TUNING GUIDE

Amazon EC2 Instances Powered by 1st-4th Gen AMD EPYC™ Processors

Publication	63841
Revision	1.0
Issue Date	October, 2024

© 2024 Advanced Micro Devices, Inc. All rights reserved.

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks

AMD, the AMD Arrow logo, AMD EPYC, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names and links to external sites used in this publication are for identification purposes only and may be trademarks of their respective companies.

* Links to third party sites are provided for convenience and unless explicitly stated, AMD is not responsible for the contents of such linked sites and no endorsement is implied.

Date	Version	Changes
Oct, 2024	1.0	Initial public release

Audience

This document is intended for a technical audience such IT professionals, cloud administrators, and developers who have:

- Admin access to the AWS instances.
- Knowledge and experience selecting and configuring AWS instances for various workloads.
- Admin OS access.

Author

Akshay Raj

Table of Contents

This page intentionally left blank.

Chapter 1 Introduction

This guide describes best practices for IT professionals, cloud architects, and developers to deploy and optimize Amazon EC2 instances powered by AMD EPYC processors. The guidelines contained herein will help drive informed decisions about instance selection and configuration.

Amazon offers a wide range of instances powered by AMD EPYC processors that deliver excellent price-performance across general-purpose, compute-optimized, memory-optimized, and high-performance computing (HPC) use cases. These instances include:

AMD EPYC Generation	Instance Types
4th Gen (AMD EPYC 9004 Series) Please see *AMD EPYC™ 9004 Series Architecture Overview* (available from the AMD Documentation Hub) to learn more about 4th Gen AMD EPYC processors.	• M7a • R7a • C7a • Hpc7a
3rd Gen (AMD EPYC 7003 Series) Please see *Overview of AMD EPYC™ 7003 Series Processors Microarchitecture* (available from the AMD Documentation Hub) to learn more about 3rd Gen AMD EPYC processors.	• M6a • R6a • C6a • Hpc6a
2nd Gen (AMD EPYC 7002 Series)	• C5a
1st Gen (AMD EPYC 7001 Series)	• M5a • R5a

Table 1-1: Amazon EC2 instance types by processor generation

1.1 Amazon EC2 Instance Families

Table 1-2 compares the Amazon EC2 instance families listed in Table 1-1, above, and helps inform instance family selection by highlighting the specialized nature of each instance type.

Feature	M-Series (General Purpose)	R-Series (Memory Optimized)	C-Series (Compute Optimized)	HPC-Series (High Performance Computing)
Latest Gen	M7a	R7a	C7a	Hpc7a
CPU Generation	4th Gen AMD EPYC 9R14			
Max vCPUs	192			
SMT	OFF (1vCPU = 1 CPU core) *Note: SMT=ON (2 vCPU = 1 CPU core) for all other instance types*			
Max Memory	768 GiB	1,536 GiB	384 GiB	768 GiB
Memory::vCPU Ratio	4 to 1	8 to 1	2 to 1	4 to 1
Max Network BW	50 Gbps			300 Gbps
Max EBS BW	40 Gbps			25 Gbps
Instance Storage	EBS-Only			
Use Cases	• Web servers • Small-medium databases • Dev environments	• Traditional databases • In-memory databases • Distributed caches • Big Data analytics	• Batch processing • Ad serving • Media and Entertainment • Financial Services	• HPC workloads • AI/ML • Financial Services
SAP-Certified	Yes	Yes	No	No
Elastic Fabric Adapter Support	No	No	Yes (48xlarge)	Yes
AVX-512 Support	Yes	Yes	Yes	No
Comments	The M-series offers a balance of compute, memory, and network resources that is suitable for a wide range of workloads.	The R-series offers the highest memory-to-vCPU ratio, making it ideal for memory-intensive workloads.	The C-series provides the lowest memory-to-vCPU ratio, optimized for compute-intensive tasks.	The HPC-series offers the highest network bandwidth and includes local NVMe SSD storage, tailored for high-performance computing workloads.
Generational Comparisons	See "General Purpose (M-Series)" on page 3.	See "Memory Optimized (R-Series)" on page 4.	See "Compute Optimized (C-Series)" on page 5.	See "High Performance Computing (HPC-Series)" on page 6.

Table 1-2: Amazon instance family comparison

1.1.1 General Purpose (M-Series)

Table 1-3 highlights some of the differences between generations of general purpose Amazon EC2 M-series instances.

