**Cloud Benchmarking**

Chief Architect for Cloud Operations and Analytics

Huawei Munich Research Centre

Prof. Jorge Cardoso

jorge.cardoso@huawei.com

16.01.2018

This document is Part C of the lectures on Cloud Benchmarking and contains a set of exercises to benchmark a public cloud platform. Part B was on OpenStack Operations and contained a set of exercises to operate OpenStack cloud platform. Part A was on OpenStack, Operations and Troubleshooting.

Table of Contents

[1 Overview 3](#_Toc503699354)

[2 ECS Benchmarking 3](#_Toc503699355)

[2.1 Preparing the VM 3](#_Toc503699356)

[2.1.1 VM Creation 3](#_Toc503699357)

[2.1.2 SSH Access 4](#_Toc503699358)

[2.1.3 OS Update 4](#_Toc503699359)

[2.2 Benchmarking 4](#_Toc503699360)

[2.2.1 CPU Benchmark 4](#_Toc503699361)

[2.2.2 Disk IO Benchmark 7](#_Toc503699362)

[2.2.3 Network Benchmark 9](#_Toc503699363)

[3 RDS Benchmarking 11](#_Toc503699364)

[3.1 Preparation 11](#_Toc503699365)

[3.1.1 DB Preparation 11](#_Toc503699366)

[3.1.2 VM Preparation 12](#_Toc503699367)

[3.1.3 Access the database 12](#_Toc503699368)

[3.2 DB Benchmarking 12](#_Toc503699369)

[3.3 Results 14](#_Toc503699370)

[4 Webapp Deployment 14](#_Toc503699371)

[4.1 Overview 14](#_Toc503699372)

[4.2 Preparation 15](#_Toc503699373)

[4.3 Configure Wordpress 16](#_Toc503699374)

# Overview

This document describes the benchmarking of AliCloud ECS and RDS. By no means, it should be considered a final study. Many more experiments and tests are needed to obtain a precise view of benchmarking. Nonetheless, it describes the main techniques and tools which are used for benchmarking exercises.

The benchmarking includes the following main activities:

* Create an ECS (Elastic Compute Service) instance
* Install performance analysis tools in VM (e.g., sysbench, netperf, etc.)
* Benchmark the CPU, the network, disk I/O, and file access
* Create RDS (Relational Database Service) instance
* Create database and accounts with privileges
* Run IOPS and connection to test the RDS
* Deploy Wordpress web application in VM
* Access Wordpress remotely

# ECS Benchmarking

The performance benchmarking and reporting of ECS placed emphasis on:

* Benchmarking of ECS
* Focus on CPU, storage and networking performance, e.g., disk IO, networking IO.
* Benchmarking of RDS
* Focus on database IOPS, QPS, connection number and others relevant index.

Students can open an AliCloud account and use the resources for the exercise.

* http://intl.aliyun.com/campaign/freetrial

## Preparing the VM

### VM Creation

An ECS instance with 1 core and 1GB memory was created.

|  |  |
| --- | --- |
| ECS | Region: EU Central 1 (Frankfurt)  I/O Optimized: IO optimized instance  Instance type: 1-core, 1GB  Network Type: VPC  Bandwidth: 30Mbps (Data transfer usage)  OS: Ubuntu 16.04 64bit  System Disk: 40GB SSD Cloud Disk  Auto-renew:  No  Server Guard:  No  Order Type: starter-package  Prompt：If you select 1M bandwidth or above, a public IP will be distributed, which is unable to unbind. |

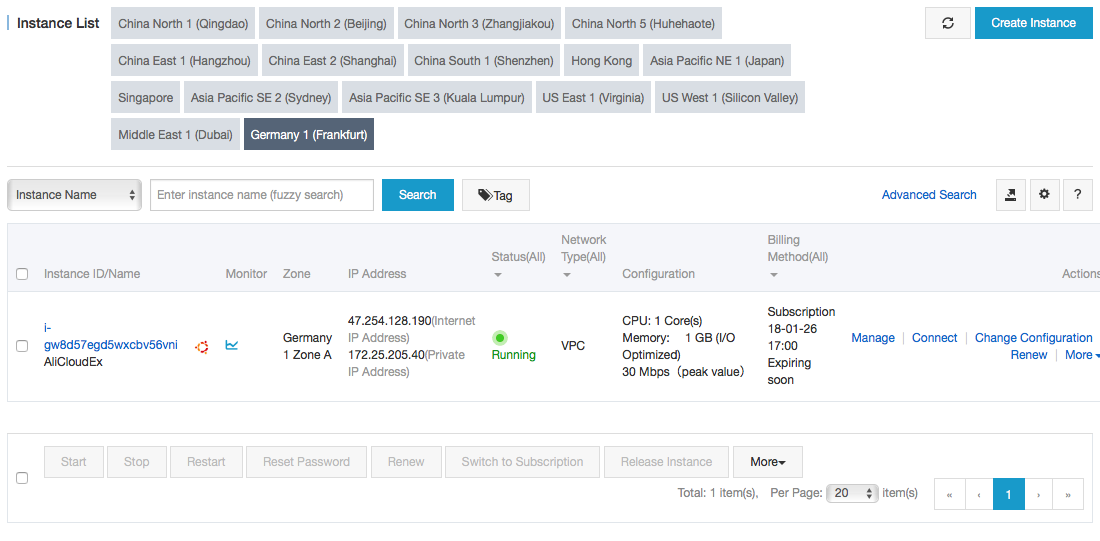


Figure . Console and instance panel

### SSH Access

To gain access to the VM created, the private key **alicloud.pem** generated using the option Key Pairs of the console was used along with ssh:

$ Jorges-iMac:.ssh jcardoso$ ssh -i alicloud.pem root@47.254.128.190

The output is:

Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-62-generic x86\_64)

 \* Documentation:  https://help.ubuntu.com

 \* Management:     https://landscape.canonical.com

 \* Support:        https://ubuntu.com/advantage

Welcome to Alibaba Cloud Elastic Compute Service !

