Running Oracle with Lightbits on AWS

May 2023

Table of Contents

1. Introduction

The Oracle database has long been the de facto most trusted relational (or ACID-based) database. It has been and is still used for many mission-critical applications —from bank transactions to health records, to human resource data in verticals such as banks, pharmaceuticals, and government organizations. We have all used applications that are using data from an Oracle database, and in the last few years, as organizations have started to migrate to the cloud, the need to run Oracle in the cloud has also increased. The "where we use Oracle" has changed from on-premise to cloud, but how we use Oracle or what applications/projects use the Oracle database has not.

So regardless of where we run Oracle, one element has not changed — the performance we are used to getting from an Oracle database. If we're used to doing 100k transactions per second (for example) in our on-premise environment, we want to get the same performance in the cloud, currently, this might be a challenge.

The main reason for that is that while the processing power (whether Intel, AMD, or ARM-based CPUs) and the memory performance are hardware components that are identical (or as close to) in the on-premise and off-premise world, the same can't be said about storage. Storage is still a bottleneck in all cloud environments.

What we're used to getting from our storage arrays or software-defined storage (SDS) on-premise is not transferable to the cloud in the high transaction-based databases domain - and this is where Lightbits® comes into play.

This paper will demonstrate the high-performance capabilities and advantages of the Lightbits Cloud Data Platform running Oracle workloads on storage-optimized EC2 instances on Amazon Web Services (AWS). The tests I've done will show that you can easily take the Oracle workloads and performance that you are used to from your on-premise setup, and successfully migrate it to AWS.

2. Lightbits on AWS Overview

The Lightbits Cloud Data Platform is a robust data storage platform for any cloud - with unmatched speed, flexibility, and efficiency. It enables organizations with a cloud-first strategy to move IO-intensive database workloads – such as Oracle – that may be running on-premises on DAS or SAN today, to the cloud without compromise. It fills a gap that native cloud storage solutions cannot deliver for these workloads: high performance, consistently low latency, enterprise data services, hybrid models, and predictable and lower costs.

Offered by the inventors of NVMe®/TCP, the Lightbits solution is a software-defined, disaggregated block storage system that works with common orchestration environments such as Kubernetes, VMware, and Openshift/Kubernetes. A single Lightbits cluster can provide disaggregated and centralized storage to 10s and 100s of Openshift/Kubernetes clusters.

Lightbits is presented as a set of software packages that run on several servers (or EC2 instances in the case of AWS), creating a cluster of logical storage consisting of all the NVMe SSDs that are attached to each of the servers in the cluster. Lightbits runs on standard Linux-based operating systems. The NVMe/TCP kernel module (nvme_tcp, the initiator on the client) is part of the Linux code from kernel version 4.10 and is available as part of the kernel on several enterprise-level Linux distributions such as RHEL, Ubuntu, and SLES).

Lightbits is easy to use. It plugs directly into Amazon EC2 instances and supports hybrid deployments — offering the flexibility to port the license between on-premises storage servers and the public cloud.

The native NMVe/TCP and clustered architecture, coupled with innovative Intelligent Flash Management and rich enterprise data services, solve the bottlenecks of cloud-native storage solutions, enabling the migration of critical workloads to the cloud.

2.1 Lightbits on AWS EC2

In AWS, Lightbits uses the storage optimized EC2 instances, currently supporting i3en.3x/6x/12x/24x/metal and i4i.2x/8x/16x/24x/metal instances.

The capacity and performance of the Lightbits cluster are impacted by the size of the NVMe devices (7.5TB in i3en and 3.75TB in i4i), the number of the NVMe devices attached to the instance, the number of cores in the instance, and the network bandwidth that the instance can provide.

Previous internal tests in Lightbits have shown the i4i.metal (8 x 3.75TB NVMe devices, 128 cores, and 75 Gbps network) as the instance with the best general performance. This is what was used in these Oracle/SLOB tests.

3. Test Environment Setup

Note: It is outside the scope of this paper to present all of the deployment options that are possible with Lightbits on AWS. For more information, see the Lightbits AWS Deployment Guide.

3.1 Intro to SLOB (the Silly Little Oracle Benchmark)

SLOB has been widely used in the last 12 years for testing the relationship between Oracle databases and the underlying storage infrastructure.

The idea behind SLOB is to act more as a database workload generation tool and less as a database benchmark. Using SLOB, you can test a platform (or compare platforms), and provide data to understand the storage **performance** capabilities of your Oracle platform.

What SLOB does not try to test is the transaction capabilities of Oracle or the platform that Oracle runs on (CPU/Memory relationship). We can all agree that Oracle is the most robust ACID-compliant database available today.

