Determining the IOPS Needs for Oracle Database on AWS

Abdul Sathar Sait

December 2014

Contents

Abstract

Amazon Web Services (AWS) provides a comprehensive set of services and tools for deploying Oracle Database on the AWS cloud infrastructure, one of the most reliable and secure cloud computing services available today. A large number of businesses of all sizes use Oracle Database to handle their data needs. Oracle Database performance relies heavily on the performance of the storage subsystem, but storage performance always comes at a price. This whitepaper will help you determine the input/output operations per second (IOPS) needed by the storage system of your database for best performance at optimal cost.

Introduction

AWS offers its customers the flexibility to run Oracle Database on Amazon Relational Database Service (Amazon RDS), the managed database service in the cloud, as well as on Amazon Elastic Compute Cloud (Amazon EC2). Many customers prefer to use Amazon RDS for Oracle Database because it provides an easy, managed option to run Oracle Database on AWS without having to think about infrastructure provisioning or installing and maintaining database software. Alternatively, you can run Oracle Database directly on Amazon EC2, which allows you full control over setup of the entire infrastructure and database environment.

To get the best performance from your database, you must configure the storage tier to provide the IOPS and throughput needed by the database. This requirement applies equally to Oracle Database on Amazon RDS and to Oracle Database on Amazon EC2. If the storage system does not provide enough IOPS to support the database workload, you will have sluggish database performance and transaction backlog. On the other hand, if you provision much higher IOPS than your database actually needs you will have unused capacity.

The elastic nature of the AWS infrastructure does allow you to increase or decrease the total IOPS available for Oracle Database on Amazon EC2, but doing this has a performance impact on the database, requires extra effort, and might require database downtime. With Amazon RDS for Oracle, you can scale up IOPS but cannot scale down IOPS in the same instance. IOPS scale-up for RDS also is a time-consuming and resource-intensive process that affects the performance of the database. Thus, determining the IOPS and throughput your Oracle Database needs to some level of accuracy before setting up the database on AWS is a good thing to do.

Storage Options for Oracle Database

For Oracle Database storage on AWS, you need to use Amazon Elastic Block Store (Amazon EBS). Amazon EBS volumes offer the consistent low-latency performance needed to run your Oracle Database. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS provides three volume types: General Purpose (SSD), Provisioned IOPS (SSD), and Magnetic. The three volume types differ in performance characteristics and cost. For the high and consistent IOPS required for Oracle Database, Amazon EBS General Purpose (GP2) or Amazon EBS Provisioned IOPS (PIOPS) volumes will be the right fit.

For GP2 volumes, IOPS performance is directly related to the provisioned capacity. GP2 volumes can deliver a consistent baseline of 3 IOPS/GB up to a maximum of 3,000 IOPS for a 1 TB volume. I/O is included in the price of General Purpose (SSD) volumes, so you pay only for each gigabyte of storage you provision. GP2 volumes also have the ability to burst to 3,000 IOPS per volume, independent of volume size, to meet the periodic spike in performance needs found for most applications. This ability is a very useful feature for a database, where you can predict normal IOPS needs well but you might still find a higher spike from time to time based on specific workloads.

GP2 volumes are sufficient for most Oracle Database workloads. If you need more IOPS and throughput than GP2 can provide, then PIOPS is the right choice. PIOPS can provide up to 4,000 IOPS per volume and can achieve maximum IOPS with volume sizes as small as 400 GB.

Stripe multiple volumes together for more IOPS and larger capacity. Use only EBS optimized instances with GP2 and PIOPS. You can use multiple EBS volumes individually for different data files, but striping them together allows better throughput, balancing, scalability, and (for GP2) burstability.

IOPS Basics

IOPS is the standard measure of input and output operations per second on a storage device. It includes both read and write operations. The amount of I/O used by Oracle Database can vary a great deal within a time period, based on the load on the server and the specific queries being executed. If you are migrating an existing Oracle Database to AWS, then you need to determine the peak IOPS used by your database and provision Amazon EBS volumes on AWS accordingly so that you get the best performance even with peak load. If you choose an IOPS number based on average IOPS used by your existing database, then you will have sufficient IOPS to service the database in most cases but database performance will suffer at peak load. This issue can be mitigated to some extent by using Amazon EBS GP2 volumes, because they have the ability to burst to higher IOPS for small periods of time.

More often than not, customers assume they need much more IOPS than they actually do. This assumption occurs because customers confuse storage system IOPS with database IOPS. Most enterprises use storage area network (SAN) systems that can provide 100,000–200,000 or more IOPS for storage. The same SAN storage is usually shared by multiple databases and file systems, thus the total IOPS provided by the storage system is used by many more applications than a single database.