Feature	4th Gen AMD EPYC	3rd Gen AMD EPYC	1st Gen AMD EPYC
Instance Name	M7a*	M6a*	M5a*
Processor	AMD EPYC 9R14	AMD EPYC 7R13	AMD EPYC 7571
Max Frequency	3.7 GHz	3.6 GHz	2.5 GHz
Max vCPUs	192	192	96
Max Memory	768 GiB	768 GiB	384 GiB
Max Network BW	50 Gbps	50 Gbps	20 Gbps
Max Elastic Block Store* (EBS) BW	40 Gbps	40 Gbps	19 Gbps
SAP Certified?	Yes	Yes	No
SIMD Processor Capabilities	• AVX-512 • VNNI • bloat16	AVX2	• SSE3 • AVX
Memory Encryption	AMD Secure Memory Encryption (AMD SME)	AMD Transparent Single Key Memory Encryption (TSME)	
Built on the AWS Nitro System*, a combination of dedicated hardware and lightweight hypervisor			

Table 1-3: Generational comparison of Amazon EC2 M-series instances

1.1.2 Memory Optimized (R-Series)

Table 1-4 highlights some of the differences between generations of memory optimized Amazon EC2 R-series instances.

Feature	4th Gen AMD EPYC	3rd Gen AMD EPYC	1st Gen AMD EPYC
Instance Name	R7a*	R6a*	R5a*
Processor	AMD EPYC 9R14	AMD EPYC 7R13	AMD EPY 7571
Max Frequency	3.7 GHz	3.6 GHz	2.5 GHz
Max vCPUs	192	192	96
Max Memory	1,568 GiB	1,568 GiB	768 GiB
Max Network BW	50 Gbps	50 Gbps	25 Gbps
Max Elastic Block Store* (EBS) BW	40 Gbps	40 Gbps	19 Gbps
SAP Certified?	Yes	Yes	No
SIMD Processor Capabilities	• AVX-512 • VNNI • bloat16	AVX2	• SSE3 • AVX
Memory Encryption	AMD Secure Memory Encryption (AMD SME)	AMD Transparent Single Key Memory Encryption (TSME)	
Built on the AWS Nitro System*, a combination of dedicated hardware and lightweight hypervisor			

Table 1-4: Generational comparison of Amazon EC2 R-series instances

1.1.3 Compute Optimized (C-Series)

Table 1-5 highlights some of the differences between generations of compute optimized Amazon EC2 C-series instances.

Feature	4th Gen AMD EPYC	3rd Gen AMD EPYC	1st Gen AMD EPYC
Instance Name	C7a*	C6a*	C5a*
Processor	AMD EPYC 9R14	AMD EPYC 7R13	AMD EPY 7571
Max Frequency	3.7 GHz	3.6 GHz	2.5 GHz
Max vCPUs	192	192	96
Max Memory	384 GiB	384 GiB	768 GiB
Max Network BW	50 Gbps	50 Gbps	25 Gbps
Max Elastic Block Store* (EBS) BW	40 Gbps	40 Gbps	19 Gbps
SAP Certified?	No	Yes	Yes
SIMD Processor Capabilities	• AVX-512 • VNNI • bloat16	AVX2	• SSE3 • AVX
Memory Encryption	AMD Secure Memory Encryption (AMD SME)	AMD Transparent Single Key Memory Encryption (TSME)	
Built on the AWS Nitro System*, a combination of dedicated hardware and lightweight hypervisor			

Table 1-5: Generational comparison of Amazon EC2 C-series instances

1.1.4 High Performance Computing (HPC-Series)

Table 1-6 highlights some of the differences between generations of compute optimized Amazon EC2 HPC-series instances. This instance family is designed for tightly coupled, compute-intensive high-performance computing (HPC) workloads such as computational fluid dynamics (CFD), weather forecasting, and multiphysics simulations. They are also designed for workloads that can take advantage of improved network throughput and packet-rate performance.

Feature	4th Gen AMD EPYC	3rd Gen AMD EPYC
Instance Name	Hpc7a*	Hpc6a*
Processor	AMD EPYC 9R14*	AMD EPYC 7R13
Max Frequency	3.7 GHz	3.6 GHz
Max vCPUs	192	96
Max Memory	768 GiB	384 GiB
Max Network BW	25 Gbps	25 Gbps
Max Elastic Block Store* (EBS) BW	300 Gbps	100 Gbps
SAP Certified?	Yes	Yes
SIMD Processor Capabilities	• AVX-512 • VNNI • bloat16	AVX2
Built on the AWS Nitro System*, a combination of dedicated hardware and lightweight hypervisor		

Table 1-6: Generational comparison of Amazon EC2 HPC-series instances

Chapter 2 Instance Selection, Optimization, and Sizing

Selecting the right AMD EPYC Amazon EC2 instance type and size is crucial for optimal performance, cost-efficiency, and resource utilization. The selection process involves carefully considering your workload characteristics, performance requirements, and budget constraints to ensure that your applications run efficiently while controlling costs.

2.1 Workload Considerations

Understanding the nature of your workload is the first step in selecting the ideal AMD EPYC instance. Different types of applications have varying compute, memory, storage and network demands. Analyzing your workload profile helps you match your workload to the most suitable Amazon EC2 instance type and size for optimal performance and cost-effectiveness. This section explores various workloads and their corresponding instance recommendations. You can also use the EPYC Instance Advisor tool.