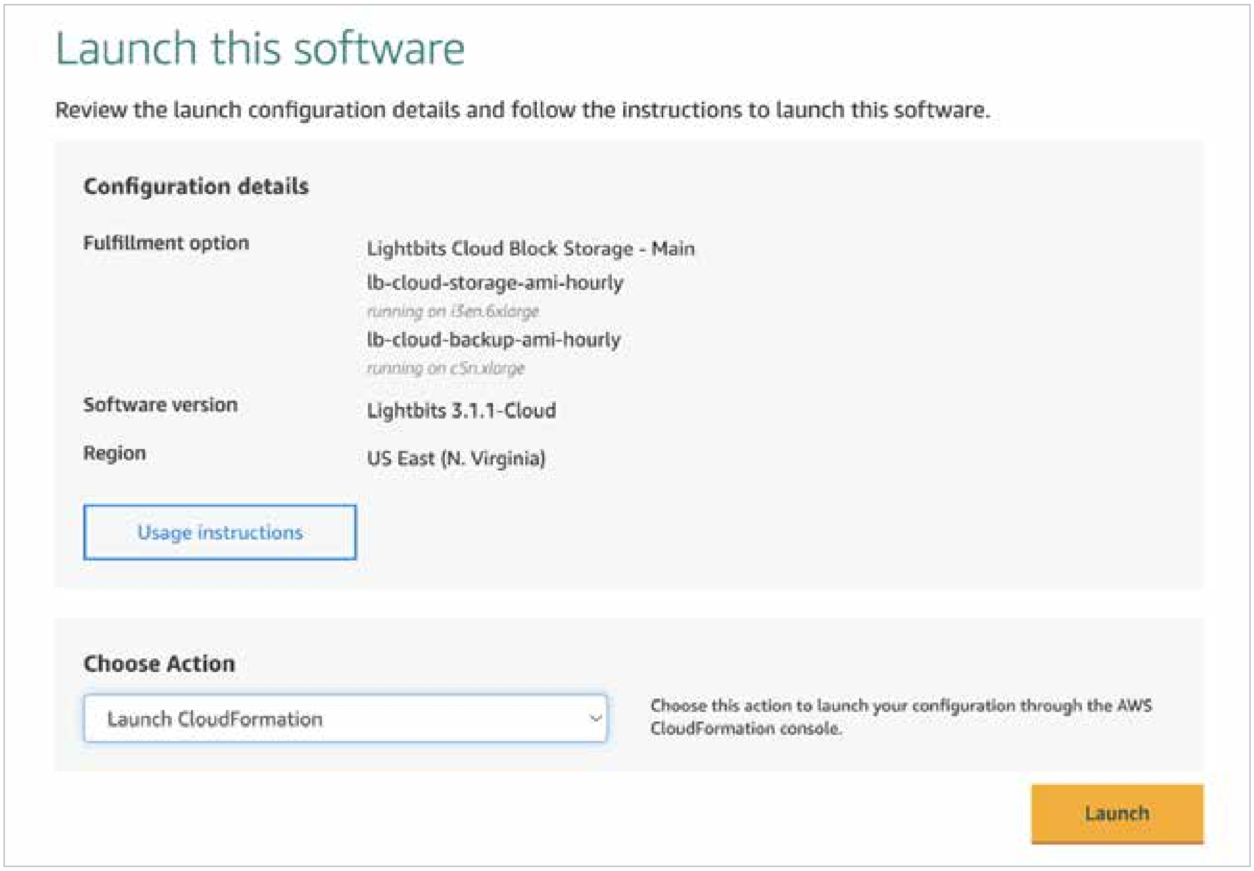
To test and use the storage directly (and this is true for any other workload generator or any other database or application), the application (Oracle) has to be forced to not use caching at all or have a data test area size that is bigger than the allowed cache. It is usually a lot easier to create a work area that is a magnitude greater than the cache, so in Oracle, it means larger than the db_buffer_cache variable.

The SLOB setup creates schemas (users) in the database, in each schema there is a set of tables and indexes that are created. You can control the table/index size via the SCALE variable in the slob.conf file.

One of the more important aspects of SLOB is the ability to control the percentage of transactions that run select-only queries (a read IO) vs transactions that will update the data (a write IO). To update a row in Oracle, it needs to reside in the cache, hence, there is a select (read IO) before an update, unless the row is cached, so a 100% update means that 50% of the physical IOs were read and 50% were write.

