Performance Efficiency Pillar

AWS Well-Architected Framework

July 2020

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

This whitepaper focuses on the performance efficiency pillar of the Amazon Web Services (AWS) Well-Architected Framework. It provides guidance to help customers apply best practices in the design, delivery, and maintenance of AWS environments.

The performance efficiency pillar addresses best practices for managing production environments. This paper does not cover the design and management of non-production environments and processes, such as continuous integration or delivery.

Introduction

The AWS Well-Architected Framework helps you understand the pros and cons of decisions you make while building workloads on AWS. Using the Framework helps you learn architectural best practices for designing and operating reliable, secure, efficient, and cost-effective workloads in the cloud. The Framework provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. We believe that having well-architected workloads greatly increases the likelihood of business success.

The framework is based on five pillars:

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization

This paper focuses on applying the principles of the performance efficiency pillar to your workloads. In traditional, on-premises environments, achieving high and lasting performance is challenging. Using the principles in this paper will help you build architectures on AWS that efficiently deliver sustained performance over time.

This paper is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, and operations team members. After reading this paper, you'll understand AWS best practices and strategies to use when designing a performant cloud architecture. This paper does not provide implementation details or architectural patterns. However, it does include references to appropriate resources.

Performance Efficiency

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements, and how to maintain efficiency as demand changes and technologies evolve.

Design Principles

The following design principles can help you achieve and maintain efficient workloads in the cloud.

- **Democratize advanced technologies:** Make advanced technology implementation easier for your team by delegating complex tasks to your cloud vendor. Rather than asking your IT team to learn about hosting and running a new technology, consider consuming the technology as a service. For example, NoSQL databases, media transcoding, and machine learning are all technologies that require specialized expertise. In the cloud, these technologies become services that your team can consume, allowing your team to focus on product development rather than resource provisioning and management.
- **Go global in minutes:** Deploying your workload in multiple AWS Regions around the world allows you to provide lower latency and a better experience for your customers at minimal cost.
- **Use serverless architectures:** Serverless architectures remove the need for you to run and maintain physical servers for traditional compute activities. For example, serverless storage services can act as static websites (removing the need for web servers) and event services can host code. This removes the operational burden of managing physical servers, and can lower transactional costs because managed services operate at cloud scale.
- **Experiment more often:** With virtual and automatable resources, you can quickly carry out comparative testing using different types of instances, storage, or configurations.
- **Consider mechanical sympathy:** Use the technology approach that aligns best with your goals. For example, consider data access patterns when you select database or storage approaches.

Definition

Focus on the following areas to achieve performance efficiency in the cloud:

- Selection
- Review
- Monitoring

- Trade-offs

Take a data-driven approach to building a high-performance architecture. Gather data on all aspects of the architecture, from the high-level design to the selection and configuration of resource types.

Reviewing your choices on a regular basis, ensures that you are taking advantage of the continually evolving AWS Cloud. Monitoring ensures that you are aware of any deviance from expected performance. Make trade-offs in your architecture to improve performance, such as using compression or caching, or relaxing consistency requirements.

Selection

The optimal solution for a particular workload varies, and solutions often combine multiple approaches. Well-architected workloads use multiple solutions and enable different features to improve performance.

AWS resources are available in many types and configurations, which makes it easier to find an approach that closely matches your needs. You can also find options that are not easily achievable with on-premises infrastructure. For example, a managed service such as Amazon DynamoDB provides a fully managed NoSQL database with single-digit millisecond latency at any scale.

Performance Architecture Selection

Often, multiple approaches are required to get optimal performance across a workload. Well-architected systems use multiple solutions and enable different features to improve performance.

Use a data-driven approach to select the patterns and implementation for your architecture and achieve a cost effective solution. AWS Solutions Architects, AWS Reference Architectures, and AWS Partner Network (APN) partners can help you select an architecture based on industry knowledge, but data obtained through benchmarking or load testing will be required to optimize your architecture.

Your architecture will likely combine a number of different architectural approaches (for example, event-driven, ETL, or pipeline). The implementation of your architecture will use the AWS services that are specific to the optimization of your architecture's performance. In the following sections we discuss the four main resource types to consider (compute, storage, database, and network).

Understand the available services and resources: Learn about and understand the wide range of services and resources available in the cloud. Identify the relevant services and configuration options for your workload, and understand how to achieve optimal performance.

