RDBMS in the Cloud: Oracle Database on AWS

Jean-Pierre Le Goaller, Carlos Conde, and Shakil Langha

October 2013

(Please consult **http://aws.amazon.com/whitepapers/** for the latest version of this paper)

Table of Contents

Abstract

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use cloud computing platform. Relational database management systems, or RDBMS, are widely deployed within the Amazon cloud. In this whitepaper, we help you understand how to deploy Oracle Database on AWS. You can run Oracle Database on Relational Database Service (Amazon RDS) or Amazon Elastic Compute Cloud (EC2).

The goal of this whitepaper is to explain how you can run Oracle Database on both Amazon RDS and Amazon EC2, and to give you an understanding of the advantages of each approach. We review in detail how to provision and monitor your Oracle database, and how to manage scalability, performance, backup and recovery, high availability and security in both Amazon RDS and Amazon EC2. We also describe how you can set up a Disaster Recovery solution between an on-premise Oracle environment and AWS, and how you can perform a migration of your existing Oracle database to AWS. After reading this whitepaper you will be able to make an educated decision and choose the solution that best fits your needs.

Oracle Database Solutions on AWS

There are two ways to run an Oracle database on AWS. One way is to use Amazon Relational Database Service (Amazon RDS). Another way is to deploy the database on Amazon Elastic Compute Cloud (EC2). Alternatively, there are other AWS solutions that you could use for your database needs.

Oracle Database on Amazon RDS

Amazon RDS is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. Amazon RDS automates installation, disk provisioning and management, patching, minor version upgrades, failed instance replacement, as well as backup and recovery of your Oracle database. Amazon RDS also offers automated Multi-AZ (Availability Zone) synchronous replication, allowing you to set up a highly available environment fully managed by AWS. If you want Amazon to handle the day-to-day management of your Oracle database, Amazon RDS is the preferred way. This enables you to focus on higher-level tasks, such as schema optimization, query tuning, and application development. For more information about Amazon RDS, see http://aws.amazon.com/documentation/rds/.

Oracle Database on Amazon EC2

Amazon EC2 is a web service that provides resizable computing capacity in the cloud. When deploying your Oracle database on Amazon EC2, you have full control over the operating system, database installation, and configuration. You also have full control over your database administration, including backups and recovery, patching of the operating system and the database, tuning of the operating system and database parameters, security management, and configuration of high availability or replication. In terms of database administration, using Amazon EC2 is similar to running an Oracle database in-house. With Amazon EC2, you can quickly provision and configure instances and storage, you can scale your environment vertically (by changing the size of your instances) or horizontally (by adding more instances), and you can provision your databases in AWS regions across the world to provide low latency to your end-users worldwide. Running your own relational database on Amazon EC2 is the ideal scenario if you require a maximum level of control and configurability. For more information about Amazon EC2, see http://aws.amazon.com/documentation/ec2/.

Other Database Scenarios

Although running your own Oracle database on Amazon EC2 or Amazon RDS is a great solution for many users, there are a number of scenarios where other AWS solutions might be a better choice.

Scenario	Description
Web Scalability	If your application needs to store huge quantities of data with a high throughput, you may find NoSQL databases to be more appropriate for your needs than relational databases. While NoSQL databases are not relational and not ACID compliant, they are typically more adept at storing and retrieving large amount of data with high performance. Also, NoSQL databases are significantly easier to manage. Many AWS customers run NoSQL databases in Amazon EC2, including Cassandra, MongoDB, Redis, CouchDB, and HBase. If you do not want to manage your NoSQL database, Amazon DynamoDB is a fast, fully-managed AWS NoSQL database that provides predictable performance and seamless scalability. All data items are stored on solid state drives (SSDs) and are automatically replicated across multiple Availability Zones (within a region) to provide built-in high availability and data durability. For more information about DynamoDB, see http://aws.amazon.com/dynamodb.
Data Warehousing	While you can use Oracle Database for data warehousing purposes, an alternative is 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. It is optimized for datasets ranging from a few hundred gigabytes to a petabyte or more and costs less than $1,000 per terabyte per year. For more information about Amazon Redshift, see http://aws.amazon.com/redshift.
Storage of Numerous Binary Large Objects (Blobs)	While the Oracle databases support blobs storage and management, you may find Amazon Simple Storage Service (Amazon S3) to be a better and simpler choice for BLOB storage if your application makes heavy use of these objects (video, audio, images, and so on). Many AWS customers have found it useful to store blob-style data in Amazon S3 while using a NoSQL database such as DynamoDB to manage blob metadata. For more information about Amazon S3, see http://aws.amazon.com/s3.

Choosing between Amazon RDS and Amazon EC2 for an Oracle Database

For an Oracle database, both Amazon RDS and Amazon EC2 have advantages. Amazon RDS is easier to set up, manage and maintain than running Oracle in Amazon EC2 and lets you focus on other tasks rather than the day-to-day administration of Oracle. Alternatively, running Oracle in Amazon EC2 gives you more control, flexibility and choice. Depending on your application and your requirements, one might be preferable over the other.

Amazon RDS might be a better choice for you if:

- You want to focus on your business and applications, and outsource the following tasks to Amazon: provisioning of the database, management of backup and recovery, management of security patches, upgrades of minor Oracle versions and storage management.
- You need a highly available database solution and want to take advantage of the push-button, synchronous Multi-AZ replication offered by Amazon RDS, without having to manually setup and maintain a standby database.
- You do not want to manage backups and, most importantly, point-in-time recoveries of your database.

- You would rather focus on high-level tasks, such as performance tuning and schema optimization, than on the daily administration of the database.

Amazon EC2 might be a better choice for you if:

- You need full control over the database, including access to the operating system.
- You have experienced database administrators that will be able to manage the database.
- Your database size exceeds the current maximum database size in Amazon RDS (3TB at the time of this writing).
- You need to use Oracle features or options not currently supported by Amazon RDS.
- You want to setup a disaster recovery solution between several AWS regions, or between your on-premise environment and AWS.

Oracle Database Feature Comparison between Amazon RDS and Amazon EC2

The capabilities of Amazon RDS for an Oracle database are constantly improving. However, there are a few Oracle Database features and options that are currently not supported by Amazon RDS. Here is a list of Oracle Database features that are not directly supported by Amazon RDS, but where Amazon RDS provides equivalent functionality:

- Oracle Data Guard and Oracle Active Data Guard allow you to create standby databases. Amazon RDS provides similar functionality when run in Multi-AZ mode.
- Oracle Enterprise Manager (OEM) Grid Control allows you to manage multiple Oracle database and application servers. While Amazon RDS does not support Grid Control, you can manage your Amazon RDS databases individually with OEM Database Control.
- Automated Storage Management (ASM) simplifies the management of Oracle data files, control files and log files. It is however unnecessary in Amazon RDS since Amazon RDS manages all the Oracle files for you.
- Streams can be configured to propagate database changes within and between Oracle databases, and can be used for replication purposes. Instead, you can use Amazon RDS Multi-AZ to create a synchronous standby database.
- Oracle XML DB provides native XML storage and retrieval capabilities. It is supported in Amazon RDS without the XML DB Protocol Server.
- Real Application Clusters (RAC) is a cluster database with a shared cache and shared disk architecture. Currently, you cannot run RAC in Amazon EC2 either. Some of the customers that require RAC choose to host their databases outside of AWS while running the rest of their infrastructure in AWS. To do this while providing good database performance, they host their RAC database at an AWS partner datacenter, which is linked via AWS Direct Connect to the AWS datacenter where the rest of their infrastructure exists. AWS Direct Connect lets you establish a dedicated network connection between the AWS Direct Connect location and the closest AWS region. It provides low-latency 1 Gbps and 10 Gbps connections, and you can easily provision multiple connections if you need more network capacity. For more information about AWS Direct Connect, see http://aws.amazon.com/directconnect/.

Currently the following Oracle options are not supported in Amazon RDS:

- Oracle Java support allows you to deploy Java server-side applications in the database.
- Oracle Locator and Oracle Spatial aid you in managing geographic and location-data in a native type within the database.

If you need to use any of these database features and options, Amazon EC2 is currently the best-suited deployment platform. This list is subject to change as we add new features to Amazon RDS. To check for recent updates, see the Amazon RDS documentation at http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.Options.html.

Oracle Licensing and Support

Before running Oracle in AWS, make sure that you have the correct licensing and support for Oracle.

Licensing/Support	Description
Oracle Licensing in Amazon RDS	You can run Amazon RDS for Oracle under two different licensing models: License Included and Bring Your Own License (BYOL). • In the License Included model, you do not need separately purchased Oracle licenses because the Oracle Database software has been licensed by AWS. The license is included in the Amazon RDS hourly cost. It is important to note however that the License Included model is currently limited to Oracle database Standard Edition One. • In the BYOL model, you can use existing Oracle database licenses for your Amazon RDS deployment, or purchase new licenses directly from Oracle. In the BYOL model, you should follow the licensing rule defined by Oracle, and described at http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf. If you decide to use the Multi-AZ mode (see the "High Availability" section for details), you may need extra licenses for the standby database instance. You should review your Oracle Software Licensing Agreement and comply with Oracle's licensing policies.
Oracle Licensing in Amazon EC2	Licensing Oracle in Amazon EC2 follows the BYOL model explained in the previous Amazon RDS description.
Oracle Support for AWS	Oracle fully supports deploying Oracle database on AWS, as described at http://www.oracle.com/technetwork/topics/cloud/faq-098970.html#support.

Starting an Oracle Database Instance on AWS

The way you start an Oracle database instance on AWS depends on your configuration.

Starting an Oracle Database Instance on Amazon RDS

Starting an Oracle Database instance on Amazon RDS is easy. You can do this by using the AWS Management Console or you just choose the Oracle engine version that you want to run, and we take care of choosing and configuring the underlying software. Using the AWS Management Console, you select the RDS tab, click **Launch a DB instance**, and then select the Oracle edition that you wish to use. Figure 1 shows an example of the Launch DB Instance Wizard.