2.1.1 Compute-Intensive

Compute-intensive workloads involve complex calculations, simulations, or algorithms that place heavy demands on CPU processing power. They typically have high CPU utilization and benefit from processors with high clock speeds, multiple cores, and advanced instruction set capabilities. Some examples include:

- Scientific computing (e.g., weather modeling, physics simulations)
- Financial modeling and risk analysis
- Video encoding, transcoding, and/or rendering
- Cryptography and encryption
- Machine learning training and inference

Suggested Amazon EC2 instance types:

- C7a (4th Gen AMD EPYC)
- HPC7a (4th Gen AMD EPYC)
- C6a (3rd Gen AMD EPYC)
- HPC6a (3rd Gen AMD EPYC)

2.1.2 Big Data and Analytics

Big data and analytics workloads involve processing and manipulating large datasets that often require significant memory resources. These applications typically have high memory usage and CPU utilization with frequent data ingestion and transformation operations. Some examples include:

- Data processing and analysis
- Real-time data processing
- Stream processing
- In-memory databases and caching systems
- Business intelligence tools

Suggested Amazon EC2 instance types:

- R7a (4th Gen AMD EPYC)
- M7a (4th Gen AMD EPYC)
- R6a (3rd Gen AMD EPYC)
- M6a (3rd Gen AMD EPYC)

2.1.3 Databases

Database workloads involve frequent disk read/write operations and can be both memory and I/O intensive. These applications constantly read and write data to disk for queries, transactions, and logging, often requiring a balance of compute, memory, and storage resources. Some examples include:

- Relational databases (MySQL, PostgreSQL, Oracle)
- NoSQL databases (MongoDB, Cassandra)
- In-memory databases (Redis, Memcached)

Suggested Amazon EC2 instance types:

- R7a (4th Gen AMD EPYC)
- M7a (4th Gen AMD EPYC)
- R6a (3rd Gen AMD EPYC)
- M6a (3rd Gen AMD EPYC)

2.1.4 Web and Application Servers

Web and application server workloads typically require a balance of compute, memory, and network resources. These applications handle multiple concurrent connections and may experience varying loads throughout the day. SOme examples include:

- Web servers (Apache, NGINX)
- Ecommerce platforms
- Cloud-native applications (containerized microservices)

Suggested Amazon EC2 instance types:

- C7a (4th Gen AMD EPYC)
- M7a (4th Gen AMD EPYC)
- C6a (3rd Gen AMD EPYC)
- M6a (3rd Gen AMD EPYC)

2.1.5 AI/ML Workloads

AI/ML workloads can be both compute and memory-intensive, depending on the specific task. These applications often involve processing large datasets and performing complex mathematical operations. Some examples include:

- Machine learning model training
- Deep learning and neural networks
- Natural Language Processing (NLP)
- Computer vision and image recognition
- Recommendation systems

Suggested Amazon EC2 instance types:

- HPC7a (4th Gen AMD EPYC)
- R7a (4th Gen AMD EPYC)

2.1.6 High Performance Computing (HPC)

HPC workloads require massive parallel processing capabilities and low-latency networking. These applications typically involve solving complex computational problems that demand high levels of processing power and memory bandwidth. Some examples include:

- Computational fluid dynamics (CFD)
- Molecular dynamics simulations
- Genomics and bioinformatics
- Financial risk modeling
- Seismic analysis in oil and gas exploration

Suggested Amazon EC2 instance types:

- HPC7a (4th Gen AMD EPYC)
- HPC6a (3rd Gen AMD EPYC)

2.2 Performance Considerations

Optimizing the performance of Amazon EC2 instances powered by AMD EPYC processors requires a deep understanding of the processor architecture and carefully tuning various system components. This section explores key performance considerations that can help you maximize the efficiency and throughput of Amazon EC2 instances powered by AMD EPYC processors, including strategies for optimizing the CPU, memory, I/O, storage, network, and operating system. Implementing the best practices described in this section helps your applications fully leverage the advanced features of AMD EPYC processors such as their high core counts, large L3 cache sizes, and ample memory bandwidth. Performance optimization is an iterative process that requires ongoing benchmarking to determine the best configuration for your application.

2.2.1 CPU Optimizations

Optimizing CPU performance is crucial for compute-intensive workloads. Some key strategies and techniques for maximizing CPU performance include:

1. Identify CPU-bound workloads:
    - Use `htop` to monitor CPU usage. Consistently high utilization (near 100%) indicates CPU-bound processes.
    - Check load averages in `htop`. If they significantly exceed the number of CPU cores/threads, then the system is struggling with CPU demand.