If you are evaluating an existing workload, you must generate an inventory of the various services resources it consumes. Your inventory helps you evaluate which components can be replaced with managed services and newer technologies.

Define a process for architectural choices: Use internal experience and knowledge of the cloud, or external resources such as published use cases, relevant

documentation, or whitepapers to define a process to choose resources and services. You should define a process that encourages experimentation and benchmarking with the services that could be used in your workload.

When you write critical user stories for your architecture, you should include performance requirements, such as specifying how quickly each critical story should execute. For these critical stories, you should implement additional scripted user journeys to ensure that you have visibility into how these stories perform against your requirements.

Factor cost requirements into decisions: Workloads often have cost requirements for operation. Use internal cost controls to select resource types and sizes based on predicted resource need.

Determine which workload components could be replaced with fully managed services, such as managed databases, in-memory caches, and other services. Reducing your operational workload allows you to focus resources on business outcomes.

For cost requirement best practices, refer to the *Cost-Effective Resources* section of the Cost Optimization Pillar whitepaper .

Use policies or reference architectures: Maximize performance and efficiency by evaluating internal policies and existing reference architectures and using your analysis to select services and configurations for your workload.

Use guidance from your cloud provider or an appropriate partner: Use cloud company resources, such as solutions architects, professional services, or an appropriate partner to guide your decisions. These resources can help review and improve your architecture for optimal performance.

Reach out to AWS for assistance when you need additional guidance or product information. AWS Solutions Architects and AWS Professional Services provide guidance for solution implementation. APN Partners provide AWS expertise to help you unlock agility and innovation for your business

Benchmark existing workloads: Benchmark the performance of an existing workload to understand how it performs on the cloud. Use the data collected from benchmarks to drive architectural decisions.

Use benchmarking with synthetic tests to generate data about how your workload's components perform. Benchmarking is generally quicker to set up than load testing and

is used to evaluate the technology for a particular component. Benchmarking is often used at the start of a new project, when you lack a full solution to load test.

You can either build your own custom benchmark tests, or you can use an industry standard test, such as TPC-DS to benchmark your data warehousing workloads. Industry benchmarks are helpful when comparing environments. Custom benchmarks are useful for targeting specific types of operations that you expect to make in your architecture.

When benchmarking, it is important to pre-warm your test environment to ensure valid results. Run the same benchmark multiple times to ensure that you've captured any variance over time.

Because benchmarks are generally faster to run than load tests, they can be used earlier in the deployment pipeline and provide faster feedback on performance deviations. When you evaluate a significant change in a component or service, a benchmark can be a quick way to see if you can justify the effort to make the change. Using benchmarking in conjunction with load testing is important because load testing informs you about how your workload will perform in production.

Load test your workload: Deploy your latest workload architecture on the cloud using different resource types and sizes. Monitor the deployment to capture performance metrics that identify bottlenecks or excess capacity. Use this performance information to design or improve your architecture and resource selection.

Load testing uses your *actual* workload so you can see how your solution performs in a production environment. Load tests must be executed using synthetic or sanitized versions of production data (remove sensitive or identifying information). Use replayed or pre-programmed user journeys through your workload at scale that exercise your entire architecture. Automatically carry out load tests as part of your delivery pipeline, and compare the results against pre-defined KPIs and thresholds. This ensures that you continue to achieve required performance.

Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch to set alarms that indicate when thresholds are breached and signal that a test is outside of the expected performance.

Using AWS services, you can run production-scale environments to test your architecture aggressively. Since you only pay for the test environment when it is needed, you can carry out full-scale testing at a fraction of the cost of using an on-

premises environment. Take advantage of the AWS Cloud to test your workload to see where it fails to scale, or scales in a non-linear way. You can use Amazon EC2 Spot Instances to generate loads at low costs and discover bottlenecks before they are experienced in production.

When load tests take considerable time to execute, parallelize them using multiple copies of your test environment. Your costs will be similar, but your testing time will be reduced. (It costs the same to run one EC2 instance for 100 hours as it does to run 100 instances for one hour.) You can also lower the costs of load testing by using Spot Instances and selecting Regions that have lower costs than the Regions you use for production.

The location of your load test clients should reflect the geographic spread of your end users.

Resources

Refer to the following resources to learn more about AWS best practices for load testing.

