed, so the storage used in any snapshot is generally much less than the storage consumed on an Amazon EBS volume.

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

Pricing information for Amazon EBS can be found at <u>Amazon EC2 Pricing</u>.[10]

Scalability and Elasticity

Using the AWS Management Console or the APIs, Amazon EBS volumes can easily and rapidly be provisioned and released to scale in and out with your total storage demands. While individual Amazon EBS volumes cannot be resized, if you find that you need additional storage, you have two ways to expand the amount of Amazon EBS space available for your Amazon EC2 instance. The simplest approach is to create and attach a new Amazon EBS volume and begin using it together with your existing ones. However, if you need to expand the size of a single Amazon EBS volume, you can effectively resize a volume using a snapshot:

1. Detach the original Amazon EBS volume.

2. Create a snapshot of the original Amazon EBS volume's data into Amazon S3.

3. Create a new Amazon 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 in place of the original. (In many cases, an OS-level utility must also be used to expand the file system.)

5. Delete the original Amazon EBS volume.

Interfaces

Amazon offers management APIs for Amazon EBS in both SOAP and REST formats. These are used to create, delete, describe, attach, and detach Amazon EBS volumes for your Amazon EC2 instances; to create, delete, and describe snapshots from Amazon EBS to Amazon S3; and to copy snapshots from one region to another. If you prefer to work with a graphical tool, the AWS Management Console gives you all the capabilities of the API in an easy-to-use browser interface. Regardless of how you create your Amazon EBS volume, note that all storage is allocated at the time of volume creation, and that you are charged for this allocated storage even if you don't write data to it.

There is no AWS data API for Amazon EBS. Instead, Amazon EBS presents a block-device interface to the Amazon EC2 instance. That is, to the Amazon EC2 instance, an Amazon EBS volume appears just like a local disk drive. To write to

and read data from Amazon EBS volumes, you therefore use the native file system I/O interfaces of your chosen operating system.

Anti-Patterns

As described previously, Amazon 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. Amazon EBS has the following anti-patterns:

- **Temporary storage**—If you are using Amazon EBS for temporary storage (such as scratch disks, buffers, queues, and caches), consider using local instance store volumes, Amazon SQS, or ElastiCache (Memcached or Redis).

- **Highly-durable storage**—If you need very highly-durable storage, use Amazon S3 or Amazon Glacier. Amazon S3 standard storage is designed for 99.999999999% annual durability per object. In contrast, Amazon EBS volumes with less than 20 GB of modified data since the last snapshot are designed for between 99.5% and 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 Amazon EBS requires a web server running on Amazon EC2, while you can deliver web content directly out of Amazon S3.

Amazon EC2 Instance Store Volumes

Amazon EC2 instance store volumes (also called ephemeral drives) provide temporary block-level storage for many Amazon EC2 instance types.[11] This storage consists of a preconfigured and pre-attached block of disk storage on the same physical server that hosts the Amazon EC2 instance.[12] . The amount of this disk storage varies by Amazon EC2 instance type. In those Amazon EC2 instance families that provide instance storage, larger instances tend to provide both more and larger instance store volumes. Note that some instance types, such as the micro instances (t1) use Amazon EBS storage only, with no instance storage provided.

In addition, the storage-optimized Amazon EC2 instance family provides special-purpose instance storage targeted to specific use cases. HI1 instances provide very fast solid-state drive (SSD)-backed instance storage capable of supporting over 120,000 random read IOPS, and are optimized for very high random I/O performance and low cost per IOPS. In contrast, HS1 instances are optimized for very high storage density, low storage cost, and high sequential I/O performance.

Ideal Usage Patterns

In general, 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 well-suited 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 Amazon EBS AMIs and instance store AMIs). This storage is usable only from a   single Amazon EC2 instance during its lifetime. Note that, unlike Amazon EBS volumes, instance store volumes cannot be detached or attached to another instance.

High I/O and high storage provide Amazon EC2 instance storage targeted to specific use cases. High I/O instances provide instance store volumes backed by SSD, and are ideally suited for many high performance database workloads. Example applications include NoSQL databases like Cassandra and MongoDB. High storage instances support much higher storage density per Amazon EC2 instance, and are ideally suited for applications that benefit from high sequential I/O performance across very large datasets. Example applications include data warehouses, Hadoop storage nodes, seismic analysis, cluster file systems, etc.

Note that applications using instance storage for persistent data generally provide data durability through replication, or by periodically copying data to durable storage.

Performance

The non-SSD-based instance store volumes in most Amazon EC2 instance families have performance characteristics similar to standard Amazon EBS volumes. 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 aggregate IOPS, or to improve sequential disk throughput, multiple instance store volumes can be grouped together using RAID 0 (disk striping) software. Because the bandwidth to the disks is not limited by the network, aggregate sequential throughput for multiple instance volumes can be higher than for the same number of Amazon EBS volumes.

The SSD instance store volumes in the Amazon EC2 high I/O instances provide from tens of thousands to hundreds of thousands of low-latency, random 4 KB random IOPS. Because of the I/O characteristics of SSD devices, write performance can be variable. For more information, see High I/O Instances in the *Amazon EC2 User Guide*.[13]

The instance store volumes on Amazon EC2 high storage instances provide very high storage density and high sequential read and write performance. High storage instances are capable of delivering 2.6 GB/sec of sequential read and write performance when using a block size of 2 MB. For more information, see High Storage Instances in the *Amazon EC2 User Guide*.[14]

Durability and Availability

Amazon EC2 local instance store volumes are not intended to be used as durable disk storage. Unlike Amazon EBS volume data, data on instance store volumes persists only during the life of the associate Amazon EC2 instance. This means that data on instance store volumes is persistent across orderly instance reboots, but if the Amazon EC2 instance is stopped and re-started, terminates, or fails, all data on the instance store volumes is lost. For more information on the lifecycle of an Amazon EC2 instance, see Instance Lifecycle.[15]

You should not use local instance store volumes for any data that must persist over time, such as permanent file or database storage, without providing for data persistence by replicating your data, or by periodically copying data to durable storage such as Amazon EBS or Amazon S3. Note that this also applies to the special-purpose SSD and high-density instance store volumes in the high I/O and high storage instance types.