### OS Update

The VM was updated with the latest Ubuntu packages.

$ sudo apt-get update

$ sudo apt-get install unzip

$ sudo apt-get install software-properties-common python-software-properties

$ sudo add-apt-repository main

$ sudo add-apt-repository universe

$ sudo add-apt-repository restricted

$ sudo add-apt-repository multiverse

## Benchmarking

### CPU Benchmark

To benchmark CPU, the byte-unixbench tools was used:

$ wget https://github.com/kdlucas/byte-unixbench/archive/master.zip

$ unzip ./master.zip

$ cd ./byte-unixbench-master/UnixBench

$ ./Run

After running the tool, the output obtained was:

========================================================================

BYTE UNIX Benchmarks (Version 5.1.3)

System: AliCloudEx: GNU/Linux

OS: GNU/Linux -- 4.4.0-62-generic -- #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC 2017

Machine: x86\_64 (x86\_64)

Language: en\_US.utf8 (charmap="UTF-8", collate="UTF-8")

CPU 0: Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz (5000.0 bogomips)

x86-64, MMX, Physical Address Ext, SYSENTER/SYSEXIT, SYSCALL/SYSRET

20:02:01 up 4 days, 20:17, 1 user, load average: 0.00, 0.00, 0.20; runlevel 2017-12-26

------------------------------------------------------------------------

Benchmark Run: Sun Dec 31 2017 20:02:01 - 20:30:05

1 CPU in system; running 1 parallel copy of tests

Dhrystone 2 using register variables 34438786.9 lps (10.0 s, 7 samples)

Double-Precision Whetstone 4186.1 MWIPS (9.9 s, 7 samples)

Execl Throughput 5661.1 lps (29.6 s, 2 samples)

File Copy 1024 bufsize 2000 maxblocks 1295671.2 KBps (30.0 s, 2 samples)

File Copy 256 bufsize 500 maxblocks 352375.9 KBps (30.0 s, 2 samples)

File Copy 4096 bufsize 8000 maxblocks 2915577.3 KBps (30.0 s, 2 samples)

Pipe Throughput 2344275.2 lps (10.0 s, 7 samples)

Pipe-based Context Switching 340783.6 lps (10.0 s, 7 samples)

Process Creation 17191.1 lps (30.0 s, 2 samples)

Shell Scripts (1 concurrent) 10418.7 lpm (60.0 s, 2 samples)

Shell Scripts (8 concurrent) 1349.6 lpm (60.0 s, 2 samples)

System Call Overhead 4089066.5 lps (10.0 s, 7 samples)

System Benchmarks Index Values BASELINE RESULT INDEX

Dhrystone 2 using register variables 116700.0 34438786.9 2951.1

Double-Precision Whetstone 55.0 4186.1 761.1

Execl Throughput 43.0 5661.1 1316.5

File Copy 1024 bufsize 2000 maxblocks 3960.0 1295671.2 3271.9

File Copy 256 bufsize 500 maxblocks 1655.0 352375.9 2129.2

File Copy 4096 bufsize 8000 maxblocks 5800.0 2915577.3 5026.9

Pipe Throughput 12440.0 2344275.2 1884.5

Pipe-based Context Switching 4000.0 340783.6 852.0

Process Creation 126.0 17191.1 1364.4

Shell Scripts (1 concurrent) 42.4 10418.7 2457.2

Shell Scripts (8 concurrent) 6.0 1349.6 2249.4

System Call Overhead 15000.0 4089066.5 2726.0

========

**System Benchmarks Index Score 1970.1**

The CPU has a score of **1970.1** for a single task. The AliCloud VM has a similar performance with an Intel Xeon Processor E5-2680 v2 made available in 2013. To compare results with other platforms, see <http://linux-bench.com/results/?table=unixbench_single>. The comparison table is not maintained and verified. Thus, entries may not reflect real performance and may be in some cases incorrect. Therefore, the tool has more utility when used to directly compare systems.

Figure 2 shows the CPU performance using the AliCloud console.

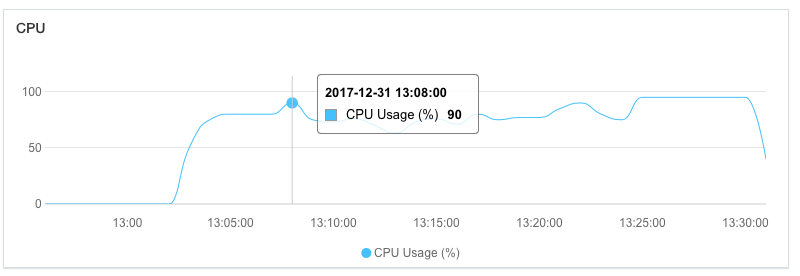


Figure . CPU benchmark using AliCloud console

Figure 3 shows the **top** unix utility to observe the performance of the CPU. The values shown by AliCloud console and the top utility generally differ in 10%-20%.

