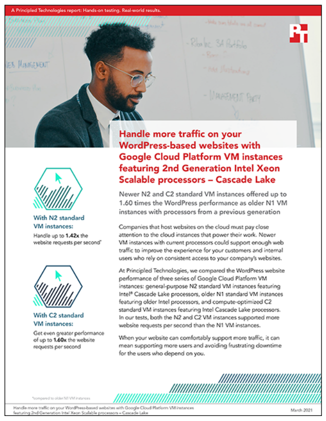

A Principled Technologies report: Hands-on testing. Real-world results.

Handle more traffic on your WordPress-based websites with Google Cloud Platform VM instances featuring 2nd Generation Intel Xeon Scalable processors – Cascade Lake

Newer N2 and C2 standard VM instances offered up to 1.60 times the WordPress performance as older N1 VM instances with processors from a previous generation

With N2 standard VM instances: Handle up to **1.42x** the website requests per second*

With C2 standard VM instances: Get even greater performance of up to **1.60x** the website requests per second

Companies that host websites on the cloud must pay close attention to the cloud instances that power their work. Newer VM instances with current processors could support enough web traffic to improve the experience for your customers and internal users who rely on consistent access to your company's websites.

At Principled Technologies, we compared the WordPress website performance of three series of Google Cloud Platform VM instances: general-purpose N2 standard VM instances featuring Intel® Cascade Lake processors, older N1 standard VM instances featuring older Intel processors, and compute-optimized C2 standard VM instances featuring Intel Cascade Lake processors. In our tests, both the N2 and C2 VM instances supported more website requests per second than the N1 VM instances.

When your website can comfortably support more traffic, it can mean supporting more users and avoiding frustrating downtime for the users who depend on you.

*compared to older N1 VM instances

Handle more traffic on your WordPress-based websites with Google Cloud Platform VM instances featuring 2nd Generation Intel Xeon Scalable processors – Cascade Lake March 2021

The science behind the report:

Handle more traffic on your WordPress-based websites with Google Cloud Platform VM instances featuring 2nd Generation Intel Xeon Scalable processors – Cascade Lake

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 Handle more traffic on your WordPress-based websites with Google Cloud Platform VM instances featuring 2nd Generation Intel Xeon Scalable processors – Cascade Lake .

We concluded our hands-on testing on January 12, 2021. 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 January 11, 2021or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Results of our WordPress testing on Google Cloud Platform.

WordPress performance	Requests per second
4 vCPU results	
n1-standard-4	114.85
n2-standard-4	156.81
c2-standard-4	184.82
8 vCPU results	
n1-standard-8	270.9
n2-standard-8	352.45
c2-standard-8	373.19
16 vCPU results	
n1-standard-16	470.06
n2-standard-16	670.23
c2-standard-16	742.17

System configuration information

Table 2: Detailed information on the N1 standard VM instances we tested.

System configuration information	4vCPU N1 standard VM instance	8vCPU N1 standard VM instance	16vCPU N1 standard VM instance
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	01/12/2021	01/12/2021	01/12/2021
CSP / Region	us-east1-b	us-east1-b	us-east1-b
Workload & version	oss-performance (Intel modified)	oss-performance (Intel modified)	oss-performance (Intel modified)
WL specific parameters	php-fpm: max_children=8	php-fpm: max_children=16	php-fpm: max_children=32
Iterations and result choice	3 runs, median	3 runs, median	3 runs, median
Server platform	n1-standard-4	n1-standard-8	n1-standard-16
BIOS name and version	Google Google, 1/1/2011	Google Google, 1/1/2011	Google Google, 1/1/2011
Operating system name and version/build number	Ubuntu Server 20.04 LTS	Ubuntu Server 20.04 LTS	Ubuntu Server 20.04 LTS
Date of last OS updates/patches applied	01/11/2021	01/11/2021	01/11/2021
Processor			
Number of processors	1	1	1
Vendor and model	Intel Xeon E5-26XX v4	Intel Xeon E5-26XX v4	Intel Xeon E5-26XX v4
Core count (per processor)	unknown	unknown	unknown
Core frequency (GHz)	2.20	2.20	2.20
Stepping	0	0	0
Hyper-Threading	Yes	Yes	Yes
Turbo	Yes	Yes	Yes
Number of vCPU per VM	4	8	16
Memory module(s)			
Total memory in system (GB)	15	30	60
NVMe memory present?	No	No	No
Total memory (DDR+NVMe RAM)	15	30	60
General hardware			
Storage: NW or Direct Att / Instance	NW Att	NW Att	NW Att
Network BW / Instance	N/A	N/A	N/A
Storage BW / Instance	N/A	N/A	N/A

