

Behind the Report:

Generation-over-Generation and Confidential Compute Comparison of Microsoft Azure® VM Instances with AMD Processors

This methodology shows the process Prowess Consulting’s engineers followed when comparing DCas_v6 instances types with DCas_v5 and Das_v6 Microsoft Azure instances powered by AMD processors.

Summary

This methodology outlines the approach Prowess Consulting used to evaluate generation-over-generation performance and security enhancements in Microsoft Azure® confidential computing instances powered by AMD EPYC™ processors. Our testing compared 4th Generation AMD EPYC processor–based confidential DCas_v6 instances with prior-generation DCas_v5 instances and 4th Gen AMD EPYC processor–based general-purpose Das_v6 instances. The testing focused on workloads including SPEC CPU® 2017, AMD STREAM, and Redis® benchmarks.

Testing Summary

Our engineers compared the instances across tests from each of the four SPEC CPU 2017 suites, the AMD STREAM memory benchmark, and Redis and the redis-benchmark utility. We ran our tests on instance sizes 16 and 96. Table 1 shows the instance sizes, tests, and operating systems we used in our testing.

Table 1. Instance sizes, tests, and operating systems

Instance Type	Operating System	Test
D16as_v6, DC16as_v6, DC16as_v5, D96as_v6, DC96as_v6, and DC96as_v5	Windows Server® 2025	SPEC CPU®
D16as_v6, DC16as_v6, and DC16as_v5	Ubuntu® Linux® 24.04	SPEC CPU
D96as_v6, DC96as_v6, and DC96as_v5	Windows Server 2025	AMD STREAM
D96as_v6, DC96as_v6, and DC96as_v5	Ubuntu Linux 24.04	redis-benchmark

In all cases, we used the operating system (OS) default volume for the OS. For tests requiring additional volumes, we used a premium solid-state drive (SSD) 2 with 80,000 input/output operations per second (IOPS) and 1,200 Mb/s throughput. In the case of all confidential compute instances, we selected the “Confidential OS disk encryption” option. All instances were set up in the same region and zone (East US zone 3).

Hardware Configuration

This section outlines the setup we used for the Windows Server® 2025 and Ubuntu® 24.04 instances. We configured all the instances via the Azure portal. Except for any additional hard disk drives needed for a specific test (see Table 2), the hardware setup steps for each test on a given OS were identical.

Table 2 shows the CPU, memory, and OS information for each of the instance types used in our testing.

Table 2. CPU, memory, and OS information for instance types used

Instance Type	DC16as_v6	D16as_v6	DC16as_v5	DC96as_v6	D96as_v6	DC96as_v5
CPU Model	80-core AMD EPYC™ 9V74 processor	80-core AMD EPYC 9V74 processor	AMD64 family 25 model 1 stepping 1	80-core AMD EPYC 9V74 processor	80-core AMD EPYC 9V74 processor	AMD64 family 25 model 1 stepping 1
CPU Count	1	1	1	1	1	1
Cores/Threads	8/16	8/16	8/16	48/96	48/96	48/96
Memory	64 GB	64 GB	64 GB	384 GB	384 GB	384 GB
OS Version	Windows Server® 2025 Datacenter: Azure Edition 24H2	Windows Server 2025 Datacenter: Azure Edition 24H2	Windows Server 2025 Datacenter: Azure Edition 24H2	Windows Server 2025 Datacenter: Azure Edition 24H2/Ubuntu® Linux® 24.04	Windows Server 2025 Datacenter: Azure Edition 24H2/Ubuntu Linux 24.04	Windows Server 2025 Datacenter: Azure Edition 24H2/Ubuntu Linux 24.04

Table 3 outlines the additional volumes that we used with specific test configurations. We provisioned all premium SSD2 volumes with the max 80,000 IOPS and 1,200 MB/s throughput.

Table 3. Additional volumes used with specific configurations

Test Type	Additional Disks Needed	Description
redis-benchmark	0	Not applicable (N/A)
AMD STREAM	0	N/A
SPEC CPU® 2017	1	1 TB premium SSD2, used for SPEC® files

Windows Server® Instance Setup

These steps apply to the setup of any test system running Windows Server 2025.

1. Log into the Azure portal.
2. From the **Azure services** section of the page, select **Virtual machines**.
3. Click the **Create** dropdown, and then select **Virtual machine**.
4. Ensure the **Subscription** box properly reflects your Azure subscription name.
5. Specify a **Resource Group** from the dropdown.
6. Add a user-chosen name for the instance in the **Virtual machine name** field.
7. From the **Region** dropdown, select **EAST US**.
8. From the **Availability Zone** selector, select **Zone 3**.
9. Select the appropriate **Security Type**:
 - a. For the confidential compute instance types, select **Confidential virtual machine**.
 - b. For the general-purpose instance types, select **Trusted launch virtual machines**.
10. To set the OS image, select **See all images** from the **Image Size** dropdown.
 - a. Select **Windows Server**.
 - b. Select **Windows Server 2025 Datacenter: Azure Edition**.
 - c. Click **Create**.

11. In the **Username** field, specify a username.
12. Enter and confirm the chosen password.
13. Leave default selection of **Allow RDP port**.
14. Click **Next : Disks**.
15. Leave the **Image default (127 GB)** selection for the OS disk:
 - a. For confidential compute instances, ensure **Confidential OS disk encryption** is selected.
16. Referencing Table 3, if additional disks are needed, add them as follows:
 - a. In the **Data Disks** section, click **Create and attach a new disk**.
 - b. Leave default selections for:
 - i. **Name**: Will default to a value based on the instance name.
 - ii. **Source Type: None (empty Disk)**.
 - c. From the **Size** section, select **Change size**.
 - i. From the **Storage Type** dropdown, select **Premium SSD v2**.
 - ii. Select the disk size as appropriate.
 - iii. Set **Disk IOPS** to **80000**.
 - iv. Set **Disk throughput** to **1200**.
 - v. Click **OK**.
 - d. Select **Delete disk with VM**.
 - e. Click **OK**.
17. Click **Review + Create**.
18. Click **Create**.
19. Once the instance has been created, click **Go to Resource** and make a note of the IP address.
20. Open a remote desktop connection to the instances IP.
21. Click the Windows icon to open the quick access menu.
22. Right-click the PowerShell® icon, and then select **Run as administrator**.
23. Run the following commands to install Windows® updates:

```
Install-Module PSWindowsUpdate -Force
Import-Module PSWindowsUpdate
Get-WUList
Install-WindowsUpdate -AcceptAll
```
24. If additional disks have been added for this instance:
 - a. Click the **Windows** icon, and then, in the search bar, enter **Disk management**.
 - b. Select **Create and manage hard disk partitions**.
 - c. For each unallocated disk, right-click the unallocated sections, and then select **New simple volume**.
 - i. Leave the default size selection, and then click **Next**.
 - ii. Note the assigned disk letter, and then click **Next**.
 - iii. Leave the default file system selections, and then click **Next**.
 - iv. To complete the process, click **Finish**.
25. Reboot the server to apply the pending changes, and then reconnect the remote desktop connection.