2. Download and install the AMD Optimizing CPU Libraries (AOCL):
   - Set the `AOCL_ROOT` environment variable to point to the AOCL installation directory.
   - Include relevant header files and link against AOCL libraries during compilation:
     ```
     $ gcc -I$AOCL_ROOT/include -L$AOCL_ROOT/lib -lamdlibm -lm your_program.c -o
     your_program
     ```
   - Use specific flags for different optimizations:
     > **Vector math:** `-lamdlibm -fveclib=AMDLIBM -lm`

     > **Faster math:** `-lamdlibm -fsclrlib=AMDLIBM -lamdlibmfast -lm`
3. Maximize L3 cache usage:
   - Use Amazon EC2 instances of .2xlarge for 4thg AMD EPYC instances and .4xlarge size or larger in older-generation instances for exclusive L3 cache access.
   - Group or pin threads that share data to the same L3 cache domain using CPU affinity techniques:
     ```
     $ taskset -c 0-3 your_application
     ```
   - Use CPU pinning to avoid OS process migration away from hot L3 cache data.
4. Optimize Docker container performance:
   - Identify CPU topology using `lscpu` or `lstopo`.
   - Set CPU affinity for Docker in `/etc/docker/daemon.json`:
     ```
     {
       "cpu-rt-runtime": 950000,
       "cpu-rt-period": 1000000,
       "default-cpu-rt-runtime": 950000
     }
     ```
   - Pin containers to specific CPUs:
     ```
     $ docker run --cpuset-cpus="1,3" my-container
     ```
5. Use the Performance CPU governor:
   ```
   $ sudo cpupower frequency-set -g performance
   ```
6. Enable profiling for performance analysis:
   ```
   $ export AOCL_PROFILE=1
   ```
7. Run your application and analyze the generated `aocl_profile_report.txt`.
8. Leverage advanced instruction sets by using compiler flags to enable AVX2 and AVX-512 instructions:
   ```
   $ gcc -mavx2 -mavx512f your_program.c -o your_program
   ```
9. Optimize for specific AMD EPYC generations:
   - Use `-march=znver4` for Amazon EC2 instances powered by 4th Gen EPYC processors.
   - Use `-march=znver3` for Amazon EC2 instances powered by 3rd Gen EPYC processors.
   - Use `-march=znver2` for Amazon EC2 instances powered by 2nd Gen AMD EPYC processors.

2.2.2 Memory Optimizations

Memory-intensive workloads involve processing and manipulating large datasets that must be loaded into memory for efficient access. These workloads thus require systems with large amounts of RAM to avoid excessive paging or swapping to disk, which can significantly degrade performance. Examples of memory-intensive workloads include:

- In-memory databases and caching systems
- Big data analytics and data mining
- Machine learning inference (e.g., recommendation systems)
- Real-time data processing and stream processing
- High-performance computing (HPC) applications

Some key memory optimization strategies include:

1. Maximize memory bandwidth.
    - Resize the VM to utilize all memory channels or full socket:
        > Each AMD EPYC processor Core Compute Die (CCD) has roughly 80 GB/s peak usable bandwidth to the I/O die and an exclusive 32MB L3 cache.

        > The peak read bandwidth is approximately twice the write bandwidth, with a total peak usable bandwidth of around 10 GB/s per core.

        > Use larger instance sizes (e.g., c6a.48xlarge with 96 cores) to fully utilize the socket capacity.
    - Spread applications across multiple CCDs by distributing memory allocations across all NUMA nodes to maximize bandwidth utilization:
      ```
      $ numactl --interleave=all your_application
      ```
      This is best for applications that need high memory bandwidth but that don't heavily use shared memory across the cores. There can be trade-offs to consider if you need both high memory bandwidth and ample shared memory.
    - Leverage Amazon EC2 instances powered by latest available generation AMD EPYC generations. For example, 4th Gen AMD EPYC processors offer up to 12 DDR5-4800 memory channels per socket for a peak raw memory bandwidth of up to 460 GB/s per socket.
2. Enable and configure large pages (hugepages) to reduce TLB misses:
   ```
   $ echo 1024 > /proc/sys/vm/nr_hugepages
   $ mount -t hugetlbfs nodev /mnt/huge
   ```
3. Use `numactl` to optimize for the AMD EPYC NUMA architecture by controlling NUMA policy:
   ```
   $ numactl --membind=0 your_application  # Bind to NUMA node 0
   ```
4. Adjust `vm.swappiness` to control swap behavior:
   ```
   $ sysctl -w vm.swappiness=10
   ```
5. Monitor and manage memory usage.
    - Use tools like `free`, `vmstat`, and `sar` to monitor memory usage and swap activity.
    - Implement proper memory management in your applications to avoid memory leaks and inefficient usage.

6. Consider disabling Transparent Huge Pages (THP) for certain latency-sensitive workloads, which can improve performance:
   ```
   $ echo never > /sys/kernel/mm/transparent_hugepage/enabled
   ```

7. Optimize application-specific settings.

   - For databases, adjust buffer pool sizes and caching mechanisms.