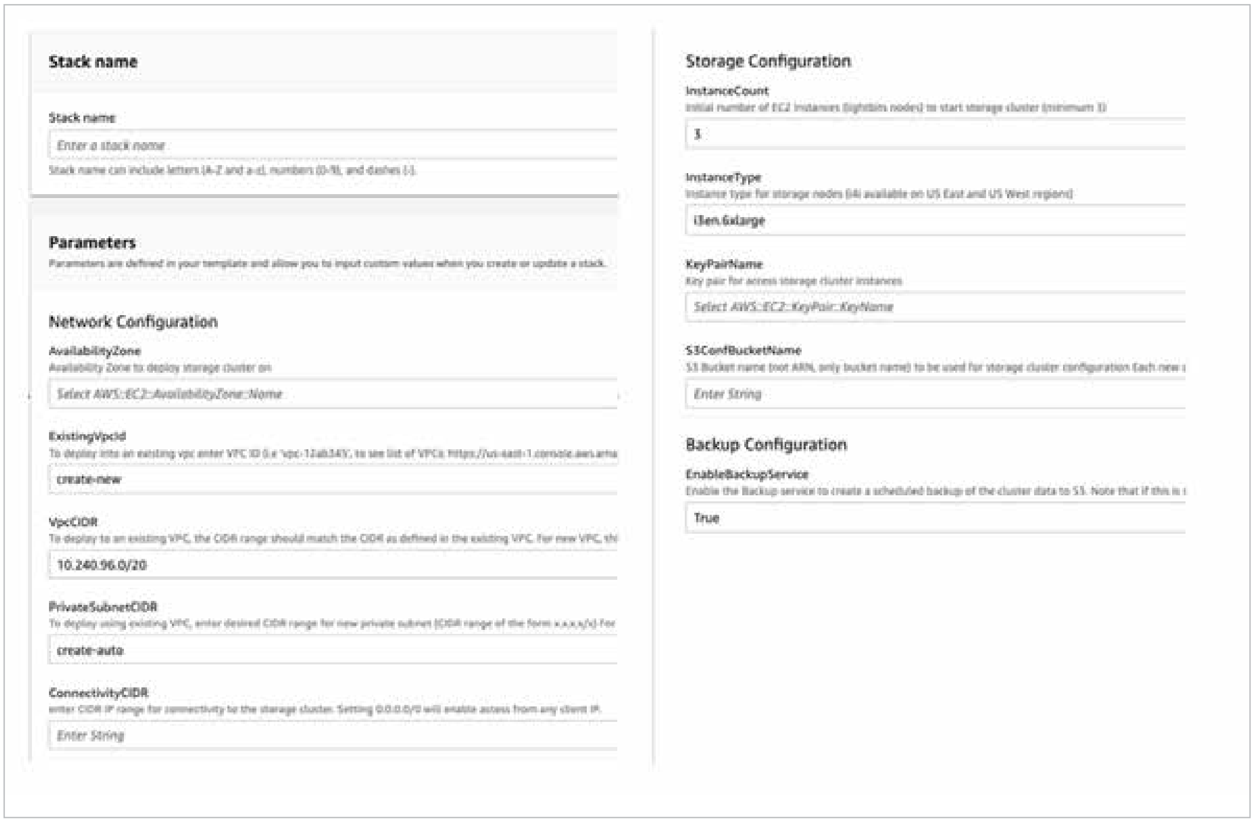
3.2 Setting Up the Lightbits Cluster in AWS

Creating a Lightbits cluster on AWS is as easy as any other marketplace offering. Just head to your AWS console and type "Lightbits" in the search box and choose "Lightbits Cloud Block Storage" from the marketplace section. Once you subscribe to the offer (like any other marketplace offer), you'll get to the section to configure the offer by choosing the region where you want to deploy the cluster.



Follow this by clicking "Launch" (the "Action" should be set to "Launch CloudFormation").

Click “Next” on the first CloudFormation page, and you will see the main configuration page of the Lightbits cluster:

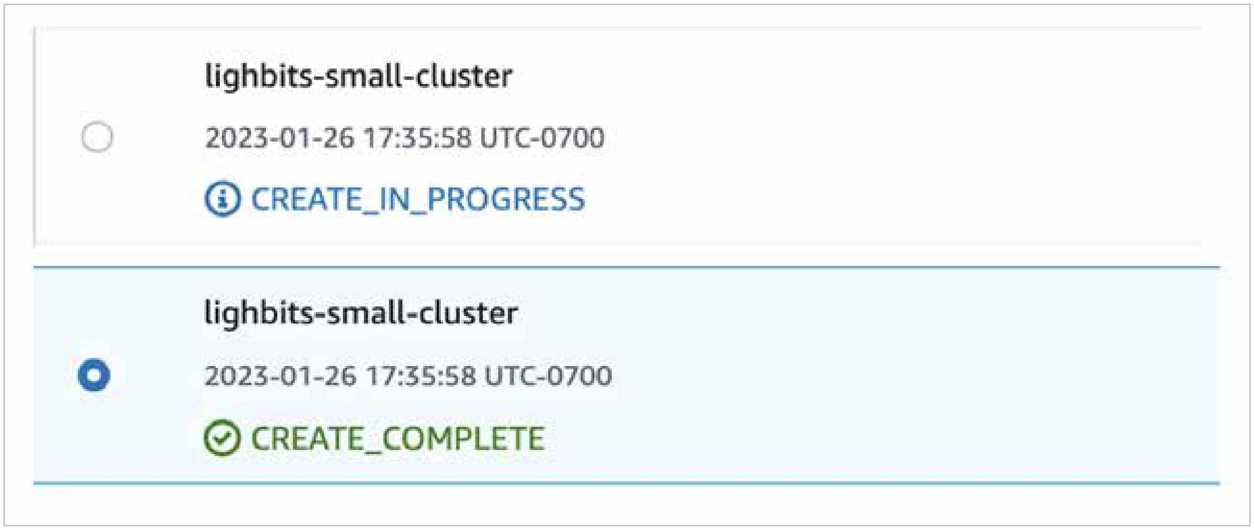


A few things to note:

1. You can either create a new VPC and create the Lightbits cluster in it (and then, of course, all your storage consumers will have to reside in the new VPC), or you can choose an existing VPC of your own and supply the VPC ID.
2. If you choose your own VPC, make sure to choose the right subnet that is associated with your VPC, and then the right CIDR for VpcCIDR. Also make sure that in PrivateSubnetCIDR, you specify a range that derives from the VPC.
3. ConnectivityCIDR controls what client IPs will be allowed to access the Lightbits cluster. You can use “0.0.0.0/0” to allow all IPs in the VPC.
4. The instance count is self-explanatory; the minimum count is 3.
5. Instance type is the most important variable, as it will impact the overall capacity and performance of the Lightbits cluster (more CPU cores and higher bandwidth will impact the overall performance of the cluster, and more NVMe devices will impact the total cluster capacity as well as the performance of the cluster). Lightbits supports a few instance store options from the i3en and i4i families of instances. All performance tests in this paper were done with 3 instances of i4i.metal.

