The science behind the report:

Realize 2.1X the data intensive workload performance with 20% less power with AMD EPYC processor-backed clusters

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 Realize 2.1X the data intensive workload performance with 20% less power with AMD EPYC processor-backed clusters.

We concluded our hands-on testing on March 6, 2024. 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 June 13, 2023 or 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 testing.

		Intel Xeon Scalable processor-based cluster	AMD EPYC processor-based cluster
Test scenario 1 - AMD EPYC™ 9534 processor-based cluster vs. Intel® Xeon® Platinum 8480+ processor-based cluster			
100% read	RPS	84,355,048	122,389,531
	Watts under load	1,002	801
	Performance/watt	84,162	152,745
80-20 read/write	RPS	76,761,057	112,596,334
	Watts under load	1,003	796
	Performance/watt	76,499	141,282
Test scenario 2 - AMD EPYC 9684X processor-based cluster vs. Intel Xeon Platinum 8480+ processor-based cluster			
100% read	RPS	84,355,048	177,763,205
	Watts under load	1,002	1,010
	Performance/watt	84,162	175,894
80-20 read/write	RPS	76,761,057	157,999,207
	Watts under load	1,003	1,016
	Performance/watt	76,499	155,396

		Intel Xeon Scalable processor-based cluster	AMD EPYC processor-based cluster
Test scenario 3 - AMD EPYC 9174F processor-based cluster vs. Intel Xeon Gold 6444Y processor-based cluster			
100% read	RPS	37,980,229	38,807,375
	Watts under load	744	709
	Performance/watt	50,990	54,702
80-20 read/write	RPS	35,450,081	36,722,284
	Watts under load	749	710
	Performance/watt	47,320	51,655

System configuration information

Table 2: Detailed information on the systems we tested.

System configuration information	4x Supermicro Hyper A+ AS-2125HS-TNR	4x Supermicro Hyper SuperServer SYS-221H-TNR
BIOS name and version		
Version	1.4	1.3
Non-default BIOS settings	Default (for AMD EPYC 9534 and 9684X processor-based clusters) Determinism Control: Disabled (for AMD EPYC 9174F processor-based cluster) TSME: Disabled (for AMD EPYC 9174F processor-based cluster) Memory and CPU reserved for all VMs (for AMD EPYC 9174F processor-based cluster)	Default (for Intel Xeon Platinum 8480+ processor-based cluster) TSME: Disabled (for Intel Xeon Gold 6444Y processor-based cluster) Memory and CPU reserved for all VMs (for Intel Xeon Gold 6444Y processor-based cluster)
Operating system name and version/build number	Red Hat CoreOS (RHCOS) 4.13	Red Hat CoreOS (RHCOS) 4.13
Date of last OS updates/patches applied	06/13/2023	06/13/2023
Power management policy	Default (for AMD EPYC 9534 and 9684X processor-based clusters) Maximum IO Performance (for AMD EPYC 9174F processor-based cluster)	Default (for Intel Xeon Platinum 8480+ processor-based cluster) Maximum IO Performance (for Intel Xeon Gold 6444Y processor-based cluster)
Processor 1		
Number of processors per server	2	2
Vendor and model	AMD EPYC 9534	Intel Xeon Platinum 8480+
Core count (per processor)	64	56
Core frequency (GHz)	2.45	2.00
Stepping	B1	E5
Processor 2		
Number of processors	2	2
Vendor and model	AMD EPYC 9684X	Intel Xeon Platinum 8480+
Core count (per processor)	96	56
Core frequency (GHz)	2.55	2.00
Stepping	B1	E5
Processor 3		
Number of processors	2	2
Vendor and model	AMD EPYC 9174F	Intel Xeon Gold 6444Y
Core count (per processor)	16	16
Core frequency (GHz)	4.15	3.60
Stepping	B1	S3