Most Oracle Database production systems fall in the range of 3,000–30,000 IOPS. A performance test environment's IOPS needs are generally identical to those of production environments, but for other test and development environments the range is usually between 200 and 2,000 IOPS. Some online transaction processing (OLTP) systems go up to 60,000 IOPS. There certainly are Oracle databases that use more than 60,000 IOPS, but that is not the norm. If you see numbers that are very different from these, it would be prudent to do further analysis to confirm your numbers.

The best way to estimate actual IOPS needed for your database is to query the system tables over a period of time and find the peak IOPS usage by the existing database.

Estimating IOPS for an Existing Database

As mentioned earlier, your goal in estimating IOPS should be to find peak IOPS for your database. Doing this requires measuring IOPS over a period of time and picking the highest value. The best way to get this information is from the gv$sysstat dynamic performance view. A *dynamic performance view* is a special kind of view made available in Oracle Database that provides database performance information. Such views are continuously updated while a database is open and in use. These views are also used by Oracle Enterprise Manager and Automatic Workload Repository (AWR) reports to gather data. There is a gv$ view for almost all v$ views; gv$ views contain data for all nodes in a Real Application Cluster (RAC) identified by an instance ID. You can also use gv$ views for non-RAC systems; in this case, the gv$ views will have only one row per performance criterion.

To determine IOPS, you can adopt the following sample Oracle PL/SQL script to fit your needs and run the script for a period of your choice during peak database load in your existing environment. For better accuracy, run this for the same peak period for a few different days and then choose the largest value from among the values obtained as the peak IOPS.

The script following will capture data and store it in a table named PEAK_IOPS_ MEASUREMENT, so you need to create the table first using the following code:

```
CREATE TABLE peak_iops_measurement (capture_timestamp date,
total_read_io number, total_write_io number, total_io number,
total_read_bytes number, total_write_bytes number, total_bytes
number);
```

The script following runs for an hour (*run_duration*) and captures data every 5 seconds (*capture_gap*). It then calculates the average IO and throughput per second for those 5 seconds and stores this information in the table. You can easily modify the values of *run_duration* and *capture_gap* to best fit your needs.

```
DECLARE

            run_duration number := 3600;
            capture_gap number := 5;
            loop_count number :=run_duration/capture_gap;

            rdio number;
            wtio number;
            prev_rdio number :=0;
            Prev_wtio number :=0;

            rdbt number;
            wtbt number;
            prev_rdbt number;
            Prev_wtbt number;


BEGIN
 FOR i in 1..loop_count LOOP
```

```
                SELECT SUM(value) INTO rdio from gv$sysstat
                WHERE name ='physical read total IO requests';

                SELECT SUM(value) INTO wtio from gv$sysstat
                WHERE name ='physical write total IO requests';

                SELECT SUM(value)* 0.000008 INTO rdbt from gv$sysstat
                WHERE name ='physical read total bytes';

                SELECT SUM(value* 0.000008) INTO wtbt from gv$sysstat
                WHERE name ='physical write total bytes';


IF i > 1  THEN
               INSERT INTO peak_iops_measurement (capture_timestamp,
total_read_io, total_write_io, total_io, total_read_bytes,
total_write_bytes, total_bytes)
VALUES (sysdate,(rdio-prev_rdio)/5,(wtio-prev_wtio)/5,((rdio-
prev_rdio)/5)+((wtio-prev_wtio))/5,(rdbt-prev_rdbt)/5,(wtbt-
prev_wtbt)/5,((rdbt-prev_rdbt)/5)+((wtbt-prev_wtbt))/5);

END IF;

                prev_rdio := rdio;
                prev_wtio := wtio;

                prev_rdbt := rdbt;
                prev_wtbt := wtbt;

                DBMS_LOCK.SLEEP(capture_gap);

 END LOOP;
 COMMIT;
 EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
END;
/
```

The values that are important from this table are *total_io* and *total_bytes;* the split of time spent in read and write operations is captured for any comparison you might want to do later. Once you have collected data for a sufficient amount of time, you can find the peak IOPS used by your database by using the following query, which simply takes the largest value from the column *total_io*.

```
SELECT MAX(total_io) PeakIOPS FROM peak_iops_measurement;
```

To cover for any unforeseen spikes, it would be a good idea to add an additional 10 percent to this peak IOPS number to account for the actual IOPS needed for your database. This actual IOPS is the total number of IOPS you should provision for your Amazon EBS volume (GP2 or PIOPS).

Estimating IOPS for a New Database

For a database that you are setting up for the first time on AWS, for which there are no existing statistics, you can go with an IOPS number based on the expected number of transactions per second. Though the IOPS needed per transaction can vary tremendously based on the amount of data involved, the number of queries in a transaction, and the query complexity, generally 30 IOPS per transaction is a good number to consider. Thus if you are expecting 100 transactions per second, you can start with 3,000-IOPS Amazon EBS volumes. Because the amount of data in such a new database is usually small, changing the IOPS associated with Amazon EBS will be relatively easy whether your database is on Amazon RDS or Amazon EC2.