Videos

- Introducing The Amazon Builders' Library (DOP328)

Documentation

- AWS Architecture Center
- Amazon S3 Performance Optimization
- Amazon EBS Volume Performance
- AWS CodeDeploy
- AWS CloudFormation
- Load Testing CloudFront
- AWS CloudWatch Dashboards

Compute Architecture Selection

The optimal compute choice for a particular workload can vary based on application design, usage patterns, and configuration settings. Architectures may use different compute choices for various components and enable different features to improve performance. Selecting the wrong compute choice for an architecture can lead to lower performance efficiency.

Evaluate the available compute options: Understand the performance characteristics of the compute-related options available to you. Know how instances, containers, and functions work, and what advantages, or disadvantages, they bring to your workload.

In AWS, compute is available in three forms: instances, containers, and functions:

Instances

Instances are virtualized servers, allowing you to change their capabilities with a button or an API call. Because resource decisions in the cloud aren't fixed, you can experiment with different server types. At AWS, these virtual server instances come in different families and sizes, and they offer a wide variety of capabilities, including solid-state drives (SSDs) and graphics processing units (GPUs).

Amazon Elastic Compute Cloud (Amazon EC2) virtual server instances come in different families and sizes. They offer a wide variety of capabilities, including solid-state drives (SSDs) and graphics processing units (GPUs). When you launch an EC2 instance, the instance type that you specify determines the hardware of the host computer used for your instance. Each instance type offers different compute, memory, and storage capabilities. Instance types are grouped in instance families based on these capabilities.

Use data to select the optimal EC2 instance type for your workload, ensure that you have the correct networking and storage options, and consider operating system settings that can improve the performance for your workload.

Containers

Containers are a method of operating system virtualization that allow you to run an application and its dependencies in resource-isolated processes.

When running containers on AWS, you have two choices to make. First, choose whether or not you want to manage servers. AWS Fargate is serverless compute for containers, or Amazon EC2 can be used if you need control over the installation,

configuration, and management of your compute environment. Second, choose which container orchestrator to use: Amazon Elastic Container Service (ECS) or Amazon Elastic Kubernetes Service (EKS)

Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that allows you to automatically execute and manage containers on a cluster of EC2 instances or serverless instances using AWS Fargate. You can natively integrate Amazon ECS with other services such as Amazon Route 53, Secrets Manager, AWS Identity and Access Management (IAM), and Amazon CloudWatch.

Amazon Elastic Kubernetes Service (Amazon EKS) is a fully managed Kubernetes service. You can choose to run your EKS clusters using AWS Fargate, removing the need to provision and manage servers. EKS is deeply integrated with services such as Amazon CloudWatch, Auto Scaling Groups, AWS Identity and Access Management (IAM), and Amazon Virtual Private Cloud (VPC).

When using containers, you must use data to select the optimal type for your workload — just as you use data to select your EC2 or AWS Fargate instance types. Consider container configuration options such as memory, CPU, and tenancy configuration. To enable network access between container services, consider using a service mesh such as AWS App Mesh, which standardizes how your services communicate. Service mesh gives you end-to-end visibility and ensures high-availability for your applications.

Functions

Functions abstract the execution environment from the code you want to execute. For example, AWS Lambda allows you to execute code without running an instance.

You can use AWS Lambda to run code for any type of application or backend service with zero administration. Simply upload your code, and AWS Lambda will manage everything required to run and scale that code. You can set up your code to automatically trigger from other AWS services, call it directly, or use it with Amazon API Gateway.

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. You can create an API that acts as a “front door” to your Lambda function. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management.

To deliver optimal performance with AWS Lambda, choose the amount of memory you want for your function. You are allocated proportional CPU power and other resources. For example, choosing 256 MB of memory allocates approximately twice as much CPU power to your Lambda function as requesting 128 MB of memory. You can control the amount of time each function is allowed to run (up to a maximum of 300 seconds).

Understand the available compute configuration options: Understand how various options complement your workload, and which configuration options are best for your system. Examples of these options include instance family, sizes, features (GPU, I/O), function sizes, container instances, and single versus multi-tenancy.