Ubuntu® Linux® Instance Setup

These steps apply to the setup of any test system that will be running Ubuntu 24.04.

1. Log in to the Azure portal.
2. From the **Azure services** section of the page, select **Virtual machines**.
3. Click the **Create** dropdown, and then select **Virtual machine**.
4. Ensure the **Subscription box** properly reflects your Azure subscription name.
5. Specify a **Resource Group** from the dropdown.
6. Add a user-chosen name for the instance in the **Virtual machine name** field.
7. From the **Region** dropdown, select **EAST US**.
8. From the **Availability zone** selector, select **Zone 3**.
9. Select the appropriate security type:
 - a. **Confidential virtual machine** for the confidential compute instance types.
 - b. **Trusted launch virtual machines** for the general-purpose instance types.

10. From the **Image Size** dropdown, select **See all images**:
 - a. Find **Ubuntu Server 24.04**, and then click the **Select** dropdown.
 - b. Select either **Ubuntu Server 24.04 LTS -x64 Gen2** or **Ubuntu Server 24.04 LTS (Confidential Compute) -x64 Gen 2**, as appropriate for the given instance type.
11. From the **Authentication Type** radial button choices, select **Password**.
12. In the **Username** field, specify a username.
13. Enter and confirm the password for the instance.
14. Leave the default selection for inbound ports, allowing access on port 22.
15. Click **Next : Disks**.
16. Leave the **Image default (30 GB)** selection for the OS disk.
 - a. For confidential compute instances, ensure the confidential OS disk encryption box is selected.
17. If this system under test (SUT) will be used for SPEC® testing:
 - a. In the **Data Disks** section, click **Create and attach a new disk**.
 - b. Leave default selections for:
 - i. **Name**: Will default to a value based on the instance name.
 - ii. **Source Type: None (empty Disk)**
 - c. From the **Size** section, select **Change** size:
 - i. From the **Storage Type** dropdown, select **Premium SSD v2**.
 - ii. Select the disk size as appropriate.
 - iii. Set **Disk IOPS** to **80000**.
 - iv. Set **Disk throughput** to **1200**.
 - v. Click **OK**.
 - d. Select **Delete disk with VM**.
 - e. Click **OK**.
18. Click **Review + Create**.
19. Click **Create**.
20. Once the instance has been created, click **Go to Resource**, and then make note of the IP address.
21. Open a Secure Shell (SSH) connection to that IP address using the previously set username and password.
22. To update the system, run the following commands:

```
sudo apt update
sudo apt upgrade -y
```
23. Run **sudo reboot** to reboot the instance and apply updates.

Benchmark-Specific Steps

This section includes the steps to set up and run each of the benchmarks. Each test scenario assumes that the required systems have been set up in line with the [Hardware Configuration](#) section of this document.

AMD STREAM

This section includes the steps to set up the [AMD STREAM](#) memory benchmark on a Windows Server 2025 system, in addition to how to start a benchmark run.

STREAM Setup

This section includes the steps to install and configure the STREAM benchmark. We performed this testing on Windows Server 2025 systems on DC96as_v6, DC96as_v5, and D96as_v6 Azure VM instance types. No additional remote volumes were needed for this test.

1. From a Remote Desktop Protocol (RDP) session to the SUT, install the following prerequisites:
 - a. To install Microsoft® Visual Studio® with C/C++ development tools:
 - i. Download Visual Studio (Community) from <https://visualstudio.microsoft.com/downloads/>.
 - ii. Open the downloaded **VisualStudioSetup.exe** file.
 - iii. Accept the User Access Control (UAC) prompt, and then click **Continue**.
 - iv. Select **Desktop development with C++**, and then click **Install**.
 - v. Wait for the Visual Studio installer to complete all downloads and installs.
 - vi. To continue without signing into Visual Studio, click **Skip this for now**.
 - vii. Click **Start Visual Studio**.

- b. To install Git:
 - i. Download the **Git-2.50.1-64-bit** version of Git for Windows from <https://gitforwindows.org/>.
 - ii. Open the resulting **Git-2.50.1-64-bit.exe** download.
 - iii. Accept the UAC prompt, and then click **Next**.
 - iv. Accept all the default values and click **Next** until reaching the **Configuring experimental options** window.
 - v. Click **Install**.
 - vi. Clear the **View Release Notes** checkbox, and then click **Finish**.
2. To clone the STREAM repo, open a PowerShell terminal, and then run **git clone https://github.com/jeffhammond/STREAM.git**.
3. Open Visual Studio.
4. Navigate to the STREAM directory, and then open **stream.c**.
5. Make the following edits to the **stream.c** file:
 - a. Add a **typedef** for **ssize_t** to support Windows:

```
#ifdef _WIN32
#include <BaseTsd.h>
typedef SSIZE_T ssize_t;
#endif
```

- b. Comment out the inclusion of **<unistd.h>**, as it is not available on Windows:

```
//# include <unistd.h>
```

- c. Replace UNIX® timing headers with timing support native to Windows:

- i. Add:

```
#include <windows.h>
```

- ii. Comment out:

```
//# include <sys/time.h>
```

- d. Replace the original **mysecond()** function with a version compatible with Windows using **QueryPerformanceCounter**:

- i. Add:

```
double mysecond()
{
    static LARGE_INTEGER frequency;
    static BOOL initialized = FALSE;
    LARGE_INTEGER counter;
    if (!initialized) {
        QueryPerformanceFrequency(&frequency);
        initialized = TRUE;
    }
    QueryPerformanceCounter(&counter);
    return (double)counter.QuadPart / (double)frequency.QuadPart;
}
```