Cost Model

The cost of the Amazon EC2 instance includes any local instance store volumes, if the instance type provides them. While there is no additional charge for data

storage on local instance store volumes, note that data transferred to and from Amazon EC2 instance store volumes from other Availability Zones or outside of an Amazon EC2 region may incur data transfer charges, and additional charges will apply for use of any persistent storage, such as Amazon S3, Amazon Glacier, Amazon EBS volumes, and Amazon EBS snapshots. Pricing information for Amazon EC2, Amazon EBS, and data transfer can be found at Amazon EC2 Pricing.[16]

Scalability and Elasticity

The number and storage capacity of Amazon EC2 local instance store volumes are fixed and defined by the instance type.

While you can't increase or decrease the number of instance store volumes on a single Amazon EC2 instance, this storage is still scalable and elastic, in that you can scale the total amount of instance store up or down by increasing or decreasing the number of running Amazon EC2 instances.

Local instance store volumes are tied to a particular Amazon EC2 instance, and are fixed in number and size for a given Amazon EC2 instance type, so the scalability and elasticity of this storage is tied to the number of Amazon EC2 instances.

However, you can achieve full storage elasticity by including one of the other suitable storage options, such as Amazon S3 or Amazon EBS, in your Amazon EC2 storage strategy.

Interfaces

There is no separate management API for Amazon EC2 instance store volumes. Instead, instance store volumes are specified using the block device mapping feature of the Amazon EC2 API and the AWS Management Console. You cannot create or destroy instance store volumes, but you can control whether or not they are exposed to the Amazon EC2 instance, and what device name is used.

There is also no separate data API for instance store volumes. Just like Amazon EBS volumes, instance store volumes present a block-device interface to the Amazon EC2 instance. That is, to the Amazon EC2 instance, an instance store volume appears just like a local disk drive. To write to and read data from

instance store volumes, you therefore use the native file system I/O interfaces of your chosen operating system.

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. Also, keep careful track of your block device mappings. There is no simple way for an application running on an Amazon EC2 instance to determine which block device is an instance store (ephemeral) volume and which is an Amazon EBS (persistent) 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, or data that is replicated for durability. In many situations, however, other AWS storage options may be more appropriate. Amazon EC2 instance store volumes have the following anti-patterns:

- **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, Amazon EBS volumes or Amazon S3 are more appropriate.

- **Relational database storage**—In most cases, relational databases require storage that persists beyond the lifetime of a single Amazon EC2 instance, making Amazon 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 Amazon EBS volumes are the better choice.

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

AWS Import/Export

[AWS Import/Export](#) accelerates moving large amounts of data into and out of AWS using portable storage devices for transport.[17] AWS transfers your data directly onto and off of storage devices using Amazon's high-speed internal network and bypassing the Internet. For significant datasets, AWS Import/Export is often faster than Internet transfer and more cost effective than upgrading your connectivity.

AWS Import/Export supports importing and exporting data into and out of several types of AWS storage, including Amazon EBS snapshots, Amazon S3 buckets, and Amazon Glacier vaults.

Ideal Usage Patterns

AWS Import/Export is ideal for transferring large amounts of data in and out of the AWS cloud, especially in cases where transferring the data over the Internet would be too slow or too costly. In general, if loading your data over the Internet would take a week or more, you should consider using AWS Import/Export. Common use cases include initial data upload to AWS, content distribution or regular data interchange to/from your customers or business associates, transfer to Amazon S3 or Amazon Glacier for off-site backup and archival storage, and quick retrieval of large backups from Amazon S3 or Amazon Glacier for disaster recovery.

Performance

Each AWS Import/Export station is capable of loading data at over 100 MB per second, but in most cases the rate of the data load will be bounded by a combination of the read or write speed of your portable storage device and, for Amazon S3 data loads, the average object (file) size.

Durability and Availability

Once the data is imported to AWS, the durability and availability characteristics of the target storage applies. Amazon EBS snapshots, Amazon S3, and Amazon Glacier are all designed for 99.999999999% (11 nines) durability. For more details, see the sections on each individual service in this whitepaper.

Cost Model

With AWS Import/Export, you pay only for what you use. AWS Import/Export has three pricing components: a per-device fee, a data load time charge (per data-loading-hour), and possible return shipping charges (for expedited shipping, or shipping to destinations not local to that AWS Import/Export region). For the destination storage, the standard Amazon EBS snapshot, Amazon S3, and Amazon Glacier request and storage pricing applies. Pricing information can be found at [AWS Import/Export Pricing](#).[18]

Scalability and Elasticity

The total amount of data you can load using AWS Import/Export is limited only by the capacity of the devices that you send to AWS. For Amazon S3, individual files will be loaded as objects in Amazon S3, and may range up to 5 terabytes in size. For Amazon Glacier, individual devices will be loaded as a single archive, and may range up to 4 terabytes in size.

The aggregate total amount of data that can be imported is virtually unlimited.

Interfaces

To upload or download data, you must create and submit an AWS Import/Export job for each storage device shipped. Each job request requires a manifest file, a YAML-formatted text file that contains a set of key-value pairs that supply the   required information—such as your device ID, secret access key, and return address—necessary to complete the job.

Jobs can be created using a command line tool (the AWS Import/Export Web Service Tool), the AWS SDK for Java, or a native REST API. The job request is tied to the storage device through a signature file in the root directory (for Amazon S3 import jobs), or by a barcode taped to the device (for Amazon EBS and Amazon Glacier jobs).

Anti-Patterns

AWS Import/Export is optimal for large data that would take too long to load over the Internet, so the anti-pattern is simply data that is more easily transferred over the Internet. If your data can be transferred over the Internet in less than one week, AWS Import/Export may not be the ideal solution.

AWS Storage Gateway

[AWS Storage Gateway](#) is a service that connects an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage infrastructure.[19] The service enables you to securely store data to the AWS cloud for scalable and cost-effective storage. AWS Storage Gateway supports industry-standard storage protocols that work with your existing applications. It provides low-latency performance by maintaining frequently accessed data on-premises while securely storing all of your data encrypted in Amazon S3. For disaster recovery scenarios, it can serve as a cloud-hosted solution, together with Amazon EC2, that mirrors your entire production environment.