Click “Next” in “Specify stack details” and on the “Configure stack option” pages, and then “Submit” on the “Review” page. And that’s it! Your Lightbits cluster will be ready in about 10 minutes.

Note: The CloudFormation Stacks page will show the initial stack and then more nested stacks will follow. You can monitor the initial stack (the one without "NESTED" in the description), and when the status changes from "CREATE_IN_PROGRESS" to "CREATE_COMPLETE", the cluster is ready for use.



3.3 Preparing the EC2 Instance to Run Oracle and Connect to the Lightbits Cluster

There is a public git repository that includes all the bash scripts that were used to create the EC2 instance/s that run Oracle, install Oracle software, and run all the performance tests shown in this paper. There will be details below on what each script does and if there are things that you need to edit or pay attention to.

Note: In this setup, the Lightbits cluster was created into an existing VPC and subnets, and also created a bastion in the VPC that we use to ssh to, and then run all scripts from it. The scripts use the pssh (parallel-ssh) package to access the Oracle instances without a password in parallel. The idea here is that everything can be run on several Oracle instances at the same time (if you choose to).

The following are the requirements to run the scripts (on the bastion):

1. Pssh package is installed
2. The jq package or binary is installed
3. A file with the private DNS name of all Lightbits instances is in your cluster (the PrivateDnsName variable in the AWS cli API)
4. The AWS CLI is installed and configured with an account that can create instances (and optionally also EBS volumes if you chose to compare vs io2.bx)
5. The ssh key pem file is paired with the key name you want to use for all the Oracle instances
6. The scripts from the aws_oracle_blog directory are in this git repo

Description of Scripts

The first script you can use is the ***create_instance*** script, which is a simple bash script that helps you create one or several instances at the same time to then use these instances to install and run Oracle.

We supplied two AMI IDs in the script (for RHEL8 and Alma8, although note that some instances support only RHEL8). Of course, you can use other AMIs as the base for the OS that will host Oracle, though we tested the scripts with RHEL-based OS.

Remember that the AMI/OS will also determine the AWS username you will use to run the rest of the scripts.

You can also use this script to create the instance/s into a specific placement group.

The ***my_config*** script is used as a config file to hold variables that are used in all the rest of the scripts. You **must** populate it with the right information to successfully run all the scripts.

The ***0.setup_ssh_access*** scripts make sure that we can ssh passwordless to all the instances that will run Oracle. This makes using pssh a lot easier. You should also run it for the Lightbits instances.

The scripts ***1.install_packages***, ***2.os_tweaks*** and ***3.oracle_user_setup*** all set up the necessary packages and binaries that are needed to install the Oracle software, create an Oracle database, and use the SLOB workload generator.

The ***4.connect_to_lb_cluster*** script collects information from the Lightbits cluster and creates a systemd service that connects to the Lighbits cluster targets. You can run this script even if you plan to use EBS storage — there is no harm in running the script. In production systems, it's recommended to use the Lighbits discovery client instead of manually creating a systemd service.

The ***5. reboot*** script reboots the instance to get the kernel changes (and everything else) to take effect.

Once the Oracle EC2 instances are up and running, we move to the next phase of installing the Oracle software — creating the Oracle database and SLOB schemas and populating the database with random SLOB data.

The ***6.copy_oracle_home*** script copies the Oracle software zip file and unzips it to the specified ORACLE_HOME path. We followed the Oracle documents on all the paths to the Oracle base and Oracle home. (You can always change to your path as well; just remember to change it in all the relevant scripts).

The ***7.create_volumes*** script helps you create volumes to use for the Oracle deployment and data. You can specify whether to use Lighbits volumes (use LB or lb for Lightbits), or you can use this script to create io2/io2.bx volumes.

For the tests here, we used either a 16 or 24-volume configuration. For Lightbits volumes, a higher core count on the Oracle instances can take advantage of more data files. Note also that this script gives the ability to create volumes in either a 35GB or 75GB size configuration, the reason for that is to keep things simple for testing purposes.

To observe Lightbits behavior with different types of EC2 instance store instances (with fewer SSD devices). You can of course use other sizes, in any case, and as explained before, we're running Oracle with a very low database buffer cache, so anyhow all/most operations are done directly via the storage.

The ***8.install_oracle*** script installs the Oracle software. It uses the ***myoracleinstall.rsp*** answer file included in the repo.

The ***9.create_slob_database*** takes care of creating an Oracle database named SLOB and then creates the IOPS tablespace using all the volumes (datafiles) that were created in step #7. The IOPS tablespace is needed to run SLOB. The script also copies all of the relevant database configuration files.

The ***10.create_slob_schemas*** script automates the SLOB data creation process (the SLOB ***setup.sh*** script). The database will bounce at the end of the script.

Note: Technically, once you have the EC2 instances that you want to run the Oracle database on, you can edit the ***run_it_all.sh*** script and all scripts will run in parallel on all of the instances.