- ii. Comment out the original **mysecond()** implementation that used **gettimeofday()**:

```
//double mysecond()
//{
//    struct timeval tp;
//    struct timezone tzp;
//    int i;
//    i = gettimeofday(&tp, &tzp);
//    return ((double) tp.tv_sec + (double) tp.tv_usec * 1.e-6);
//}
```

- e. To avoid compiler warning c4477, update the **printf** statement for array-size reporting:

- i. Add:

```
printf("Array size = %zu (elements), Offset = %d (elements)\n", (size_t)STREAM_ARRAY_SIZE, OFFSET);
```

- ii. Comment out the original line:

```
//printf("Array size = %llu (elements), Offset = %d (elements)\n", (unsigned long long) STREAM_ARRAY_SIZE, OFFSET);
```

- f. Update the **printf** statement for **sizeof(STREAM_TYPE)** to use the correct format specifier:

```
printf("WEIRD: sizeof(STREAM_TYPE) = %zu\n", sizeof(STREAM_TYPE));
```

6. Save the changes to the stream.c file.
7. Create a new text file, **stream-runner.ps1**, and enter the following contents:

```
$threadCounts = @(2, 32, 64, 96)
$command = ".\stream.exe"
$timestamp = Get-Date -Format "yyyyMMdd_HH:mm:ss"
$logFile = "stream_log_$timestamp.txt"
"STREAM Thread Scaling Log - $(Get-Date)" | Out-File $logFile
foreach ($threads in $threadCounts) {
    Write-Host "`n=== Running with OMP_NUM_THREADS=$threads ==="
    $env:OMP_NUM_THREADS = $threads
    Add-Content $logFile "`n--- OMP_NUM_THREADS=$threads ---"
    & $command | Add-Content $logFile
}
```

Running STREAM

This section includes the steps to run the STREAM benchmark. For any given array size, the stream-runner.ps1 script will run tests with 2, 32, 64, and 96 thread counts. For our testing, we used array sizes 2,000,000, 11,000,000, 20,000,000, and 80,000,000.

1. Open Visual Studio.
2. Navigate to the STREAM directory.
3. Open the Developer command prompt.
4. To compile stream.c for a given array size, replace **\$ArraySize** with the array size value, and then run the following command:

```
cl /O2 /arch:AVX2 /favor:AMD64 /openmp /DSTREAM_ARRAY_SIZE=$ArraySize stream.c
```
5. Open a PowerShell terminal and navigate to the directory containing stream-runner.ps1.
6. To start a test iteration, run **./stream-runner.ps1**.
7. After completion, results can be found in **./stream_log_\$timestamp.txt**.
8. Repeat from step 4 for each array size value to be tested.

Redis-benchmark

This section covers the steps to install and configure the Redis testing environments. Tests consisted of one or more pairs of redis-server and redis-benchmark processes.

Redis® Setup

This section includes the steps to install and configure Redis and the redis-benchmark tool. We ran these tests on Ubuntu Linux 24.04 systems with hardware configurations of DC96as_v6, DC96as_v5, and D96as_v6 Azure VM instance types. No additional volumes are needed for this test.

1. Create an SSH connection into the targeted device.
2. Update the environment with the following commands:

```
sudo apt-get update
sudo apt-get upgrade -y
```

3. Install Redis Open Source on Ubuntu using APT with the following commands:

```
sudo apt-get install lsb-release curl gpg
curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis -archive-
keyring.gpg
sudo chmod 644 /usr/share /keyrings/redis-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg]
https://packages.redis.io/deb $(lsb_release -cs) main" | sudo tee
sudo apt-get update
sudo apt-get install redis
```

4. Disable transparent huge pages (THP) with the following command:

```
sudo vi /etc/systemd/system/disable-thp.service
```

5. Add the following content to the **disable-thp.service** file:

```
[Unit]
Description=Disable Transparent Huge Pages
[Service]
Type=oneshot
ExecStart=/bin/sh -c 'echo never > /sys/kernel/mm/transparent_hugepage/enabled'
ExecStart=/bin/sh -c 'echo never > /sys/kernel/mm/transparent_hugepage/defrag'
[Install]
WantedBy=multi-user.target
```

6. Save the file.

7. Execute the following commands:

```
sudo systemctl daemon-reload
sudo systemctl enable disable-thp.service
sudo systemctl start disable-thp.service
```

8. Configure **/etc/sysctl.conf** with the following optimizations:

- Set **vm.overcommit_memory = 1**
- Set **net.core.somaxconn = 65535**

9. Configure **/etc/redis/redis.conf** with the following optimizations:

- Uncomment **save ""** to disable backgrounds saves for consistent performance.

```
maxmemory-policy allkeys-lru
tcp-backlog 65535
timeout 0
tcp-keepalive 60
maxclients 65000
```

10. Configure remaining Redis configuration files with the following script:

```
for i in {1..48}; do PORT=$((6379 + i))
mkdir -p /var/lib/redis-$i
mkdir -p /var/log/redis
cp /etc/redis/redis.conf /etc/redis/redis-$i.conf
sed -i "s/^port 6379/port $PORT/" /etc/redis/redis-$i.conf
sed -i "s|^dir /var/lib/redis|dir /var/lib/redis-$i|" /etc/redis/redis-$i.conf
sed -i "s|^logfile /var/log/redis/redis-server.log|logfile /var/log/redis/redis-server-$i.log|" /etc/redis/redis-$i.conf
done
```

11. Configure the following system limits in **/etc/security/limits.conf**:

```
* soft nofile 65535
* hard nofile 65535
```

12. Copy the **runner.sh** script to the instance.

- Be sure to initialize the script with **chmod +x**.
- See the [Appendix](#) for full script contents.

Running Redis

This section includes the steps to run the Redis benchmark. We ran benchmark tests both with single processes leveraging pipelining, and with 48 concurrent server and redis-benchmark processes.

- Create an SSH connection into the targeted instance.
- Prior to starting a test run, execute the following command:

```
redis-cli flushall
```

- Start the performance monitoring tools.
- Run tests with 50 parallel clients and 1 million total requests with payload sizes of 64, 512, and 1,024.
- Run the following command to initialize the runner.sh script:

```
./runner.sh ./<RUN#>
```

- Record the responses per second (RPS) and average latency.
- Final scores will be a median value of at least three runs.