Considering Throughput

In addition to determining the right IOPS, it is also important to make sure your instance configuration can handle the throughput needs of your database. *Throughput* is the measure of the transfer of bits across the network between the Amazon EC2 instance running your database and the Amazon EBS volumes that store the data. The throughput possible relates directly to the network bandwidth available to the Amazon EC2 instance and the capability of Amazon EBS to receive data.

To choose the right Amazon EC2 instance for your database, take the maximum value for *total_bytes* from the PEAK_IOPS_MEASUREMENT table generated by the preceding script and compare it to the instance bandwidth table following. This comparison lets you choose an Amazon EC2 instance type that can provide the necessary bandwidth to move that much data. Note that the script preceding converts the throughput value from bytes per second to megabits per second before storing it in the PEAK_IOPS_MEASUREMENT table. Note also that the instance types shown in the table must be launched as Amazon EBS–optimized to consistently achieve the given level of performance.

Instance Type	Dedicated Amazon EBS Throughput (Mbps)	Max 16K IOPS*
c1.xlarge	1,000	8,000
c3.xlarge	500	4,000
c3.2xlarge	1,000	8,000
c3.4xlarge	2,000	16,000
g2.2xlarge	1,000	8,000
i2.xlarge	500	4,000
i2.2xlarge	1,000	8,000
i2.4xlarge	2,000	16,000
m1.large	500	4,000
m1.xlarge	1,000	8,000
m2.2xlarge	500	4,000
m2.4xlarge	1,000	8,000
m3.xlarge	500	4,000
m3.2xlarge	1,000	8,000
r3.xlarge	500	4,000
r3.2xlarge	1.000	8,000
r3.4xlarge	2.000	16,000

* This value is a rounded approximation based on a 100 percent read-only workload and is provided as a baseline configuration aid. Amazon EBS–optimized connections are full-duplex and can drive more IOPS in a 50/50 read/write workload where both communication lanes are used. In some cases, network and file system overhead can reduce the maximum IOPS available.

You can find more about Amazon EC2–Amazon EBS configuration in the *Amazon EC2 User Guide*. In addition to bandwidth availability, there are certain other considerations that need to go into choosing the Amazon EC2 instance for your Oracle Database. These considerations include your database license, virtual CPUs available, and memory size.

Verifying Your Configuration

Once you configure your environment based on the IOPS and throughput numbers derived using the methods explained preceding, you can verify your configuration before actually installing the database by using a tool named Oracle Orion, available from Oracle. Oracle Orion simulates Oracle Database I/O workloads using the same I/O software stack as Oracle Database, thus providing a measurement of IOPS and throughput that is just like what your database will experience. You can find more details on this tool and download it from the Oracle website.

Further Reading

For additional information on using Oracle Database with AWS services, consult the following resources.

Oracle Database on AWS:

- Advanced Architectures for Oracle Database on Amazon EC2
  http://d0.awsstatic.com/enterprise-marketing/Oracle/AWSAdvancedArchitecturesforOracleDBonEC2.pdf
- Strategies for Migrating Oracle Database to AWS
  http://d0.awsstatic.com/whitepapers/strategies-for-migrating-oracle-database-to-aws.pdf
- Choosing the Operating System for Oracle Workloads on Amazon EC2
  http://d0.awsstatic.com/whitepapers/choosing-os-for-oracle-workloads-on-ec2.pdf
- Best Practices for Running Oracle Database on AWS
  http://d0.awsstatic.com/whitepapers/best-practices-for-running-oracle-database-on-aws.pdf
- AWS Case Study: Amazon.com Oracle DB Backup to Amazon S3
  http://aws.amazon.com/solutions/case-studies/amazon-oracle/

Oracle on AWS:

- http://aws.amazon.com/oracle/
- http://aws.amazon.com/rds/oracle/

Oracle on AWS FAQ:

- http://www.oracle.com/technetwork/topics/cloud/faq-098970.html

Oracle on AWS Test Drives:

- http://aws.amazon.com/solutions/global-solution-providers/oracle/labs/

Oracle licensing on AWS:

- http://www.oracle.com/us/corporate/pricing/cloud-licensing-070579.pdf

Getting started with Oracle RMAN backups and Amazon S3:

- http://aws.amazon.com/backup-recovery/getting-started/

AWS service details:

- http://aws.amazon.com/products/
- http://aws.amazon.com/documentation/
- http://aws.amazon.com/whitepapers/

AWS pricing information:

- http://aws.amazon.com/pricing/
- http://calculator.s3.amazonaws.com/index.html