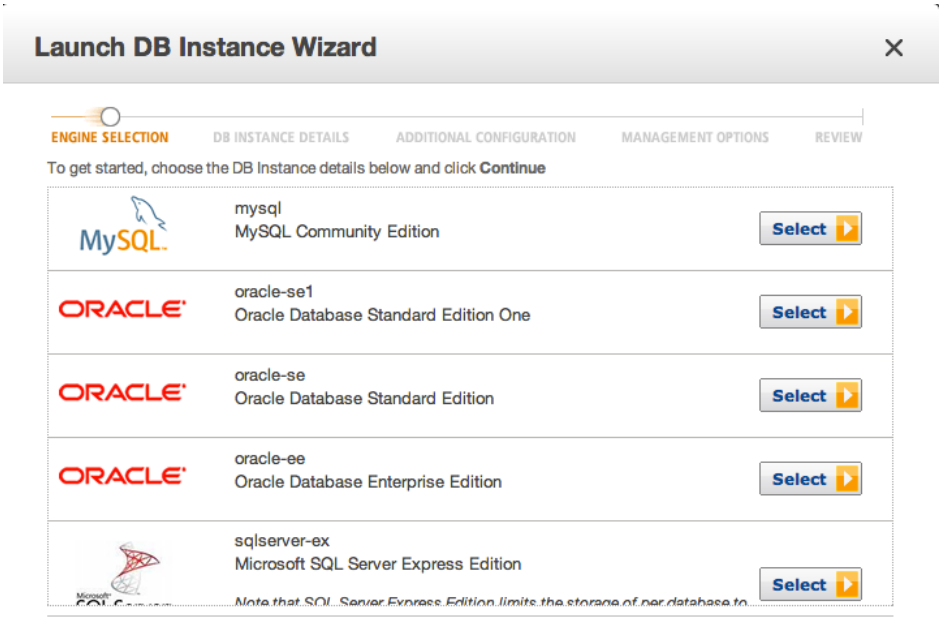

Figure 1: AWS Management Console Launch DB Instance Wizard

You then configure your instance by specifying a few parameters, such as the size of the Amazon RDS instance, the size of the database, and the I/O settings. This configuration will be explained more later in this whitepaper. You can also accomplish all of this by using the command-line interface (CLI) or by invoking the application programming interface (API) from several programming languages including Java, Node.js, PHP, Python, Ruby and .NET.

Starting an Oracle Database instance in Amazon EC2

To start an Oracle Database instance in Amazon EC2, you first have to do is to choose an Amazon Machine Image (AMI). An AMI is a special type of pre-configured operating system and virtual application software (in this case, Oracle software), which is used to create a virtual machine in Amazon EC2. You can start an Amazon EC2 instance using the AWS Management Console, the CLI or the API.

Choosing a Linux AMI in Amazon EC2

We recommend that you use the AMIs available at http://aws.amazon.com/amis in the Oracle category. In addition to the Oracle Applications AMIs, you will find AMIs containing Oracle Enterprise Linux and the Oracle Database in the following editions: Standard Edition One, Standard Edition and Enterprise Edition. When starting an Amazon EC2 instance using one of these AMIs, you get the benefit of having an Oracle database pre-installed within of a few minutes.

Alternatively, you can start an Amazon EC2 instance running the operating system of your choice, and install Oracle Database manually.

An AMI attribute to watch out for is the virtualization type. For example, you can find out whether an AMI uses PV (Para Virtualization) or HVM (Hardware Virtual Machine) by parsing the output of the CLI command `ec2-describe-images -v <AMI ID>` and looking for the value of the `virtualizationType` field, which will be either “hvm” or “paravirtual”. Some instance types, such as cluster compute instances, only support HVM AMIs. Several operating systems have AMIs running on HVM, including Amazon Linux, SUSE and Microsoft Windows.

Choosing a Microsoft Windows AMI in Amazon EC2

In order to run Oracle on an Amazon EC2 instance, you can start a virtual machine using a Windows AMI, and then install Oracle Database on top of it, at which point you can bundle your own AMI that now includes Oracle. You can find a selection of Windows AMIs in the AWS marketplace at http://aws.amazon.com/marketplace.

Performance Management

The performance of a relational database instance on AWS depends on many factors, including the Amazon RDS or Amazon EC2 instance type, the configuration of the database software, the application workload, and for Oracle databases running on Amazon EC2 instances, the storage configuration. The following sections describe various options that are available to you to tune the performance of the AWS infrastructure on which your Oracle database is running.

Instance Sizing

Increasing the performance of a database requires an understanding of which of the server's resources is the performance constraint. If the database performance is limited by CPU, memory or network throughput, you can scale up the memory, compute, and network performance by choosing a larger instance type.

For many customers, increasing the performance of a single DB instance is the easiest way to increase the performance of their application overall. One way to achieve this is with vertical scalability. You do this by scaling up (or down) the instance size to address the hardware performance requirements of the database. In the Amazon RDS and Amazon EC2 environments, vertical scalability is very easy.

Instance Sizing in Amazon RDS

You have the option in Amazon RDS of choosing the instance class that fits your workload. Amazon RDS supports several database instance classes. At the time of this writing, they range from the very small Micro to the High-Memory Quadruple Extra Large, which features eight virtual cores, 68GB of memory and a high I/O capacity. The Amazon RDS instance classes are essentially identical, in terms of CPU, memory and I/O capacity, to the Micro, Standard and High-Memory Amazon EC2 instance types. For complete and up-to-date information, see the Amazon RDS for Oracle home page at http://aws.amazon.com/rds/oracle/.

The first time that you start your database instance, pick the instance type that seems the most relevant to your use case (in terms of number of cores and amount of memory) and use that as the starting point. You can then monitor the performance to determine whether it is a good fit or whether you need to pick a bigger or smaller instance type. For more information about how to monitor your Oracle database performance, see the "Monitoring and Management" section.

If you want to change the size of your instance, you can modify the Amazon RDS instance, using the AWS Management Console, the CLI or the API. One of the parameters that you can modify is the Database Instance Class. This will cause a restart of your database instance, which you can choose to occur right away or during the next weekly maintenance window that you specified while creating the instance (which can also be changed).

Instance Sizing in Amazon EC2

Amazon EC2 instances are grouped into eight families: Standard (first and second generation), Micro, High-Memory, High-CPU, Cluster Compute, Cluster GPU, High I/O, High Storage, and High Memory Cluster. For complete, up-to-date information about Amazon EC2 instance types, see http://aws.amazon.com/ec2/instance-types/.

When running high-performance databases, the High-Memory Instances can be a good option because they allow you to maximize the amount of memory available to the SGA (System Global Area) of the database. The Cluster Compute Instances combine very high CPU capability and high-memory. You should also consider the High I/O Instances because they feature local SSD drives and will offer the most I/O of any instance type. The High Memory Cluster Instances feature a high amount of memory with local SSD storage and can be good choice for the largest database instances. For more information about the impact of the instance size on I/O performance, see the "Disk I/O Management" sections.

In AWS it is very easy and quick to scale vertically (change the instance type and size), if you find out that you undersized or oversized your instance. The method to change the size of the instance depends on the type of AMI that you selected:

- EBS-backed instances are instances where the root device is stored in Amazon Elastic Block Store (Amazon EBS). In this case, you can just stop the instance, change the instance type (either through the AWS Management Console, the CLI, or an API), and restart the instance.
- Instance-store backed instances are instances where the root device is stored on the instance internal storage. In this case, you would save any changes to the root device (for example by rebundling an AMI), terminate the instance and start a new one.
- In both scenarios, the instance size change can be accomplished within a few minutes.

Note: To determine whether your instance is EBS-backed or instance-store backed, you can look at the Amazon EC2 instances dashboard in the AWS Management Console, or parse the output of the CLI command `ec2-describe-images -v <AMI ID>` and look for the value of the `rootDeviceType` field.

Disk I/O Management in Amazon RDS

Amazon RDS uses Amazon EBS for database and log storage. Depending on the size of storage requested, Amazon RDS automatically stripes across multiple EBS volumes to enhance IOPS performance. For a high-level description about EBS, seethe "EBS Volumes" section.

Increasing Amazon RDS Storage Size

Amazon RDS makes it easy to scale up your storage by using the AWS Management Console, the CLI (`rds-modify-db-instance`) or the `ModifyDBInstance` API. Note that Amazon RDS adds storage without restarting the DB Instance and without interrupting active processes. The main reason for increasing the Amazon RDS storage size is to accommodate the database growth, but this can also be done to increase I/O. For an existing DB Instance, you may observe some I/O capacity improvement if you scale up your storage.

Increasing Amazon RDS Instance Size

Increasing the size of the Amazon RDS instance may provide better performance because larger instances offer the following:

- More memory (which allows for more data to be cached).
- More network bandwidth (which can help provide better EBS performance).

Amazon RDS Provisioned IOPS

Amazon RDS Provisioned IOPS (I/O operations per second) is a storage option that gives you control over your database storage performance by allowing you to specify your IOPS rate at database creation time. This feature is designed to deliver fast, predictable, and consistent I/O performance. When you create new Oracle database instances using the

AWS Management Console or the Amazon RDS APIs, you can provision from 1,000 IOPS to 30,000 IOPS with corresponding storage ranging from 100GB to 3TB. You can start small and scale up in increments of 1,000 IOPS. Here are some important things to know about provisioned IOPS in Amazon RDS:

- The ratio between the amount of storage and Provisioned IOPS should be between 3 and 10. For example, a 100GB database could have provisioned IOPS set between 300 and 1,000.
- Oracle RDS uses a page size of 8 KB. On a DB instance with a full duplex I/O channel bandwidth of 1000 megabits per second (Mbps), such as the m2.4xl instance, the maximum IOPS for page I/O is about 25,000 IOPS in each direction for 8 KB I/O. A workload consisting of 50% reads and 50% writes could reach 25,000 IOPS with 16 KB I/O or 25,000 IOPS with 8 KB I/O.
- If you are using Provisioned IOPS storage, we recommend that you use the instances optimized for Provisioned IOPS (currently the m1.large, m1.xlarge, m2.2xlarge, and m2.4xlarge instance classes). The available network bandwidth for Provisioned IOPS for m1.large instance class is 500 megabits per second (Mbps) compared to 1000 Mbps for an m1.xlarge, m2.2xlarge, or m2.4xlarge instance. For a similar IOPS-intensive workload, the number of realized IOPS for instances with a 1000 Mbps storage network bandwidth will be higher than for other instances.
- You can convert an Amazon RDS database using standard storage to use provisioned IOPS storage. The actual amount of your I/O throughput may vary depending on your workload. For Oracle, the maximum IOPS rate is 25,000 with a 8KB page size.