SPEC CPU® 2017

This section outlines the steps taken to set up and run the SPEC CPU 2017 suite of tests. We conducted this testing on both Windows Server 2025 and Ubuntu Linux 24.04 systems.

SPEC CPU 2017 on Windows Server

This section includes the steps to install and configure SPEC CPU 2017 benchmark suites on Windows via Windows Subsystem for Linux (WSL). This test requires one additional volume. This document assumes that volume has been mounted to drive letter E:. We ran these tests on hardware configurations of DC16as_v6, DC16as_v5, D16as_v6, DC96as_v6, DC96as_v5, and D96as_v6. We tested Ubuntu configurations on hardware configurations DC96as_v6, DC96as_v5, and D96as_v6.

1. From within an RDP connection to the instance, open a PowerShell terminal as an administrator.
2. To install the WSL subsystem requirements, run **Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux**.
3. To apply the changes, reboot the system and then reconnect the RDP session.
4. From within an RDP connection to the instance, open a PowerShell terminal as an administrator.
5. To install Ubuntu 24.04 via WSL 1, run the following commands:

```
wsl --set-default-version 1
#press any key when prompted
wsl --install -d Ubuntu-24.04
```

6. Open a file explorer window and navigate to the location of the SPEC CPU 2017 ISO image.
7. Right-click the ISO image, and then select **Mount**. If prompted to confirm, select **Yes**.
8. From the mounted ISO image folder, copy all contents.
9. Create a new folder, **E:\specISO**.
10. Paste the copied contents into **E:\specISO**.
11. From the open PowerShell terminal, run **wsl** to enter the Ubuntu WSL terminal.
12. To switch to the root user and install system requirements, run the following:

```
sudo su -
apt-mark hold system
apt update
apt install -y build-essential gcc g++ gfortran make cmake libomp-dev ibnetcdf-dev libnetcdf-dev netcdf-
bin libgfortran-11-dev libgfortran5 libblas-dev liblapack-dev libhdf5-dev clang libfftw3-dev libgsl-dev
```

13. To set system environment variables, replace **\$VCPU_COUNT** with the core count of the instance being tested, and then run the following:

```
cat >> ~/.bashrc << 'EOF'
export CC=/usr/bin/gcc
export CXX=/usr/bin/g++
export FC=/usr/bin/gfortran
export NETCDF_ROOT=/usr
export OMP_NUM_THREADS=$VCPU_COUNT
export OMP_STACKSIZE=128M
export OMP_SCHEDULE=static
export OMP_PROC_BIND=true
export MAKEFLAGS="-j$VCPU_COUNT"
EOF
```

14. To bring in the environment values, run the following:

```
source ~/.bashrc
```

15. To change to the directory of the additional volume, run the following:

```
cd /mnt/e/specIso
```

16. To create a SPEC installation directory, run the following:

```
mkdir ../specInstall
```

17. To start the installation process, run the following:

```
./install.sh -d /mnt/e/specInstall/.
```


18. Wait while the installation attempts complete. To work around the failing Perl® test, run the following:

```
cd /mnt/e/specInstall/tools/bin/linux-x86_64/  
tar -xf tools-linux-x86_64.tar.xz  
cd ../../..  
cp -r ./tools/bin/linux-x86_64/bin/* ./bin/  
SPEC=/mnt/e/specInstall ./bin/packagetools linux-x86_64
```

19. To enter the SPEC installation directory, run the following:

```
cd /mnt/e/specInstall
```

20. To open the configuration file, run the following:

```
vim ./config/config.cfg
```

21. To set paste mode in vim, enter **:set paste**.

22. Press **A** to enter editing mode.

23. Referencing the content in the SPEC CPU 2017 configuration file's appendix, paste the contents of the appendix into this file. Note the updates required for **\$VCPU_COUNT** and **Label**.

24. To exit vim, enter **:wq**.

25. To bring in SPEC CPU set environment variables, run the following:

```
source ./shrc
```

26. To build the SPEC tests, run the following:

```
runcpu --action=build --config=config --tune=base,peak all -I
```

27. To start a test run for a given test suite, run the following:

- a. For integer rate tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak intrate -I
```

- b. For integer speed tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak intspeed -I
```

- c. For floating point rate tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak fprate -I
```

- d. For floating point speed tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak fpspeed -I
```

28. Results can be found in the results folder *.txt files. Final scores are a median of three runs.

SPEC CPU 2017 Setup for Ubuntu

This section includes the steps to install and configure SPEC CPU 2017 on Ubuntu 24.04.

1. Connect to the SUT via SSH.
2. Run the following commands to format and mount the additional volume:

```
sudo fdisk -l # and note the disk ID  
sudo mkfs.ext4 /dev/DISK_ID  
mkdir /spec  
sudo mount /dev/DISK_ID /spec
```

3. Reconnect to the SUT via SSH.

4. To install apt-based requirements, run the following:

```
apt install -y build-essential make cmake libc6-dev linux-libc-dev zlib1g-dev libxml2-dev libxslt1-dev  
libbz2-dev liblzma-dev libssl-dev libffi-dev unzip imagemagick libmagickcore-dev libmagickwand-dev  
gfortran gcc g++ libomp-dev libnetcdf-dev libnetcdf-dev netcdf-bin libgfortran-11-dev libgfortran5  
libblas-dev liblapack-dev libhdf5-dev libfftw3-dev libgsl-dev
```

5. From the local workstation web browser, obtain the AOCC 5.0 compiler from [AOCC 5.0 EULA](#).

6. From the local workstation, use PowerShell to copy the compiler file to the instance with SCP:

```
aocc-compiler-5.0.0.tar username@$Instance_IP_Address:~/
```

7. From the SSH connection to the SUT, to extract the tar file, run the following:

```
tar -xvf aocc-compiler-5.0.0.tar
```

8. To install the compiler, run the following:

```
sudo aocc-compiler-5.0.0/install.sh
```

9. To activate compiler environment variables, run the following:

```
source /tmp/setenv_AOCC.sh
```

10. To remove the stack size limit, run the following:

```
ulimit -s unlimited
```

11. To set the concurrence for make operations, run the following:

```
export MAKEFLAGS="-j96"
```

12. From the local workstation, to upload the SPEC CPU 2017 ISO file to the instance via SCP, run the following:

```
scp ./spec2017.iso username@$Instance_IP_address:/spec/
```

