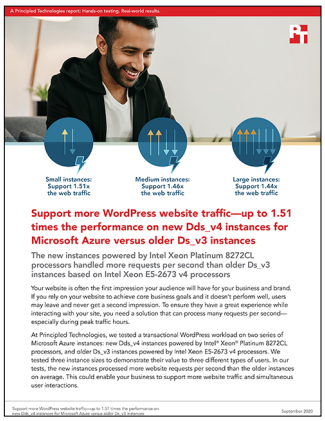

The science behind the report:

Support more WordPress website traffic—up to 1.51 times the performance on new Dds_v4 instances for Microsoft Azure versus older Ds_v3 instances

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Support more WordPress website traffic—up to 1.51 times the performance on new Dds_v4 instances for Microsoft Azure versus older Ds_v3 instances.

We concluded our hands-on testing on August 7, 2020. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on August 5, 2020 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

Table 1: Average number of requests per second each instance achieved. Source: Principled Technologies.

	D4s_v3	D4ds_v4	D4ds_v4 % increase
Small VMs (4 vCPUs)			
Requests per second	117.5	178.5	51.9%
Medium VMs (16 vCPUs)			
Requests per second	487.5	715.7	46.8%
Large VMs (64 vCPUs with MySQL query cache disabled)			
Requests per second	1,499.7	2,164.1	44.3%

System configuration information

Table 2: Detailed configuration information for the Ds_v3 instances we tested. Source: Principled Technologies.

Server configuration information	4vCPU Broadwell VM	16vCPU Broadwell VM	64vCPU Broadwell VM
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	08/06/2020	08/06/2020	08/06/2020
CSP / region	Microsoft Azure East US (Zone 2)	Microsoft Azure East US (Zone 2)	Microsoft Azure East US (Zone 2)
Workload and version (see the How we tested section for individual software component versions)	oss-performance (Note that Intel provided a script to enable the use of PHP rather than HHVM.)	oss-performance (with Intel script)	oss-performance (with Intel script)
Workload-specific parameters	php-fpm: max_children=8	php-fpm: max_children=32	php-fpm: max_children=128
Iterations and result choice	Three runs, median	Three runs, median	Three runs, median
Server platform	D4s_v3	D16s_v3	D64s_v3
BIOS name and version	Microsoft Corporation Hyper-V UEFI Release v2.0, 8/26/2016	Microsoft Corporation Hyper-V UEFI Release v2.0, 8/26/2016	Microsoft Corporation Hyper-V UEFI Release v2.0, 8/26/2016
Operating system name and version/build number	Ubuntu Server 20.04 LTS Gen2	Ubuntu Server 20.04 LTS Gen2	Ubuntu Server 20.04 LTS Gen2
Date of last OS updates/ patches applied	08/04/2020	08/04/2020	08/04/2020
Processor			
Number of processors	1	1	2
Vendor and model	Intel® Xeon® CPU E5-2673 v4	Intel Xeon CPU E5-2673 v4	Intel Xeon CPU E5-2673 v4
Core count (per processor)	20	20	20
Core frequency (GHz)	2.30	2.30	2.30
Stepping	1	1	1
Hyper-threading	Yes	Yes	Yes
Turbo	Yes	Yes	Yes
Number of vCPUs per VM	4	16	64
Memory module(s)			
Total memory in system (GB)	16	64	256
NVMe memory present?	No	No	No
Total memory (DDR + NVMe RAM)	16	64	256
General hardware			
Storage: network or direct-attached / instance	Direct-attached	Direct-attached	Direct-attached
Local storage (OS)			
Number of drives	1	1	1
Drive size (GB)	30	30	30

Server configuration information	4vCPU Broadwell VM	16vCPU Broadwell VM	64vCPU Broadwell VM
Drive information	Standard SSD	Standard SSD	Standard SSD
Temporary drive			
Number of drives	1	1	1
Drive size (GB)	32	128	512
Network adapter			
Vendor and model	Microsoft Hyper-V Network Adapter	Microsoft Hyper-V Network Adapter	Microsoft Hyper-V Network Adapter
Number and type of ports	1x 40Gb	1x 50Gb	1x 50Gb

Table 3: Detailed configuration information for the Dds_v4 instances we tested. Source: Principled Technologies.