Disk I/O Management in Amazon EC2

In this section, we review the Amazon EC2 disk storage options that you can choose from for your Oracle database.

Instance Storage

An Amazon EC2 instance comes with a certain amount of “local” storage, which is ephemeral. Any data saved on an instance will not be available after that instance is terminated by the customer, or if the underlying hardware fails, which would cause an instance restart to happen on a different server. This characteristic makes instance storage a challenging option for database persistent storage. However, Amazon EC2 instances can have the following benefits:

- Ephemeral disks offer good performance for sequential disk access, and don’t impact your network connectivity. Some customers have found it useful to use ephemeral disks to store temporary files to conserve network bandwidth.
- High I/O instances (discussed later) offer unmatched I/O performance and are recommended for database workloads, provided you implement a backup or replication strategy that addresses the ephemeral nature of this storage.
- High Storage instances offer 48TB of internal storage, which can allow you to run very large databases on instance storage. Just like for High I/O instances, you should mitigate the risk of losing your local storage with a backup or replication strategy.

EBS Volumes

AWS offers a storage service called Amazon Elastic Block Store (Amazon EBS), which provides persistent block-level storage volumes. Amazon EBS volumes are off-instance storage that persist independently from the life of an instance. Amazon EBS volume data is mirrored across multiple servers in an Availability Zone (datacenter) to prevent the loss of data from the failure of any single component. It is easy to back them up to Amazon Simple Storage Service (Amazon S3)

using snapshots. These attributes make EBS volumes suitable for data files, log files and for the flash recovery area. The maximum size of an EBS volume is 1TB, but you can address larger database sizes by striping your data across multiple volumes. There are two types of EBS volumes, standard EBS volumes and Provisioned IOPS volumes, as described in the following sections.

Standard EBS Volumes

Standard EBS volumes provide about 100 IOPS on average, with the ability to burst to hundreds of IOPS on a best-effort basis. Standard EBS volumes are great for applications with moderate or bursty I/O requirements, as well as for boot volumes. Standard EBS volumes tend to perform better for sequential reads and writes (as opposed to random reads and writes).

Provisioned IOPS Volumes

Provisioned IOPS volumes are designed to deliver predictable and consistent high performance for I/O intensive workloads such as databases. With Provisioned IOPS volumes, you specify an IOPS (I/O operations per second) rate when creating a volume, and then Amazon EBS provisions that rate for the lifetime of the volume. Here are some important characteristics of Provisioned IOPS volumes:

- Amazon EBS currently supports up to 4,000 IOPS per Provisioned IOPS volume.
- By striping across 10 volumes, you would consistently provide your database with up to 40,000 IOPS.
- The number of provisioned IOPS applies to I/O operations with a size of 16KB or less. Beyond 16KB, the number of IOPS will decrease proportionally with the size of the I/O. For example, if you provision a 4,000 IOPS volume and your average I/O size is 32KB, you should expect 2,000 IOPS. If your I/O size is 64KB, then you should expect 1,000 IOPS.
- There is a maximum ratio of 10 between the volume size (in GB) and the provisioned IOPS. For example, if you provision a 50GB volume, the maximum provisioned IOPS that you could request would be 500.
- While providing more consistent performance, Provisioned IOPS can be more cost effective than standard EBS volumes if your database consistently generates a high I/O workload. With Provisioned IOPS you pay for the number of provisioned IOPS (whereas you pay for actual usage for standard EBS volumes). This has the added benefit of making your I/O cost more predictable.

EBS-Optimized Instances

EBS-optimized instances enable Amazon EC2 instances to fully utilize the provisioned IOPS on an EBS volume. 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. When attached to EBS-optimized instances, Provisioned IOPS volumes are designed to deliver within 10% of their provisioned performance 99.9% of the time. The combination of EBS-optimized instances and Provisioned IOPS volumes helps to ensure that instances are capable of consistent and high EBS I/O performance. Most databases with high I/O requirements should benefit from this feature. You can also use EBS-optimized instances with standard EBS volumes if you need predictable bandwidth between your instances and EBS. Note that EBS-optimized instances are currently available only on some of the larger instance types. For more details, see http://aws.amazon.com/ec2/instance-types/.

Choosing the Right Instance Type

If your performance is limited by disk I/O, changes to the configuration of your disk resources may be in order. EBS volumes are connected via the network, and therefore the network throughput available to your instance can have an impact on disk performance. A rule of thumb is that the larger the instance, the more network throughput it can provide. To take full advantage of standard or Provisioned IOPS volumes, you could use the instances running on a 10

Gigabit network, such as the Cluster Compute, High Memory Cluster and High I/O Instances, or choose the EBS-optimized instances described in the previous section. An increase in network throughput can have a significant impact on network-attached storage performance, so be sure to choose the appropriate instance type.

Spreading the I/O Activity on Several EBS Volumes

To scale up random I/O performance, you can increase the number of EBS volumes your data resides on, for example by using 8 x 100GB EBS volumes instead of 1 x 800GB EBS volume. Aggregating multiple EBS volumes increases the total IOPS of the logical volume. EBS volumes can be aggregated using different techniques like Linux software RAID, Logical Volume Manager (LVM) or Oracle Automatic Storage Management (ASM).

A single standard EBS volume can provide an average of approximately 100 IOPS, and single instances with arrays of 10+ attached EBS disks can average 1,000 IOPS. Provisioned IOPS offer an easier and more predictable way to provide your database with high I/O. You can stripe two or more volumes together in order to reach multiple thousands of IOPS By striping across 10 Provisioned IOPS volumes on EBS-optimized instances, you could provide your database with storage volumes capable of up to 20,000 IOPS.

However, remember that utilizing striping techniques generally reduces the operational durability of the logical volume by a degree inversely proportional to the number of EBS volumes in the stripe set. EBS volume data is natively replicated, so using RAID 0 (striping) might provide you with sufficient redundancy and availability. You could use RAID 10 (striping and mirroring) instead of RAID 0 to improve the availability of your aggregate, but in most cases using RAID 10 will decrease your I/O performance compared to RAID 0. The choice between RAID 0 and RAID 10 will depend on your availability requirements and on the number of volumes of your aggregate. At any rate, you should take snapshots of your EBS volumes as often as possible.

Data, logs, and temporary files will benefit from being stored on independent EBS volumes or volume aggregates because they present different I/O patterns. In order to take advantage of additional EBS volumes, be sure to evaluate the network load to help ensure that your instance size is sufficient to provide the network bandwidth required.

Oracle ASM on Amazon EBS

Oracle Automatic Storage Management (ASM) is an integrated, high-performance file system and disk manager designed specifically for Oracle databases. Compared to other file systems, ASM presents several advantages specifically for Oracle databases:

- Chunks of data are distributed across all available logical disks in a disk group, thereby removing potential performance “hot-spots”.
- ASM does not perform any I/O itself and does no read-ahead (like file systems) to push data in cache that is never used by the database.
- No intensive tuning, such as setting fragment sizes and file system journals is required.
- No journal is required for consistency because Oracle redo logs already cover this function.
- Adding or removing storage is very easy. After adding storage, ASM automatically rebalances the volumes so they will all be utilized equally. This increases performance and is particularly in an environment like AWS where you can provision new EBS volumes on-demand.

Oracle ASM disk groups provide three types of redundancy: normal, high, and external. With normal and high redundancy, files are replicated within the disk group. With external redundancy, ASM does not provide any redundancy for the disk group. When setting up ASM for a group of volumes, we recommend using external redundancy since

Amazon EBS volumes are already redundant within an Availability Zone. Oracle ASM best practices, like using different disk groups for data and log files, work and recovery areas, also apply in Amazon EBS.

High I/O and Cluster High Memory Instances

For high I/O workloads, an alternative to Provisioned IOPS volumes is to use High I/O instances, which contain SSD drives as internal storage and address the most demanding database workloads. The High I/O Quadruple Extra Large instance can provide up to 120,000 random read IOPS and 85,000 random write IOPS. The High Memory Cluster Eight Extra Large Instance offers 244 GB of memory in addition to 240 GB of local SSD storage. Note however that this SSD storage is internal to the instance and will be lost if the instance is stopped or if the underlying hardware fails. When using this type of storage for databases, you should make sure that you have a solid strategy to avoid loss of data, for example by frequently backing up your data to Amazon S3. In addition to storage performance, High I/O and High Memory Cluster Instances also have very high I/O performance via 10 Gigabit Ethernet, which allows for increased EBS performance.

Reducing I/O with Oracle Advanced Compression

Since 11g Oracle provides an option called Advanced Compression. It allows for compressing structured data (numbers, characters) as well as unstructured data (documents, spreadsheets, XML and other files). Using advanced compression can result in up to 2 to 4x disk space savings. It will also result in less disk writes, and faster full scan and range scan operations as compressed rows can be retrieved with less disk I/O. Memory will also be more effectively utilized as the blocks will remain compressed in memory. However, there is a trade-off. Using advanced compression will reduce I/O and storage footprint, but the compression and decompression activities may cause some performance overhead for some workloads.

Benchmarking

As described previously, Amazon EC2 offers many options to optimize and tune your I/O subsystem. We encourage you to benchmark your actual application on several instance types and storage configurations in order to select the most appropriate configuration. For example, you could use the Oracle benchmarking tool Orion (Oracle I/O Numbers), an Oracle I/O calibration tool, which mimics the type of I/O performed by Oracle databases and allows you to measure I/O performance for storage systems.

Caching

Whether using Oracle on Amazon EC2 or Amazon RDS, Oracle users confronted with heavy workloads should look into reducing this load by caching data so that the web and application servers do not have to repeatedly access the database. There are several tools (including tools from Oracle) that can address your caching needs:

- **Memcached:** An open-source, high-performance, distributed memory object caching system. It is an in-memory key-value store for small chunks of arbitrary data (strings, objects) such as results of database calls. Memcached is widely adopted and mostly used to speed up dynamic web applications by alleviating database load.
- **Amazon ElastiCache:** A web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service 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 is protocol-compliant with Memcached, so code, applications, and popular tools that you use today with existing Memcached environments will work seamlessly with the service. ElastiCache simplifies and offloads the management, monitoring, and operation of a Memcached environment, enabling you to focus on the differentiating parts of your applications.