AWS Storage Gateway's software appliance is available for download as a virtual machine (VM) image that you install on a host in your datacenter. Once you've installed your gateway and associated it with your AWS account through our activation process, you can use the AWS Management Console to create either gateway-cached or gateway-stored volumes that can be mounted as iSCSI devices by your on-premises applications.

Gateway-cached volumes allow you to utilize Amazon S3 for your primary data, while retaining some portion of it locally in a cache for frequently accessed data. These volumes minimize the need to scale your on-premises storage infrastructure, while still providing your applications with low-latency access to their frequently accessed data. You can create storage volumes up to 32 TBs in size and mount them as iSCSI devices from your on-premises application servers. Data written to these volumes is stored in Amazon S3, with only a cache of recently written and recently read data stored locally on your on-premises storage hardware.

Gateway-stored volumes store your primary data locally, while asynchronously backing up that data to AWS. These volumes provide your on-premises applications with low-latency access to their entire datasets, while providing durable, off-site backups. You can create storage volumes up to 1 TB in size and mount them as iSCSI devices from your on-premises application servers. Data written to your gateway-stored volumes is stored on your on-premises storage hardware, and asynchronously backed up to Amazon S3 in the form of Amazon EBS snapshots.

Ideal Usage Patterns

Organizations are using AWS Storage Gateway to support a number of use cases. These include corporate file sharing, enabling existing on-premises backup applications to store primary backups on Amazon S3, disaster recovery, and data mirroring to cloud-based compute resources.

Performance

As the AWS Storage Gateway VM sits between your application, Amazon S3, and underlying on-premises storage, the performance you experience will be dependent upon a number of factors, including the speed and configuration of your underlying local disks, the network bandwidth between your iSCSI initiator and gateway VM, the amount of local storage allocated to the gateway VM, and the bandwidth between the gateway VM and Amazon S3. For gateway-cached volumes, to provide low-latency read access to your on-premises applications, it's important that you provide enough local cache storage to store your recently accessed data. Our AWS Storage Gateway documentation provides guidance on how to optimize your environment setup for best performance, including how to properly size your local storage.[20]

AWS Storage Gateway efficiently uses your Internet bandwidth to speed up the upload of your on-premises application data to AWS. AWS Storage Gateway only uploads data that has changed, which minimizes the amount of data sent over the Internet. You can also use AWS Direct Connect to further increase throughput and reduce your network costs by establishing a dedicated network connection between your on-premises gateway and AWS.[21]

Durability and Availability

AWS Storage Gateway durably stores your on-premises application data by uploading it to Amazon S3. Amazon S3 stores data in multiple facilities and on multiple devices within each facility. Amazon S3 also performs regular, systematic data integrity checks and is built to be automatically self-healing.

Cost Model

With AWS Storage Gateway, you pay only for what you use. AWS Storage Gateway has four pricing components: gateway usage (per gateway per month), snapshot storage usage (per GB per month), volume storage usage (per GB per

month), and data transfer out (per GB per month). Pricing information can be found at [AWS Storage Gateway Pricing](#).[22]

Scalability and Elasticity

In both gateway-cached and gateway-stored volume configurations, AWS Storage Gateway stores data in Amazon S3, which has been designed to offer a very high level of scalability and elasticity automatically. Unlike a typical file system that encounters issues when storing large number of files in a directory, Amazon S3 supports a virtually 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 a virtually 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

The AWS Management Console can be used to download the AWS Storage Gateway VM image. You can then select between a gateway-cached or gateway-stored configuration, activate your on-premises by associating your gateway's IP Address with your AWS account, select an AWS region, and create AWS Storage Gateway volumes and attach these volumes as iSCSI devices to your on-premises application servers.

You can begin using the AWS Storage Gateway in just a few steps. To get started, you simply:

1. Download the AWS Storage Gateway VM image from the [AWS Management Console](#).[23]

2. Select between a gateway-cached or gateway-stored configuration. With gateway-cached volumes, you store your primary data in Amazon S3, and retain frequently accessed data locally. With gateway-stored volumes, you store your primary data locally, and upload backups to Amazon S3.

3. Allocate local storage to your installed on-premises gateway from direct-attached storage (DAS), network attached storage (NAS), or storage area network (SAN) storage. For gateway-cached configurations, this local

storage will be used for your frequently accessed data. For gateway-stored configurations, it will be used for your   primary data.

4. Activate your on-premises by associating your gateway's IP Address with your AWS account and select an AWS region for your gateway to store uploaded data.

5. Use the AWS Management Console to create AWS Storage Gateway volumes and attach these volumes as iSCSI devices to your on-premises application servers.

By following these steps, you can begin using your existing on-premises applications to seamlessly store data in Amazon S3. These applications can now write data to their attached AWS Storage Gateway volumes. Your application data will either be stored directly in Amazon S3 for substantial cost savings on primary storage (gateway-cached volumes), or will be stored locally and backed up to Amazon S3 for durable and cost-effective backups (gateway-stored volumes).

Anti-Patterns

AWS Storage Gateway has the following anti-patterns:

- **Database storage**—Amazon EC2 instances using Amazon EBS volumes are a natural choice for database storage and workloads.

Amazon CloudFront

Amazon CloudFront content delivery network (CDN) is a web service for content delivery.[24] Amazon CloudFront makes your website's dynamic, static, and streaming content available from a global network of edge locations. When a visitor requests a file from your website, he or she is invisibly redirected to a copy of the file at the nearest edge location, which   results in faster download times than if the visitor had accessed the content from a data center farther away. Amazon CloudFront caches content at edge locations for a period of time that you specify.

Amazon CloudFront supports all files that can be served over HTTP. This includes dynamic web pages, such as HTML or PHP pages, any popular static files that are a part of your web application, such as website images, audio, video, media files or software downloads. For on-demand media files, you can

also choose to stream your content using Real-Time Messaging Protocol (RTMP) delivery. Amazon CloudFront also supports delivery of live media over HTTP.

Amazon CloudFront is optimized to work with other Amazon Web Services, like Amazon S3, Amazon EC2, Amazon Elastic Load Balancing, and Amazon Route 53. Amazon CloudFront also works seamlessly with any non-AWS origin server, which stores the original, definitive versions of your files.