Server configuration information	4vCPU Cascade Lake VM	16vCPU Cascade Lake VM	64vCPU Cascade Lake VM
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	08/06/2020	08/06/2020	08/06/2020
CSP / Region	Microsoft Azure East US (Zone 2)	Microsoft Azure East US (Zone 2)	Microsoft Azure East US (Zone 2)
Workload & version	oss-performance (with Intel script)	oss-performance (with Intel script)	oss-performance (with Intel script)
WL specific parameters	php-fpm: max_children=8	php-fpm: max_children=32	php-fpm: max_children=128
Iterations and result choice	3 runs, median	3 runs, median	3 runs, median
Server platform	D4ds_v4	D16ds_v4	D64ds_v4
BIOS name and version	Microsoft Corporation Hyper-V UEFI Release v4.0, 3/12/2019	Microsoft Corporation Hyper-V UEFI Release v4.0, 3/12/2019	Microsoft Corporation Hyper-V UEFI Release v4.0, 3/12/2019
Operating system name and version/build number	Ubuntu Server 20.04 LTS Gen2	Ubuntu Server 20.04 LTS Gen2	Ubuntu Server 20.04 LTS Gen2
Date of last OS updates/ patches applied	08/04/2020	08/04/2020	08/04/2020
Processor			
Number of processors	1	1	2
Vendor and model	Intel Xeon Platinum 8272CL	Intel Xeon Platinum 8272CL	Intel Xeon Platinum 8272CL
Core count (per processor)	26	26	26
Core frequency (GHz)	2.60	2.60	2.60
Stepping	7	7	7
Hyper-Threading	Yes	Yes	Yes
Turbo	Yes	Yes	Yes
Number of vCPU per VM	4	16	64
Memory module(s)			
Total memory in system (GB)	16	64	256
NVMe memory present?	No	No	No

Server configuration information	4vCPU Cascade Lake VM	16vCPU Cascade Lake VM	64vCPU Cascade Lake VM
Total memory (DDR+NVMe RAM)	16	64	256
General HW			
Storage: NW or Direct Att / Instance	Direct Att	Direct Att	Direct Att
Network BW / Instance	N/A	N/A	N/A
Storage BW / Instance	N/A	N/A	N/A
Local storage (OS)			
Number of drives	1	1	1
Drive size (GB)	30	30	30
Drive information	Standard SSD	Standard SSD	Standard SSD
Temporary drive			
Number of drives	1	1	1
Drive size (GB)	150	600	2,400
Drive information	Not listed	Not listed	Not listed
Network adapter			
Vendor and model	Microsoft Hyper-V Network Adapter	Microsoft Hyper-V Network Adapter	Microsoft Hyper-V Network Adapter
Number and type of ports	1x 50Gb	1x 50Gb	1x 50Gb

How we tested

Testing overview

We created a baseline web VM in our Microsoft Azure account with Ubuntu Server 20.04 LTS. On this base VM, we upgraded to the latest Ubuntu packages, installed Nginx (version 1.18.0), PHP (version 7.4.3), Mariadb (version 10.3.22), and installed the oss-performance benchmark (Facebook commit revision cb766d2). The software connected to the web VM over the basic Azure network. We created a snapshot of this base VM and used it to create a specialized image. We then created all of our test instances using this image to ensure that our base settings were consistent across all testing.

We used the same size of database and workload for all instances. Due to the different number of vCPUs between instances, some tuning settings differ among the three pairs. However, within each pair, we kept everything as identical as possible outside of the instance types themselves. See below for the steps we followed. The benchmark does a quick warm-up as part of the testing. Because our databases fit within the RAM, we didn't need to worry about fluctuations in disk performance. Even so, we used Standard SSD storage for all instances to ensure no delay in loading the database into memory.