- **Oracle In-Memory Database Cache:** A cache built using Oracle TimesTen In-Memory Database that can be deployed in the application tier as an in-memory cache database. Applications perform read/write operations on the cache tables using SQL and PL/SQL with automatic persistence, transactional consistency, and data synchronization with the underlying Oracle database. The writes on the cache can be synchronously or asynchronously replicated to the database.
- **Oracle Coherence:** A highly scalable and fault-tolerant distributed cache engine. It's an in-memory data grid designed to improve reliability, scalability and performance. You can provision and configure a Coherence cluster that caches data, and automatically and transparently fails over and redistributes its data management services when a server becomes inoperative or is disconnected from the network.

Database Replicas

A technique to provide higher performance is to spread the database query load across multiple instances. This technique is often referred to as *scaling out* or *horizontal scalability*.

Creating Read Replicas in Amazon RDS

Amazon RDS currently does not support read-replicas for Oracle. To increase your database throughput, you can scale vertically (use larger instance types). Alternatively you can also "shard" your database (create horizontal partitions of your database across multiple Amazon RDS instances). Data could be shared based on real-world criteria (such as product type and customer geography) or it could be spread across multiple shards using a hashing algorithm.

Creating Read Replicas in Amazon EC2

Oracle Active Data Guard is an Oracle Database add-on, which allows you to set up standby databases that can be open for read-only requests, while continuing to archive transactions from the primary database. The standby databases can be used as read replicas of your primary database. The replication between the primary and the standby databases can be configured to be synchronous. This allows you to scale your database layer horizontally by adding read replicas and to offload read-only queries from the primary database. This setup is often valuable because most applications generate more reads to the database than writes. Also, read-heavy clients like business intelligence applications can be executed against a standby instance, with no impact on the primary production database.

You can use Oracle Active Data Guard to build an elastic database infrastructure. By monitoring the usage of the primary database with Amazon CloudWatch, you can receive notifications indicating that a heavy load threshold has been met or exceeded. In this situation, you can create on-demand new standby databases to lower the load on the primary. Once this heavy usage period is over, standby instances and the resources they consume can be disposed.

> **Note:** Oracle Active Data Guard is only available for Oracle Database Enterprise Edition, not for Standard Edition and Standard Edition One.

It is also possible to use active-active replication to boost performance. In this scenario, you create one or more database replicas that can be both written to and read from, in effect implementing a distributed database where all replicas are synchronized. These technologies are covered in the "High Availability" section.

Figure 2 shows Oracle Active Data Guard being used to create a scalable reader farm made of several read replicas.

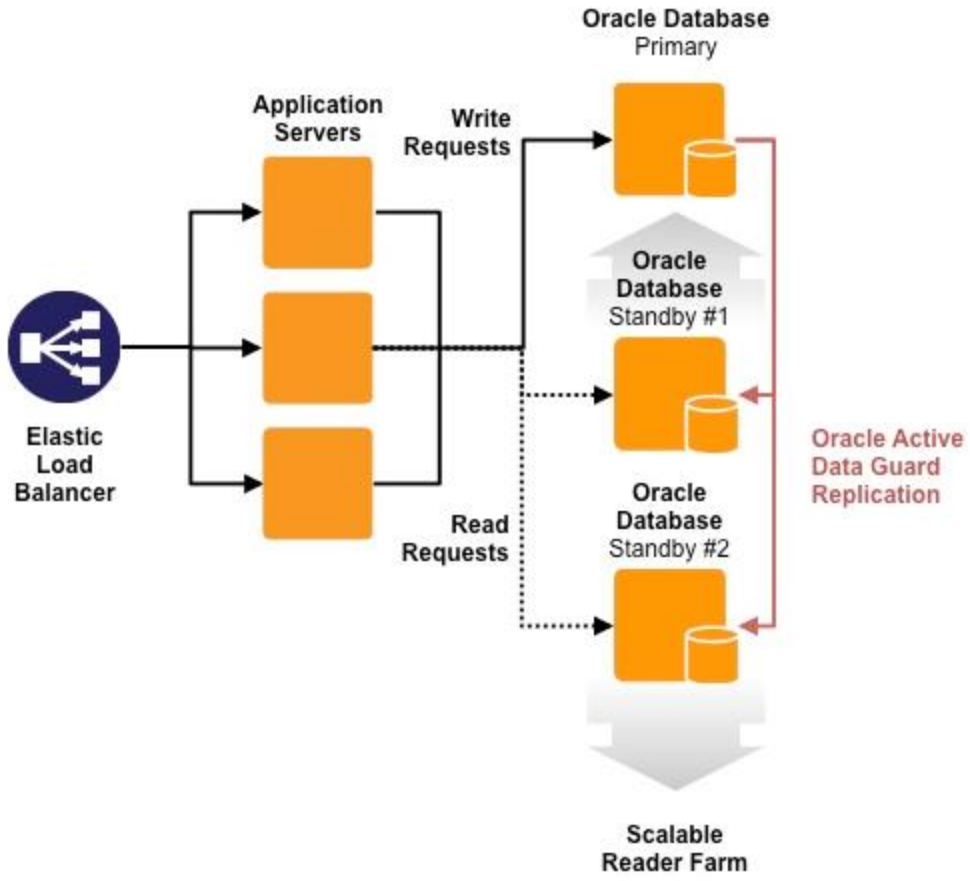

Figure 2: Oracle Active Data Guard being used to create a scalable reader farm

High Availability

By combining the tools provided to you by Oracle and by AWS, you can build highly available databases that will be a solid foundation for your applications.

High Availability Features in AWS

AWS provides you with several features to build highly durable and highly available database solutions. The following table describes some of the features and scope.

Scope	High Availability Features
Storage	Amazon EBS volumes are highly durable, high-performance, network-attached block device resources. These virtual disks can be attached to your instances, and can persist when instances are stopped or terminated, thus providing highly durable storage for databases. Each storage volume is automatically replicated within the same Availability Zone, which prevents data loss due to failure of any single hardware component. It is very important however to take regular snapshots of your EBS volumes for several reasons:

	• Snapshots are the easiest way to back up the data contained in your EBS volumes and Amazon RDS databases. • EBS volumes that operate with 20GB or less of modified data since their most recent Amazon EBS snapshot can expect an annual failure rate (AFR) of between 0.1% and 0.5%. EBS volumes with no snapshots, or modifications greater than 20GB since their last snapshot was taken, will be less durable. • EBS volumes are contained within a given Availability Zone. The snapshots are stored in Amazon S3 and are designed to survive the concurrent loss of data in two facilities. In the unlikely event of the failure of a full Availability Zone, you would be able to create new volumes from your most recent snapshots.
Backup	EBS snapshots and Amazon RDS backups are stored in Amazon S3, which is designed to provide 99.999999999% durability, and can sustain the loss of a full Availability Zone. Backups are discussed more in the "Backup and Restore" section.
Instance	You can detach volumes and re-attach to new instances of Amazon EC2 servers in the event of an instance failure. You can also quickly start new instances.
Region	You can create zone independence by utilizing multiple Availability Zones. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones.
Global	Since AWS is global, you can implement disaster recovery architectures spanning multiple regions.
System	The Amazon EC2 API control layer design supports redundancy and fault tolerance. The Amazon EC2 API has a 99.95% Annual Uptime Percentage SLA.

High Availability Features in Oracle

The Oracle database engine provides a variety of features to enhance the availability of your databases.

The following features can be used both in Amazon RDS and in Amazon EC2:

- Oracle Flashback technologies enable reversing human errors by selectively and efficiently undoing the effects of a mistake. Amazon RDS and Amazon EC2 supports multiple types of data recovery:
    - Flashback Table recovers tables to a specific point-in-time, which can be helpful when a logical corruption is limited to one or a set of tables instead of the entire database.
    - Flashback Transaction Query allows you to see all the changes made by a specific transaction.
    - Flashback Query lets you query any data at some point-in-time in the past.
- Total Recall is based on the Flashback feature and allows users to query data "AS OF" an earlier time in the past. This allows companies to "archive" data for auditing and regulatory compliance. The main difference between

Flashback and Total Recall is that with Total Recall data will be permanently stored in an archive tablespace and will only age out after a user-defined retention time.

In addition, the following features can be used if you run your Oracle database on Amazon EC2:

- In addition to the Flashback capabilities described in the previous list, the following additional Flashback features are also available:
    - Flashback Drop recovers an accidentally dropped table.
    - Flashback Transaction lets you undo the effects of a single transaction.
    - Using Flashback Database, you can restore the entire database to a specific point-in-time, using Oracle-optimized flashback logs (without using a backup). Amazon RDS has similar functionality with point-in-time restore.
- Online Data Reorganization and Definition offers you the flexibility to modify table physical attributes and transform both data and table structure while allowing users full access to the database. For example, you can move a table to a different tablespace, partition it, add or drop a column, and create and rebuild indexes, while the table remains open for read and write operations.
- Transportable Tablespaces is a feature that allows you to quickly move a user tablespace across Oracle databases. It can help you reduce database upgrade time by moving all user tablespaces from a database running an earlier software release to an empty destination database running a current software release.
- Edition-based Redefinition allows you to achieve online applications upgrades. Code changes are installed in the privacy of a new edition. Data changes are made safely by writing only to new columns or new tables not seen by the old edition. An “editioning” view exposes a different projection of a table into each edition to allow each to see just its own columns. A cross-edition trigger propagates data changes made by the old edition into the new edition’s columns, or vice-versa.

In addition to these engine features, you should design an architecture that protects you against hardware failures, datacenter problems, and disasters by using replication technologies. We will review how to set up Oracle replication in the following sections.

High Availability Architecture in Amazon RDS

Amazon RDS makes it easy to create a high availability architecture. First, Amazon RDS will automatically replace the compute instance powering your deployment in the event of a hardware failure. Second, Amazon RDS supports Multi-AZ (Availability Zone) deployments, where a secondary Oracle database is provisioned in a different Availability Zone within the same region. This architecture allows the database to survive the failure of the database instance, network and storage, or even of the Availability Zone. The rest of the infrastructure, including application and web servers, should also be deployed in at least two Availability Zones to make sure that your applications continue to operate in the event of a failure of an Availability Zone. In the design of your High Availability implementation, you can also take advantage of Elastic Load Balancing, which will automatically distribute the load across multiple Availability Zones if the instances attached to it (for example, web servers) are located in multiple Availability Zones. The replication between the two Oracle instances is synchronous, helping to ensure that all the data written to disk on the primary instance is replicated to the second instance. This feature is available for all editions of Oracle, including the ones that do not include Oracle Data Guard. This allows customers out-of-the-box high availability at a very competitive cost. Figure 3 shows an example of a high availability architecture in Amazon RDS.