   - For big data frameworks like Apache Spark, tune executor memory and other memory-related parameters.

8. Use memory-optimized instance types. For extremely memory-intensive workloads, consider using Amazon EC2 R-series instances (e.g., R7a) which offer higher memory-to-vCPU ratios.

2.2.3 Storage Optimizations

IO-intensive workloads are characterized by frequent disk read/write operations that result in high disk I/O activity. These workloads can cause performance bottlenecks because of the relatively slower speed of disk operations compared to CPU and memory operations. Optimizing storage performance is crucial for these workloads. Some examples include:

- Databases (MySQL, PostgreSQL, Oracle)

- Data processing applications

To optimize I/O performance on AMD EPYC instances:

1. Ensure that your selected Amazon EC2 instance is EBS-optimized for dedicated throughput between Amazon EBS and EC2.

2. Choose the right EBS volume type.

   - For high-performance workloads, use Provisioned IOPS SSD (io1/io2) volumes.

   - For general-purpose workloads, use General Purpose SSD (gp2/gp3) volumes.

3. Consider using RAID 0 (striping) across multiple EBS volumes for increased I/O performance:
   ```
   $ mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/xvdf /dev/xvdg
   ```

4. For workloads with high I/O requirements, utilize the local NVMe instance store volumes that are available on certain Amazon EC2 instance types.

5. Optimize the I/O scheduler. For example, for SSDs, use the `noop` or `deadline` scheduler:
   ```
   $ echo noop > /sys/block/nvme0n1/queue/scheduler
   ```

6. Utilize RAM disks for extremely I/O-intensive operations:

   - Create a tmpfs RAM disk:
     ```
     $ sudo mount -t tmpfs -o size=4G tmpfs /mnt/ramdisk
     ```

   - Move frequently accessed data to the RAM disk:
     ```
     $ sudo mv /var/lib/mysql /mnt/ramdisk/
     $ sudo ln -s /mnt/ramdisk/mysql /var/lib/mysql
     ```

7. Monitor I/O performance by using the following tools to identify I/O bottlenecks:
   - **iotop:** Monitor real-time I/O usage of processes.
   - **pidstat:** Print per-process I/O statistics.
   - **iostat:** Monitor system I/O device loading.
   - **vmstat:** Report virtual memory statistics.
8. Tune application-specific settings
   - For databases, adjust buffer pool sizes and I/O-related parameters.
   - For file servers, optimize caching mechanisms and network-related settings.
9. Consider using Amazon EFS for shared file systems
   - For workloads requiring shared access across multiple instances, use Amazon EFS with the **Max I/O** performance mode.
10. Implement proper I/O patterns in your application"
    - Use asynchronous I/O operations where possible.
    - Implement buffering and caching mechanisms to reduce disk access.
    - Optimize data access patterns to minimize random I/O operations.

2.2.4 Latency Sensitive Optimizations

Latency-sensitive workloads are applications that require low and predictable response times that typically range from microseconds to tens of microseconds. These workloads are often found in areas such as financial trading, online gaming, real-time analytics, and high-performance computing. Optimizing for such workloads involves minimizing sources of latency and variability in the system. Some examples of latency-sensitive workloads include:

- High-frequency trading systems
- Real-time bidding platforms
- Online gaming servers

To optimize latency:

1. Prioritize tasks using `chrt` to identify latency-sensitive tasks and set higher priorities:
```
# Set SCHED_FIFO policy for process with PID 1234 and priority 90
$ sudo chrt -f -p 90 1234
# Start a new process with SCHED_RR policy and priority 50
$ sudo chrt -r -p 50 /path/to/my_latency_sensitive_app
```

2. Disable deeper CPU C-states:

   - Install the `cpupower` tool:
     ```
     $ sudo apt install linux-tools-common
     ```

   - Disable C2 state on all cores:
     ```
     $ sudo cpupower idle-set -d 2
     ```

3. Disable Simultaneous Multi-threading (SMT)

   - Check SMT support:
     ```
     $ ls /sys/devices/system/cpu/smt
     ```

   - Disable SMT:
     ```
     $ sudo echo off > /sys/devices/system/cpu/smt/control
     ```

   - Verify SMT is disabled:
     ```
     $ cat /sys/devices/system/cpu/smt/active
     ```

4. Set the maximum core frequency:

   - Check available CPU frequency scaling options:
     ```
     $ cpupower frequency-info
     ```

   - Set the performance governor:
     ```
     $ sudo cpupower frequency-set -g performance
     ```

   - Verify the maximum frequency:
     ```
     $ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
     ```

   - Make the settings persistent:
     ```
     $ sudo systemctl enable cpupower
     $ sudo systemctl start cpupower
     ```

5. Disable Transparent Huge Pages (THP):

   - Disable THP at runtime:
     ```
     $ echo never > /sys/kernel/mm/transparent_hugepage/enabled
     $ echo never > /sys/kernel/mm/transparent_hugepage/defrag
     ```

   - For permanent disabling, add `transparent_hugepage=never` to the kernel boot parameters in the GRUB configuration file.