Creating the Ubuntu Server 20.04 LTS baseline image

This section contains the steps we took to create our baseline image.

Creating the baseline image VM

1. Log into the Azure Portal, and navigate to the Virtual Machines service.
2. To open the Add VM wizard, click Add.
3. On the Basics tab, set the following:
    a. For Subscription, choose your from the drop-down menu.
    b. For Resource, choose your research group from the drop-down menu.
    c. Name the Virtual Machine.
    d. For Region, choose your region from the drop-down menu.
    e. Leave the Availability options set to No infrastructure redundancy required.
    f. Click Browse all public and private images.
    g. In the Search field, enter Ubuntu Server 20.04 LTS. From the list of results, select "Ubuntu Server 20.04 LTS - Gen2."
    h. Leave Azure Spot instance set to No.
    i. Select the instance size you wish to use. We used Standard B4ms.
    j. Leave the Authentication type set to SSH public key.
    k. Choose either a new Username, or leave the default.
    l. Choose Generate new key pair for the SSH public key source.
    m. Enter a name for the Key pair name.
    n. Leave Public inbound ports set to Allow selected ports.
    o. For Select inbound ports, choose SSH (22).
4. On the Disks tab, set the following:
    a. For the OS disk type, choose Standard HDD from the drop-down menu.
    b. Leave the default Encryption type.
5. On the Networking tab, set the following:
    a. From the drop-down menu, choose your virtual network.
    b. To create a new Public IP, choose Create new.
    c. Leave the rest of the settings at defaults.
6. On the Management tab, set the following:
    a. Choose your Diagnostics storage account from the dropdown menu.
    b. Leave the rest set to defaults.
7. On the Advanced tab, leave all defaults.
8. On the Tags tab, add any tags you wish to use.
9. On the Review + create tab, review your settings, and click Create.

Configuring Ubuntu Server 20.04 LTS

1. Using the SSH key generated during Azure instance creation, log in as testuser. For example:

```
ssh -i wordpress_key.pem testuser@<INSTANCE_PUBLIC_IP_ADDRESS>
```

2. Add the hostname entry:

```
echo "127.0.1.1 wordpress-server" | sudo tee /etc/hosts
```

3. Install the latest update packages and reboot the VM:

```
sudo apt update
sudo apt upgrade -y
sudo reboot
```

4. Install additional tools:

```
sudo apt install -y nmon sysstat numactl ksh
```

Installing oss-performance benchmark and its prerequisites

1. Download or copy the benchmark archive and extract it:

```
tar -xf oss-performance.tar.gz
```

2. To install prerequisites, run the included setup.sh script:

```
cd oss-performance/scripts
./setup.sh
```

3. Disable services that conflict with the benchmark:

```
sudo systemctl disable --now apache2
sudo systemctl disable --now nginx
sudo systemctl disable --now php7.4-fpm phpsessionclean.timer phpsessionclean
```

4. Configure Mariadb:

```
sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf
max_connections        = 1000
sudo systemctl restart mariadb.service
sudo systemctl enable mariadb.service
```

5. Finish Mariadb setup:

```
sudo mysql_secure_installation
Enter current password for root (enter for none):
Set root password? [Y/n]: Y
New password: <PASSWORD>
Re-enter new password: <PASSWORD>
Remove anonymous users? [Y/n]: Y
Disallow root login remotely? [Y/n]: Y
Remove test database and access to it? [Y/n]:  Y
Reload privilege tables now? [Y/n]:  Y
```

6. Set up Mariadb permissions for the benchmark:

```
sudo mysql -u root -p
USE mysql;
UPDATE user SET plugin='mysql_native_password' WHERE User='root';
GRANT ALL ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
EXIT;
```

Creating the WordPress database

In this section, you will create a WordPress database comprising the following types of content that we determined specifically for our testing purposes.