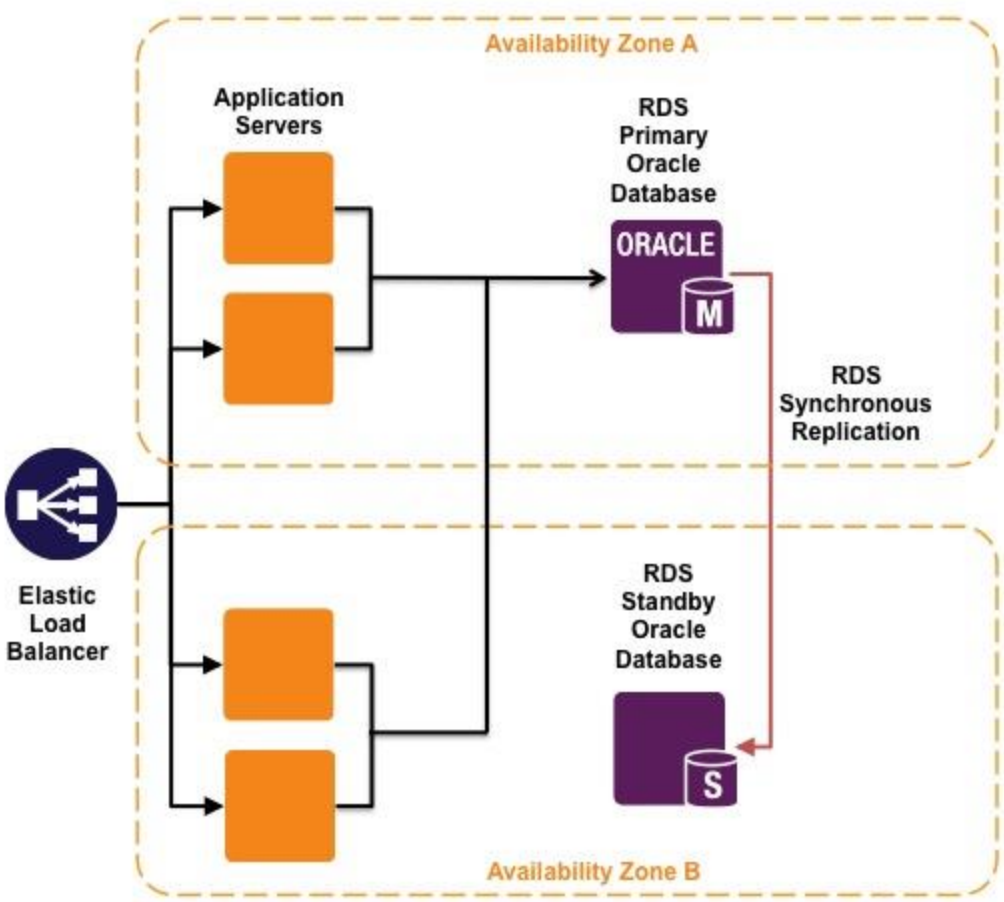

Figure 3: High availability architecture in Amazon RDS

A failover to the standby typically takes three minutes, and will occur in the event of any of the following:

- Loss of availability in primary Availability Zone
- Loss of network connectivity to primary
- Compute unit failure on primary
- Storage failure on primary
- Scaling of the compute class of your DB Instance up or down
- Software patching

To choose the Multi-AZ option, all you have to do is to enable it while creating the database. Figure 4 shows how it looks in the AWS Management Console.

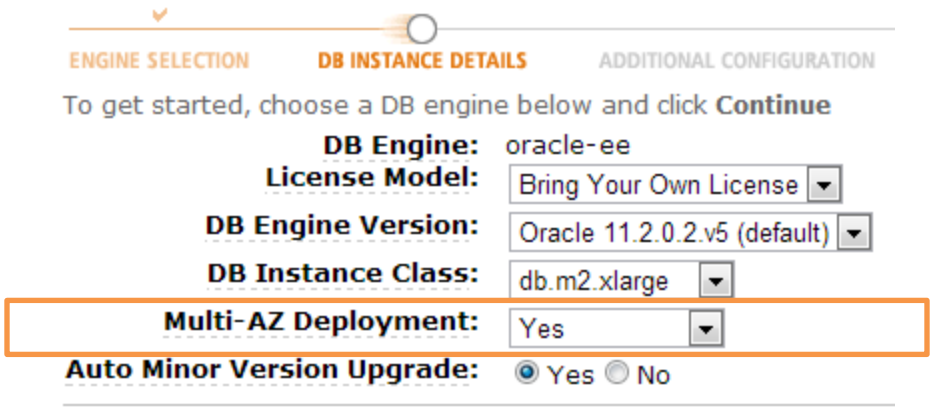

Figure 4: Multi-AZ Deployment option

You can also convert an existing Oracle RDS database to the Multi-AZ mode.

Running Amazon RDS in Multi-AZ has additional benefits:

- The Amazon RDS daily backups are taken from the standby instance, which means that there is no I/O impact to your primary Amazon RDS instance.
- When the database engine needs to be upgraded (and if you enabled the automatic minor version upgrade), the patches are applied to the standby instance first. When complete, the standby is promoted to be the new primary. The availability impact is then limited to the failover time, resulting in a shorter maintenance window.

High Availability Architecture in Amazon EC2

In a similar fashion to Amazon RDS Multi-AZ, you can increase the availability of your Oracle Database by provisioning a standby database, preferably in a different Availability Zone. If the primary database fails, you can then failover to the standby database, which then becomes the new primary. Oracle offers two related products to set up standby databases:

- Oracle Data Guard allows for setting up several standby databases. A standby database is a transactionally consistent copy of an Oracle production database that is initially created from a backup copy of the primary database. Once the standby database is created and configured, Oracle Data Guard automatically maintains the standby database by transmitting primary database redo data to the standby system, where the redo data is applied to the standby database. You can set up three kinds of standby databases with Oracle Data Guard:
    - **Physical:** A physical standby database replicates the exact contents of its primary database. The data in the database will be exactly the same as the primary database.
    - **Logical:** A Logical standby database converts the redo generated at the primary database into data and SQL and then re-applies those SQL transactions on the logical standby. Thus, physical structures and organization will be different from the primary database. Users can read from the logical standby databases while the changes are being applied and can even write to tables in the logical standby database that are not being maintained by the replication. However there are a number of unsupported objects that can make this mode unusable for some databases.
    - **Snapshot:** A snapshot standby database receives and archives, but does not apply, redo data from a primary database, so it cannot be used for read replicas or for high availability.

Because of the limitations of the logical and snapshot standby databases, we will focus on physical standbys in this discussion.

Oracle Data Guard maintains the standby databases as transaction-consistent copies of the primary database, and the replication between the primary and the standby databases can be configured to be either synchronous or asynchronous. Oracle Data Guard has three protection modes that allow you to maximize protection, availability or performance. To take full advantage of the AWS environment, these instances should be placed in distinct Availability Zones. If the production database becomes unavailable, you can switch any standby database to become the new primary, which minimizes the downtime associated with the incident.

Oracle Data Guard does not support setting up read replicas because the physical standby database cannot be open for reads while, at the same time, archiving transactions from the primary database. Oracle Data Guard can function either in managed-recovery mode or in read-only mode, but not in both modes at the same time. Oracle Data Guard is meant for data protection, high availability and disaster recovery.

- Oracle Active Data Guard, which we described previously in the "Creating Read Replicas in Amazon EC2" section, is an option built upon Oracle Data Guard that allows for the setup of read replicas. In addition to the functionality described previously, Oracle Active Data Guard enables read-only access to the standby databases while at the same time being kept up-to-date by archiving transactions from the primary database. This allows you to run read queries and reports on the standby instances, and to perform your backups from a standby instance.

Both Oracle Data Guard and Oracle Active Data Guard are often used as the foundation of highly available Oracle environments, allowing you to set up one or several slave databases to which the primary database can failover in the event of a failure or a disaster. Figure 5 shows an example of this architecture.

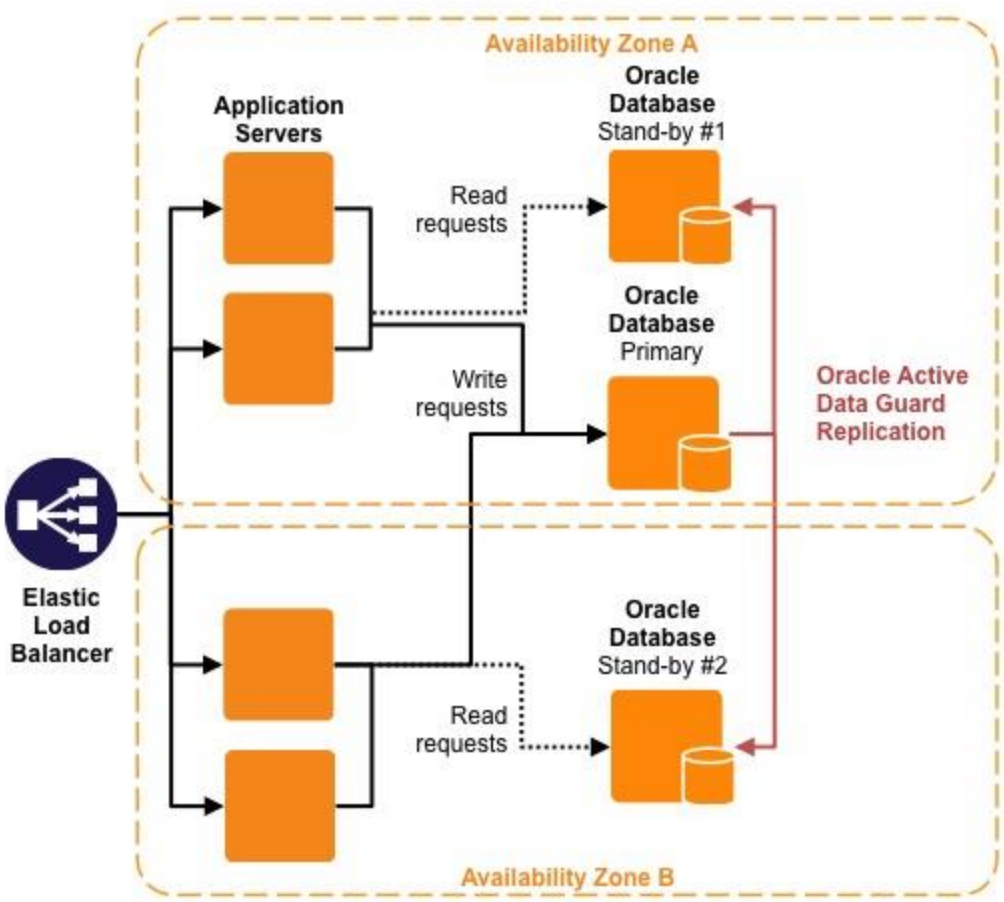

Figure 5: High availability architecture for Oracle

Third-Party High Availability Tools

Besides Oracle Data Guard, there are other third-party tools that you can use to set up a highly available database on Amazon EC2. For example, Dbvisit Standby can be used to create and managed Oracle standby databases for Oracle standard and enterprise editions.