When selecting instance families and types, you must also consider the configuration options available to meet your workload's needs:

- **Graphics Processing Units (GPU) —** Using general purpose computing on GPUs (GPGPU), you can build applications that benefit from the high degree of parallelism that GPUs provide by leveraging platforms (such as CUDA) in the development process. If your workload requires 3D rendering or video compression, GPUs enable hardware-accelerated computation and encoding, making your workload more efficient.
- **Field Programmable Gate Arrays (FPGA) —** Using FPGAs, you can optimize your workloads by having custom hardware-accelerated execution for your most demanding workloads. You can define your algorithms by leveraging supported general programming languages such as C or Go, or hardware-oriented languages such as Verilog or VHDL.
- **AWS Inferentia (Inf1) —** Inf1 instances are built to support machine learning inference applications. Using Inf1 instances, customers can run large scale machine learning inference applications like image recognition, speech recognition, natural language processing, personalization, and fraud detection. You can build a model in one of the popular machine learning frameworks such as TensorFlow, PyTorch, or MXNet and use GPU instances such as P3 or P3dn to train your model. After your machine learning model is trained to meet your requirements, you can deploy your model on Inf1 instances by using AWS Neuron, a specialized software development kit (SDK) consisting of a compiler, run-time, and profiling tools that optimize the machine learning inference performance of Inferentia chips.

- **Burstable instance families —** Burstable instances are designed to provide moderate baseline performance and the capability to burst to significantly higher performance when required by your workload. These instances are intended for workloads that do not use the full CPU often or consistently, but occasionally need to burst. They are well suited for general-purpose workloads, such as web servers, developer environments, and small databases. These instances provide CPU credits that can be consumed when the instance must provide performance. Credits accumulate when the instance doesn't need them.
- **Advanced computing features —** Amazon EC2 gives you access to advanced computing features, such as managing C-state and P-state registers and controlling turbo-boost of processors. Access to co- processors allows cryptography operations offloading through AES-NI, or advanced computation through AVX extensions.

The AWS Nitro System is a combination of dedicated hardware and lightweight hypervisor enabling faster innovation and enhanced security. Utilize AWS Nitro Systems when available to enable full consumption of the compute and memory resources of the host hardware. Additionally, dedicated Nitro Cards enable high speed networking, high speed EBS, and I/O acceleration.

Collect compute-related metrics: One of the best ways to understand how your compute systems are performing is to record and track the true utilization of various resources. This data can be used to make more accurate determinations about resource requirements.

Workloads (such as those running on microservices architectures) can generate large volumes of data in the form of metrics, logs, and events. Determine if your existing monitoring and observability service can manage the data generated. Amazon CloudWatch can be used to collect, access, and correlate this data on a single platform from across all your AWS resources, applications, and services running on AWS and on-premises servers, so you can easily gain system-wide visibility and quickly resolve issues.

Determine the required configuration by right-sizing: Analyze the various performance characteristics of your workload and how these characteristics relate to memory, network, and CPU usage. Use this data to choose resources that best match your workload's profile. For example, a memory-intensive workload, such as a database, could be served best by the r-family of instances. However, a bursting workload can benefit more from an elastic container system.

Use the available elasticity of resources: The cloud provides the flexibility to expand or reduce your resources dynamically through a variety of mechanisms to meet changes in demand. Combined with compute-related metrics, a workload can automatically respond to changes and utilize the optimal set of resources to achieve its goal.

Optimally matching supply to demand delivers the lowest cost for a workload, but you also must plan for sufficient supply to allow for provisioning time and individual resource failures. Demand can be fixed or variable, requiring metrics and automation to ensure that management does not become a burdensome and disproportionately large cost.

With AWS, you can use a number of different approaches to match supply with demand. The Cost Optimization Pillar whitepaper describes how to use the following approaches to cost:

- Demand-based approach
- Buffer-based approach
- Time-based approach

You must ensure that workload deployments can handle both scale-up and scale-down events. Create test scenarios for scale-down events to ensure that the workload behaves as expected.

Re-evaluate compute needs based on metrics: Use system-level metrics to identify the behavior and requirements of your workload over time. Evaluate your workload's needs by comparing the available resources with these requirements and make changes to your compute environment to best match your workload's profile. For example, over time a system might be observed to be more memory-intensive than initially thought, so moving to a different instance family or size could improve both performance and efficiency.

Resources

Refer to the following resources to learn more about AWS best practices for compute.