6. Optimize the network:

   - Use Elastic Network Adapter (ENA) for improved network performance

   - Consider using Elastic Fabric Adapter (EFA) for ultra-low latency networking in HPC and machine learning applications

7. Tune the OS:
   - Use a real-time kernel for critical low-latency applications
   - Adjust kernel parameters for network and I/O performance:
     ```
     bash
     sysctl -w net.core.rmem_max=16777216
     sysctl -w net.core.wmem_max=16777216
     sysctl -w vm.swappiness=0
     ```
8. Apply application-level optimizations:
   - Use lock-free data structures and algorithms where possible.
   - Implement efficient memory management to avoid garbage collection pauses.
   - Use asynchronous I/O operations to prevent blocking on I/O.
9. Monitor and profile your application:
   - Use tools like `perf` and `ftrace` to identify sources of latency in your application.
   - Monitor system-wide latency using `cyclictest` or the `rt-tests` suite.

2.2.5 Operating System Optimizations

Optimizing the operating system is crucial for maximizing the performance of Amazon EC2 instances powered by AMD EPYC processors. This section covers general OS tuning techniques, with a focus on sysctl parameters and tuned profiles.

2.2.5.1 Sysctl

`Sysctl` allows you to modify kernel parameters at runtime. Here are some recommended `sysctl` settings for optimizing AMD EPYC instances:

1. Network optimizations:
   ```
   # Increase the maximum number of open file descriptors
   sysctl -w fs.file-max=2097152
   # Increase network buffer sizes
   sysctl -w net.core.rmem_max=16777216
   sysctl -w net.core.wmem_max=16777216
   sysctl -w net.ipv4.tcp_rmem="4096 87380 16777216"
   sysctl -w net.ipv4.tcp_wmem="4096 65536 16777216"
   # Enable TCP Fast Open
   sysctl -w net.ipv4.tcp_fastopen=3
   # Increase the maximum number of incoming connections
   sysctl -w net.core.somaxconn=65535
   ```

2. Virtual memory optimizations:
   ```
   # Reduce swappiness
   sysctl -w vm.swappiness=10
   # Increase the amount of dirty data before writing to disk
   sysctl -w vm.dirty_ratio=60
   sysctl -w vm.dirty_background_ratio=2
   # Optimize for database workloads
   sysctl -w vm.overcommit_memory=2
   sysctl -w vm.overcommit_ratio=95
   ```

3. File system and I/O optimizations:
   ```
   # Increase the maximum number of asynchronous I/O requests
   sysctl -w fs.aio-max-nr=1048576
   # Optimize for high I/O workloads
   sysctl -w vm.dirty_bytes=1073741824
   sysctl -w vm.dirty_background_bytes=536870912
   ```

4. NUMA-specific optimizations:
   ```
   # Enable automatic NUMA balancing
   sysctl -w kernel.numa_balancing=1
   ```

You can make these changes persistent across reboots by either adding them to `/etc/sysctl.conf` or by creating a new file in `/etc/sysctl.d/`.

2.2.5.2 TuneD Profiles

TuneD is a daemon that monitors your system and optimizes its performance under certain workloads and can benefit users who simply want the best "out of the box" optimizations. TuneD includes several profiles (with a few listed below) that optimize your OS for particular applications. You can also create a custom profile for your specific needs.

- **General purpose:** balanced
- **Compute intensive:** throughput-performance
- **Latency sensitive:** latency-performance

Implementing these optimization strategies can significantly enhance the performance of your workloads on Amazon EC2 instances powered by AMD EPYC processors. Remember to benchmark your specific applications to find the optimal configuration for your use case.

2.3 Cost Considerations

Optimizing costs is a crucial aspect of running workloads on Amazon EC2 instances. This section presents several strategies for optimizing your costs when running Amazon EC2 instances powered by AMD EPYC processors.

2.3.1 Instance Right-Sizing

Instance right-sizing is the process of matching instance types and sizes to your workload performance and capacity requirements at the lowest possible cost. Consider the following for Amazon EC2 instances powered by AMD EPYC processors:

1. **Analyze current usage:** Use AWS CloudWatch* to monitor the CPU, memory, network, and disk usage metrics of your current instances.

2. **Identify underutilized resources:** Look for instances with consistently low utilization (e.g., below 40% CPU usage) as candidates for downsizing.

3. **Consider workload patterns:** Understand your application's performance requirements and usage patterns over time.