Ideal Usage Patterns

Amazon CloudFront is ideal for distribution of frequently-accessed static content that benefits from edge delivery—like popular website images, videos, media files or software downloads. Amazon CloudFront can also be used to deliver dynamic web applications over HTTP. These applications may include static content, dynamic content, or a whole site with a mixture of the two. Amazon CloudFront is also commonly used to stream audio and video files to web browsers and mobile devices.

Performance

Amazon CloudFront is designed for low-latency and high-bandwidth delivery of content. Amazon CloudFront speeds up the distribution of your content by routing end users to the edge location that can best serve the end user's request in a worldwide network of edge locations. Typically, requests are routed to the nearest Amazon CloudFront edge location in terms of latency. This dramatically reduces the number of networks that your users' requests must pass through and improves performance. End users get both lower latency—the time it takes to load the first byte of the object—and higher sustained data transfer rates needed to deliver popular objects to end user at scale.

Durability and Availability

Because a CDN is an edge cache, Amazon CloudFront does not provide durable storage. Durable file storage is provided by the origin server, such as Amazon S3 or a web server running on Amazon EC2. Amazon CloudFront provides high availability by using a distributed global network of edge locations. Origin requests from the edge locations to AWS origin servers (e.g., Amazon EC2, Amazon S3, etc.) are carried over network paths that Amazon constantly monitors and optimizes for both availability and performance. This edge network provides increased reliability and availability because there is no longer

a central point of failure. Copies of your files are now held in edge locations around the world.

Cost Model

With Amazon CloudFront, there are no long-term contracts or required minimum monthly commitments—you pay only for as much content as you actually deliver through the service. Amazon CloudFront has two pricing components: regional data transfer out (per GB) and requests (per 10,000). Note that it is often cheaper (as well as faster) to deliver popular content from Amazon S3 through Amazon CloudFront rather than directly from Amazon S3. Pricing information can be found at [Amazon CloudFront Pricing](#).[25]

Scalability and Elasticity

Amazon CloudFront is designed to provide seamless scalability and elasticity. You can easily start very small, and grow to massive numbers of global connections. With Amazon CloudFront, you don't need to worry about maintaining expensive web server capacity to meet the demand from potential traffic spikes for your content. The service automatically responds as demand increases or decreases without any intervention from you. Amazon CloudFront also uses multiple layers of caching at each edge location and collapses simultaneous requests for the same object before contacting your origin server. These optimizations further help reduce the need to scale your origin infrastructure as your website becomes more popular.

Interfaces

There are several ways to manage and configure Amazon CloudFront. The AWS Management Console provides an easy way to manage Amazon CloudFront. All the features of the Amazon CloudFront API are supported. For example, you can enable or disable distributions, configure CNAMEs, and enable end-user logging. You can also use the Amazon CloudFront command line tools, the native REST API, or one of the supported SDKs.

There is no data API for Amazon CloudFront, no command to preload data. Instead, data is automatically pulled into Amazon CloudFront edge locations on the first access of an object from that location.

Anti-Patterns

Amazon CloudFront is optimal for delivery of popular static or dynamic. However, in a number of situations Amazon CloudFront is not the optimal solution. Amazon CloudFront has the following anti-patterns:

- **Programmatic cache invalidation**—While Amazon CloudFront supports cache invalidation, AWS recommends   using object versioning rather than programmatic cache invalidation.

- **Infrequently requested data**—It may be better to serve infrequently-accessed data directly from the origin   server, avoiding the additional cost of origin fetches for data that is not likely to be reused at the edge.

Amazon Simple Queue Service (Amazon SQS)

[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.[26] 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 a virtually 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 class of 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 Patterns

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 Amazon SQS worker queue, you create an Amazon SQS 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 pulls the task messages from the queue

2. Retrieves the named file

3. Processes the conversion

4. Writes the image back to Amazon S3

5. Writes a "task complete" message to another queue

6. Deletes the original task message

7. Checks 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 stored 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 Model

With Amazon SQS, you pay only for what you use and there is no minimum fee. 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). Pricing information can be found at Amazon SQS Pricing.[27]

Scalability and Elasticity

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

Interfaces

Amazon SQS can be accessed through an HTTP Query web services interface, as well as through SDKs for Java, PHP, Ruby, and .NET. The Amazon SQS APIs provides both management and data interfaces. Five APIs make it easy for developers to get started with Amazon SQS: CreateQueue, SendMessage, ReceiveMessage, ChangeMessageVisibility, and DeleteMessage. Additional APIs provide advanced functionality.

Anti-Patterns

Amazon SQS has the following anti-patterns:

- **Binary or large messages**—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

Amazon 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.

- **High-speed message queuing or very short tasks**—If your application requires a very high-speed message send and receive response from a single producer or consumer, use of Amazon DynamoDB or a message-queuing system hosted on Amazon EC2 may be more appropriate.

Amazon Relational Database Service (Amazon RDS)

Amazon Relational Database Service (Amazon RDS) is a web service that provides the capabilities of MySQL, Oracle, or Microsoft SQL Server relational database as a managed, cloud-based service.[28] It also eliminates much of the administrative overhead associated with launching, managing, and scaling your own relational database on Amazon EC2 or in another computing environment.

Ideal Usage Patterns

Amazon RDS is ideal for existing applications that rely on MySQL, Oracle, or SQL Server traditional relational database engines. Since Amazon RDS offers full compatibility and direct access to native database engines, most code, libraries, and tools designed for these databases should work unmodified with Amazon RDS. Amazon RDS is also optimal for new applications with structured data that requires more sophisticated querying and joining capabilities than that provided by Amazon's NoSQL database offering, Amazon DynamoDB.

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 Amazon EC2 compute unit (ECU)) up to a quadruple extra-large instance (64-bit platform with 68 GB of RAM and 26 ECUs).

Amazon RDS Provisioned IOPS provides a high-performance storage option designed to deliver fast, predictable, and consistent performance for I/O intensive transactional database workloads. When creating a new DB instance using the Amazon RDS Provisioned IOPS storage, you can specify the IOPS your instance needs from 1,000 IOPS to 30,000 IOPS and Amazon RDS provisions that IOPS rate for the lifetime of the instance.