Videos

- Amazon EC2 foundations (CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)

- Optimize performance and cost for your AWS compute (CMP323-R1)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)

Documentation

- Instances:
  - Instance Types
  - Processor State Control for Your EC2 Instance
- EKS Containers: EKS Worker Nodes
- ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration

Storage Architecture Selection

The optimal storage solution for a particular system varies based on the kind of access method (block, file, or object), patterns of access (random or sequential), throughput required, frequency of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-architected systems use multiple storage solutions and enable different features to improve performance.

In AWS, storage is virtualized and is available in a number of different types. This makes it easier to match your storage methods with your needs, and offers storage options that are not easily achievable with on-premises infrastructure. For example, Amazon S3 is designed for 11 nines of durability. You can also change from using magnetic hard disk drives (HDDs) to SSDs, and easily move virtual drives from one instance to another in seconds.

Performance can be measured by looking at throughput, input/output operations per second (IOPS), and latency. Understanding the relationship between those measurements will help you select the most appropriate storage solution.

Storage	Services	Latency	Throughput	Shareable
Block	Amazon EBS, EC2 instance store	Lowest, consistent	Single	Mounted on EC2 instance, copies via snapshots
File system	Amazon EFS, Amazon FSx	Low, consistent	Multiple	Many clients
Object	Amazon S3	Low-latency	Web scale	Many clients
Archival	Amazon S3 Glacier	Minutes to hours	High	No

From a latency perspective, if your data is only accessed by one instance, then you should use block storage, such as Amazon EBS. Distributed file systems such as Amazon EFS generally have a small latency overhead for each file operation, so they should be used where multiple instances need access.

Amazon S3 has features than can reduce latency and increase throughput. You can use cross-region replication (CRR) to provide lower-latency data access to different geographic regions.

From a throughput perspective, Amazon EFS supports highly parallelized workloads (for example, using concurrent operations from multiple threads and multiple EC2 instances), which enables high levels of aggregate throughput and operations per second. For Amazon EFS, use a benchmark or load test to select the appropriate performance mode.

Understand storage characteristics and requirements: Understand the different characteristics (for example, shareable, file size, cache size, access patterns, latency, throughput, and persistence of data) that are required to select the services that best fit your workload, such as object storage, block storage, file storage, or instance storage.

Determine the expected growth rate for your workload and choose a storage solution that will meet those rates. Object and file storage solutions, such as Amazon S3 and Amazon Elastic File System, enable unlimited storage; Amazon EBS have pre-determined storage sizes. Elastic volumes allow you to dynamically increase capacity, tune performance, and change the type of any new or existing current generation volume with no downtime or performance impact, but it requires OS filesystem changes.

Evaluate available configuration options: Evaluate the various characteristics and configuration options and how they relate to storage. Understand where and how to use provisioned IOPS, SSDs, magnetic storage, object storage, archival storage, or ephemeral storage to optimize storage space and performance for your workload.

Amazon EBS provides a range of options that allow you to optimize storage performance and cost for your workload. These options are divided into two major categories: SSD-backed storage for transactional workloads, such as databases and boot volumes (performance depends primarily on IOPS), and HDD-backed storage for throughput-intensive workloads, such as MapReduce and log processing (performance depends primarily on MB/s).

SSD-backed volumes include the highest performance provisioned IOPS SSD for latency-sensitive transactional workloads and general purpose SSD that balance price and performance for a wide variety of transactional data.

Amazon S3 transfer acceleration enables fast transfer of files over long distances between your client and your S3 bucket. Transfer acceleration leverages Amazon CloudFront globally distributed edge locations to route data over an optimized network

path. For a workload in an S3 bucket that has intensive GET requests, use Amazon S3 with CloudFront. When uploading large files, use multi-part uploads with multiple parts uploading at the same time to help maximize network throughput.

Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. To support a wide variety of cloud storage workloads, Amazon EFS offers two performance modes: general purpose performance mode, and max I/O performance mode. There are also two throughput modes to choose from for your file system, Bursting Throughput, and Provisioned Throughput. To determine which settings to use for your workload, see the Amazon EFS User Guide.

Amazon FSx provides two file systems to choose from: Amazon FSx for Windows File Server for enterprise workloads and Amazon FSx for Lustre for high-performance workloads. FSx is SSD-backed and is designed to deliver fast, predictable, scalable, and consistent performance. Amazon FSx file systems deliver sustained high read and write speeds and consistent low latency data access. You can choose the throughput level you need to match your workload’s needs.