13. From the SHT connection to the instance, to mount the ISO, run the following:

```
mkdir /mnt/specIso  
mount -o loop /spec/cpu2017-1.1.9.iso /mnt/speciso
```

14. To install SPEC, run the following:

```
/mnt/speciso/install.sh -d /spec/specInstall
```

15. To change to the install directory, run the following:

```
cd /spec/specInstall
```

16. To open the configuration file, run the following:

```
vim ./config/config.cfg
```

17. To set the paste mode in vim, press :, and then type the following:

```
set paste
```

18. Press **Enter** to set the mode.

19. Press **A** to enter editing mode.

20. Referencing the content in the SPEC CPU 2017 configuration file's appendix, paste the contents of the appendix into the edited config.cfg file. Note the updates required for **\$VCPU_COUNT** and **Label**, as mentioned in the **Appendix** of this guide.

21. To exit vim, run the following:

```
:wq
```

22. To activate the required environment variables, run the following:

```
source ./shrc .
```

23. To build the test executables, run the following:

```
runcpu --action=build --config=config --tune=base,peak all -I
```

24. To start a test run for a given test suite, run the following:

a. For integer rate tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak intrate -I
```

b. For integer speed tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak intspeed -I
```

c. For floating point rate tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak fprate -I
```

d. For floating point speed tests:

```
runcpu --config=config --action=run --size=ref --tune=base,peak fpspeed -I
```

25. Results can be found in the results folder *.txt files. Final scores are a median of three runs.

Appendix

This section contains additional content as referenced in the main document.

Redis Runner Script Contents

This runner.sh script is used as a comprehensive Redis performance benchmarking tool. It is configured to run with 64, 512, and 1,024 payload sizes with scaling parallel pipeline levels for testing both single-instance and multiple-instance setups.

```
resultsDir=$1
payload_sizes=(64 512 1024)
clients=50
requests=1000000
sleep 2 # Give Redis time to start
# Run benchmarks
echo "Running benchmarks..."
for p in 1 16 24 36 48 96; do
for size in "${payload_sizes[@]}; do
    echo "Running benchmark with payload size: ${size} bytes"
    redis-benchmark -c $clients -n $requests -d $size -P $p --csv | tee -a $resultsDir/Redis_C-
${clients}_N-${requests}_D-${size}_P-${p}.csv
    echo "-----"
sleep 2
done
done
sleep 60
# Start Redis instances
echo "Starting Redis instances..."
for i in {1..48}; do
    PORT=$((6379 + i))
    DIR="/tmp/redis${PORT}"
    mkdir -p "$DIR"
    # redis-server --port "$PORT" --dir "$DIR" --daemonize yes
    redis-server /etc/redis/redis-${i}.conf &
done
sleep 5
benchmark_pids=()
for size in "${payload_sizes[@]}; do
    benchmark_pids=()
    for i in {1..48}; do
        PORT=$((6379 + i))
        echo "Starting Redis server $i"
        redis-benchmark -h localhost -c "$clients" -n "$requests" -d "$size" -p "$PORT" --csv \
| tee -a "$resultsDir/Redis_C-${clients}_N-${requests}_D-${size}_I-${i}.csv" &
        benchmark_pids+=($!)
    done
    for pid in "${benchmark_pids[@]}; do
        wait "$pid"
    done
done
echo "Benchmarking complete."
```

SPEC CPU 2017 Configuration File

This configuration file is used for the config/config.cfg file with all SPEC CPU tests. Of note, the copies, threads, and label values should be adjusted to reflect the system being tested (**\$VCPU_COUNT**). Copies and threads should be equal to the vCPU count of the instance, and **\$INSTANCE_NAME** should be replaced by a meaningful descriptor.

```
%define label $INSTANCE_NAME
%ifndef %{copies}
%   define copies $VCPU_COUNT
%endif
intspeed,fpspeed:
    threads          = $VCPU_COUNT
iterations          = 1
label               = %{label}
mean_anyway         = 1
output_format       = txt
reportable          = 0
tune                = base,peak

intrate,fprate:
    copies           = %{copies}
default:
    CC               = clang
    CLD              = clang
    CXX              = clang++
    CXXLD            = clang++
    FC               = gfortran
    F77              = gfortran
    F90              = gfortran
    CC_VERSION_OPTION = --version
    CXX_VERSION_OPTION = --version
    FC_VERSION_OPTION = --version
#-----
# CONSERVATIVE BASE
#-----
default=base:
    OPTIMIZE          = -O1
    COPTIMIZE          = -std=gnu99 -Wno-implicit-function-declaration -fno-strict-aliasing
    CXXOPTIMIZE        = -std=c++11 -Wno-deprecated -fno-strict-aliasing
    FOPTIMIZE          = -O1 -fallow-argument-mismatch

    LDOPTIMIZE         = -O1
    EXTRA_LDFLAGS      = -Wl,--allow-multiple-definition
EXTRA_LIBS           = -lm
#-----
# AGGRESSIVE PEAK
#-----
default=peak:
    OPTIMIZE          = -O3
    COPTIMIZE          = -std=gnu99 -march=native -fno-strict-aliasing
    CXXOPTIMIZE        = -std=c++14 -march=native -fno-strict-aliasing
    FOPTIMIZE          = -O3 -march=native -funroll-loops -fallow-argument-mismatch
    LDOPTIMIZE         = -O3
    EXTRA_LDFLAGS      = -Wl,--allow-multiple-definition
fpspeed=peak:
    EXTRA_CFLAGS       = -fopenmp -DSPEC_OPENMP -flto=thin
    EXTRA_CXXFLAGS     = -fopenmp -DSPEC_OPENMP -flto=thin
```