Replication

Replication is the process of copying and maintaining database objects, such as tables, in multiple databases that make up a distributed database system. Changes applied at one database are captured and stored locally before being forwarded and applied at each of the remote databases. Replication uses distributed database technology to share data between multiple sites and the same data is available at multiple locations. Replication can increase availability of applications because alternate data access options are available. For example, the application can continue to function if parts of the distributed database are down as replicas of the data might still be accessible. In Amazon EC2, replication could be used to setup multiple copies of a database across multiple Availability Zones, or could be used as part of a disaster recovery strategy to replicate data across multiple AWS regions. While the primary focus of this section is high

availability, the replication technologies described in the following list can also be used to increase performance by spreading the workload across multiple databases:

- Oracle Basic and Advanced Replication

  Oracle Basic replication can replicate data, but cannot replicate other objects such as procedures and indexes. Replication is one-way and snapshot copies are read-only.

  Oracle Advanced replication supports multi-master configuration and allows for data to be updated on any replicated instance. It allows data and other database objects, like indexes and procedures, to be replicated.

- Oracle GoldenGate

  Oracle GoldenGate is a high-performance software application for real-time transactional change data capture, transformation, and delivery. It includes log-based bidirectional data replication. It can help organizations eliminate the downtime caused by both unplanned and planned outages, and improve system performance and scalability. The software can be configured to minimize downtime during system upgrade, migration, and maintenance activities. It can also be used in the context of disaster recovery by creating and maintaining an immediate failover with up-to-the-minute data. It synchronizes data for distributed applications in real-time across geographies. For AWS, this could be Availability Zones within a region or across different regions.

- Third-party replication solutions

  Besides the Oracle tools mentioned previously, there are several third-party solutions which you can use to set up replication between several Oracle databases. Dbvisit Replicate and Quest SharePlex for Oracle are examples of third-party solutions.

Backup and Restore

Depending on how you deployed your Oracle database, several backup strategies are available.

Backup and Restore on Amazon RDS

One of the important benefits of Amazon RDS is that it automates the backup and recovery process, and also gives you the ability to take additional point-in-time snapshots. You do not need any backup and recovery expertise and can rely on AWS to perform these tasks for you.

Automated Backups

When automated backups are turned on for your DB Instance, Amazon RDS automatically performs a full daily snapshot of your data (during your preferred backup window) and captures transaction logs as updates to your DB Instance are made. You can retain your automated database backups for up to 35 days. During the backup window, storage I/O may be suspended while your data is being backed up. This I/O suspension typically lasts a few minutes at most. This I/O suspension is avoided with Multi-AZ DB deployments, as in this case the backup is taken from the standby instance.

Automated Point-in-Time Recovery

Being able to recover a database from a backup is critical, but can be complicated when you are running your database on your premises. The automated backup feature of Amazon RDS enables point-in-time recovery for your DB Instance through the AWS Management Console or with the API. Amazon RDS backs up your database logs, transaction logs, and store for a user-specified retention period. This allows you to restore your DB Instance to any point in time during your

retention period, up to the last five minutes. Your automatic backup retention period can be configured to up to 35 days.

Database Snapshots

DB Snapshots are user-initiated backups of your DB Instance. These full database backups will be stored by Amazon RDS until you explicitly delete them. You can create a new DB Instance from a DB Snapshot whenever you want, for example to create a development or a test database identical to the production database or to upgrade your production database. Note that when you perform a restore operation to a point in time or from a DB Snapshot, a new DB Instance is created with a new endpoint.

Backup and Restore in Amazon EC2

When you deploy an Oracle database on Amazon EC2 instances, you are responsible for database backups just like you would be if you deployed it on your premises.

Oracle Recovery Manager (RMAN)

Oracle RMAN (Recovery Manager) is the backup and recovery manager tool provided by Oracle. It includes both command-line and graphical interfaces. Most Oracle users use Oracle RMAN to back up their database. AWS gives you several options as to where to store your backups.

Backup to Amazon S3

Amazon Simple Storage Service (Amazon S3) is Amazon's storage cloud. It provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, using authenticated web services requests. Amazon S3 has been designed for 99.999999999% durability, which makes it an excellent choice for a backup repository. Your Amazon S3 objects are redundantly stored on multiple devices across multiple Availability Zones in the Amazon S3 region where your database is running. Availability Zones are distinct locations within a region that are engineered to be isolated from failures in other Availability Zones, which means that your backups would still be available if a failure occurs in one of the Availability Zones of the region where the backups are stored. In addition to replicating the stored objects several times, Amazon S3 regularly verifies the integrity of data stored using checksums. If a corruption is detected, it is repaired using redundant data.

Backup to Amazon S3 using Oracle Secure Backup Cloud Module

You can back up your Oracle database directly to Amazon S3 by using Oracle Secure Backup (OSB) Cloud Module, which is a Media Management Library (MML) for RMAN. OSB Cloud Module integrates with Oracle Enterprise Manager Database Control and Grid Control (which will be described later in the “Monitoring and Management” section), and with the RMAN CLI, which allows you to write customized RMAN scripts. OSB Cloud Module supports Oracle Database 9i Release 2 or higher, including Oracle Database 11g.

There are multiple advantages to using this feature when running your database on Amazon EC2:

- OSB Cloud Module Backups stored on Amazon S3 are easily accessible to your Amazon EC2 instances. The ability to have immediate access to your backups substantially reduces database restoration times.
- OSB allows you to encrypt backups to help provide data security, and you also benefit from RMAN compression capabilities.
- Using Amazon S3 means having a geo-redundant and highly durable storage service for database backups at a very affordable price.

- You benefit from the virtually unlimited capacity of Amazon S3.

Note that the Oracle Amazon Machine Images provided by Oracle already includes the Oracle Secure Backup Cloud module install tool. If you used an AMI to provision your database, you can find the install tool in the/home/oracle/scripts/osbws directory. If not, you can download it from the Oracle Technology Network (OTN) website.

Backup to EBS Volumes

As explained previously in the "Performance Management" section, Amazon EBS volumes are off-instance block devices that persist independently from the life of the instance, and can directly attach to your instances. Using RMAN to back up your database to EBS volumes is the closest thing to backing up to disk in an on-premise environment, and an alternative to using the Cloud Backup module described in the previous section. Another benefit is that once your backups to EBS volumes are done, you can create snapshots of your EBS volumes through the AWS Management Console, CLI or API. EBS volumes snapshots are stored in Amazon S3, which provides a highly durable and reliable repository for your data.

EBS Snapshots

If your database is stored in EBS volumes, you can, in addition to RMAN backups, put your tablespaces in hot backup mode and take snapshots of the underlying Amazon EBS volumes. EBS snapshots can be done through the AWS Management Console, the CLI and the API.

Long-Term Archival in Amazon Glacier

Amazon Glacier is an extremely low-cost storage service that provides highly secure and highly durable storage for data archiving and backup. Amazon Glacier is very similar to Amazon S3 and is also designed for 99.999999999% durability. Two key differences between Amazon S3 and Amazon Glacier are:

- Amazon Glacier has significantly lower storage costs than Amazon S3.
- Amazon Glacier is optimized for data that is infrequently accessed and for which retrieval times of several hours are suitable.

Amazon S3 and Amazon Glacier are integrated in such a way that you can store your backups in Amazon S3 and have them automatically moved to Amazon Glacier after a period of time that you specify for long-term archiving. You can also specify an expiration time that defines how long the backups will be kept in Amazon Glacier. For example, you could store each new database backup in Amazon S3 for 15 days (or any period of time during which you are most likely to use it to restore your database). You could then have the backups automatically moved to Amazon Glacier after 15 days, with an expiration time of 365 days, which would cause the backup to be removed from Amazon Glacier after a year. This would allow you to store your older backups at a minimal cost while being able to retrieve them within a few hours if needed.

Monitoring and Management

Amazon RDS and Amazon EC2 provide several monitoring and management tools.

Amazon RDS Monitoring

Amazon CloudWatch collects many Amazon RDS-specific metrics. You can look at these metrics using the graphical interface in the AWS Management Console, the command line (using the `mon-get-stats` command) or the API. In addition to the system-level metrics collected on Amazon EC2 instances (such as CPU usage, disk and network I/O), the

Amazon RDS metrics include many database-specific metrics, such as database connections, free storage space, read and write I/O per second, read and write latency, read and write throughput, and available RAM. For a full list of Amazon RDS metrics, see the CloudWatch documentation at http://docs.amazonwebservices.com/AmazonCloudWatch/latest/DeveloperGuide/CW_Support_For_AWS.html#rds-metricscollected. We also wrote an Amazon RDS monitoring guide, available at http://aws.amazon.com/articles/2934.

In addition, you can use Oracle Enterprise Manager 11g Database Control (OEM) when you enable the OEM Database Control option for your DB Instance, which can be done in the AWS Management Console or with the API. For example, if you chose the default OEM port 1158 (and configured the database security group to allow this traffic to get through the firewall), you would be able to access OEM through https://<your-RDS-CNAME>:1158/em to manage and monitor your database.