Make decisions based on access patterns and metrics: Choose storage systems based on your workload's access patterns and configure them by determining how the workload accesses data. Increase storage efficiency by choosing object storage over block storage. Configure the storage options you choose to match your data access patterns.

How you access data impacts how the storage solution performs. Select the storage solution that aligns best to your access patterns, or consider changing your access patterns to align with the storage solution to maximize performance.

Creating a RAID 0 (zero) array allows you to achieve a higher level of performance for a file system than what you can provision on a single volume. Consider using RAID 0 when I/O performance is more important than fault tolerance. For example, you could use it with a heavily used database where data replication is already set up separately.

Select appropriate storage metrics for your workload across all of the storage options consumed for the workload. When utilizing filesystems that use burst credits, create alarms to let you know when you are approaching those credit limits. You must create storage dashboards to show the overall workload storage health.

For storage systems that are a fixed sized, such as Amazon EBS or Amazon FSx, ensure that you are monitoring the amount of storage used versus the overall storage

size and create automation if possible to increase the storage size when reaching a threshold

Resources

Refer to the following resources to learn more about AWS best practices for storage.

Videos

- Deep dive on Amazon EBS (STG303-R1)
- Optimize your storage performance with Amazon S3 (STG343)

Documentation

- Amazon EBS:
  - Amazon EC2 Storage
  - Amazon EBS Volume Types
  - I/O Characteristics
- Amazon S3: Request Rate and Performance Considerations
- Amazon Glacier: Amazon Glacier Documentation
- Amazon EFS: Amazon EFS Performance
- Amazon FSx:
  - Amazon FSx for Lustre Performance
  - Amazon FSx for Windows File Server Performance

Database Architecture Selection

The optimal database solution for a system varies based on requirements for availability, consistency, partition tolerance, latency, durability, scalability, and query capability. Many systems use different database solutions for various sub-systems and enable different features to improve performance. Selecting the wrong database solution and features for a system can lead to lower performance efficiency.

Understand data characteristics: Understand the different characteristics of data in your workload. Determine if the workload requires transactions, how it interacts with data, and what its performance demands are. Use this data to select the best

performing database approach for your workload (for example, relational databases, NoSQL Key-value, document, wide column, graph, time series, or in-memory storage).

You can choose from many purpose-built database engines including relational, key-value, document, in-memory, graph, time series, and ledger databases. By picking the best database to solve a specific problem (or a group of problems), you can break away from restrictive one-size-fits-all monolithic databases and focus on building applications to meet the needs of your customers.

Relational databases store data with predefined schemas and relationships between them. These databases are designed to support ACID (atomicity, consistency, isolation, durability) transactions, and maintain referential integrity and strong data consistency. Many traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), and e-commerce use relational databases to store their data. You can run many of these database engines on Amazon EC2, or choose from one of the AWS managed database services: Amazon Aurora, Amazon RDS, and Amazon Redshift.

Key-value databases are optimized for common access patterns, typically to store and retrieve large volumes of data. These databases deliver quick response times, even in extreme volumes of concurrent requests.

High-traffic web apps, e-commerce systems, and gaming applications are typical use-cases for key-value databases. In AWS, you can utilize Amazon DynamoDB, a fully managed, multi-Region, multi-master, durable database with built-in security, backup and restore, and in-memory caching for internet-scale applications

In-memory databases are used for applications that require real-time access to data. By storing data directly in memory, these databases deliver microsecond latency to applications for whom millisecond latency is not enough. You may use in-memory databases for application caching, session management, gaming leaderboards, and geospatial applications. Amazon ElastiCache is a fully managed in-memory data store, compatible with Redis or Memcached.

A document database is designed to store semi structured data as JSON-like documents. These databases help developers build and update applications such as content management, catalogs, and user profiles quickly. Amazon DocumentDB is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads.

A wide column store is a type of NoSQL database. It uses tables, rows, and columns, but unlike a relational database, the names and format of the columns can vary from row to row in the same table. You typically see a wide column store in high scale industrial apps for equipment maintenance, fleet management, and route optimization. Amazon Managed Apache Cassandra Service is a wide column scalable, highly available, and managed Apache Cassandra–compatible database service.