```
EXTRA_FFLAGS      = -fopenmp -DSPEC_OPENMP
EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -fopenmp -flto=thin
EXTRA_LIBS        = -lm -lomp
#-----
# Base Portability Settings
#-----
default:
PORTABILITY       = -DSPEC_LP64
500.perlbench_r,600.perlbench_s:
CPORTABILITY      = -DSPEC_LINUX -DSPEC_LINUX_X64
502.gcc_r,602.gcc_s:
CPORTABILITY      = -DHOST_WIDE_INT="long long" -DSPEC_LINUX
COPTIMIZE         = -std=gnu99 -Wno-implicit-function-declaration -Wno-int-conversion
505.mcf_r,605.mcf_s:
CPORTABILITY      = -DSPEC_LINUX
508.namd_r:
CXXPORTABILITY    = -DSPEC_LINUX
510.parest_r:
CXXPORTABILITY    = -DSPEC_LINUX
CXXOPTIMIZE       = -std=c++11 -Wno-deprecated -Wno-narrowing
511.povray_r:
CXXPORTABILITY    = -DSPEC_LINUX
CXXOPTIMIZE       = -std=c++11 -Wno-deprecated -Wno-narrowing
519.lbm_r,619.lbm_s:
CPORTABILITY      = -DSPEC_LINUX
520.omnetpp_r,620.omnetpp_s:
CXXPORTABILITY    = -DSPEC_LINUX
CXXOPTIMIZE       = -std=c++11 -Wno-deprecated -Wno-narrowing
523.xalancbmk_r,623.xalancbmk_s:
CXXPORTABILITY    = -DSPEC_LINUX
CXXOPTIMIZE       = -std=c++11 -Wno-deprecated -Wno-narrowing -Wno-unused
525.x264_r,625.x264_s:
CPORTABILITY      = -DSPEC_LINUX
526.blender_r:
CXXPORTABILITY    = -DSPEC_LINUX
CPORTABILITY      = -funsigned-char -DSPEC_LINUX
CXXOPTIMIZE       = -std=c++11 -Wno-deprecated -Wno-narrowing
COPTIMIZE         = -std=gnu99 -Wno-implicit-function-declaration -Wno-sign-conversion
531.deepsjeng_r,631.deepsjeng_s:
CXXPORTABILITY    = -DSPEC_LINUX
538.imagick_r,638.imagick_s:
CPORTABILITY      = -DSPEC_LINUX
541.leela_r,641.leela_s:
CXXPORTABILITY    = -DSPEC_LINUX
CXXOPTIMIZE       = -std=c++11 -Wno-deprecated
557.xz_r,657.xz_s:
CPORTABILITY      = -DSPEC_NO_SSIZE_T -DSPEC_LINUX
544.nab_r,644.nab_s:
CPORTABILITY      = -DSPEC_LINUX
548.exchange2_r,648.exchange2_s:
FC                = gfortran
PORTABILITY       = -DSPEC_LP64
#-----
# PEAK OPTIMIZATIONS - Integer Benchmarks
#-----
```

```
500.perlbench_r=peak:
    OPTIMIZE           = -O2
    COPTIMIZE          = -march=native -fno-strict-aliasing -fno-unsafe-math-optimizations -fno-finite-math-
only
    EXTRA_CFLAGS      = -fno-lto # Override global LTO setting
600.perlbench_s=peak:
    OPTIMIZE           = -O2
    COPTIMIZE          = -march=native -fno-strict-aliasing -fno-unsafe-math-optimizations -fno-finite-math-
only
    EXTRA_CFLAGS      = -fno-lto # Override global LTO setting
502.gcc_r=peak:
    OPTIMIZE           = -O2
    COPTIMIZE          = -march=native -fgnu89-inline -fno-strict-aliasing
    EXTRA_LDFLAGS      = -Wl,--allow-multiple-definition
    EXTRA_CFLAGS      = -fno-lto
602.gcc_s=peak:
    OPTIMIZE           = -O2
    COPTIMIZE          = -march=native -fgnu89-inline -fno-strict-aliasing
    EXTRA_LDFLAGS      = -Wl,--allow-multiple-definition
    EXTRA_CFLAGS      = -fno-lto
505.mcf_r=peak:
    OPTIMIZE           = -O1
    COPTIMIZE          = -march=native -fno-strict-aliasing -std=gnu99 -Wno-implicit-function-declaration
605.mcf_s=peak:
    OPTIMIZE           = -O1
    COPTIMIZE          = -march=native -fno-strict-aliasing -std=gnu99 -Wno-implicit-function-declaration
520.omnetpp_r=peak:
    OPTIMIZE           = -O3
    CXXOPTIMIZE        = -march=native -std=c++11 -fno-strict-aliasing
620.omnetpp_s=peak:
    OPTIMIZE           = -O3
    CXXOPTIMIZE        = -march=native -std=c++11 -fno-strict-aliasing
523.xalancbmk_r=peak:
    OPTIMIZE           = -O1
    CXXOPTIMIZE        = -march=native -std=c++11 -fno-strict-aliasing -Wno-implicit-function-declaration
623.xalancbmk_s=peak:
    OPTIMIZE           = -O1
    CXXOPTIMIZE        = -march=native -std=c++11 -fno-strict-aliasing -Wno-implicit-function-declaration
525.x264_r=peak:
    OPTIMIZE           = -O3
    COPTIMIZE          = -march=native -funroll-loops -fcommon -fno-strict-aliasing
    EXTRA_CFLAGS      = -fno-lto # Override global LTO setting
625.x264_s=peak:
    OPTIMIZE           = -O3
    COPTIMIZE          = -march=native -funroll-loops -fcommon -fno-strict-aliasing
    EXTRA_CFLAGS      = -fno-lto # Override global LTO setting
531.deepsjeng_r=peak:
    OPTIMIZE           = -O3
    CXXOPTIMIZE        = -march=native -std=c++11
631.deepsjeng_s=peak:
    OPTIMIZE           = -O3
    CXXOPTIMIZE        = -march=native -std=c++11
541.leela_r=peak:
    OPTIMIZE           = -O3
    CXXOPTIMIZE        = -march=native -std=c++11
```