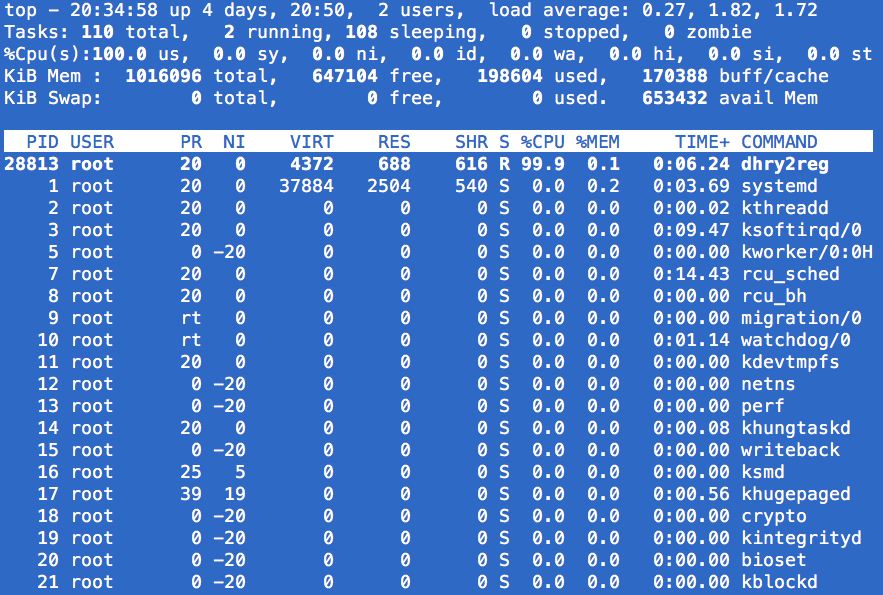


Figure . CPU benchmark using top unix utility

Another tool which can be used for CPU benchmarking is **sysbench**. For example, the following command enables to study how threaded process scales for the VM.

$ for each in 1 2 4 8 16 32 64; do sysbench --test=cpu --cpu-max-prime=20000 --num-threads=$each run; done

The results were close to 26 seconds. Naturally, since the VM we are benchmarking has only one core (run: $ cat /proc/cpuinfo), increasing the number of threads can in some cases even slowdown the computation time. But for multicore CPUs, the true performance will be measured when running the benchmark with the correct amount of threads. A CPU with 8 cores should run an 8 thread CPU benchmark to get the true performance number.

Ideally the benchmark should also be done for Amazon AWS, MS Azure, and Google GCP to establish a baseline and calculate an offset of performance.

Other programs to benchmark a VM can be found at:

* <https://wiki.mikejung.biz/Benchmarking>

### Disk IO Benchmark

To measure disk IO performance, **sysbench** was used. First, a test file of 30GB (we are using a SSD with 40GB) was created. The file needs to be much bigger than the RAM, otherwise the system will use RAM for caching which tampers with the benchmark results.

$ sysbench --test=fileio --file-total-size=30G prepare

The output was:

sysbench 0.4.12:  multi-threaded system evaluation benchmark

128 files, 245760Kb each, 30720Mb total

Creating files for the test...

Afterwards, benchmark itself can be run:

$ sysbench --test=fileio --file-total-size=30G --file-test-mode=rndrw --init-rng=on --max-time=300 --max-requests=0 run

The output was:

sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:

Number of threads: 1

Initializing random number generator from timer.

Extra file open flags: 0

128 files, 240Mb each

30Gb total file size

Block size 16Kb

Number of random requests for random IO: 0

Read/Write ratio for combined random IO test: 1.50

Periodic FSYNC enabled, calling fsync() each 100 requests.

Calling fsync() at the end of test, Enabled.

Using synchronous I/O mode

Doing random r/w test

Threads started!

Time limit exceeded, exiting...

Done.

**Operations performed: 428100 Read, 285400 Write, 913193 Other = 1626693 Total**

**Read 6.5323Gb Written 4.3549Gb Total transferred 10.887Gb (37.161Mb/sec)**

2378.33 Requests/sec executed

Test execution summary:

total time: 300.0002s

total number of events: 713500

total time taken by event execution: 157.0585

per-request statistics:

min: 0.00ms

avg: 0.22ms

max: 33.67ms

approx. 95 percentile: 0.32ms

Threads fairness:

events (avg/stddev): 713500.0000/0.00

execution time (avg/stddev): 157.0585/0.00

The important number is the *Kb/sec* value:

* Operations performed: … 1626693. (i.e., 1626693/300s=5422 IOPS)
* Total transferred 10.887Gb (37.161Kb/sec)

Analysing the disk monitoring interface of the AliCloud console, the following results for IOPS were observed (Figure 4).