Graph databases are for applications that must navigate and query millions of relationships between highly connected graph datasets with millisecond latency at large scale. Many companies use graph databases for fraud detection, social networking, and recommendation engines. Amazon Neptune is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets.

Time-series databases efficiently collect, synthesize, and derive insights from data that changes over time. IoT applications, DevOps, and industrial telemetry can utilize time-series databases. Amazon Timestream is a fast, scalable, fully managed time series database service for IoT and operational applications that makes it easy to store and analyze trillions of events per day.

Ledger databases provide a centralized and trusted authority to maintain a scalable, immutable, and cryptographically verifiable record of transactions for every application. We see ledger databases used for systems of record, supply chain, registrations, and even banking transactions. Amazon Quantum Ledger Database (QLDB) is a fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log owned by a central trusted authority. Amazon QLDB tracks every application data change and maintains a complete and verifiable history of changes over time.

Evaluate the available options: Evaluate the services and storage options that are available as part of the selection process for your workload's storage mechanisms. Understand how, and when, to use a given service or system for data storage. Learn about available configuration options that can optimize database performance or efficiency, such as provisioned IOPs, memory and compute resources, and caching.

Database solutions generally have configuration options that allow you to optimize for the type of workload. Using benchmarking or load testing, identify database metrics that matter for your workload. Consider the configuration options for your selected database approach such as storage optimization, database level settings, memory, and cache.

Evaluate database caching options for your workload. The three most common types of database caches are the following:

- **Database integrated caches:** Some databases (such as Amazon Aurora) offer an integrated cache that is managed within the database engine and has built-in write-through capabilities.
- **Local caches:** A local cache stores your frequently used data within your application. This speeds up your data retrieval and removes network traffic associated with retrieving data, making data retrieval faster than other caching architectures.
- **Remote caches:** Remote caches are stored on dedicated servers and typically built upon key/value NoSQL stores such as Redis and Memcached. They provide up to a million requests per second per cache node.

For Amazon DynamodDB workloads, DynamoDB Accelerator (DAX) provides a fully managed in-memory cache. DAX is an in-memory cache that delivers fast read performance for your tables at scale by enabling you to use a fully managed in-memory cache. Using DAX, you can improve the read performance of your DynamoDB tables by up to 10 times — taking the time required for reads from milliseconds to microseconds, even at millions of requests per second.

Collect and record database performance metrics: Use tools, libraries, and systems that record performance measurements related to database performance. For example, measure transactions per second, slow queries, or system latency introduced when accessing the database. Use this data to understand the performance of your database systems.

Instrument as many database activity metrics as you can gather from your workload. These metrics may need to be published directly from the workload or gathered from an application performance management service. You can use AWS X-Ray to analyze and debug production, distributed applications, such as those built using a microservices architecture. An X-Ray trace can include segments which encapsulate all the data points for single component. For example, when your application makes a call to a database in response to a request, it creates a segment for that request with a sub-segment representing the database call and its result. The sub-segment can contain data such as the query, table used, timestamp, and error status. Once instrumented, you should enable alarms for your database metrics that indicate when thresholds are breached.

Choose data storage based on access patterns: Use the access patterns of the workload to decide which services and technologies to use. For example, utilize a relational database for workloads that require transactions, or a key-value store that provides higher throughput but is eventually consistent where applicable.

Optimize data storage based on access patterns and metrics: Use performance characteristics and access patterns that optimize how data is stored or queried to achieve the best possible performance. Measure how optimizations such as indexing, key distribution, data warehouse design, or caching strategies impact system performance or overall efficiency.

Resources

Refer to the following resources to learn more about AWS best practices for databases.

Videos

- AWS purpose-built databases (DAT209-L)
- Amazon Aurora storage demystified: How it all works (DAT309-R)
- Amazon DynamoDB deep dive: Advanced design patterns (DAT403-R1)

Documentation

- AWS Database Caching
- Cloud Databases with AWS
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices
- Amazon DynamoDB Accelerator

Network Architecture Selection

The optimal network solution for a workload varies based on latency, throughput requirements, jitter, and bandwidth. Physical constraints, such as user or on-premises

resources, determine location options. These constraints can be offset with edge locations or resource placement.