- 50 WordPress front pages
- 40 RSS feeds
- 66 posts, consisting of:
  - 6 copies of 2 unique popular posts (12 total)
  - 3 copies of 3 unique, slightly less popular post (9 total)
  - 2 copies of 13 unique, moderately popular posts (26 total)
  - 1 copy of 19 unique, unpopular posts (19 total)

1. Start the benchmark with a pause:

```
php perf.php --wordpress --php5=/usr/bin/php-cgi --wait-after-warmup &
```

2. Open a web browser to the VM running the paused benchmark using the assigned network port. The default is port 8090. For example:

```
http://example.com:8090/
```

3. Follow the setup wizard instructions.
4. Select Language, click Continue.
5. Enter username, password, and email address, then click Continue.
6. Make the following changes to the WordPress default settings and pages:
   a. Settings -> Permalinks
   b. Under "Common Settings", Select "Plain" Permalink style and then click "Save Changes" button
   c. Pages -> All Pages
   d. Select Page "Privacy Policy" -> Quick Edit ->
7. Status: Published
8. Click the Update button
9. Select Pages -> All Pages
10. Select all -> Bulk Actions -> Move to Trash -> Click "Apply." A message will appear alerting you to the pages' removal.
11. Click "Trash" Link
12. Click "Empty Trash" button. A message will appear alerting you to the pages' permanent deletion.
13. Select Posts -> All Posts
14. Select all -> Bulk Actions -> Move to Trash -> Click "Apply." A message will appear alerting you to the post's removal..
15. Click "Trash" Link
16. Click "Empty Trash" button. A message will appear alerting you to the post's permanent deletion.
17. Install the Demo Data Creator 1.3.4 plugin.
18. Run the plugin by navigating to Tools -> Demo Data Creator
19. To create the initial demo data, you must create settings for Users, Categories, Pages, Posts, Comments, and Links. Start by adjusting the Users settings to the following:
    - Number of users: 200 (default 100)
    - User email template (with [x] for the user ID): demouser[x]@example.com
20. Click Create users. A message will appear alerting you to the creation of 200 demo users.
21. Now, adjust the settings for Categories:
    - Maximum number of categories (max 25): 10
22. Click Create categories. A message will appear alerting you to the creation of 7 demo categories.
23. Skipping over the Posts section for now, adjust the Pages settings to the following:
    - Maximum number of pages (max 50): 25
    - Maximum number of top-level pages (max 10): 5
    - Maximum number of level to nest pages (max 5): 3
    - Maximum number of blog page paragraphs (min 1, max 50): 10
24. Click Create pages. A message will appear alerting you to the creation of 10 demo pages.
25. Now, adjust the settings for Posts:
    - Maximum number of posts (max 100): 50
    - Maximum number of blog post paragraphs (min 1, max 50): 10

26. Click Create posts repeatedly until the total number of pages and posts is 52 or greater. Several message will appear alerting you to the number of demo posts and pages you have just created.
27. Now, adjust the settings for Comments:
    - Maximum number of comments per post (max 50): 10
28. Click Create comments. A message will appear alerting you to the creation of demo comments.
29. Now, adjust the settings for Links:
    - Maximum number of links in blogroll (max 100): 25
30. Click Create links. A message will appear alerting you to the creation of demo links.
31. Log out from the admin user.
32. Back up the database the benchmark will use:

```
cd oss-benchmark/targets/wordpress
mysqldump -u root -p wp_bench > dbdump.sql
gzip dbdump.sql
```

Creating a snapshot of your baseline VM

1. In your Azure portal, navigate to the Snapshots service.
2. To open the Snapshot wizard, click Add.
3. On the Basics tab, set the following:
    a. For Subscription, choose your Subscription.
    b. For Resource group, choose your resource group.
    c. Enter a name for your snapshot.
    d. For Region, choose your region.
    e. Select Full - make a complete read-only copy of the selected disk for the Snapshot type.
    f. Choose the OS disk from your baseline VM.
    g. For the Storage type, choose Standard HDD.
4. On the Encryption tab, leave all defaults.
5. On the Tags tab, add any tags you wish to use.
6. On the Review + create tab, review your settings, and click Create.