```
641.leela_s=peak:
    OPTIMIZE           = -O3
    CXXOPTIMIZE        = -march=native -std=c++11
548.exchange2_r=peak:
    FC                 = gfortran
    OPTIMIZE           = -O3
    FOPTIMIZE          = -march=native -funroll-loops -fallow-argument-mismatch
    PORTABILITY        = -DSPEC_LP64
648.exchange2_s=base,peak:
    FC                 = gfortran
    LD                 = gfortran
    PORTABILITY        = -DSPEC_LP64
648.exchange2_s=peak:
    OPTIMIZE           = -O3
    FOPTIMIZE          = -march=native -funroll-loops -fopenmp -DSPEC_OPENMP -fallow-argument-mismatch
    EXTRA_FFLAGS      = -fopenmp -DSPEC_OPENMP
    EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -fopenmp
    EXTRA_LIBS        = -lm -lgomp # gfortran linker needs lgomp
557.xz_r=peak:
    OPTIMIZE           = -O3
    COPTIMIZE          = -march=native -funroll-loops
657.xz_s=peak:
    OPTIMIZE           = -O3
    COPTIMIZE          = -march=native -funroll-loops
    EXTRA_CFLAGS      = -fopenmp -DSPEC_OPENMP -flto=thin
    EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -fopenmp -flto=thin
    EXTRA_LIBS        = -lm -lomp
#-----
# PEAK OPTIMIZATIONS - Floating Point Benchmarks
#-----
503.bwaves_r=base,peak:
    FC                 = gfortran
    F77                = gfortran
    F90                = gfortran
    PORTABILITY        = -DSPEC_LP64
503.bwaves_r=peak:
    OPTIMIZE           = -O3
    FOPTIMIZE          = -march=native -funroll-loops -fallow-argument-mismatch
603.bwaves_s=base,peak:
    FC                 = gfortran
    F77                = gfortran
    F90                = gfortran
    LD                 = gfortran
    PORTABILITY        = -DSPEC_LP64
603.bwaves_s=peak:
    OPTIMIZE           = -O3
    FOPTIMIZE          = -march=native -funroll-loops -fopenmp -DSPEC_OPENMP -fallow-argument-mismatch
    EXTRA_FFLAGS      = -fopenmp -DSPEC_OPENMP
    EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -fopenmp
    EXTRA_LIBS        = -lm -lgomp
507.cactuBSSN_r=base,peak:
    CC                 = clang
    CXX                = clang++
    FC                 = gfortran
    F77                = gfortran
```

```

F90                = gfortran
CLD                = gfortran
LD                = gfortran
CXXPORTABILITY    = -DSPEC_LINUX
PORTABILITY       = -DSPEC_LP64
507.cactuBSSN_r=peak:
OPTIMIZE          = -O3
CXXOPTIMIZE       = -march=native -std=c++11
FOPTIMIZE         = -march=native -funroll-loops -fallow-argument-mismatch
607.cactuBSSN_s=base,peak:
CC                = clang
CXX               = clang++
FC                = gfortran
F77               = gfortran
F90               = gfortran
CLD               = gfortran
LD                = gfortran
CXXPORTABILITY    = -DSPEC_LINUX
PORTABILITY       = -DSPEC_LP64
607.cactuBSSN_s=peak:
OPTIMIZE          = -O3
CXXOPTIMIZE       = -march=native -std=c++11
FOPTIMIZE         = -march=native -funroll-loops -fallow-argument-mismatch
508.namd_r=peak:
OPTIMIZE          = -O3
CXXOPTIMIZE       = -march=native -std=c++11 -ffast-math
510.parest_r=peak:
OPTIMIZE          = -O3
CXXOPTIMIZE       = -march=native -std=c++11 -fno-strict-aliasing -fno-unsafe-math-optimizations
OPTIMIZE          = -O3
CXXOPTIMIZE       = -march=native -std=c++11 -funroll-loops
519.lbm_r=peak:
OPTIMIZE          = -O1
COPTIMIZE         = -march=native
619.lbm_s=peak:
OPTIMIZE          = -O1
COPTIMIZE         = -march=native
521.wrf_r=base,peak:
CC                = clang
FC                = gfortran
F77               = gfortran
F90               = gfortran
CLD               = gfortran
LD                = gfortran
CPORTABILITY      = -DSPEC_LINUX
PORTABILITY       = -DSPEC_LP64
FPORTABILITY      = -fconvert=big-endian
EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -Wl,--no-as-needed
EXTRA_LIBS        = -lnetcdff -lnetcdf -lm -lstdc++
EXTRA_FFLAGS      = -fallow-argument-mismatch
521.wrf_r=peak:
OPTIMIZE          = -O2
FOPTIMIZE         = -march=native -fallow-argument-mismatch
621.wrf_s=base:
CC                = clang

```



```

FC                = gfortran
F77               = gfortran
F90               = gfortran
CLD               = gfortran
LD                = gfortran
CPORTABILITY      = -DSPEC_LINUX
PORTABILITY       = -DSPEC_LP64
FPORTABILITY      = -fconvert=big-endian
EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -Wl,--no-as-needed
EXTRA_LIBS        = -lnetcdff -lnetcdf -lm -lstdc++
EXTRA_FFLAGS      = -fallow-argument-mismatch
621.wrf_s=peak:
CC                = clang
FC                = gfortran
F77               = gfortran
F90               = gfortran
CLD               = gfortran
LD                = gfortran
CPORTABILITY      = -DSPEC_LINUX
PORTABILITY       = -DSPEC_LP64
FPORTABILITY      = -fconvert=big-endian
OPTIMIZE          = -O2
FOPTIMIZE         = -march=native -fopenmp -DSPEC_OPENMP -fallow-argument-mismatch
EXTRA_CFLAGS      = -fno-lto
EXTRA_FFLAGS      = -fopenmp -DSPEC_OPENMP
EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -Wl,--no-as-needed -fopenmp
EXTRA_LIBS        = -lnetcdff -lnetcdf -lm -lstdc++ -lgomp # gfortran linker needs lgomp
526.blender_r=peak:
OPTIMIZE          = -O3
COPTIMIZE         = -march=native -funsigned-char
CXXOPTIMIZE       = -march=native -std=c++11
527.cam4_r=base,peak:
CC                = clang
FC                = gfortran
F77               = gfortran
F90               = gfortran
CLD               = gfortran
LD                = gfortran
CPORTABILITY      = -DSPEC_LINUX -Wno-error=implicit-int -Wno-implicit-const-int-float-conversion
PORTABILITY       = -DSPEC_LP64
EXTRA_LDFLAGS     = -Wl,--allow-multiple-definition -Wl,--no-as-needed
EXTRA_LIBS        = -lnetcdff -lnetcdf -lm -lstdc++
EXTRA_FFLAGS      = -I/usr/include -fallow-argument-mismatch
527.cam4_r=peak:
OPTIMIZE          = -O2
FOPTIMIZE         = -march=native -I/usr/include -fallow-argument-mismatch
627.cam4_s=base:
CC                = clang
FC                = gfortran
F77               = gfortran
F90               = gfortran
CLD               = gfortran
LD                = gfortran
CPORTABILITY      = -DSPEC_LINUX -DUSE_COSP -Wno-error=implicit-int -Wno-implicit-const-int-float-
conversion

```