3.4 Running SLOB

We created a wrapper for SLOB's ***runit.sh*** script called ***loop_over_runit.sh***. You can find it under the SLOB root directory (if you have not changed the path to the Oracle user on your EC2 instance, it should be /home/oracle/SLOB).

The reason behind creating the wrapper script is to automate the testing of different variables, such as the number of schemas, the number of threads that SLOB will use, the length of the runs, and also the update percentage (UPDATE_PCT). Using this script gives you the ability to test several permutations and find the right combination for your Oracle instance and your Lightbits cluster.

The ***loop_over_runit.sh*** script takes one variable as a run name and creates a directory with the same name. This directory will include all the output files (or all variable permutations) that you choose to run. SLOB's output includes the AWR report calculated between before and after snapshots, and the directory will also include a tar file with the AWR report, iostat, vmstat and mpstat taken during the run, the slob.conf file used for the run, the awr report in html format, and also the database init file. This gives you a complete picture of all the variables that were used and also how the Oracle database was configured at the time of the run.

3.5 SLOB Runs Performance When Using Lightbits

As explained previously, the performance of your SLOB runs when using Lightbits will be impacted by the type of EC2 instance you run Oracle on (mainly via the number of cores and the network bandwidth), and the type of EC2 instances you chose to deploy the Lightbits cluster with.

Like other storage solution testing, we recommend performing a solid write operation before starting to test random read/write operations. Using the loop_over_runit.sh script, we would recommend running it with the following variables:

Once your initial run is done, you can start testing your Oracle environment using SLOB with different variables. For example, in the following setting of loop_over_runit.sh, we will test SLOB with four options of update % (100%, 25%, 50% and 0%), with 16 and 24 schemas, a runtime of 5 minutes, using 24, 48 and 64 threads. This sums up to 24 different runs. (for published performance results, I recommend at least 30 minutes of runtime - preferably 60 minutes. So RUNTIME should be set to 1800 or 3600).

```
#!/bin/bash

readonly RUN_NAME=${1}
readonly SLEEP=120
readonly SLOB_HOME=/home/oracle/SLOB
readonly SLOB_CONF_PATH=/home/oracle/SLOB/slob.conf
readonly UPDATE_PCT=(100 25 50 0)
readonly SCHEMAS=(16 24)
readonly RUNTIME=(300)
readonly THREADS=(24 48 64)
readonly THREADS_ELEMENTS=${#THREADS[@]}
```

Once your runs are done, you can then use the print_slob_runs.sh script to print the results. The output is comma-separated and ready to be pasted into a spreadsheet. The output will be sorted by the number of schemas, threads, update percentage, and last by instance name.

(Note that the image below is cut because the output is very long).

```
[fedora@ip-10-0-0-118 02092023_ip-10-0-1-7]$ bash /tmp/SLOB/print_slob_runs.sh testrun
node name,number of schemas,number of threads,update %,SQL executions per seconds,physical reads iops,physical reads(MB/s),
le parallel write(us),log file parallel write(us)
ip-10-0-1-7.us-east-2.compute.internal,24,72,0,16164,1026258, 8017.7,7841, 62,.8, 1592,0,0, 530, 3341
ip-10-0-1-7.us-east-2.compute.internal,24,72,25,7213,455911, 3561.9,118147, 1062,95.4, 1308,0,0, 353, 2327
ip-10-0-1-7.us-east-2.compute.internal,24,72,50,4312,270464, 2113.1,130655, 1176,105.9, 816,0,0, 247, 1841
ip-10-0-1-7.us-east-2.compute.internal,24,72,100,1181,64650, 530.7,60717, 555,52.8, 489,0,0, 224, 832
ip-10-0-1-7.us-east-2.compute.internal,24,80,0,16725,1061468, 8292.8,2131, 17,.2, 1705,0,0, 637, 2946
ip-10-0-1-7.us-east-2.compute.internal,24,80,25,6469,407581, 3184.3,106029, 954,85.8, 1275,0,0, 335, 1960
ip-10-0-1-7.us-east-2.compute.internal,24,80,50,3999,250289, 1955.5,122135, 1099,98.8, 843,0,0, 258, 1632
ip-10-0-1-7.us-east-2.compute.internal,24,80,100,1935,117576, 918.6,113980, 1036,96.1, 603,0,0, 242, 1224
ip-10-0-1-7.us-east-2.compute.internal,24,96,0,15699,995030, 7773.7,527, 4,.1, 2150,0,0, 1075, 2299
ip-10-0-1-7.us-east-2.compute.internal,24,96,25,2533,155563, 1215.4,48132, 435,39.4, 887,0,0, 294, 903
ip-10-0-1-7.us-east-2.compute.internal,24,96,50,1430,84993, 664.1,42176, 383,34.6, 452,0,0, 209, 775
ip-10-0-1-7.us-east-2.compute.internal,24,96,100,2071,124851, 975.6,121647, 1103,102.4, 669,0,0, 239, 1501
```

4. Performance Results

Before reviewing the results, it's prudent to understand the Lightbits technology and how it is capable of achieving high performance.

The first thing to understand is that a volume in a Lightbits cluster resides entirely on a Lightbits node. If you created a volume with a replication factor of 2 or 3, it means that the copies also reside on their own Lightbits nodes (Lightbits uses the term protection groups when setting up a cluster to decide where and how to place secondary copies of a volume).