Durability and Availability

As with other relational databases on Amazon EC2, Amazon RDS leverages Amazon EBS volumes as its data store. For enhanced durability, Amazon 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, Amazon 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 Amazon RDS Multi-AZ deployment feature enhances both the durability and the availability of your database by synchronously replicating your data between a primary Amazon 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, Amazon 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 helps to prevent loss of data.

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

Cost Model

With Amazon RDS, you pay only for what you use and there is no minimum fee. 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 I/O requests), additional backup storage (per GB-month), and data transfer in / out (per GB per month). Pricing information can be found at [Amazon Relational Database Service Pricing](#).[29]

Scalability and Elasticity

Amazon RDS resources can be scaled elastically in several dimensions: database storage size, database storage IOPS rate, database instance compute capacity, and the number of read replicas.

Amazon RDS supports "pushbutton scaling" of both database storage and compute resources. To scale the Amazon RDS database storage elastically, you can use the command line tools, API, or AWS Management Console to provision additional storage. Depending on your needs, this additional storage can be added either immediately or during the next maintenance window. For databases needing consistent I/O performance, you can scale the throughput of your database by specifying the IOPS rate, from 1,000 to 30,000 IOPS, with corresponding storage size of 100 GB to 3 TB.

Scaling computational resources up or down by switching from one DB instance type to another 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.

Amazon RDS for MySQL also enables you to scale out beyond the capacity of a single database deployment for read-heavy database workloads by creating one or more read replicas. Read replicas use MySQL's built-in asynchronous replication capability, and can be used in conjunction with the synchronous replication provided by Amazon RDS Multi-AZ deployments.

Finally, in the more general case, administrators may configure multiple Amazon RDS instances and leverage database partitioning or sharding to spread the workload over multiple DB instances, achieving even greater database scalability and elasticity.

Interfaces

Amazon RDS APIs and the AWS Management Console provide a management interface that allows you to create, delete, modify, and terminate Amazon RDS DB instances; to create DB snapshots; and to perform point-in-time restores. To start using Amazon RDS, you simply use the AWS Management Console or Amazon RDS APIs to launch a database instance (*DB instance*), selecting the DB engine (MySQL, Oracle or SQL Server), license type, DB instance class, and storage capacity that best meets your needs.

There is no AWS data API for Amazon RDS. Once the database is created, AWS provides a DNS endpoint through which you can connect to your DB instance using your favorite database tool or programming language. Since you have direct access to a native MySQL, Oracle, or SQL Server database engine, most tools designed for these engines should work unmodified with Amazon RDS. After your schema and data are in place, you interact with your information via standard SQL, as well as JDBC and other popular APIs, and any graphical tools that can work with relational data. There are no code changes to be made to let your application interact with Amazon RDS. You simply replace your database server's address (e.g. dbserver.example.com) in you database connection string 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, even after failover of an Amazon RDS Multi-AZ deployment. Aside from configuring the endpoint, everything else about your database-centric application is unchanged.

Anti-Patterns

Amazon RDS is a great solution for cloud-based fully-managed relational database, but in a number of scenarios it may not be the right choice. Amazon RDS has the following anti-patterns:

- **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 DynamoDB to be more appropriate for your needs.

- **Numerous BLOBs**—While all of the database engines provided by Amazon RDS support binary large objects (BLOBs), if your application makes heavy use of them (audio files, videos, images, and so on), you may find Amazon S3 to be a better choice.

- **Automated scalability**—As stated previously, Amazon RDS provides pushbutton scaling. If you need fully-automated scaling, Amazon DynamoDB may be a better choice.

- **Other database platforms**—At this time, Amazon RDS provides a MySQL, Oracle, and SQL Server databases. If you need another database platform (such as IBM DB2, Informix, PostgreSQL, or Sybase) you need to deploy a self-managed database on an Amazon EC2 instance by using a relational database AMI, or by installing database software on an Amazon EC2 instance.

- **Complete control**—If your application requires complete, OS-level control of the database server, with full root or admin login privileges (for example, to install additional third-party software on the same server), a self-managed database on Amazon EC2 may be a better match.

Amazon DynamoDB

Amazon DynamoDB is a fast, fully-managed NoSQL database service that makes it simple and cost-effective to store and retrieve any amount of data, and serve any level of request traffic.[30] Amazon DynamoDB helps offload the administrative burden of operating and scaling a highly-available distributed database cluster. This storage alternative meets the latency and throughput requirements of highly demanding applications by providing extremely fast and predictable performance with seamless throughput and storage scalability.

Amazon DynamoDB stores structured data in tables, indexed by primary key, and allows low-latency read and write access to items ranging from 1 byte up to 64 KB. Amazon DynamoDB supports three data types: number, string, and binary, in both scalar and multi-valued sets. Tables do not have fixed a schema, so each data item can have a different number of attributes. The primary key can either be a single-attribute hash key or a composite hash-range key. Local

secondary indexes provide additional flexibility for querying against attributes other than the primary key. Amazon DynamoDB provides both eventually-consistent reads (by default), and strongly-consistent reads (optional), as well as implicit item-level transactions for item put, update, delete, conditional operations, and increment/decrement.

Amazon DynamoDB is integrated with other services, such as Amazon Elastic MapReduce (Amazon EMR), Amazon Redshift, Amazon Data Pipeline, and Amazon S3, for analytics, data warehouse, data import/export, backup, and archive.

Ideal Usage Patterns

Amazon DynamoDB is ideal for existing or new applications that need a flexible NoSQL database with low read and write latencies, and the ability to scale storage and throughput up or down as needed without code changes or downtime.

Common use cases include: mobile apps, gaming, digital ad serving, live voting and audience interaction for live events, sensor networks, log ingestion, access control for web-based content, metadata storage for Amazon S3 objects, e-commerce shopping carts, and web session management. Many of these use cases require a highly available and scalable database because downtime or performance degradation has an immediate negative impact on an organization's business.

Performance

SSDs and limiting indexing on attributes provides high throughput and low latency (single-digit milliseconds typical for average server-side response times), and drastically reduces the cost of read and write operations. As the datasets grow, predictable performance is required so that low-latency for the workloads can be maintained. This predictable performance can be achieved by defining the provisioned throughput capacity required for a given table. Behind the scenes, the service handles the provisioning of resources to achieve the requested throughput rate, which takes the burden away from the customer to have to think about instances, hardware, memory, and other factors that can affect an application's throughput rate. Provisioned throughput capacity reservations are elastic and can be increased or decreased on demand.