```

PORTABILITY          = -DSPEC_LP64
EXTRA_LDFLAGS         = -Wl,--allow-multiple-definition -Wl,--no-as-needed
EXTRA_LIBS            = -lnetcdff -lnetcdf -lm -lstdc++
EXTRA_FFLAGS          = -I/usr/include -fallow-argument-mismatch
627.cam4_s=peak:
  CC                  = clang
  FC                  = gfortran
F77                   = gfortran
  F90                 = gfortran
  CLD                 = gfortran
  LD                  = gfortran
  CPORTABILITY        = -DSPEC_LINUX -DUSE_COSP -Wno-error=implicit-int -Wno-implicit-const-int-float-
conversion
PORTABILITY          = -DSPEC_LP64
OPTIMIZE              = -O1
FOPTIMIZE             = -march=native -fopenmp -DSPEC_OPENMP -I/usr/include -fallow-argument-mismatch
EXTRA_CFLAGS          = -fopenmp -DSPEC_OPENMP -fno-lto # Override global LTO
EXTRA_FFLAGS          = -fopenmp -DSPEC_OPENMP -I/usr/include
EXTRA_LDFLAGS         = -Wl,--allow-multiple-definition -Wl,--no-as-needed -fopenmp -pthread
EXTRA_LIBS            = -lnetcdff -lnetcdf -lm -lstdc++ -lgomp # gfortran linker needs lgomp
preENV_OMP_STACKSIZE = 512M
538.imagick_r=peak:
  OPTIMIZE            = -O3
  COPTIMIZE           = -march=native -funroll-loops -ffast-math
638.imagick_s=peak:
  OPTIMIZE            = -O3
  COPTIMIZE           = -march=native -funroll-loops -ffast-math
544.nab_r=peak:
  OPTIMIZE            = -O3
  COPTIMIZE           = -march=native -ffast-math
644.nab_s=peak:
  OPTIMIZE            = -O3
  COPTIMIZE           = -march=native -ffast-math
EXTRA_CFLAGS          = -Wno-unknown-pragmas
EXTRA_LDFLAGS         = -Wl,--allow-multiple-definition
EXTRA_LIBS            = -lm
549.fotonik3d_r=base,peak:
  FC                  = gfortran
549.fotonik3d_r=peak:
  OPTIMIZE            = -O3
FOPTIMIZE             = -march=native -funroll-loops -fallow-argument-mismatch
649.fotonik3d_s=base,peak:
  FC                  = gfortran
  LD                  = gfortran
649.fotonik3d_s=peak:
  OPTIMIZE            = -O3
  FOPTIMIZE           = -march=native -funroll-loops -fopenmp -DSPEC_OPENMP -fallow-argument-mismatch
EXTRA_FFLAGS          = -fopenmp -DSPEC_OPENMP
EXTRA_LDFLAGS         = -Wl,--allow-multiple-definition -fopenmp
EXTRA_LIBS            = -lm -lgomp # gfortran linker needs lgomp
554.roms_r=base,peak:
  FC                  = gfortran
554.roms_r=peak:
  OPTIMIZE            = -O3
  FOPTIMIZE           = -march=native -funroll-loops -fallow-argument-mismatch

```

Methodology | Generation-over-Generation and Confidential Compute Comparison of Microsoft Azure® VM Instances with AMD Processors

```
654.roms_s=base,peak:
    FC                = gfortran
    F77               = gfortran
    F90               = gfortran
    LD                = gfortran
    PORTABILITY       = -DSPEC_LP64
654.roms_s=peak:
    OPTIMIZE          = -O3
    FOPTIMIZE         = -march=native -funroll-loops -fopenmp -DSPEC_OPENMP -fallow-argument-mismatch
    EXTRA_FFLAGS     = -fopenmp -DSPEC_OPENMP
    EXTRA_LDFLAGS    = -Wl,--allow-multiple-definition -fopenmp
    EXTRA_LIBS       = -lm -lgomp # gfortran linker needs lgomp
628.pop2_s=base:
    CC                = clang
    FC                = gfortran
    F77               = gfortran
    F90               = gfortran
    CLD               = gfortran
    LD                = gfortran
    CPORTABILITY      = -DSPEC_LINUX
    PORTABILITY       = -DSPEC_LP64
    EXTRA_LDFLAGS    = -Wl,--allow-multiple-definition -pthread
    EXTRA_LIBS       = -lnetcdff -lnetcdf -lm -lstdc++
    EXTRA_FFLAGS     = -fconvert=big-endian
628.pop2_s=peak:
    CC                = clang
    FC                = gfortran
    F77               = gfortran
    F90               = gfortran
    CLD               = gfortran
    LD                = gfortran
    CPORTABILITY      = -DSPEC_LINUX
    PORTABILITY       = -DSPEC_LP64
    OPTIMIZE          = -O2
    FOPTIMIZE         = -march=native -fopenmp -DSPEC_OPENMP -fconvert=big-endian -fallow-argument-
mismatch
    EXTRA_CFLAGS     = -fopenmp -DSPEC_OPENMP -fno-lto # Override global LTO
    EXTRA_FFLAGS     = -fopenmp -DSPEC_OPENMP -fconvert=big-endian
    EXTRA_LDFLAGS    = -Wl,--allow-multiple-definition -fopenmp -pthread
    EXTRA_LIBS       = -lnetcdff -lnetcdf -lm -lstdc++ -lgomp # gfortran linker needs lgomp
#-----
# Environment Variables for OpenMP
#-----
intspeed,fpspeed:
    preENV_OMP_STACKSIZE = 128M
    preENV_OMP_SCHEDULE  = static
    preENV_OMP_PROC_BIND = true
```



Legal Notices and Disclaimers

The analysis in this document was done by Prowess Consulting and commissioned by Microsoft Corporation. Results have been simulated and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance. Prowess and the Prowess logo are trademarks of Prowess Consulting, LLC. Copyright © 2025 Prowess Consulting, LLC. All rights reserved. Other trademarks are the property of their respective owners.