To maximize the performance you can get from a Lightbits cluster, you have two options:

1. Get your single client (in our case the EC2 instance running Oracle) to consume enough volumes from as many Lightbits nodes as possible (meaning, use more than one Lightbits volume for your Oracle tablespace). The storage performance of a single EC2 instance running an Oracle database will also be affected by the number of cores the EC2 instance has and the network bandwidth that is available to the instance.
2. Get several clients (in our case, several EC2 instances, each running an Oracle database) to run in parallel and via that consume many Lightbits volumes.

Both of the above options are based on NVMe/TCP's excellent ability to connect to multiple IP addresses, using local queues (on the client and the targets) to serve IOs as if the NVMe devices are just regular local NVMe devices.

While the first option is good to understand what maximum storage performance you can get from a single instance, the 2nd option allows you to test the maximum storage performance you can get from a single Lightbits cluster (remember that Lightbits can scale out very easily, and even automatically based on capacity used, so increasing the total performance of the cluster is as easy as just adding more Lightbit nodes/EC2 instances to the cluster).

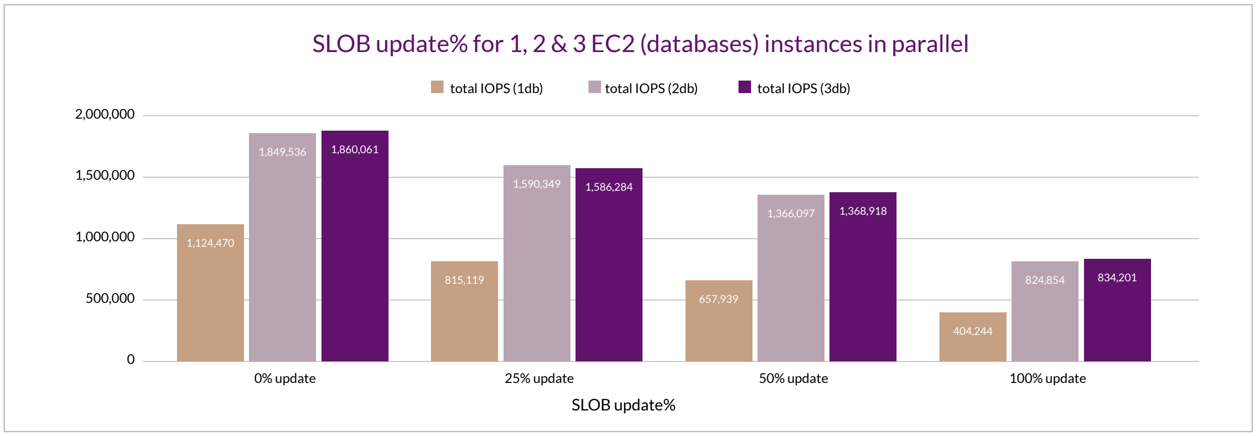
4.1 Single vs Multiple R6i.24xlarge

Note: All numbers refer to physical IO operations directly from the Lightbits volumes.

In this configuration, 24 Lightbits volumes were used to create the SLOB IOPS tablespace, for a single instance of Oracle running on R6i.24xlarge.

Another identical instance was added using the same layout (24 Lightbits volumes) SLOB was run in parallel together with the first instance. Once data collection was done for two EC2 instances running Oracle, a 3rd instance was added with the same configuration.

The results show that a single EC2 instance (single Oracle database), is capable of about 815k physical IOPS in the 25% update configuration. When adding another instance (and another database) the total physical IOPS from the two databases is close to 1.6M. When adding a 3rd instance, the results show that the physical IOPS number is unchanged since the total performance ability of this minimal Lightbits cluster was reached. To increase the total Lightbits cluster performance, just add more Lightbits instances.



4.2 Comparison of Different EC2 Instance Types Running a Single Database

This comparison tested three different types of EC2 instances: the R6in.24xlarge that were used previously, the X2idn.32xlarge, and the R5n.16xlarge.

These three instances are different in some variables:

1. The R6in.24x and X2idn.32x are both based on Intel's Ice Lake CPU, while the R5n.16x is based on the previous generation of CPUs - Cascade Lake.
2. Both the X2idn.32x have 128 cores, while the R6in.24x have 96 cores and the R5n.16x has 64 cores. They also differ in memory size, but since the testing is with very low buffer_cache, the memory size has a minimal impact.
3. The R6in.24x (the latest instance to join the EC2 range of instances) has a network limit of 150 Gbps. The x2idn.32x has a bandwidth limit of 100 Gbps, and the more modest R5n.16x has a limit of 75 Gbps.

As always, it's highly recommend testing with your dataset as performance could vary from setup to setup, but testing does reveal some very interesting data points:

1. As expected, the R5n.16x is slower when doing many selects than the other two instances, however, it is not significantly slower.
2. The X2idn.32x in the tests of select only (read-only) have reached the bandwidth limit that the instance can provide, adding more volumes to the database, or even more core (if it was possible), would not help here. As a comparison, the R6in.24x - which has fewer cores than the X2idn.32x but 50% more network bandwidth - did not reach the network limit.
3. When doing many writes (100% update of SLOB), the R5n.16x had better results than the R6in.24x. The reasons for this are beyond the scope of this paper or the knowledge that we have of how exactly the physical attributes of these instances are implemented.