Durability and Availability

Amazon DynamoDB has built-in fault tolerance that automatically and synchronously replicates data across three Availability Zones in a region for high availability and to help protect data against individual machine, or even facility failures.

Cost Model

With Amazon DynamoDB, you pay only for what you use and there is no minimum fee. Amazon DynamoDB has three pricing components: provisioned throughput capacity (per hour), indexed data storage (per GB per month), data transfer in or out (per GB per month). New customers can start using Amazon DynamoDB for free as part of the AWS Free Usage Tier. Pricing information can be found at Amazon DynamoDB Pricing.[31]

Scalability and Elasticity

Amazon DynamoDB is both highly-scalable and elastic. There is no limit to the amount of data you can store in an Amazon DynamoDB table, and the service automatically allocates more storage as you store more data using the Amazon DynamoDB write APIs. Data is automatically partitioned and re-partitioned as needed, while the use of SSDs provides predictable low-latency response times at any scale. The service is also elastic, in that you can simply "dial-up" or "dial-down" the read and write capacity of a table as your needs change.

Interfaces

Amazon DynamoDB provides a low-level REST API, as well as higher-level SDKs for Java, .NET, and PHP that wrap the low- level REST API and provide some object-relational mapping (ORM) functions. These APIs provide both a management and data interface for Amazon DynamoDB. The API currently offers thirteen operations that enable table management (creating, listing, deleting, and obtaining metadata) and working with attributes (getting, writing, and deleting attributes; query using an index, and full scan). While standard SQL isn't available for Amazon DynamoDB, you may use the Amazon DynamoDB select operation to create SQL-like queries that retrieve a set of attributes based on criteria that you provide. You can also work with Amazon DynamoDB using the AWS Management Console.

Anti-Patterns

Amazon DynamoDB has the following anti-patterns:

- **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 (MySQL, Oracle, or SQL Server), or one of the many preconfigured Amazon EC2 database AMIs. You are also free to create your own Amazon EC2 instance, and install your own choice of database software.

- **Joins and/or complex transactions**—While many solutions are able to leverage Amazon DynamoDB 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 a self-managed database.

- **BLOB data**—If you plan on storing large (greater than 64 KB) BLOB data, such as digital video, images, or music, you'll want to consider Amazon S3. However, Amazon DynamoDB 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.

- **Large data with low I/O rate**—Amazon DynamoDB uses SSD drives and is optimized for workloads with a high I/O rate per GB stored. If you plan to store very large amounts of data that are infrequently accessed, other storage options, such as Amazon S3, may be a better choice.

Amazon ElastiCache

[ElastiCache](#) is a web service that makes it easy to deploy, operate, and scale a distributed, in-memory cache in the cloud.[32] ElastiCache improves the performance of web applications by allowing you to retrieve information from a fast, managed, in-memory caching system, instead of relying entirely on slower disk-based databases. ElastiCache supports two popular open-source caching engines: Memcached and Redis.

- **Memcached**–ElastiCache is protocol-compliant with Memcached, a widely adopted memory object caching system, so code, applications, and popular tools that you use today with existing Memcached environments will work seamlessly with the service.

- **Redis**–A popular open-source in-memory key-value store that supports data structures such as sorted sets and lists. ElastiCache supports Redis master / slave replication which can be used to achieve cross Availability Zone redundancy.

Ideal Usage Patterns

ElastiCache improves application performance by storing critical pieces of data in memory for low-latency access. It is frequently used as a database front end in read-heavy applications, improving performance and reducing the load on the database by caching the results of I/O-intensive queries. It is also frequently used to manage web session data, to cache dynamically-generated web pages, and to cache results of computationally-intensive calculations, such as the output of recommendation engines. For applications that need more complex data structures than strings, such as lists, sets, hashes, and sorted sets, the Redis engine is often used as an in-memory NoSQL database.

Performance

Performance of a cache layer is very dependent on the caching strategy and the hit rate at the application level, so it is difficult to provide general guidance. Choose the total amount of cache memory needed based on the size of your dataset and the expected access pattern. Then divide this by the memory per cache node to get the required number of cache nodes. Make sure that you can maintain acceptable performance without overloading the database backend in the event of the failure and replacement of one or more cache nodes. You can easily add or remove cache nodes from a running cluster, but you cannot change the cache node type in a running cluster.

Durability and Availability

By definition, an in-memory cache stores transient data, or transient copies of durable data, so data durability is managed elsewhere. It is still desirable to have your cache be as fault-tolerant and highly-available as possible.

With the Memcached engine, all ElastiCache nodes in a single cache cluster are provisioned in a single Availability Zone. ElastiCache automatically monitors the health of your cache nodes and replaces them in the event of network partitioning, host hardware, or software failure. In the event of cache node failure, the cluster remains available, but performance may be reduced due to time needed to repopulate the cache in the new "cold" cache nodes. To provide enhanced fault-tolerance for Availability Zone failures or cold-cache effects, you can run redundant cache clusters in different Availability Zones.

With the Redis engine, ElastiCache supports replication to up to five read replicas for scaling. To improve availability, you can place read replicas in other Availability Zones. ElastiCache monitors the primary node, and if the node becomes unavailable, ElastiCache will repair or replace the primary node if possible, using the same DNS name. If the primary cache node recovery fails or its Availability Zone is unavailable, you can failover from the primary node to one of the read replicas with an API call.

Cost Model

With ElastiCache, you pay only for what you use and there is no minimum fee. ElastiCache has only a single pricing component: pricing is per cache node-hour consumed. For new customers, AWS provides a [free usage tier](#) that includes up to 750 hours usage of a micro cache node. Pricing information can be found at [Amazon ElastiCache Pricing](#).[33]

Scalability and Elasticity

ElastiCache is highly scalable and elastic. You can choose from a range of cache node types supporting from 213 MB up to 68 GB of memory per node. You can add or delete cache nodes to your cache cluster at any time to meet your application load. Auto Discovery enables automatic discovery of Memcached cache nodes by ElastiCache Clients when the nodes are added to or removed from an ElastiCache cluster.