Creating your image with the baseline snapshot

To create an image, you must first have a Shared Image Gallery. The steps below will walk you through the creation of the gallery as well as the image creation steps. Once you have created your gallery, you will not need to do so again to add new images.

1. In your Azure portal, navigate to the Shared image galleries service.
2. To open the Add gallery wizard, click Add.
3. On the Basics tab, set the following:
    a. For Subscription, choose your subscription.
    b. For Resource group, choose your resource group.
    c. Name your gallery.
    d. For Region, choose your region.
    e. If you would like, enter a description.
4. On the Tags tab, add any tags you wish to use.
5. On the Review + create tab, review your settings, and click Create.
6. Click on your new image gallery, and click Add new image definition to open the wizard.
7. On the Basics tab, set the following:
    a. Set the Operating System to Windows.
    b. Set the VM generation to Gen 2.
    c. Set the Operation system state to Specialized.
    d. Enter whatever you wish for the Publisher, Offer, and SKU entries.
8. Skip the Version tab.
9. Skip the Publishing options tab.
10. On the Tags tab, add any tags you wish to use.
11. On the Review + create tab, review your settings, and click Create.
12. Click on the image definition you've created, and click Add version to open the wizard.

13. On the Basics tab, set the following:
    a. Enter a version number such as `1.0.0`
    b. From the drop-down menu, choose the OS disk snapshot of the baseline VM you created.
    c. Leave the rest as defaults.
14. On the Encryption tab, leave defaults.
15. On the Tags tab, add any tags you wish to use.
16. On the Review + create tab, review your settings, and click Create.

Creating the VMs under test

In this section, we list the steps required to create a VM from the image we created previously. Follow the steps six times to create the three Dds_v4 and three Ds_v3 VMs (see Table 2 and Table 3 for full instance details). In addition, create a D4ds_v4 instance for the benchmark client (named `wordpress-client`). For our testing, we used the East US Region and Availability Zone 2.

Creating the VMs from the specialized image

1. Open the Azure Portal and navigate to the Share image galleries service.
2. Click on the Shared image gallery you created.
3. Navigate to the image version you created (this should be something like 1.0.0), and click Create VM.
4. On the Basics tab, set the following:
    a. For Subscription, choose your subscription..
    b. For Resource group, choose your resource group..
    c. Enter a Virtual machine name.
    d. Choose your Availability Zone, and set the Zone you desire.
    e. Select the instance size you want.
    f. Leave the rest as defaults.
5. On the Disks tab, set the following:
    a. Change the OS disk type to Standard SSD.
    b. Click OK.
6. Skip the Networking, Management, and Advanced tabs.
7. On the Tags tab, assign any tags you wish to use.
8. On the Review + create tab, review your settings, and click Create.
9. Once the VM creation is complete, click Go to resource (or, navigate to the virtual machine service, and click the new VM).

Running the tests

In this section, we list the steps to run the oss-performance benchmark on the VMs under test. We created a script to automate the run instructions from the oss workload documentation as well as to start and stop the monitoring tools we used to track CPU utilization. We set the runtime parameter to 300 seconds, the step parameter to 5, and the client_thread parameter to 200. For the full script we used, see the Run_test.sh section on page 10.

1. On the VM you're testing, configure the hostname to include a unique identifier. For example:

```
sudo hostnamectl set-hostname wp-D4s-v3-2673v4
```

2. On the client, run the benchmark script substituting the hostname of the VM under test.

```
./run_test.sh <HOSTNAME_OF_VM_UNDER_TEST>
```

3. Repeat the benchmark test three times across all VM instance types. The benchmark will automatically save all results in the results/ subdirectory of the client VM.