Instance Type	Update %	Physical Reads	Read Bandwidth (MB/s)	Physical Writes	Write Bandwidth (MB/s)	Total Physical IOPS	Total Bandwidth (MB/s)
R5n.16x	0	919040	7188	7	0	919,047	7,188
X2idn.32x	0	1422618	11041	6	0	1422624	11041
R6in.24	0	1124470	9101	15	0	1124485	9101
R5n.16x	25	571242	4463	141828	1386	713070	5849
X2idn.32x	25	712344	5565	182600	1762	894944	7328
R6in.24	25	648806	5069	166313	1602	815119	6671
R5n.16x	100	268492	2120	260633	2010	529125	4129
X2idn.32x	100	415041	3245	416856	3245	831897	6490
R6in.24	100	204239	1606	200005	1601	404244	3207

4.3 Comparison to io2 Block Express (io2.bx)

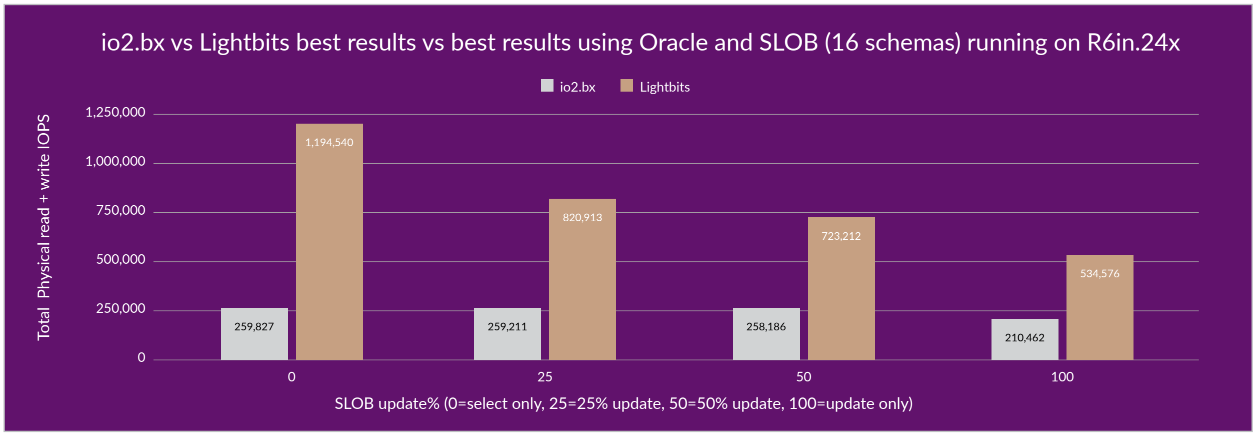
In AWS, io2.bx is the high-end (or top) EBS storage option. With io2.bx you choose not only the size of the volume that you want to use but also the number of IOPS that are allowed on the volume ("Provisioned IOPS"). This means that volumes have a limit of IO operations that are allowed on them. Since not all IOs are equal because of different block sizes (for example 8k vs 128k block size), this is a major variable to consider.

There is also a limit on what an EC2 instance can consume from EBS (per AWS specifications, it is 350k). This of course gives you a lot more freedom to use storage that is consumed via the network path vs EBS — which is exactly what Lightbits provides.

Several tests were run using 16 io2.bx volumes each with a limit of 16k IOPS, 16 io2.bx volumes with each 25k provisioned IOPS, 4 io2.bx volumes each with 90k provisioned IOPS, and lastly, 2 io2.bx volumes with each allowing 255k provisioned IOPS.

Note: for the graph below the highest numbers were used from each SLOB update %, regardless of the io2.bx permutation ratio of volumes/provisioned IOPS.

The SLOB tests using Lightbits also used 16 volumes (schemas in SLOB).



In these tests, the R6in.24x instance using io2.bx maxed out at a consumption rate of an average of 256k IOPS, while the Lightbits numbers are significantly higher.

5. Summary and Conclusions

This technical paper provides you with the knowledge to understand the performance capabilities of the Lightbits Cloud Data Platform on AWS, and how to test the performance of storage via SLOB using different types of EC2 instances running Oracle.

The results come from a Lightbits cluster based on i4i.metal with the minimum number of nodes (3).

The results have shown that this minimal Lightbits configuration can supply around 1.4M 8k physical reads per second to Oracle (when doing select only and using X2idn.32x as the Oracle EC2 instance).

All tests were done on Oracle version 19c (19.3).