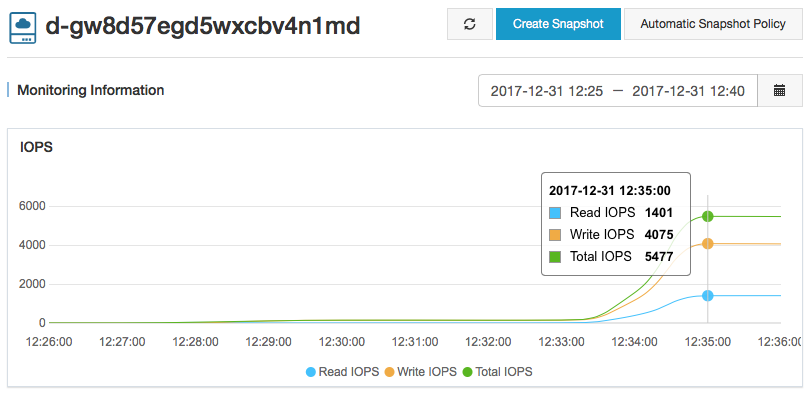


Figure . Disk IO benchmark (IOPS)

It can be seen that the benchmarking done using sysbench provides similar values (5422 vs 5477) when compared to the results of the console.

Analysing the disk monitoring interface of the console, the following results for the throughput can be observed (Figure 5).

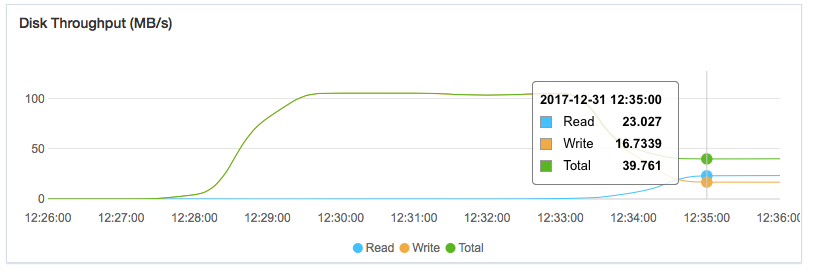


Figure . Disk IO benchmark (throughput)

It can be seen that the benchmarking done provides similar values (36Mb/s vs 39Mb/s) when compared to the results of the console. The difference is most like due to buffering being ignored by sysbench.

After the benchmark, the 30GB test file can be deleted from the system.

$ sysbench --test=fileio --file-total-size=30G cleanup

A proper performance evaluation would require to run the benchmark with different block sizes (e.g., 4, 8, 12, and 16Kb) and with different number of threads (e.g., 1, 2, 3, and 4) assuming the use of a VM with a multicore CPU.

### Network Benchmark

#### Point-to-point Connection

The network was tested using the unix utility **netperf.**

$ wget http://archive.ubuntu.com/ubuntu/pool/multiverse/n/netperf/netperf\_2.6.0-2\_i386.deb

$ sudo dpkg -i  netperf\_2.6.0-2\_i386.deb

Since netperf works with a client and a server, the server run in the AliCloud VM:

$ netserver -D -4 -L 0.0.0.0 -p 12865

The client run in a remote MacOS computer:

Jorges-iMac:osk jcardoso$ sudo netperf -H 47.254.128.190 -l 60 -t TCP\_STREAM -p 12865

The results show that the network has a throughput of 1.43 \*10^6 bits /sec when communicating between ECS and the remote MacOS.

MIGRATED TCP STREAM TEST from (null) (0.0.0.0) port 0 AF\_INET to (null) () port 0 AF\_INET

Recv   Send    Send

Socket Socket  Message  Elapsed

Size   Size    Size     Time     Throughput

bytes  bytes   bytes    secs.    10^6bits/sec

 87380 131072 131072    61.05       **1.43**

Analysing the network monitoring interface of the console, the following results for the throughput can be observed (Figure 6).

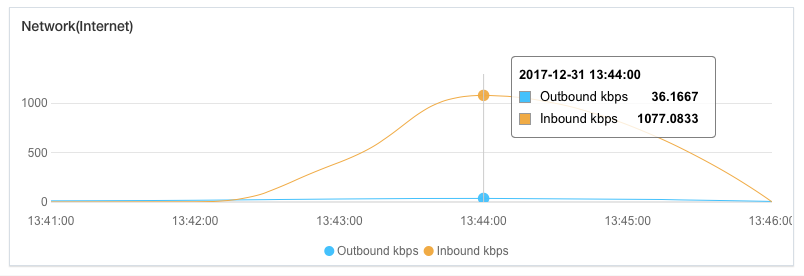


Figure . Network benchmark (client-server throughput)

Both values are roughly similar (a thorough investigation would be needed to understand why the difference exists). The throughput is mainly limited by the speed of the internet connection on the client side, i.e., the MacOS system.

The same benchmark was reexecuted using a remote client located in another cloud provider (https://www.cloud.mwn.de). The results were higher with an AliCloud VM network throughput of 104.33 \*10^6 bits /sec.

MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF\_INET to 47.254.128.190 () port 0 AF\_INET : demo

Recv   Send    Send

Socket Socket  Message  Elapsed

Size   Size    Size     Time     Throughput

bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    60.23     **104.33**

#### Internet Connection

I also tested the upload and download network speed to the internet connection of the AliCloud VM. I used speedtest.

$ wget https://raw.githubusercontent.com/sivel/speedtest-cli/master/speedtest.py

$ python speedtest.py

The results of speedtest.py were:

Retrieving speedtest.net configuration...

Testing from Alibaba (47.254.128.190)...

Retrieving speedtest.net server list...