System configuration information	4vCPU N1 standard VM instance	8vCPU N1 standard VM instance	16vCPU N1 standard VM instance
Local storage			
OS			
Number of drives	1	1	1
Drive size (GB)	30	30	30
Drive information (speed, interface, type)	Standard SSD	Standard SSD	Standard SSD
Temporary drive			
Number of drives	0	0	0
Drive size (GB)	N/A	N/A	N/A
Network adapter			
Vendor and model	Google VirtIO Ethernet Adapter	Google VirtIO Ethernet Adapter	Google VirtIO Ethernet Adapter
Number and type of ports	1x 100Gb	1x 100Gb	1x 100Gb

Table 3: Detailed information on the N2 standard VM instances we tested.

System configuration information	4vCPU N2 standard VM instance	8vCPU N2 standard VM instance	16vCPU N2 standard VM instance
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	01/12/2021	01/12/2021	01/12/2021
CSP / Region	us-east1-b	us-east1-b	us-east1-b
Workload & version	oss-performance (Intel modified)	oss-performance (Intel modified)	oss-performance (Intel modified)
WL specific parameters	php-fpm: max_children=8	php-fpm: max_children=16	php-fpm: max_children=32
Iterations and result choice	3 runs, median	3 runs, median	3 runs, median
Server platform	n2-standard-4	n2-standard-8	n2-standard-16
BIOS name and version	Google Google, 1/1/2011	Google Google, 1/1/2011	Google Google, 1/1/2011
Operating system name and version/build number	Ubuntu Server 20.04 LTS	Ubuntu Server 20.04 LTS	Ubuntu Server 20.04 LTS
Date of last OS updates/patches applied	01/11/2021	01/11/2021	01/11/2021
Processor			
Number of processors	1	1	1
Vendor and model	Intel Xeon Cascade Lake	Intel Xeon Cascade Lake	Intel Xeon Cascade Lake
Core count (per processor)	unknown	unknown	unknown
Core frequency (GHz)	2.80	2.80	2.80
Stepping	7	7	7
Hyper-Threading	Yes	Yes	Yes
Turbo	Yes	Yes	Yes

System configuration information	4vCPU N2 standard VM instance	8vCPU N2 standard VM instance	16vCPU N2 standard VM instance
Number of vCPU per VM	4	8	16
Memory module(s)			
Total memory in system (GB)	16	32	64
NVMe memory present?	No	No	No
Total memory (DDR+NVMe RAM)	16	32	64
General hardware			
Storage: NW or Direct Att / Instance	NW Att	NW Att	NW 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 (speed, interface, type)	Standard SSD	Standard SSD	Standard SSD
Temporary drive			
Number of drives	0	0	0
Drive size (GB)	N/A	N/A	N/A
Network adapter			
Vendor and model	Google VirtIO Ethernet Adapter	Google VirtIO Ethernet Adapter	Google VirtIO Ethernet Adapter
Number and type of ports	1x 100Gb	1x 100Gb	1x 100Gb

Table 4: Detailed information on the C2 standard VM instances we tested.