For more information on how to power your Oracle database on AWS with Lightbits, see https://www.lightbitslabs.com/oracle/

Additional Resources

Lightbits on the AWS Marketplace

Webinar with AWS: Get Unmatched Performance on Oracle DBs on AWS Storage via SLOB

Appendix

Deploying Lightbits on AWS:

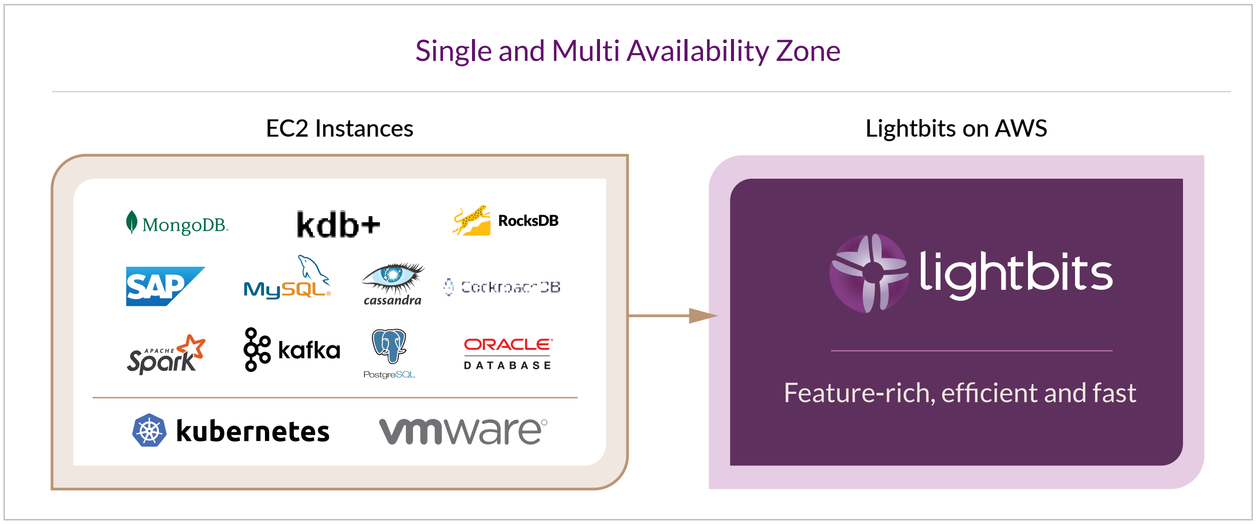
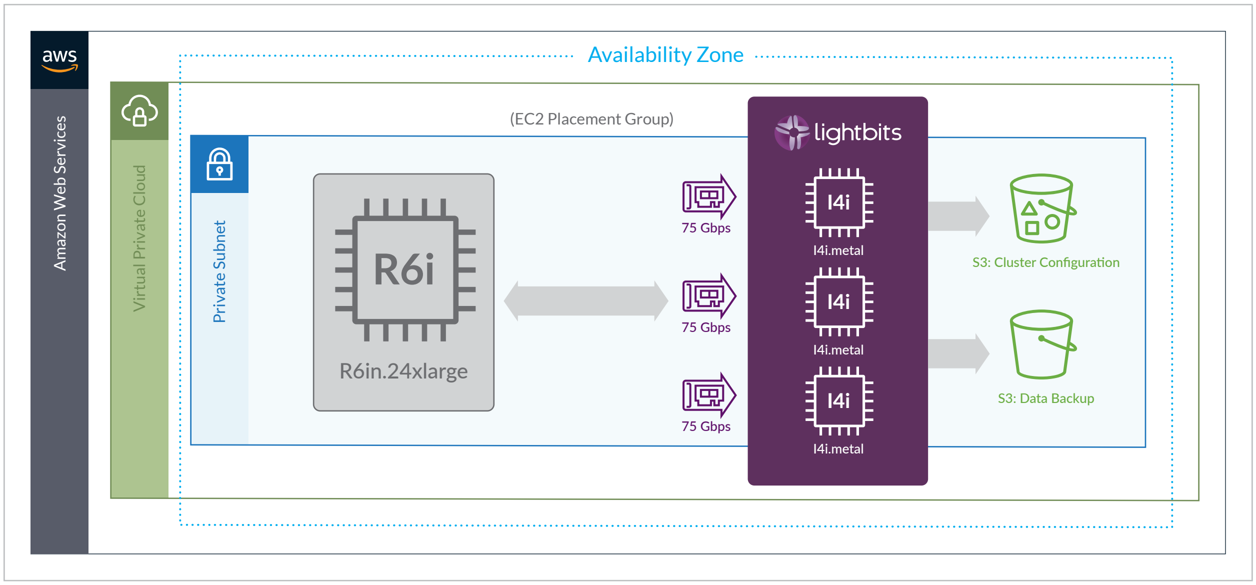


Diagram of the test setup:



About Lightbits Labs

Lightbits Labs® (Lightbits), offers a Complete Data Platform for any cloud in VMware, Kubernetes, and OpenStack orchestration environments. It enables organizations with a cloud-first strategy to move IO-intensive database, analytic, transactional, and streaming applications that may be on DAS or SAN to the cloud without compromise. It delivers high performance, consistent low latency, and predictable low costs that cloud-native block storage solutions cannot provide. The native NMVe/TCP and clustered architecture, coupled with innovative Intelligent Flash Management and rich enterprise data services solve the storage challenges endemic to cloud-native applications. Lightbits is backed by enterprise technology leaders [Cisco Investments, Dell Technologies Capital, Intel Capital, Lenovo, and Micron] and is on a mission to deliver a robust cloud platform with unmatched efficiency, agility, and flexibility.

www.lightbitslabs.com

info@lightbitslabs.com

US Office
1830 The Alameda,
San Jose, CA 95126, USA

Israel (Kfar Saba) Office
17 Atir Yeda Street,
Kfar Saba, Israel 4464313

Israel (Haifa) Office
3 Habankim Street,
Haifa, Israel 3326115