Interfaces

To control and manage ElastiCache—to create, describe, reboot, modify, and destroy cache clusters—you can use the AWS Management Console, the ElastiCache command line tools, the HTTP Query API, and various SDKs.

To read and write data to ElastiCache cache cluster, you simply use the normal Memcached and Redis APIs. Existing applications using Memcached or Redis can use ElastiCache with almost no modifications other than changing the port or DNS name used to connect. For a Memcached application, you use standard operations like get, set, incr and decr in exactly the same way as you would in your existing Memcached deployments. For a Redis application, you use GET, SET, EXPIRE, and the various flavors of PUSH and POP exactly as you do with existing Redis application.

ElastiCache for Memcached is protocol-compliant with Memcached, supporting both the text and the binary protocols. As a result, you can switch to using ElastiCache without recompiling or re-linking your applications–the libraries you use will continue to work. Similarly, ElastiCache for Redis is protocol-compliant with open source Redis. Code, applications, drivers and tools that customers use today with their existing standalone Redis data store will continue to work with ElastiCache for Redis.

Anti-Patterns

Amazon ElastiCache has the following anti-patterns:

- **Persistent data**—If you need very fast access to data, but also need strong data durability (persistence), Amazon DynamoDB is probably a better choice.

Amazon Redshift

Amazon Redshift is a fast, fully-managed, petabyte-scale data warehouse service that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools.[34] It is optimized for datasets that range from a few hundred gigabytes to a petabyte or more.

Traditional data warehouses require significant time and resources to administer, especially for large datasets. The financial cost associated with building, maintaining, and growing self-managed, on-premises data warehouses is very high. Amazon Redshift manages the work needed to set up, operate, and scale a data warehouse, from provisioning the infrastructure capacity to automating ongoing administrative tasks such as backups and patching.

Ideal Usage Patterns

Amazon Redshift is ideal for analyzing large datasets using your existing business intelligence tools. Organizations are using Amazon Redshift to do the following:

- Analyze global sales data for multiple products

- Store historical stock trade data

- Analyze ad impressions and clicks

- Aggregate gaming data

- Analyze social trends

- Measure clinical quality, operation efficiency, and financial performance in the health care space

Performance

Amazon Redshift uses a variety of innovations to obtain very high query performance on datasets ranging in size from hundreds of gigabytes to a petabyte or more. It uses columnar storage, data compression, and zone maps to reduce the amount of I/O needed to perform queries. Amazon Redshift has a massively parallel processing (MPP) architecture that parallelizes and distributes SQL operations to take advantage of all available resources. The underlying hardware is designed for high performance data processing that uses local attached storage to maximize throughput.

Durability and Availability

Amazon Redshift has multiple features that enhance the reliability of your data warehouse cluster. Amazon Redshift stores three copies of your data—all data written to a node in your cluster is automatically replicated to other nodes within the cluster, and all data is continuously backed up to Amazon S3. Snapshots are automated, incremental, and continuous. Amazon Redshift stores your snapshots for a user-defined period, which can be from one to thirty-five days. At any time, you can create one or more manual snapshots, which are retained until explicitly deleted. Amazon Redshift also continuously monitors the health of the cluster and automatically re-replicates data from failed drives and replaces nodes as necessary.

Cost Model

With Amazon Redshift, you can pay as you go and there are no upfront costs. Amazon Redshift has three pricing components: data warehouse node hours, backup storage, and data transfer. Compute node hours are the total number of hours run across all compute nodes for the billing period. Backup storage is the storage associated with automated and manual snapshots for an Amazon Redshift data warehouse cluster. Increasing the backup retention period or taking additional snapshots increases the backup storage consumed by the Amazon Redshift data warehouse cluster. There is no additional charge for backup storage up to 100% of your provisioned storage for an active data warehouse cluster.

There is no data transfer charge for data transferred to or from Amazon Redshift outside of Amazon Virtual Private Cloud (Amazon VPC). Data transfer to or from Amazon Redshift in Amazon VPC accrues standard AWS data transfer charges.

Pricing information can be found at [Amazon Redshift Pricing](#).[35]

Scalability and Elasticity

Amazon Redshift provides "pushbutton scaling" of compute nodes within a data warehouse cluster. With a few clicks of the AWS Management Console or a simple API call, you can easily scale the number of nodes in your data warehouse cluster up or down as your performance or capacity needs change. An Amazon Redshift data warehouse cluster can be started with as little as a single 2 TB XL node and scale all the way to a hundreds 16 TB 8XL nodes for 1.6 PB of compressed user data. Amazon Redshift will place your existing cluster into read-only mode, provision a new cluster of your chosen size, and then copy data from your old cluster to your new one in parallel. Queries can continue running against the old cluster while the new one is being provisioned. Once the data has been copied to the new cluster, Amazon Redshift will automatically redirect queries to the new cluster and remove the old cluster.

Interfaces

The Amazon Redshift Query API provides a management interface to manage data warehouse clusters programmatically. Additionally, the AWS SDKs for Java, .NET, and other languages provide class libraries that wrap the underlying Amazon Redshift API to simplify your programming tasks. If you prefer a more

interactive way of managing clusters, you can use the Amazon Redshift console and the AWS CLI.

The Amazon Redshift APIs do not provide a data interface. Amazon Redshift is a SQL data warehouse and uses industry standard ODBC and JDBC connections and PostgreSQL drivers. Once you've provisioned your cluster, you can connect to it, start loading data, and run queries using the same SQL-based tools and business intelligence applications you use today. For more information, see the Amazon Redshift Partners page.[36]

Data can be loaded into Amazon Redshift from a range of data sources including Amazon S3, Amazon DynamoDB, and AWS Data Pipeline. Amazon Redshift attempts to load data in parallel into each compute node to maximize the rate at which data can be ingested into the data warehouse cluster. For more information on loading data into Amazon Redshift, see the Amazon Redshift Getting Started Guide.[37]

Anti-Patterns

Amazon Redshift has the following anti-patterns:

- **OLTP workloads**—Amazon Redshift is a column-oriented database suited to data warehouse and analytics, where queries are typically performed over very large datasets. If your application involves online transaction processing, a traditional row-based database system, such as Amazon RDS, is a better match.