System configuration information	4vCPU C2 standard VM instance	8vCPU C2 standard VM instance	16vCPU C2 standard VM instance
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	01/12/2021	01/12/2021	01/12/2021
CSP / Region	us-east1-b	us-east1-b	us-east1-b
Workload & version	oss-performance (Intel modified)	oss-performance (Intel modified)	oss-performance (Intel modified)
WL specific parameters	php-fpm: max_children=8	php-fpm: max_children=16	php-fpm: max_children=32
Iterations and result choice	3 runs, median	3 runs, median	3 runs, median
Server platform	n2-standard-4	n2-standard-8	n2-standard-16
BIOS name and version	Google Google, 1/1/2011	Google Google, 1/1/2011	Google Google, 1/1/2011
Operating system name and version/build number	Ubuntu Server 20.04 LTS	Ubuntu Server 20.04 LTS	Ubuntu Server 20.04 LTS

System configuration information	4vCPU C2 standard VM instance	8vCPU C2 standard VM instance	16vCPU C2 standard VM instance
Date of last OS updates/patches applied	01/11/2021	01/11/2021	01/11/2021
Processor			
Number of processors	1	1	1
Vendor and model	Intel® Xeon® Cascade Lake	Intel® Xeon® Cascade Lake	Intel® Xeon® Cascade Lake
Core count (per processor)	unknown	unknown	unknown
Core frequency (GHz)	3.10	3.10	3.10
Stepping	7	7	7
Hyper-Threading	Yes	Yes	Yes
Turbo	Yes	Yes	Yes
Number of vCPU per VM	4	8	16
Memory module(s)			
Total memory in system (GB)	16	32	64
NVMe memory present?	No	No	No
Total memory (DDR+NVMe RAM)	16	32	64
General hardware			
Storage: NW or Direct Att / Instance	NW Att	NW Att	NW 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 (speed, interface, type)	Standard SSD	Standard SSD	Standard SSD
Temporary drive			
Number of drives	0	0	0
Drive size (GB)	N/A	N/A	N/A
Network adapter			
Vendor and model	Google VirtIO Ethernet Adapter	Google VirtIO Ethernet Adapter	Google VirtIO Ethernet Adapter
Number and type of ports	1x 100Gb	1x 100Gb	1x 100Gb

How we tested

Testing overview

We created a baseline Web VM in our Google Cloud account with Ubuntu Server 20.04 LTS. On this base VM, we upgraded to the latest Ubuntu packages, installed Nginx, PHP, Mariadb, and installed the oss-performance benchmark. The benchmark uses HHVM which no longer supports PHP code. Thus, Intel provided us a script to modify the benchmark to use PHP instead.

We created a snapshot of the 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 database and workload for all instances. See below for the steps we followed as well as tables to show the VM configurations and settings. The benchmark does a quick warmup as part of the testing and since our databases fit in RAM we didn't need to worry about fluctuations in disk performance, although we used Standard SSD storage for all instances just to ensure there was no delay 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 Instance

1. Log into Google Cloud and click on Go to console.
2. Click Compute engine, then click VM instances.
3. Click the Create.
4. In the left window, select New VM instance.
5. Add the following information:
    a. Name: Name your VM instance.
    b. Labels: Use any appropriate labels.
    c. Region: Select your desired region.
    d. Zone: Select your desired zone.
    e. Machine Configuration:
    f. Machine family: General-purpose
    g. Series: N1
    h. Machine type: n1-standard-4
    i. CPU platform: Automatic
    j. Keep Turn on display device unchecked.
    k. Keep Confidential VM Service and Container unchecked.
    l. Boot Disk, click Change.
        i. Operating System: Ubuntu
        ii. Version: Ubuntu 20.04 LTS
        iii. Boot disk type: SSD persistent disk
        iv. Size: 30GB
        v. Click Select
    m. Identity and API access: Compute Engine default service account.
    n. Firewall: Check Allow HTTP traffic and Allow HTTPs traffic.
    o. Click Create.