System configuration information	4x Supermicro Hyper A+ AS-2125HS-TNR	4x Supermicro Hyper SuperServer SYS-221H-TNR
Memory module(s)		
Total memory in system (GB)	1,024	1,024
Number of memory modules	16	16
Vendor and model	Micron MTC40F2046S1RC48BA1	Micron MTC40F2046S1RC48BA1
Size (GB)	64	64
Type	PC5-4800B	PC5-4800B
Speed (MHz)	4,800	4,800
Speed running in the server (MHz)	4,800	4,800
Local storage		
Number of drives	2	2
Drive vendor and model	Micron MTFDKBA480TFR	Micron MTFDKCC960TFR
Drive size (GB)	480	960
Drive information	M.2 PCIe Gen4 NVMe SSD	PCIe Gen4 NVMe SSD
Number of drives	2	2
Drive vendor and model	Micron MTFDKCB1T6TFS	Micron MTFDKCB1T6TFS
Drive size (TB)	1.6	1.6
Drive information	PCIe Gen4 NVMe SSD	PCIe Gen4 NVMe SSD
Number of drives	4	4
Drive vendor and model	Micron MTFDKCB7T6TFR	Micron MTFDKCB7T6TFR
Drive size (TB)	7.6	7.6
Drive information	PCIe Gen4 NVMe SSD	PCIe Gen4 NVMe SSD
Network adapter		
Vendor and model	Mellanox MCX613106A-VDAT	Mellanox MCX613106A-VDAT
Number and type of ports	2x 200GbE	2x 200GbE
Cooling fans		
Vendor and model	Delta PFM0812HE-01	San Ace 9HV0812P1H6071
Number of cooling fans	4	4
Power supplies		
Vendor and model	Supermicro PWS-1K63A-1R	Supermicro PWS-1K24A-iR
Number of power supplies	2	2
Wattage of each (W)	1,600	1,200

How we tested

We used four worker nodes, or servers, in each cluster.

The VMs we tested had eight cores, and we ran more or fewer VMs to maximize performance on each configuration. For the first and second comparisons, we ran 16 VMs per host for the AMD processors (64 VMs total on the cluster) and 15 per host for the Intel processors (60 VMs total). In the last comparison, we ran 6 VMs per host, or 24 total, for each cluster.

Prepare infrastructure

DNS

We used a Windows Server DNS for our name resolution. Feel free to use your preferred DNS.

Table 3: The following is an example DNS table for the AMD EPYC processor-based testbeds.

Name	IP address
api.amd-ocp	10.209.60.201
api-int.amd-ocp	10.209.60.201
*.apps.amd-ocp	10.209.60.202
bootstrap.amd-ocp	10.209.60.10
etcd-0.amd-ocp	10.209.60.11
etcd-1.amd-ocp	10.209.60.12
etcd-2.amd-ocp	10.209.60.13
master-0.amd-ocp	10.209.60.11
master-1.amd-ocp	10.209.60.12
master-2.amd-ocp	10.209.60.13
worker-0.amd-ocp	10.209.60.21
worker-1.amd-ocp	10.209.60.22
worker-2.amd-ocp	10.209.60.23
worker-3.amd-ocp	10.209.60.24
worker-4.amd-ocp	10.209.60.25

Table 4: The following is an example DNS table for the Intel Xeon Scalable processor-based testbed.

Name	IP address
api.intel-ocp	10.209.61.201
api-int.intel-ocp	10.209.61.201
*.apps.intel-ocp	10.209.61.202
bootstrap.intel-ocp	10.209.61.10
etcd-0.intel-ocp	10.209.61.11
etcd-1.intel-ocp	10.209.61.12
etcd-2.intel-ocp	10.209.61.13
master-0.intel-ocp	10.209.61.11
master-1.intel-ocp	10.209.61.12
master-2.intel-ocp	10.209.61.13
worker-0.intel-ocp	10.209.61.21
worker-1.intel-ocp	10.209.61.22
worker-2.intel-ocp	10.209.61.23
worker-3.intel-ocp	10.209.61.24
worker-4.intel-ocp	10.209.61.25

Deploying pfsense

We combined our DHCP and gateway servers by deploying pfsense. Once you have those two aspects functioning, you will need to set DHCP reservations for the network connections you are using on your systems under test so that your hosts IP addresses match the DNS entries you created in the previous section.

Prepare systems under test

Configure all systems to boot from CD first, and follow the instructions below.

Using OpenShift Assisted Installer

1. Log into the Red Hat Hybrid Cloud Console.
2. Make sure you're on the OpenShift tab, and click Scale your applications.
3. Select Clusters, and click Next.
4. Click Create cluster.
5. On the top of the page, click the Datacenter tab.
6. Under Assisted Installer, click Create Cluster.
7. In the cluster name field, type either `amd-ocp` or `intel-ocp` as the cluster name (this part is used as part of the FQDN for the hosts).
8. In the base domain field, enter the domain you are using (we used testbed.local).
9. Choose OpenShift Container Platform 4.13.
10. Click Install OpenShift Virtualization, and click Next.
11. In the cluster view, click Add hosts.
12. In the image file selection, choose Minimal image file: Provision with virtual media, add an SSH public key, and click Generate Discovery ISO.
13. Download the discovery ISO.
14. Look into the BMC of the systems you are using for your OpenShift cluster.
15. Open a remote console.
16. Mount the discovery ISO to the system, and boot into the discovery ISO.
17. Wait for the installation to complete.

Post-install configuration

Disks

We used the following steps to format the disks of the systems under test to work best with the OpenShift Local Storage operator.