4. **Leverage AMD EPYC advantages:** Amazon EC2 instances powered by AMD EPYC processors provide optimal price-performance ratios. Learn more at Amazon EC2 Instances Powered by AMD EPYC™ Processors.

5. **Regular review:** Set up a process to regularly review and adjust your instance choices, ideally every 3-6 months or after significant application changes.

2.3.2 Reserved Instances and Savings Plan

You may be able to further optimize the cost of running Amazon EC2 instances powered by AMD EPYC processors as follows:

- **Reserved Instances:** Reserving instances can reduce costs compared to on-demand pricing in exchange for a time commitment, such as one or three year(s).
- **EC2 Instance Savings Plans:** Committing to use individual Amazon EC2 instance families in a given region can optimize costs.
- **Compute Savings Plans:** These can significantly optimize costs and can be automatically applied to Amazon EC2, Fargate, or Lambda usage regardless of instance family, size, AZ, region, OS, or tenancy.
- **Spot Instances:** Take advantage of unused Amazon EC2 capacity at a potentially significant discount compared to On-Demand prices. This may be ideal for fault-tolerant or flexible applications.

Note: AMD does not control the amount and/or applicability of any program and/or cost savings. Please consult Amazon directly for details.

2.3.3 EPYC Instance Advisor Tool

The AMD EPYC Instance Advisor (EIA) tool, helps optimize instance selection by helping you:

- Understand performance and cost across your current deployments.
- Make workload-specific recommendations by identifying the most suitable instance type based on your workload characteristics.
- Calculate potential cost savings by switching to Amazon EC2 instances powered by AMD EPYC processors.
- Deploy quickly and get 1:1 instance recommendations with cost saving estimates.

2.3.4 Cloud Cost Advisor

The AMD Cloud Cost Advisor (CCA) tool offers comprehensive cost optimization estimations:

- Receive suggestions for cost-optimized AMD EPYC instances that are comparable to your existing deployments.
- Estimate potential savings by migrating to instances powered by AMD EPYC processors.
- Generate comprehensive reports to help inform decision-making.
- Analyze and optimize costs across multiple cloud providers.

Chapter 3 Monitoring and Observability

Monitoring and observability are crucial aspects of managing Amazon EC2 instances powered by AMD EPYC processors. These practices help ensure optimal performance, identify bottlenecks, and effective troubleshooting.

3.1 AWS CloudWatch

AWS CloudWatch* is the primary monitoring service for AWS resources, including Amazon EC2 instances powered by AMD EPYC processors. It collects and tracks metrics, which are variables you can measure for your resources and applications. Some key features of AWS CloudWatch Metrics include:

- Basic and Detailed monitoring:
  - Basic monitoring provides data at 5-minute intervals at no charge.
  - Detailed monitoring offers data at 1-minute intervals for an additional cost.
- Available metrics that (among others) include:
  - CPU utilization
  - Disk read/write operations
  - Network in/out
  - Status check failed
  - EBS volume read/write bytes
- You can publish your own custom metrics to CloudWatch using the AWS CLI or API.
- Metric Math can perform mathematical operations on metrics to derive new insights.
- CloudWatch retains metric data for 15 months.
- The CloudWatch console includes graphs that help you visualize metrics.
- You can also set alarms on metrics to trigger notifications or automated actions.

To view metrics for Amazon EC2 instances powered by AMD EPYC processors:

1. Open the AWS CloudWatch console.
2. In the navigation pane, select **Metrics > All metrics**.
3. Select the **EC2** namespace.
4. Choose a metric dimension (e.g., **Per-Instance Metrics**).

3.2 Resource Monitoring Tools

Several tools can help monitor the performance of AMD EPYC instances:

- **Amazon EC2 Dashboard:** Provides a high-level overview of your EC2 instances, including those powered by AMD EPYC processors.
- **AWS Systems Manager:** Offers a unified user interface to view operational data from multiple AWS services and automate tasks across your AWS resources.
- **AWS X-Ray:** Helps developers analyze and debug distributed applications, providing insights into application performance and behavior.
- **Third-party monitoring solutions:**
    - **Datadog:** Offers comprehensive monitoring for cloud environments, including specific features for AMD EPYC instances. Learn more here*.
    - **Dynatrace:** Offers AI-powered, full-stack monitoring with specific integrations for AWS services. Lean more here*.
- Open-source tools:
    - **Prometheus:** A popular open-source monitoring and alerting toolkit. Learn more here*.
    - **Grafana:** An open-source platform for monitoring and observability, often used in conjunction with Prometheus. Learn more here*.

3.3 Instance Profiling Tools

Profiling tools help identify performance bottlenecks and optimize code execution on AMD EPYC instances:

- Virtual Memory Statistics (`vmstat`) provides information about system processes, memory, paging, block I/O, traps, and CPU activity.
- Input/Output Statistics (`iostat`) reports CPU statistics and input/output statistics for devices and partitions.
- Network Statistics (`netstat`) provides information about network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- Linux profiling with performance counters (`perf`) is a powerful Linux profiling tool that provides detailed CPU performance analysis.