Determining CPU vulnerability mitigation

We ran the following command on each VM to determine the Intel processor mitigation settings that Azure employs:
`lscpu | grep Vulnerability`

Figures 1 and 2 display the output for the D4s_v3 VM and the D4ds_v4 VM.

Figure 1: D4s_v3

```
Vulnerability Itlb multihit: KVM: Mitigation: Split huge pages
Vulnerability L1tf: Mitigation; PTE Inversion; VMX conditional cache flushes
Vulnerability Mds: Mitigation; Clear CPU buffers; SMT Host state unknown
Vulnerability Meltdown: Mitigation; PTI
Vulnerability Spec store bypass: Vulnerable
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2: Mitigation; Full generic retpoline
Vulnerability Srbds: Not affected
Vulnerability Tsx async abort: Mitigation; Clear CPU buffers; SMT Host state unknown
```

Figure 2: D4ds_v4

```
Vulnerability Itlb multihit: KVM: Mitigation: Split huge pages
Vulnerability L1tf: Not affected
Vulnerability Mds: Not affected
Vulnerability Meltdown: Not affected
Vulnerability Spec store bypass: Vulnerable
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2: Mitigation; Full generic retpoline
Vulnerability Srbds: Not affected
Vulnerability Tsx async abort: Vulnerable: Clear CPU buffers attempted
```

Run_test.sh script

Below is the full script we used to automate our tests.

```
#!/bin/bash
RUNTIME=300
STEP=5
COUNT=$((RUNTIME/STEP))
TIMESTAMP=$(date '+%Y%m%d_%H%M%S')
CLIENT_THREADS=200

TEST_HOST=${1}
REMOTE_SIEGE=${2:-$(hostname -s)}
#REMOTE_SIEGE=$(hostname -s)
#REMOTE_SIEGE=wordpress-server

echo -n "siege host: "
ssh ${TEST_HOST} "ssh ${REMOTE_SIEGE} 'echo ${REMOTE_SIEGE}'"
sleep 1

scp -p 50-server.cnf ${TEST_HOST}:
ssh ${TEST_HOST} "sudo mv -f 50-server.cnf /etc/mysql/mariadb.conf.d/50-server.cnf ; sync ; sudo
systemctl restart mysqld"

RESULTS_DIR=results/${REMOTE_SIEGE}_${TEST_HOST}_${TIMESTAMP}
mkdir -p ${RESULTS_DIR}
RESULTS_FILE=${TEST_HOST}_${TIMESTAMP}

killall -w nmon
sleep 1
if [ "${REMOTE_SIEGE}" == "$(hostname -s)" ]; then
  nmon -F ${RESULTS_DIR}/wordpress-client_${TIMESTAMP}.nmon -s${STEP} -J -t -C nginx:mysqld:php:siege
:nmon:systemd-resolve
fi
```

```
ssh ${TEST_HOST} "cd oss-performance ; php perf.php \
  --wordpress \
  --php=/usr/sbin/php-fpm7.4 \
  --remote-siege=testuser@${REMOTE_SIEGE} \
  --i-am-not-benchmarking \
  --client-threads=${CLIENT_THREADS} \
  --benchmark-time=${RUNTIME} \
  --exec-after-warmup='nmon -F /tmp/${TEST_HOST}.nmon -s${STEP} -c${COUNT} -J -t -C nginx:mysqld:php
:siege:nmon:systemd-resolve' \
  --trace 2>&1" | tee ${RESULTS_DIR}/${RESULTS_FILE}.txt

killall -w nmon
scp ${TEST_HOST}:/tmp/${TEST_HOST}.nmon ${RESULTS_DIR}/${RESULTS_FILE}.nmon
ssh ${TEST_HOST} "pkill nmon ; sleep 1 ; rm -f /tmp/${TEST_HOST}.nmon"
for nmonfile in `find ${RESULTS_DIR}/*.nmon`;
do
  ./nmonchart $nmonfile
done
cp -pf ${0} ${RESULTS_DIR}/
```

Read the report at http://facts.pt/yhxzcsy

This project was commissioned by Intel.