Selecting best server based on ping...

Hosted by SoftLayer Technologies, Inc. (Frankfurt) [0.65 km]: 1.68 ms

Testing download speed................................................................................

**Download: 414.37 Mbit/s**

Testing upload speed...............................................................................................

**Upload: 35.15 Mbit/s**

The results show that the VM network has a throughput of:

* Download 414.37 Mbits/s
* Upload 35.15 Mbit/s

The tests would need to be complemented by varying the number of concurrent connections.

# RDS Benchmarking

To test the performance of the RDS service, the focus was on database IOPS, QPS, connection number and others relevant index.

## Preparation

### DB Preparation

The first step is to start launching a new DBMS.

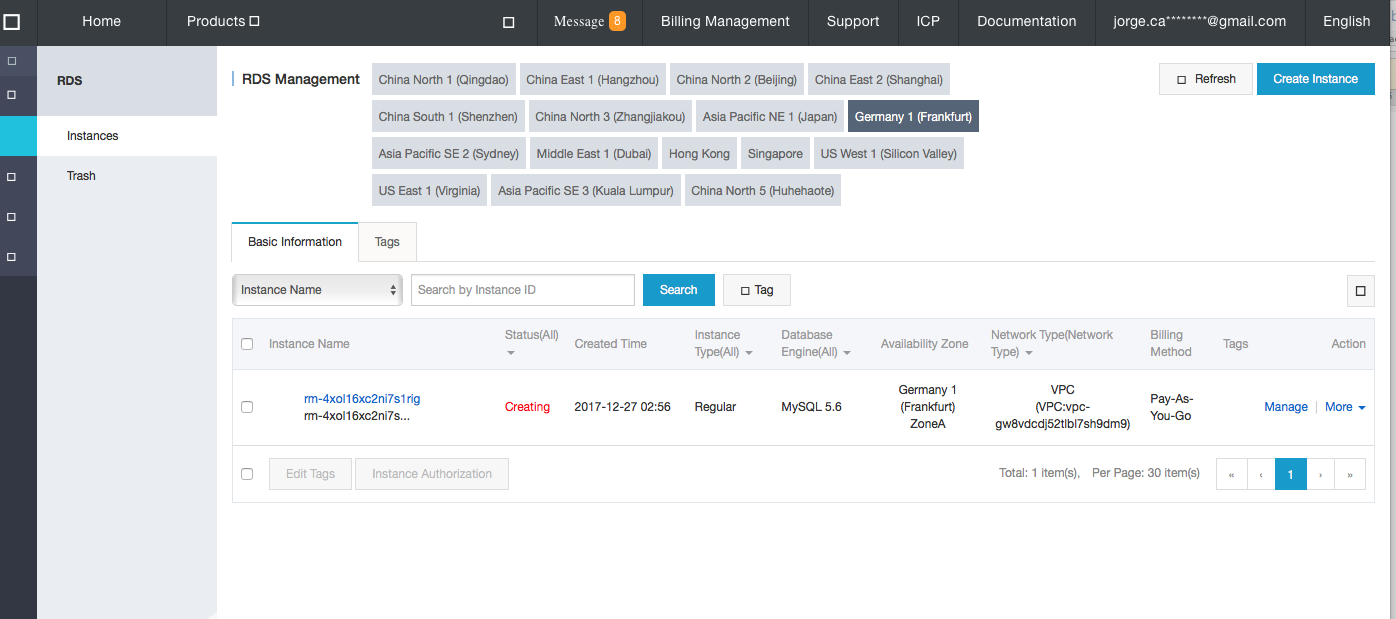


Figure . RDS creation

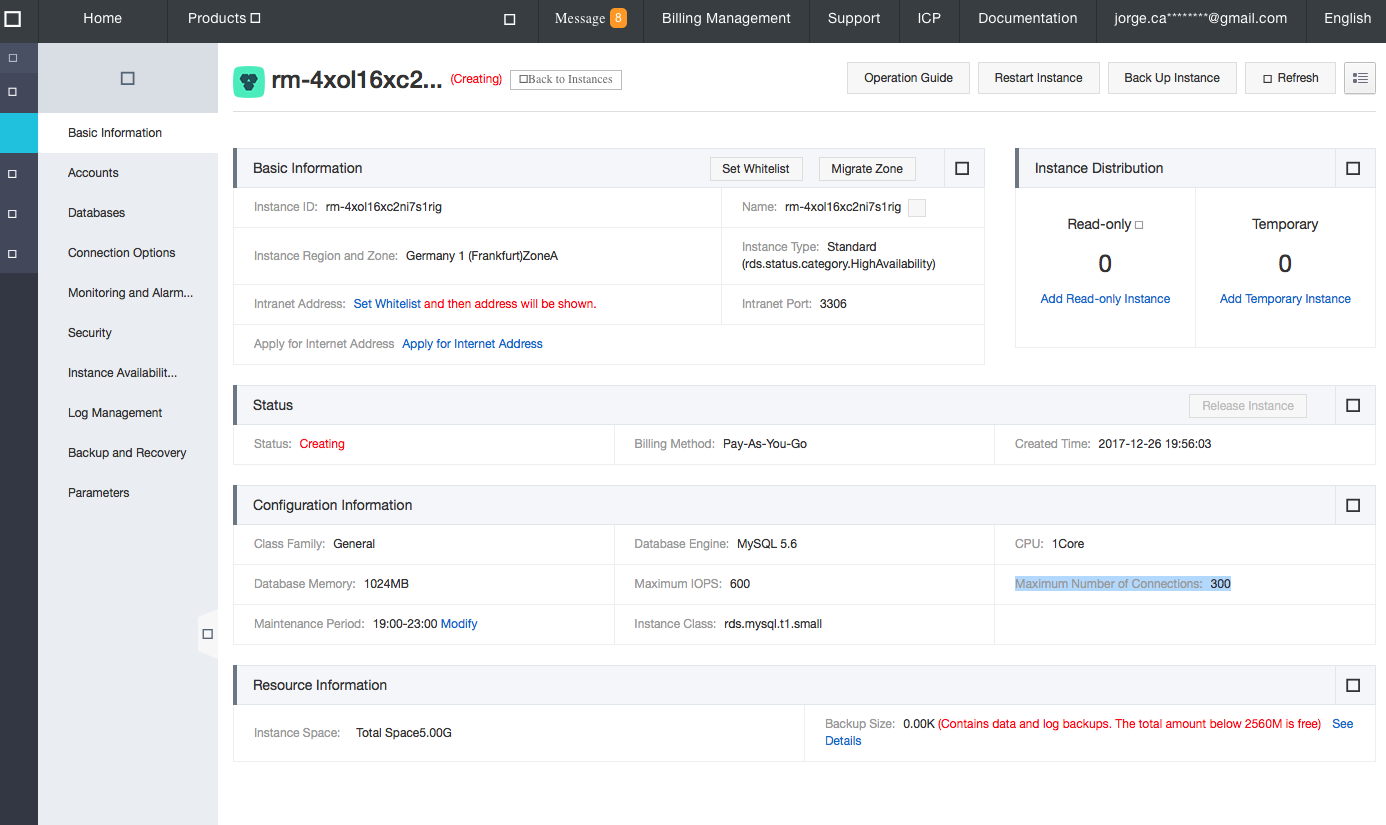


Figure . RDS view

Afterwards, create a new database called mydb.

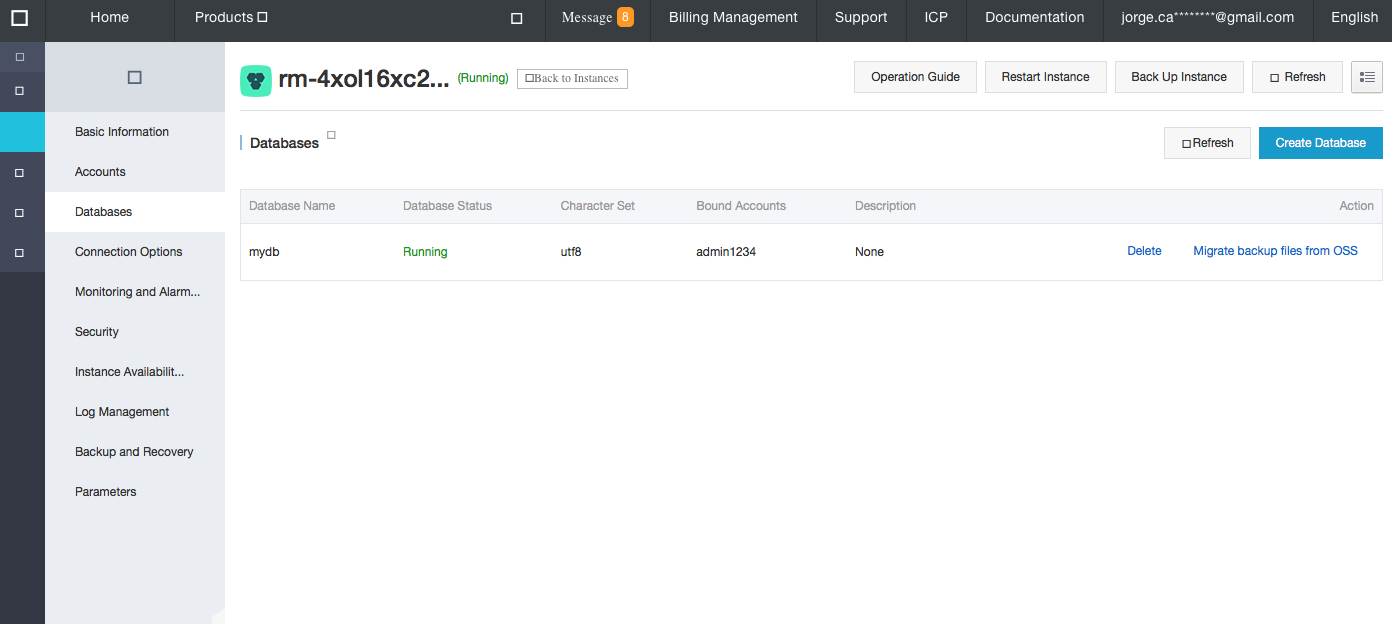


Figure . The mydb database

### VM Preparation

Install the required packages to get access to the RDS:

$ sudo apt install mysql-client-core-5.7

$ sudo apt install mariadb-client-core-10.0

### Access the database

Use the Intranet ip of database to access the database:

* rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com

$ mysql -u admin1234 -p -h rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com mydb