Configuring Ubuntu Server 20.04 LTS

1. Log into Google Cloud and click on Go to console.
2. Click Compute engine, then click VM instances.
3. In the dropdown menu next to your baseline VM instance, click SSH.
4. Once logged in, add hostname entry:

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

5. Install the latest update packages and reboot the VM.

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

6. Install additional tools:

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

Installing the oss-performance benchmark and its prerequisites

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

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

2. Run the included setup.sh script to install prerequisites:

```
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

n 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. 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 the Plain permalink style, and click Save Changes
    c. Pages→All Pages
    d. Select Page "Privacy Policy"→Quick Edit→
    e. Status: Published
7. Click "Update" button
8. Select Pages→All Pages
9. Select all→Bulk Actions→Move to Trash→Click "Apply"
10. Message: 2 pages moved to the Trash.
11. Click "Trash" Link
12. Click "Empty Trash" button
13. Message: 2 pages permanently deleted.
14. Posts→All Posts
15. Select all→Bulk Actions→Move to Trash→Click "Apply"
16. Message: 1 post moved to the Trash.
17. Click "Trash" Link
18. Click "Empty Trash" button
19. Message: 1 post permanently deleted.
20. Install the Demo Data Creator 1.3.4 plugin.
21. Run the plugin by navigating to Tools→Demo Data Creator
22. 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
23. Click Create users. A message will appear.
24. Adjust the settings for categories:
    - Maximum number of categories (max 25): 10
25. Click Create categories. A message will appear.
26. 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
27. Click Create pages. A message will appear.
28. Now, adjust the settings for Posts:
    - Maximum number of posts (max 100): 50
    - Maximum number of blog post paragraphs (min 1, max 50): 10
29. 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.
30. Now, adjust the settings for Comments:
    - Maximum number of comments per post (max 50): 10
31. Click Create comments. A message will appear.
32. Adjust the settings for Links:
    - Maximum number of links in blogroll (max 100): 25
33. Click Create links. A message will appear.
34. Log out of the admin user account.
35. Back up the database for use by the benchmark:

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

Creating a snapshot of your baseline VM instance boot disk

1. Log into Google Cloud and click Go to console.
2. Click Compute engine, then click Snapshots
3. At the top of the page, click the Create snapshot button.
4. Enter a snapshot name.
5. Optionally, enter a Description of the snapshot.
6. Select the Source disk from the drop-down menu. This is the boot disk for the VM instance created above.
7. Determine your snapshot storage location.
8. Under Location, select whether you want to store your snapshot in a Multi-regional location or a Regional location. We chose Regional.
9. Select which specific region or multi-region that you want to use. To use the region or multi-region that is closest to your source disk, select Based on disk's location (default). We chose us-east1.
10. Add any appropriate labels.
11. Leave everything else default.
12. Click Create to create the snapshot.

Creating your image with the baseline snapshot

1. Log into Google Cloud and click Go to console.
2. Click Compute engine, and click Images.
3. Click the Create image button at the top of the page.
4. Specify the Name of your image.
5. Specify the Source from which you want to create an image. In our case, we used the snapshot created in the previous step.
6. Specify the Location at which to store your image. We chose us-east1.
7. Specify a family, if you wish.
8. Enter a description, if you wish.
9. Add any appropriate labels.
10. Leave the default encryption choice.
11. To create the image, click Create.

Creating the VM instances under test

In this section we list the steps required to create a VM instance from the image we created previously. Follow the steps nine times to create the three n1, three n2, and three c2 VM instances under test. Additionally, create an n1-standard-4 VM instance for the benchmark client (named wordpress-client). For our testing, we used us-east1-b.

Creating the VM instances from the specialized image

1. Log into Google Cloud and click Go to console.
2. Click Compute engine, then click Images.
3. Select the baseline image.
4. Click Create instance at the top of the page.
5. Add the following information:
    a. Name: Name your VM instance.
    b. Labels: Use any appropriate labels.
    c. Region: Select your desired region.
    d. Zone: Select your desired zone.
    e. Machine Configuration:
    f. Machine family: General-purpose
    g. Series: [series type]
    h. Machine type: [machine type]
    i. CPU platform: Automatic for n2/c2, Broadwell or higher for n1
    j. Keep Turn on display device unchecked.
    k. Keep Confidential VM Service and Container unchecked.
    l. Boot Disk: leave default (should be your specialized image)
    m. Identity and API access: Compute Engine default service account.
    n. Firewall: Check Allow HTTP traffic and Allow HTTPs traffic.
    o. Click Create.

Setting up SSH keys in the GCP Mmtadata

To get our client and VM instance under test ready for ssh, we created ssh keys on each and added the public keys to the GCP SSH Key repository for our project.

1. Log into Google Cloud and click Go to console.
2. Click Compute engine, then click VM instances.
3. Click SSH in the dropdown next to your client VM instance.
4. Once logged in, run the following command:

```
ssh-keygen
```

5. Press return following each of the terminal prompts.
6. Once your ssh key has been generated, run the following command to output the key to the terminal:

```
cat ~/.ssh/id_rsa.pub
```

7. Copy the output to your clipboard.
8. Return to the Google cloud console.
9. Click Compute engine, then click Metadata.
10. Click the SSH Keys tab at the top of the page.
11. Click Edit.
12. Click Add item, and paste the contents of your clipboard into the textbox.
13. Click Save.
14. Repeat steps 2-13 for the VM instance under test.
15. Once both keys have been added to the GCP SSH Key repository for the project, reboot both VM instances.

Determining CPU vulnerability mitigation

We ran the following command on each VM instance to determine the Intel processor mitigation settings that Google Cloud employs:

```
lscpu | grep Vulnerability
```

Output for the n1 VM instances

```
Vulnerability Itlb multihit: Not affected
Vulnerability L1tf: Mitigation; PTE Inversion
Vulnerability Mds: Mitigation; Clear CPU buffers; SMT Host state unknown
Vulnerability Meltdown: Mitigation; PTI
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2: Mitigation; Full generic retpoline, IBPB conditional, IBRS_FW, STIBP
    conditional, RSB filling
Vulnerability Srbds: Not affected
Vulnerability Tsx async abort: Mitigation; Clear CPU buffers; SMT Host state unknown
```

Output for the n2 VM instances

```
Vulnerability Itlb multihit: Not affected
Vulnerability L1tf: Not affected
Vulnerability Mds: Mitigation; Clear CPU buffers; SMT Host state unknown
Vulnerability Meltdown: Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2: Mitigation; Enhanced IBRS, IBPB conditional, RSB filling
Vulnerability Srbds: Not affected
Vulnerability Tsx async abort: Mitigation; Clear CPU buffers; SMT Host state unknown
```

Output for the c2 VM instances

```
Vulnerability Itlb multihit: Not affected
Vulnerability L1tf: Not affected
Vulnerability Mds: Mitigation; Clear CPU buffers; SMT Host state unknown
Vulnerability Meltdown: Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2: Mitigation; Enhanced IBRS, IBPB conditional, RSB filling
Vulnerability Srbds: Not affected
Vulnerability Tsx async abort:   Mitigation; Clear CPU buffers; SMT Host state unknown
```

Run the tests

In this section, we list the steps to run the oss-performance benchmark on the VMs under test. The benchmark is started form the client VM using a script that automates the entire process.

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

   ```
   sudo hostnamectl set-hostname wp-n1-standard-4
   ```

2. Update the /etc/hosts file to have the correct loopback hostname on the VM instance under test, as well as the correct hostname and IP for the client VM instance.
3. Repeat step 2 for the client VM instance, adding the IP and hostname of the VM instance under test instead of the client information.
4. From the client VM instance, run the benchmark script substituting the hostname of the VM instance under test:

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

5. Results are automatically saved in the results/ subdirectory of the client VM instance.
6. Repeat all tests 3 times for all VM instance types.

Read the report at http://facts.pt/Eba6JZo ▶

This project was commissioned by Intel.