Amazon RDS Built-in Management Features

In addition to the management features that we already reviewed, such as automated backup and recovery, DB snapshots, automated storage management, push-button scaling and replication, Amazon RDS manages several other tasks:

- **Automatic host replacement:** Amazon RDS will automatically replace the compute instance powering your deployment in the event of a hardware failure.
- **Automatic minor version upgrade:** Amazon RDS will keep your database software up to date. The patching activity will occur during the weekly, 30-minute maintenance window that you specify when you provision your database (and that you can alter at any time). Such patching occurs infrequently, typically every few months. Your database will become unavailable during part of your maintenance window. You can disable this capability if you prefer to apply the database patches yourself. You can also minimize the downtime associated to automatic patching if you run in Multi-AZ mode. In this case, the maintenance is performed on the standby. When it is complete, the standby is promoted to primary. The maintenance is then performed on the old primary, which becomes the standby.
- **Pre-configured parameters:** Amazon RDS chooses an optimal configuration for your Oracle database configuration parameters, based on the size of your database and of the instance that you selected. These parameters are stored in a DB Parameter Group. You can choose your own parameters by creating custom DB Parameter Groups.

Amazon RDS Database Administration

In order to provide you with a secure and stable managed database experience, Amazon RDS does not provide shell access to DB Instances, and it restricts access to certain system procedures and tables that require advanced privileges.

Managing users, roles and privileges in Amazon RDS Oracle instances works almost the same way as in self-managed Oracle instances. Oracle commands, such as create user, rename user, grant, revoke and set password, work as they do in Amazon EC2 (or on-premise) databases. The master user has the DBA role. One difference is that the master user cannot run the following statements in Amazon RDS:

- Alter database
- Alter system
- Create any directory

- Drop any directory
- Grant the “any privilege” privilege (the master user can grant privileges, just not the “any privilege” privilege)
- Grant any role

Instead, Amazon RDS provides wrapper procedures for many common DBA tasks that require advanced privileges, including:

- Setting the time zone of the database
- Managing tablespaces
- Flushing the shared pool or the buffer cache
- Switching online log files
- Checkpointing the database
- Managing the redo logs
- Accessing the alert and listener logs
- Managing tracefiles
- Killing a session

For an exhaustive list of all the administrative commands that you can run in Amazon RDS, see the Amazon RDS documentation at
http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.CommonDBATasks.html

Monitoring and Management in Amazon EC2

Oracle provides the several monitoring tools. Oracle Enterprise Manager Database Control allows you to monitor and manage your databases individually. For managing many databases running in Amazon EC2, you could use Oracle Enterprise Manager Grid Control. Both tools allow you to detect and get notified of impending database problems in a timely manner. Using the Diagnostics and Tuning packs, you can identify root causes of various issues with the database (such as hanging sessions and long running queries) as well as get recommendations on optimizing and tuning database performance.

Amazon CloudWatch is an AWS instance monitoring service that provides detailed CPU, disk, and network utilization metrics for each Amazon EC2 instance and EBS volume, which allows for detailed reporting and management. This data is available in the web-based AWS Management Console as well as the API. This allows for infrastructure automation and orchestration based on the availability and load metrics.

In addition, you can use any third-party monitoring tools that have built-in Oracle monitoring capabilities, such as the open-source monitoring frameworks Nagios and Zabbix, and run them on Amazon EC2 to monitor your whole AWS environment, including your Oracle databases.

The management of an Oracle database on Amazon EC2 is similar to the management of an on-premise database. You can use Oracle Enterprise Manager and command-line tools to perform administration tasks.

Security

There are different ways that you can control access to your Oracle database. You can use Amazon VPC, security groups, and Oracle security features.

Amazon VPC

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a private, isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define. Provisioning your Oracle databases in Amazon VPC gives you more options to secure your Oracle environment and more flexibility in the management of your network topology. You can use Amazon VPC both for Amazon RDS and for deploying your database on Amazon EC2 instances. Here are some key advantages of deploying your databases in Amazon VPC:

- You create your own subnets and you can configure routing tables, networking gateways and network Access Control Lists (ACL).
- You can define different inbound and outbound rules in an Amazon VPC security group.
- You can deploy your databases on private subnets that are not reachable from the Internet. Your databases run in a secure, private environment, where they can be accessed by web servers and/or application servers, but not from the Internet. You can choose whether you want to have your web and application servers on public or private subnets.
- You can establish IPSEC VPN connections between AWS and your own datacenters, which helps to safely encrypt the network traffic between AWS and the customer premises. This allows your administrators and you applications to securely access your databases from your internal network.

In addition, the following features are applicable to Amazon EC2 in Amazon VPC, but not to Amazon RDS in Amazon VPC:

- The private IP addresses of your Amazon EC2 instances are persistent and you can choose them. The permanent private IP address associated to an Amazon VPC instance facilitates the configuration of the application servers that access the database.
- If your databases are deployed on private subnets that are not reachable from the Internet, you can use Network Address Translation (NAT) to let your database instances access the Internet to download patches or new software. With NAT, your instances have outbound Internet access, but are unreachable from the Internet.
- You can assign several network interfaces (called Elastic Network Interfaces), and multiple IP addresses per interface, so that your database instances can have several private (and public) IP addresses. Your Oracle database could be part of several subnets. For example, you can have an application subnet that is reachable by your application servers, a backup subnet, and an administration subnet.

If you want to use Amazon VPC, you have three options:

- Use Amazon RDS.
- Use an Oracle AMI running on Xen virtualization. You can find out what hypervisor is an AMI designed for, for example by parsing the output of the CLI command `ec2-describe-images -v <AMI ID>` and looking for the value of the `hypervisor` field. Oracle AMIs are either designed to run on the Xen hypervisor or on the Oracle VM (OVM) hypervisor. Since OVM is currently not supported in Amazon VPC, you can choose an Oracle AMI based on Xen.

- Install Oracle on top of Linux AMI running on Xen virtualization.

Amazon VPC offers many more features than listed here. For more information, see the Amazon VPC documentation at http://aws.amazon.com/vpc/.

Oracle Security in Amazon RDS

Amazon RDS allows you to control network access to your DB Instances using security groups. By default, network access is turned off to your DB Instances.

Using AWS Identity and Access Management (IAM), you can manage access to your Amazon RDS databases. For example, you can authorize (or deny) administrative users under your AWS account to create, describe, modify, or delete an Amazon RDS database. You can also enforce Multi-Factor Authentication (MFA). For more information on using IAM to manage administrative access to Amazon RDS, see http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAM.html.

In addition, Amazon RDS supports several Oracle security features under the Bring Your Own License (BYOL) model:

- Transparent Data Encryption (TDE) protects data at rest. It provides transparent encryption of stored data to support your privacy and compliance efforts. Applications do not have to be modified and will continue to work as before. Data is automatically encrypted before it is written to disk and automatically decrypted when reading from storage. Key management is built-in, which eliminates the task of creating, managing, and securing encryption keys. You can choose to encrypt tablespaces or specific table columns using industry standard encryption algorithms including Advanced Encryption Standard (AES) and Data Encryption Standard (Triple DES).
- Native Network Encryption protects data in motion using the Oracle Net Services. You can choose between AES, Triple DES, and RC4 encryption.
- Oracle Virtual Private Database (VPD) enables you to create security policies to control database access at the row and column level. Essentially, Oracle VPD adds a dynamic WHERE clause to a SQL statement that is issued against the table, view, or synonym to which an Oracle Virtual Private Database security policy was applied. Oracle Virtual Private Database enforces security, to a fine level of granularity, directly on database tables, views, or synonyms. Because you attach security policies directly to these database objects and the policies are automatically applied whenever a user accesses data, there is no way to bypass security.
- Fine Grained Auditing (FGA) can be understood as policy-based auditing. It lets you specify the conditions necessary for an audit record to be generated. FGA policies are programmatically bound to a table or a view. They allow you to audit an event only when conditions that you define are true, for example only if a specific column has been selected or updated. Since every access to table is not always recorded, this creates more meaningful audit trails.

Oracle Security in Amazon EC2

When running Oracle on Amazon EC2 instances, you have the responsibility to effectively protect network access to your instances with security groups, adequate operating system settings, and best practices, such as limiting access to open ports and using strong passwords. In addition, you can also configure a host-based firewall on your instances.

Using AWS Identity and Access Management (IAM), you can control access to your Amazon EC2 resources, and authorize (or deny) some users the ability to manage your instances running the Oracle database and the corresponding EBS volumes. For example, you could restrict the ability to start or stop your Amazon EC2 instances to a subset of your administrators. You can also assign Amazon EC2 roles to your instances, giving them privileges to access other AWS

resources that you control. For more information on how to use IAM to manage administrative access to your instances, see http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/UsingIAM.html.

You also have the option of deploying most Oracle security features, such as:

- The Oracle security features described previously in the "Oracle Security in Amazon RDS" section.
- Encrypted backups as mentioned previously in the "Backup and Restore" section. The Oracle backup utility, RMAN, can transparently encrypt data written to backup sets and decrypt those backup sets when they are needed for database restore operations. To create encrypted backups, the database must be configured to use Advanced Security or Oracle Secure Backup. Note that Oracle Secure Backup includes Cloud Module, which allows you to back up your database directly to Amazon S3, as described in the "Backup and Restore" section.
- Oracle Label Security is a tool for classifying data and for managing access to data on a "need to know" basis.
- Oracle Database Vault restricts access to specific areas in an Oracle database from any user, including users who have administrative access. For example, you can restrict administrative access to critical data such as employee salaries, customer medical records, or other sensitive information.

This list is not exhaustive. Oracle has many other security features that are not listed here and can also be used in Amazon EC2.

AWS for On-Premise Oracle Environments

This section explores how you can take advantage of AWS for your on-premise Oracle environments and how you can migrate your existing in-house Oracle environments to AWS.

Backing up On-Premise Oracle Databases in AWS

Because Amazon Simple Storage Service (Amazon S3) is accessible not only from Amazon EC2, but from anywhere on the web over HTTP and HTTPS, it is possible to back up your on-premise databases, wherever they are running, into Amazon S3. This allows you to automate your off-site backup retention process, rather than (or in addition to) relying on offsite tape storage. As stated previously in the "Backup and Restore" section, Amazon S3 is designed for 99.999999999% durability, which makes it a much more reliable medium for off-site backups than tapes. Using the integration between Amazon S3 and Amazon Glacier, you have your backups automatically transferred to the more cost-effective Amazon Glacier after a period of time that you specify, as described previously in the section "Long-Term Archival in Amazon Glacier". The Amazon S3 expiration feature allows you to set the retention duration of your backup at creation time.