## DB Benchmarking

To measure MySQL performance, **sysbench** was used*.*

First, a test table in the database *test*with 1,000,000 rows of data was created:

$ root@AliCloudEx:~# sysbench --test=oltp --oltp-table-size=1000000 --mysql-db=test --mysql-user=admin1 --mysql-host=rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com --mysql-password=H\*\*\*\*2#$ --db-driver=mysql prepare

Afterwards, launch a **read test** with 8 threads using the database *test*:

$ sysbench --test=oltp --oltp-table-size=1000000 --mysql-host=rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com --mysql-db=test --mysql-user=admin1 --mysql-password=H\*\*\*\*2#$ --db-driver=mysql --max-time=60 **--oltp-read-only=on** --max-requests=0 --num-threads=8 run

The output was:

sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:

Number of threads: 8

Doing OLTP test.

Running mixed OLTP test

Doing read-only test

Using Special distribution (12 iterations,  1 pct of values are returned in 75 pct cases)

Using "BEGIN" for starting transactions

Using auto\_inc on the id column

Threads started!

Time limit exceeded, exiting...

(last message repeated 7 times)

Done.

OLTP test statistics:

    queries performed:

        read:                            322364

        write:                           0

        other:                           46052

        total:                           368416

**transactions:                        23026  (383.72 per sec.)**

    deadlocks:                           0      (0.00 per sec.)

    read/write requests:                 322364 (5372.11 per sec.)

    other operations:                    46052  (767.44 per sec.)

Test execution summary:

    total time:                          60.0070s

    total number of events:              23026

    total time taken by event execution: 479.9380

    per-request statistics:

         min:                                  6.00ms

         avg:                                 20.84ms

         max:                                 88.00ms

         approx.  95 percentile:              43.15ms

Threads fairness:

    events (avg/stddev):           2878.2500/74.80

    execution time (avg/stddev):   59.9922/0.00

The important number is the transactions per second value:

* *transactions:****23026****(383.72 per sec.)*

Afterwards, launch a **read/write test** with 8 threads using the database *test*:

$ sysbench --test=oltp --oltp-table-size=1000000 --mysql-host=rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com --mysql-db=test --mysql-user=admin1 --mysql-password=H\*\*\*\*2#$ --db-driver=mysql --max-time=60 **--oltp-read-only=off** --max-requests=0 --num-threads=8 run

The output was:

sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:

Number of threads: 8

Doing OLTP test.

Running mixed OLTP test

Using Special distribution (12 iterations,  1 pct of values are returned in 75 pct cases)

Using "BEGIN" for starting transactions

Using auto\_inc on the id column

Threads started!

Time limit exceeded, exiting...

(last message repeated 7 times)

Done.

OLTP test statistics:

    queries performed:

        read:                            174454

        write:                           62305

        other:                           24922

        total:                           261681

**transactions:                        12461  (207.64 per sec.)**

    deadlocks:                           0      (0.00 per sec.)

    read/write requests:                 236759 (3945.11 per sec.)

    other operations:                    24922  (415.27 per sec.)

Test execution summary:

    total time:                          60.0134s

    total number of events:              12461

    total time taken by event execution: 480.0241

    per-request statistics:

         min:                                  8.67ms

         avg:                                 38.52ms

         max:                                108.30ms

         approx.  95 percentile:              91.47ms

Threads fairness:

    events (avg/stddev):           1557.6250/24.65

    execution time (avg/stddev):   60.0030/0.00

The important number is the transactions per second value:

* transactions: **12461**(**207.64** per sec.)

To clean up the system afterwards (i.e., remove the test table), run:

sysbench --test=oltp --mysql-db=test --mysql-user=admin1 --mysql-host=rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com --mysql-password=H\*\*\*\*2#$ cleanup

## Results

* Using 1 threads (results not shown)
  + The number of transactions is almost 140 for read only operations
  + The number of transactions is close to 100 for r/w operations
* Using 4 threads (results not shown)
  + The number of transactions is almost 400 for read only operations
  + The number of transactions is close to 200 for r/w operations
* Using 8 threads
  + The number of transactions is almost 400 for read only operations
  + The number of transactions is close to 200 for r/w operations

The database reaches maximum number of transactions supported using 4 threads. The IOPs were not measured, only the query performance. For short tests (e.g., a few seconds to up to a few minutes), the monitoring charts of the console did not show results.

# Webapp Deployment

## Overview

Application deployment and reporting involved deploying one web application to AliCloud. The reporting includes the deployment procedure and the problems experienced. The deployment is minimal and involved using of one ECS and one RDS instances.

The application deployment was tested using WordPress web application on the cloud. The simplest deployment was carried out involving one compute instance ECS and one database service RDS. The following actions were executed:

* Apache, PHP, modules, and WordPress were installed in the ECS VM
* Create WordPress database in RDS
* Configure WordPress to use the RDS database created
* The database used was the RDS benchmarked in the previous tasks

## Preparation

To install Apache web server:

$ sudo apt-get install apache2 apache2-utils

Enable Apache2 web server to start at system boot time:

$ sudo systemctl enable apache2

$ sudo systemctl start apache2

To test that the server is running. Open the web browser and enter:

* **http://47.254.128.19**.

In case the web server is up and running, the Apache2 default index page is displayed.