1. In a Linux system that has installed OpenShift tools and has access to the testbed, type the following to log into one of the workers:

```
oc debug node/worker-0
```

2. Type the following to list the block devices currently mounted in the system:

```
lsblk
```

3. Choose the disk you plan to use for VM drives and run the following command (in our example, the disk we used is /dev/nvme4n1):

```
for n in `oc get nodes -l node-role.kubernetes.io/worker | awk '{print $1}'`; do oc debug node/$n
-- bash -c "(echo g; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo
; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo
; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo
; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo
; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo n; echo ;
echo ; echo +40G; echo n; echo ; echo ; echo +40G; echo w) | fdisk /dev/nvme4n1"; done
```

4. Create `local-storage.yaml` and put the following into it:

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: vmi-nvme-disks
  namespace: openshift-local-storage
spec:
  logLevel: Normal
  managementState: Managed
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - worker0-2
        - worker1-2
        - worker2-2
        - worker3-2
  storageClassDevices:
  - devicePaths:
    - /dev/nvme4n1p1
    - /dev/nvme4n1p2
    - /dev/nvme4n1p3
    - /dev/nvme4n1p4
    - /dev/nvme4n1p5
    - /dev/nvme4n1p6
    - /dev/nvme4n1p7
    - /dev/nvme4n1p8
    - /dev/nvme4n1p9
    - /dev/nvme4n1p10
    - /dev/nvme4n1p11
    - /dev/nvme4n1p12
    - /dev/nvme4n1p13
    - /dev/nvme4n1p14
    - /dev/nvme4n1p15
    - /dev/nvme4n1p16
    - /dev/nvme4n1p17
    - /dev/nvme4n1p18
    - /dev/nvme4n1p19
    - /dev/nvme4n1p20
    - /dev/nvme4n1p21
    - /dev/nvme4n1p22
    - /dev/nvme4n1p23
    - /dev/nvme4n1p24
    - /dev/nvme4n1p25
    - /dev/nvme4n1p26
    - /dev/nvme4n1p27
    - /dev/nvme4n1p28
    - /dev/nvme4n1p29
    - /dev/nvme4n1p30
    - /dev/nvme4n1p31
    - /dev/nvme4n1p32
    - /dev/nvme4n1p33
    - /dev/nvme4n1p34
    - /dev/nvme4n1p35
    - /dev/nvme4n1p36
    - /dev/nvme4n1p37
    - /dev/nvme4n1p38
    - /dev/nvme4n1p39
    - /dev/nvme4n1p40
    - /dev/nvme4n1p41
    - /dev/nvme4n1p42
    - /dev/nvme4n1p43
    - /dev/nvme4n1p44
    - /dev/nvme4n1p45
    - /dev/nvme4n1p46
    storageClassName: vmi-nvme-sc
    volumeMode: Filesystem
```

5. Add the disks to an OpenShift storage class by typing the following:

```
oc apply -f local-storage.yaml
```

VM creation

We used the following settings for our client and Redis server VMs:

- CPUs: 8
- RAM: 16 GB
- Storage: 40 GB

We installed Red Hat Enterprise Linux 8.8 on the VMs using the following steps:

1. Click Workloads → Virtualization.
2. In Virtualization, click Create Virtual Machine.
3. In Select template, choose Red Hat Enterprise Linux 8, and click Next.
4. In boot source, choose Online, and click Customize virtual machine.
5. Name your VM.
6. Under Flavor, choose Custom, 16 GiB Memory, 8 CPU, Workload Type as High-performance, and click Next.
7. Leave networking settings at defaults, and click Next.
8. In Storage, create the following disks:
   - OS disk (modified from rootdisk: 40 GiB)
9. Click Next.
10. In Advanced, click Next.
11. In Review, verify your configuration, and click Create Virtual Machine.
12. Click into the VM.

Memtier client

Clone the base VM, and use the following steps to create a Memtier client:

```
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
sudo dnf install autoconf automake make gcc-c++ pcre-devel zlib-devel libevent-devel openssl-devel -y
sudo dnf install git -y
git clone https://github.com/RedisLabs/memtier_benchmark.git
cd memtier_benchmark/
autoreconf -ivf
./configure
sudo make
sudo make install
```

Redis server

Use the following steps to create a Redis server:

```
sudo subscription-manager register --auto-attach
sudo vi /etc/security/limits.conf
sudo dnf update -y
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
sudo dnf groupinstall "Development Tools" -y
sudo dnf install autoconf automake make gcc-c++ pcre-devel zlib-devel libevent-devel
openssl-devel wget -y
wget https://download.redis.io/redis-stable.tar.gz
ls
tar -xzvf redis-stable.tar.gz
cd redis-stable
sudo make
sudo make install
sudo vi /etc/sysctl.conf
sudo mkdir -p /var/lib/redis
sudo mkdir -p /etc/redis/cluster
sudo mkdir -p /var/log/redis
sudo useradd -m redis -p Password1
sudo usermod -a -G redis redis
sudo chown -R redis:redis /var/lib/redis
sudo chown -R redis:redis /etc/redis/cluster
sudo chown -R redis:redis /var/log/redis
sudo su - redis
sudo nmtui
sudo systemctl restart NetworkManager
ip addr
history
sudo vi /etc/security/limits.conf
sudo vi /etc/sysctl.conf
sudo nmtui
sudo su - redis
cat /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
exit
ip addr
cat /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
ip addr
sudo vi /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
sudo systemctl restart NetworkManager
for n in `oc get vmi | grep redis | awk '{print $1}'`; do virtctl ssh cloud-user@$n --command='sudo
mkdir /home/redis/.ssh/; sudo cp /home/cloud-user/.ssh/authorized_keys /home/redis/.ssh/; sudo chown
redis:redis /home/redis/.ssh/authorized_keys; sudo systemctl restart sshd'; done
for i in {7001..7007}
do
 mkdir -p /var/lib/redis/$i
 mkdir -p /etc/redis/cluster/$i
 cat <<EOF >/etc/redis/cluster/$i/redis_$i.conf
#!/bin/bash
port $i
dir /var/lib/redis/$i/
appendonly no
protected-mode no
cluster-enabled yes
cluster-node-timeout 5000
cluster-config-file /etc/redis/cluster/$i/nodes_$i.conf
pidfile /var/run/redis/redis_$i.pid
logfile /var/log/redis/redis_$i.log
loglevel notice
bind 0.0.0.0
EOF
done
for n in `oc get vmi | grep redis | awk '{print $1}'`; do virtctl ssh redis@$n --command='./initial_
config.sh'; done
for n in `oc get vmi | grep redis | awk '{print $1}'`; do virtctl ssh redis@$n --command='for i in
{7001..7007}; do redis-server /etc/redis/cluster/$i/redis_$i.conf --daemonize yes; done'; done
for n in `oc get vmi | grep redis | awk '{print $1}'`; do virtctl ssh redis@$n --command='redis-cli -p
7001 -c flushdb'; done
for ip in {1..64}; do virtctl ssh cloud-user@redis-$ip --command='export ip='"'$ip'"'; redis-
cli --cluster create 172.16.61.$ip:7001 172.16.61.$ip:7002 172.16.61.$ip:7003 172.16.61.$ip:7004
172.16.61.$ip:7005 172.16.61.$ip:7006 172.16.61.$ip:7007'; done
```

Running the test

Use the following commands to run the tests:

1. Run the following command to start Redis on all of the Redis VMs:

```
for n in `oc get vmi | grep redis | awk '{print $1}'`; do virtctl ssh redis@$n --command='for i in
{7001..7007}; do redis-server /etc/redis/cluster/$i/redis_$i.conf --daemonize yes; done'; done
```

2. Run the following command to kick off a Memtier_benchmark test on all client VMs targeting their respective host (note that the individual flags depend on the test):

```
ip=0; for n in `oc get vmi | grep hammerdb- | awk '{print $1}'`; do ip=$((ip+1)); virtctl ssh cloud-
user@$n --command='export ip='"'$ip'"'; nohup memtier_benchmark -s 172.16.21.$ip -p 7001 -t 6 -c 1
--key-maximum=60000000 -d 100 --randomize --test-time=900 --pipeline=35 --ratio=0:1 --cluster-mode
&>memtier-result.txt &'; done
```

Special changes for the AMD EPYC 9174F processor-based cluster and Intel Xeon Gold 6444Y processor-based comparison

BIOS changes

We changed the following BIOS settings to increase performance on the AMD and Intel processor-based systems:

Intel processor-based system:

- Power profile changed to high performance

AMD processor-based system:

- Power profile changed to high performance
- Transparent Secure Memory Encryption (TSME) disabled
- Determinism disabled

Guaranteed performance tagging

To guarantee all resources for the VMs under test, we configured the VMs running during the AMD EPYC 9174F processor-based cluster and Intel Xeon Gold 6444Y processor-based comparison with the following settings:

1. In the Virtualization view of the OpenShift console, click VirtualMachines.
2. Select one of the VMs.
3. Click Configuration.
4. Search for Scheduling, and click the edit icon beside Dedicated Resources.
5. Select Schedule this workload with dedicated resources (guaranteed policy), and click Save.
6. Complete steps 2 through 5 for all VMs used in the test.

Read the report at https://facts.pt/u5sWIvB ▶

This project was commissioned by AMD.