- `Valgrind` is an instrumentation framework for building dynamic analysis tools, useful for memory debugging and profiling.
- AMD uProf is AMD's proprietary profiling tool for detailed performance analysis of AMD processors.
- `eBPF` (extended Berkeley Packet Filter) is a powerful and flexible Linux kernel technology that can be used for performance analysis and monitoring.

Consider the following best practices when using these tools:

- Profile in production-like environments to get accurate results.
- Focus on hot paths and frequently executed code.
- Use a combination of tools to get a comprehensive view of performance.
- Regularly profile your applications to catch performance regressions early.

This page intentionally left blank.

Chapter 4

Troubleshooting and Support

Leveraging the support resources and best practices described in this chapter will help you effectively troubleshoot and resolve issues with Amazon EC2 instances powered by AMD EPYC processors, thereby ensuring optimal workload performance and reliability.

4.1 Noisy Neighbor Mitigation

The “noisy neighbor” is a common issue with cloud instances where running multiple VMs on the same physical host can impact the performance of other instances on the same physical host. To check if you're experiencing noisy neighbor effects on memory bandwidth:

1. Run the STREAM benchmark to measure memory bandwidth.
2. Compare your results to the expected bandwidth for your instance type.

If your bandwidth is significantly lower, you may be experiencing noisy neighbor effects. To resolve this issue:

- Request a new instance and test again. Keep requesting new instances until you get one with expected performance.
- Consider using dedicated or bare metal instances for consistent performance.

4.2 Instance Placement Verification

The instance vCPUs might not be ideally placed across CCDs, which can impact performance for some workloads. To verify your instance placement:

1. Use the core-to-core latency tool*.
2. Analyze the output. You should see lower latencies between cores on the same CCD.

If you see unexpectedly high latencies between cores that should be on the same CCD, then your instance may be poorly placed. To resolve this issue:

- Request a new instance and test again until you get one with optimal core placement.
- For critical workloads, consider using dedicated hosts or bare metal instances to ensure consistent placement.

4.3 Auto Scaling and Load Balancing

Issues with auto scaling and load balancing can lead to performance problems and increased costs. Common issues include:

- Scaling too slowly or quickly in response to demand.
- Uneven load distribution across instances.
- Scaling based on inappropriate metrics.

Some solutions and best practices include:

- **Use appropriate scaling metrics:** Choose metrics that directly correlate with your application's performance, such as request latency or queue length, rather than just CPU utilization.
- **Set appropriate scaling thresholds:** Configure scaling policies to react quickly enough to demand changes without causing oscillation.
- **Implement proper health checks:** Ensure your load balancer and auto scaling group use appropriate health checks to detect and replace unhealthy instances.
- **Use target tracking scaling policies:** These policies automatically adjust capacity to maintain a specific metric at a target value.
- **Implement gradual scaling:** Use step scaling policies to add or remove capacity in increments based on alarm breach size.
- **Optimize instance warm-up:** Set appropriate cooldown periods and health check grace periods to allow new instances to warm up before receiving traffic.
- **Use multiple Availability Zones:** Distribute your Auto Scaling group across multiple AZs for better fault tolerance and performance.
- **Consider using Spot Instances:** For flexible workloads, use a mix of On-Demand and Spot Instances to optimize costs.
- **Implement proper application-level load balancing:** Ensure your application can distribute work evenly across instances, especially for stateful workloads.
- **Monitor and adjust:** Regularly review your auto scaling and load balancing performance using AWS CloudWatch and adjust settings as needed.

4.4 AWS Support Resources

- AWS Documentation*
- AWS Support Center*
- AWS re:Post*
- AWS Trusted Advisor*
- AWS Health Dashboard*
- AWS Support API*

4.5 AMD Support Resources

In addition to AWS support, AMD provides resources specifically for EPYC processors:

- AMD Developer Central offers optimization guides, technical documentation, and best practices for AMD EPYC processors.
- AMD Enterprise Support provides direct support channels for hardware-specific issues to enterprise customers.
- AMD Support Forum is a platform where developers and system administrators can discuss AMD-specific topics and share solutions.
- AMD works with various software vendors across the AMD Data Center Partner Ecosystem to ensure compatibility and optimization. Check with your software provider for AMD-specific support.

AMD recommends using Amazon support channels first when troubleshooting issues with Amazon EC2 instances powered by AMD EPYC processors. If the issue is determined to be specific to the AMD processor architecture, then you may need to engage AMD's support resources.

Remember to provide detailed information when seeking support, including:

- Instance type and AMI used
- Exact error messages or symptoms
- Steps to reproduce the issue
- Any recent changes to your environment
- Relevant AWS CloudWatch metrics or logs

This page intentionally left blank.