- **BLOB data**—If you plan on storing binary (e.g., video, pictures, or music), you'll want to consider Amazon S3.

Databases on Amazon EC2

Amazon EC2, together with Amazon EBS volumes, provides an ideal platform for you to operate your own self-managed relational database in the cloud.[38] 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 Patterns

Running a relational database on Amazon EC2 and Amazon EBS is the ideal scenario for users whose application requires a specific traditional relational database not supported by Amazon RDS, 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 Amazon 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-premises equipment. We encourage you to benchmark your actual application on several Amazon EC2 instance types using several storage configurations 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. For database storage, it is usually best to use Amazon EBS Provisioned IOPS volumes. To scale up I/O performance, you can increase the Provisioned IOPS, change the number of Amazon EBS volumes, or use software RAID 0 (disk striping) across multiple Amazon EBS volumes, which will aggregate 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 Amazon EBS volumes as the data store, so all the notes about the durability and availability of Amazon EBS data apply here as well. And, again, the basic durability and availability of relational data stored on Amazon EBS volumes can be further enhanced by using Amazon EBS snapshots, or by using third-party database backup utilities (such as Oracle's RMAN) to store database backups in Amazon S3.

Cost Model

By running a database on Amazon EC2, you pay only for what you use and there is no minimum fee. 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 Amazon 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. Pricing information for Amazon EC2, Amazon EBS, and data transfer can be found at [Amazon EC2 Pricing](#).[39]

Scalability and Elasticity

In many cases, users of traditional relational database solutions on Amazon EC2 can take advantage of the scalability and elasticity 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 Amazon EBS AMIs. You can then create multiple new instances of your database configuration in a few moments.

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. Self-managed relational databases on Amazon EC2 have the following anti-patterns:

- **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 DynamoDB to be more appropriate for your needs, and significantly easier to manage.

- **Numerous 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. You can use a database to manage the metadata.

- **Automatic scaling**—Users of relational databases on AWS can, in many cases, leverage the scalability and elasticity of the underlying AWS platform, but this requires system administrators or DBAs to perform a manual or scripted task. If you need pushbutton scaling or fully-automated scaling, you may opt for another storage choice such as Amazon DynamoDB or Amazon RDS.

- **MySQL, Oracle, SQL Server**—If you are running a self-managed MySQL, Oracle, or SQL Server database on Amazon EC2, you should consider the automated backup, patching, Provisioned IOPS, replication, and pushbutton scaling features offered by a fully-managed Amazon RDS database.

Contributors

The following individuals and organizations contributed to this document:

- Joseph Baron

- Sanjay Kotecha

Further Reading

For additional help, see the following sources.

Cloud Storage Use Cases

For illustrations of real-world usage of AWS storage options, see the companion Storage Options in the AWS Cloud: Use Cases whitepaper.

AWS Storage Services

- Amazon S3

- Amazon Glacier

- Amazon EBS

- [Amazon EC2 Instance Store Volumes](#)
  (See sections on instance types, instance storage, and block device mapping)

- [AWS Import/Export](#)

- [AWS Storage Gateway](#)

- [Amazon CloudFront](#)

- [Amazon SQS](#)

- [Amazon RDS](#)

- [Amazon DynamoDB](#)

- [Amazon ElastiCache](#)

- [Amazon Redshift](#)

Other Resources

- [AWS SDKs, IDE Toolkits, and Command Line Tools](#)

- [Amazon Web Services Simple Monthly Calculator](#)

- [Amazon Web Services Blog](#)

- [Amazon Web Services Forums](#)

- [Running Databases on AWS](#)

- [Database AMIs in AWS Marketplace](#)
  (Search AWS Marketplace for "database")

- [Public Database AMIs](#)
  (Search public AMI list for "database" or for a particular databases)

- [AWS Free Usage Tier](#)

- [Public Data Sets on AWS](#)

- [AWS Case Studies](#)

Notes

[1] http://media.amazonwebservices.com/AWS_Storage_Use_Cases.pdf

[2] https://aws.amazon.com/s3/

[3] http://aws.amazon.com/free/

[4] http://aws.amazon.com/s3/pricing/

[5] http://aws.amazon.com/glacier/

[6] http://aws.amazon.com/glacier/pricing/

[7] http://aws.amazon.com/sns/

[8] http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html

[9] http://aws.amazon.com/ebs/

[10] http://aws.amazon.com/ec2/pricing/#EBS

[11] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/

[12] Most Amazon EC2 instance types provide local instance storage volumes; however, the micro and the M3 instance are Amazon EBS-only, and do not provide local instance storage. Also, instances that use Amazon EBS for the root device (boot from Amazon EBS) do not expose the instance store volumes by default. If desired, you can expose the instance store volumes at instance launch time by specifying a block device mapping. For more information, see the Amazon EC2 User Guide.

[13] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/storage_instances.html

[14] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/high_storage_instances.html

[15] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html

[16] http://aws.amazon.com/ec2/pricing/

[17] http://aws.amazon.com/importexport/

[18] http://aws.amazon.com/importexport/pricing/

[19] http://aws.amazon.com/storagegateway/

[20] http://docs.amazonwebservices.com/storagegateway/latest/userguide/

[21] http://aws.amazon.com/directconnect/

[22] http://aws.amazon.com/storagegateway/pricing/

[23] https://console.aws.amazon.com/storagegateway

[24] http://aws.amazon.com/cloudfront/

[25] http://aws.amazon.com/cloudfront/pricing/

[26] http://aws.amazon.com/sqs/

[27] http://aws.amazon.com/sqs/pricing/

[28] http://aws.amazon.com/rds/

[29] http://aws.amazon.com/rds/pricing/

[30] http://aws.amazon.com/dynamodb/

[31] http://aws.amazon.com/dynamodb/pricing/

[32] http://aws.amazon.com/elasticache/

[33] http://aws.amazon.com/elasticache/pricing/

[34] http://aws.amazon.com/redshift/

[35] http://aws.amazon.com/redshift/pricing/

[36] http://aws.amazon.com/redshift/partners/

[37] http://docs.aws.amazon.com/redshift/latest/gsg/welcome.html

[38] http://aws.amazon.com/ec2/

[39] http://aws.amazon.com/ec2/pricing/