Install PHP and modules for it to work with the web and database servers using the command below:

$ sudo apt-get install php7.0 php7.0-mysql libapache2-mod-php7.0 php7.0-cli php7.0-cgi php7.0-gd

Test if php is working with the web server, create a **info.php** file inside /var/www/html with the content:

<?php

phpinfo();

?>

$ sudo vi /var/www/html/info.php

To test that the server is running with php support, open the web browser and enter:

* **http://47.254.128.19**/**info.php**

Install WordPress CMS by download the latest package and extract:

$ wget -c http://wordpress.org/latest.tar.gz

$ tar -xzvf latest.tar.gz

Move the WordPress files from the extracted folder to the Apache default root directory:

$ sudo rsync -av wordpress/\* /var/www/html/

Set the permissions on the website directory to give ownership of the WordPress files to the web server as follows:

$ sudo chown -R www-data:www-data /var/www/html/

$ sudo chmod -R 755 /var/www/html/

Create the WordPress database within RDS:

$ mysql -u admin1 -p -h rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com

At the mysql shell, type the following commands:

**mysql>** CREATE DATABASE wp\_myblog;

**mysql>** GRANT ALL PRIVILEGES ON wp\_myblog.\* TO admin1@'47.254.128.190' IDENTIFIED BY 'H\*\*\*\*2#$';

**mysql>** FLUSH PRIVILEGES;

**mysql>** EXIT;

Rename the file **wp-config-sample.php** to **wp-config.php**:

$ sudo mv /var/www/html/wp-config-sample.php /var/www/html/wp-config.php

Update the file with database information:

// \*\* MySQL settings - You can get this info from your web host \*\* //

/\*\* The name of the database for WordPress \*/

define('DB\_NAME', '**wp\_myblog**'); /\*\* MySQL database username \*/ define('DB\_USER', '**admin1**'); /\*\* MySQL database password **/ define('DB\_PASSWORD', 'H\*\*\*\*2#$**'); /\*\* MySQL hostname \*/ define('DB\_HOST', '**rm-4xol16xc2ni7s1rig.mysql.germany.rds.aliyuncs.com** '); /\*\* Database Charset to use in creating database tables. \*/ define('DB\_CHARSET', '**utf8**'); /\*\* The Database Collate type. Don't change this if in doubt. \*/

define('DB\_COLLATE', **''**);

Restart the web server and mysql service:

$ sudo systemctl restart apache2.service

$ sudo systemctl restart mysql.service

## Configure Wordpress

Open web browser, enter the server address to get the welcome page to configure the system:

* **http://47.254.128.190/index.php**

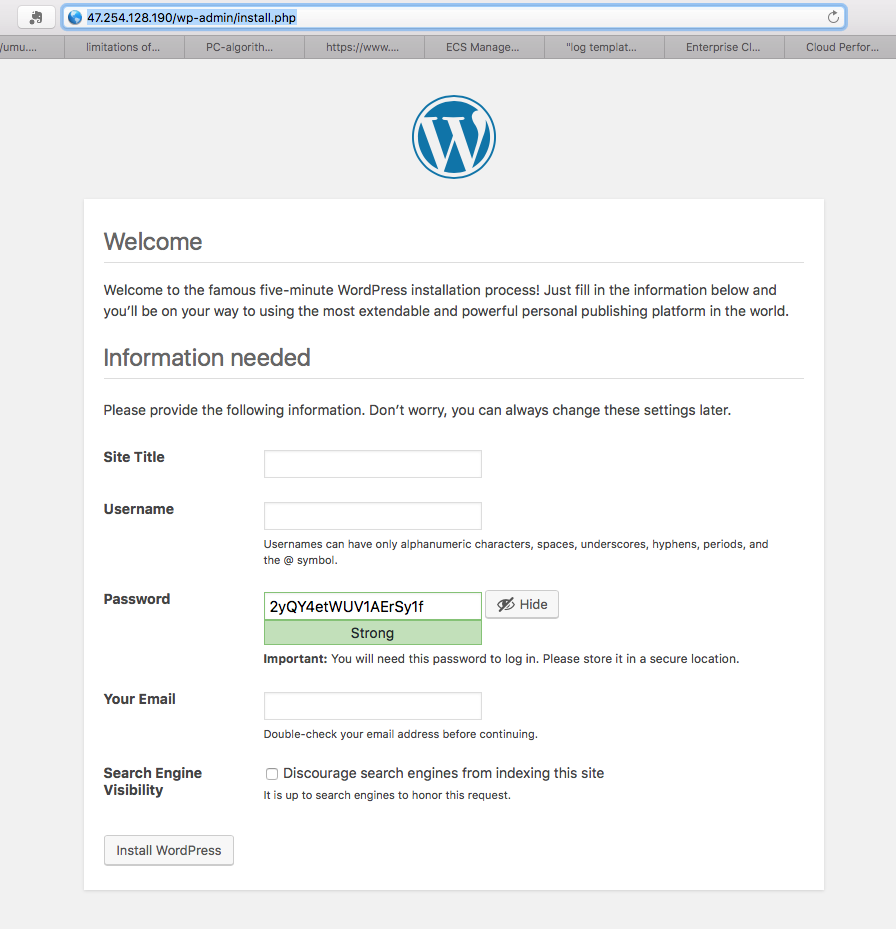


Figure . WordPress Welcome page