There are several ways to automate backup an on-premise Oracle databases in AWS:

- **Use the Oracle Secure Backup Cloud Module:** The Oracle Secure Backup Cloud Module described in the "Backup and Restore" section works for on-premise databases just like it does for databases running on Amazon EC2. It can be used to have RMAN back up the database over the network to Amazon S3. For security purposes, RMAN can encrypt the backup, which can also be sent over an encrypted link to Amazon S3. RMAN can also compress the backup in order to increase network performance.
- **Use a storage gateway:** You can back up your on-premise databases locally to a storage gateway. The storage gateway is a physical or virtual appliance to which you allocate on-premise storage. You back up your databases to this local storage and take advantage of the low latency and high bandwidth due to the fact that the storage is on-premise. The storage gateway then securely moves the data to Amazon S3 over HTTPS. You can use the

AWS Storage Gateway, described at http://aws.amazon.com/storagegateway. You can also choose a third-party gateway, such as the ones provided by Aspera, Panzura, Riverbed and TwinStrata. Features of storage gateways may include deduplication, encryption, and compression of your data, in order to secure and to speed up the transfer to Amazon S3. You can use the storage gateway to move non-Oracle data, such as data stored in files.

- **Use the AWS Import/Export service:** If the database is very large, moving the backup or export files over the network might be time-prohibitive, even when using a storage gateway. In this case, it can be faster and more reliable to store an Oracle backup on local drives and physically ship them to AWS using the AWS Import/Export service. Upon reception of the drives, AWS engineers copy the data to Amazon EBS, Amazon S3, or Amazon Glacier (your choice). We then ship your drives back to you. For more information about the AWS Import/Export service, see http://aws.amazon.com/importexport/.

Disaster Recovery on AWS for On-Premise Oracle Databases

Amazon AWS is a cost-effective platform to host your Disaster Recovery (DR) site. Instead of having to invest in a parallel set of infrastructure that is used only a few times a year, you can just transfer your database backups to Amazon S3 (or Amazon Glacier), or replicate the on-premise production database to Amazon EC2. Whenever you need to test your DR processes, or whenever you need to failover to AWS because of a failure or a disaster at your primary site, you can recover your database and start up the rest of the application stack. At all times, you only pay for what you use on AWS. This can save a lot of money compared to hosting your DR site in a datacenter where you own or lease floor space and physical infrastructure. Typically, DR environments do not run very often and benefit tremendously from AWS pay-as-you-go model. Cost-wise, all of these scenarios also benefit from the fact that incoming data transfer into AWS is free. The following sections describe two cost-efficient scenarios to use AWS as a DR site.

Disaster Recovery from Backups Stored in AWS

Storing Oracle database backups in the Cloud, as described previously, can also be a first step towards implementing a Disaster Recovery (DR) environment on AWS. You transfer your database backups to Amazon S3 (or Amazon Glacier), using one of the scenarios described previously. In the event of a failure or disaster at your primary site, you can start Amazon EC2 instances, create EBS volumes for your Oracle data files, and recover your database from Amazon S3 with Oracle RMAN OSB Cloud Module. This is a good solution for smaller databases, if your Recovery Time Objective (RTO) can accommodate the time it takes to restore a full backup of your database, and if your Recovery Point Objective (RPO) can tolerate losing all the data created since the last backup was sent to AWS. While the RPO and RTO are relatively long in this scenario, they might be acceptable for some applications. It is infinitely preferable to have a DR plan with long RTO and RPO than having any DR plan at all, or to having a DR plan solely based on tapes. This scenario is very cost-effective because its cost is limited to the cost of storing your backups (and AMIs) in AWS, and should allow all the organizations that have resisted setting up a DR site for cost reasons to finally implement a DR solution.

Disaster Recovery using a Standby Database

In the “High Availability Architecture in Amazon EC2” section, we explained how to use a standby database to design a high availability solution across multiple AWS Availability Zones. The same strategy can be used for disaster recovery, between your premises and AWS. You can set up a standby database on Amazon EC2 using Oracle Data Guard (or a third-party product with similar functionality) to replicate data between the on-premise Oracle database and a database running in Amazon EC2. After the initial configuration of Oracle Data Guard on the source side, a full backup is shipped to AWS, as described in the “Backing up On-Premise Oracle Databases in AWS” section, and the slave database is restored in Amazon EC2. The databases are then synchronized by shipping the logs generated on the primary database during this window of time and applying them to the Amazon EC2 standby instance. Later, the log shipping and

replication continue until you are ready to switch over and make the Amazon EC2 database the primary instance. For a diagram and more details about how to set up a standby database on AWS, see the "High Availability Architecture in Amazon EC2" section. Note that the standby database running in Amazon EC2 can also be used to perform database backups to Amazon S3 for added protection.

In this scenario, the RTO (Recovery Time Objective) is minimized because in case of an outage, all you have to do is to failover. To failover, switch the standby database to become the primary and start the rest of your applications on Amazon EC2 using AMIs that you have already prepared and regularly tested. The RPO (Recovery Point Objective) is also minimized because the standby database can be configured to be only several minutes behind the primary database. Both the RTO and the RPO can be down to a few minutes. In short, this solution offers much improved RTO and RPO at a cost barely greater than in the previous, backup-based scenario.

Other Disaster Recovery Considerations

Even if your application is based on an Oracle database, there is probably more data stored in flat files that you need to replicate to AWS to make your application fully functional. You can do this by uploading your flat files over HTTPS to Amazon S3 using the AWS Management Console, the CLI or the API, or, as mentioned earlier, by using an AWS (or third-party) storage gateway.

You also need to build the rest of the environment around the database, for example application servers and web servers. You should create AMIs for all the servers that will need to be started in AWS in the event of a disaster, so that they are ready to be started the day you need them.

If your application is used on the Internet, you can modify the application DNS information to make it point to AWS. To prepare for this, you would make sure that the Time-To-Live (TTL) of your DNS records does not exceed your RTO. If your application is internal to your company, you can access it over an IPSEC-encrypted VPN link, as described previously in the "Amazon VPC" section.

Finally, you need to test your DR processes regularly to make sure that they are fully functional.

For more information and guidance about implementing Disaster Recovery solutions on AWS, see the "Using AWS for Disaster Recovery" whitepaper available at http://aws.amazon.com/whitepapers/.

Migrating your On-Premise Oracle Database to AWS

The Disaster Recovery strategies described in the previous section can be used to migrate your on-premise Oracle database into the AWS cloud. The following are some ways to migrate:

- **Upload files to Amazon S3 or Amazon EBS:** If the database is small or if can sustain extended downtime, an easy way to migrate it to AWS is to take a backup of the database and then to upload the backup or the export files to Amazon S3 or Amazon EBS, as described in the "Backing Up On-Premises Oracle Databases in AWS" and "Disaster Recovery from Backups Stored in AWS" sections. An alternative would be to export your database, using Oracle's export utility or Oracle Data Pump, and to upload the export files to Amazon S3 or Amazon EBS. Your database can then be restored or imported into an Oracle database running on Amazon EC2. Note that Amazon RDS supports Oracle Data Pump, which can be used to import an on-premise database over the network into Amazon RDS. For more information, see the Amazon RDS Data Import Guide for Oracle at http://aws.amazon.com/articles/4173109646282306.

- **Use the AWS Import/Export service:** If your database is large and can sustain extended downtime, it might be easier and faster to use the AWS Import/Export service and ship us disk drives containing a backup or an export of your database, as described in the "Backing up On-Premise Oracle Databases in AWS" section.
- **Use a standby database in Amazon EC2:** If the database cannot sustain extended downtime, an option is to set up a standby database in Amazon EC2, as described in the "Disaster Recovery using a Standby Database" section. A great advantage of this scenario is that it minimizes the amount of downtime that you will occur during the migration, down to a few minutes. After you have switched over to AWS, you can setup the on-premise database as the new standby, allowing you to temporarily switch back to your on-premise environment if an issue was discovered in your new AWS environment.

Managing Cost

Managing the cost of the IT infrastructure is often an important driver for cloud adoption. The cost advantages inherent to AWS should make running Oracle Database on Amazon EC2 a cost-effective proposition. Amazon RDS should further reduce your costs by reducing the management and administration tasks that you have to perform. In addition, you can take advantage of several strategies to manage your database cost effectively.

Reducing Cost with Reserved Instances

Amazon RDS and Amazon EC2 reserved instances allow you to lower costs and reserve capacity. Reserved Instances can save you up to 70% over On-Demand rates when used in steady state, which tend to be the case for many databases. They can be purchased for one or three year terms. If your Oracle database is going to be running more than 25% of the time, you will most likely financially benefit from using a reserved instance. Note that in Amazon RDS you cannot stop an instance, but instead you can take a final snapshot prior to termination, and recreate a new Amazon RDS instance from the snapshot when you need it. For more information about Reserved Instances, see http://aws.amazon.com/rds/reserved-instances/ and http://aws.amazon.com/ec2/reserved-instances/.

Other Options to Reduce Costs

The following options can be combined to reduce your cost:

- Use the License Included model to save on Oracle licenses where the Oracle license is included in the Amazon RDS hourly cost. This is especially effective for databases that are not running 24*7 and for short projects.
- Shutdown database instances when they are not needed. For example, some development and test databases could be shutdown at night and on weekends, and restarted on weekdays in the morning.
- Scale down the size of your databases during off-peak times.

Conclusion

AWS provides you with two deployment platforms to deploy your Oracle databases: Amazon EC2 and Amazon RDS. In this whitepaper we have explained how to manage performance, high availability, monitoring and security in both environments. Whether you choose to deploy Oracle in Amazon EC2 or in Amazon RDS, you will benefit from the advantages inherent to AWS, including:

- Ease and speed of provisioning Oracle instances and storage in AWS
- No capital expense

- High security
- High availability and durability
- Low cost
- Pay-as-you-go pricing

Further Reading

In addition to this whitepaper, we recommend that you consult the following documents and web sites:

- Amazon RDS Getting Started Guide:
  http://docs.aws.amazon.com/AmazonRDS/latest/GettingStartedGuide/
- Working with Oracle on Amazon RDS:
  http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Oracle.html
- Amazon EC2 Getting Started Guide:
  http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/
- AWS Security Center:
  http://aws.amazon.com/security/
- AWS Architecture Center:
  http://aws.amazon.com/architecture/
- AWS Cloud Computing Whitepapers:
  http://aws.amazon.com/whitepapers/