On AWS, networking is virtualized and is available in a number of different types and configurations. This makes it easier to match your networking methods with your needs. AWS offers product features (for example, Enhanced Networking, Amazon EC2 networking optimized instances, Amazon S3 transfer acceleration, and dynamic Amazon CloudFront) to optimize network traffic. AWS also offers networking features (for example, Amazon Route 53 latency routing, Amazon VPC endpoints, AWS Direct Connect, and AWS Global Accelerator) to reduce network distance or jitter.

Understand how networking impacts performance: Analyze and understand how network-related features impact workload performance. For example, network latency often impacts the user experience, and not providing enough network capacity can bottleneck workload performance.

Since the network is between all application components, it can have large positive and negative impacts on application performance and behavior. There are also applications that are heavily dependent on network performance such as High Performance Computing (HPC) where deep network understanding is important to increase cluster performance. You must determine the workload requirements for bandwidth, latency, jitter, and throughput.

Evaluate available networking features: Evaluate networking features in the cloud that may increase performance. Measure the impact of these features through testing, metrics, and analysis. For example, take advantage of network-level features that are available to reduce latency, network distance, or jitter.

Many services commonly offer features to optimize network performance. Consider product features such as EC2 instance network capability, enhanced networking instance types, Amazon EBS-optimized instances, Amazon S3 transfer acceleration, and dynamic CloudFront to optimize network traffic.

AWS Global Accelerator is a service that improves global application availability and performance using the AWS global network. It optimizes the network path, taking advantage of the vast, congestion-free AWS global network. It provides static IP addresses that make it easy to move endpoints between Availability Zones or AWS Regions without needing to update your DNS configuration or change client-facing applications

Amazon S3 content acceleration is a feature that lets external users benefit from the networking optimizations of CloudFront to upload data to Amazon S3. This makes it easy to transfer large amounts of data from remote locations that don't have dedicated connectivity to the AWS Cloud.

Newer EC2 instances can leverage enhanced networking. N-series EC2 instances, such as M5n and M5dn, leverage the fourth generation of custom Nitro card and Elastic Network Adapter (ENA) device to deliver up to 100 Gbps of network throughput to a single instance. These instances offer 4x the network bandwidth and packet process compared to the base M5 instances and are ideal for network intensive applications. Customers can also enable Elastic Fabric Adapter (EFA) on certain instance sizes of M5n and M5dn instances for low and consistent network latency.

Amazon Elastic Network Adapters (ENA) provide further optimization by delivering 20 Gbps of network capacity for your instances within a single placement group. Elastic Fabric Adapter (EFA) is a network interface for Amazon EC2 instances that enables you to run workloads requiring high levels of inter-node communications at scale on AWS. With EFA, High Performance Computing (HPC) applications using the Message Passing Interface (MPI) and Machine Learning (ML) applications using NVIDIA Collective Communications Library (NCCL) can scale to thousands of CPUs or GPUs.

Amazon EBS optimized instances use an optimized configuration stack and provide additional, dedicated capacity for Amazon EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between Amazon EBS I/O and other traffic from your instance.

Latency-based routing (LBR) for Amazon Route 53 helps you improve your workload's performance for a global audience. LBR works by routing your customers to the AWS endpoint (for EC2 instances, Elastic IP addresses, or ELB load balancers) that provides the fastest experience based on actual performance measurements of the different AWS Regions where your workload is running.

Amazon VPC endpoints provide reliable connectivity to AWS services (for example, Amazon S3) without requiring an internet gateway or a Network Address Translation (NAT) instance.

Choose appropriately sized dedicated connectivity or VPN for hybrid workloads: When there is a requirement for on-premise communication, ensure that you have adequate bandwidth for workload performance. Based on bandwidth requirements, a single dedicated connection or a single VPN might not be enough, and you must enable traffic load balancing across multiple connections.

You must estimate the bandwidth and latency requirements for your hybrid workload. These numbers will drive the sizing requirements for AWS Direct Connect or your VPN endpoints.

AWS Direct Connect provides dedicated connectivity to the AWS environment, from 50 Mbps up to 10 Gbps. This gives you managed and controlled latency and provisioned bandwidth so your workload can connect easily and in a performant way to other environments. Using one of the AWS Direct Connect partners, you can have end-to-end connectivity from multiple environments, thus providing an extended network with consistent performance.

The AWS Site-to-Site VPN is a managed VPN service for VPCs. When a VPN connection is created, AWS provides tunnels to two different VPN endpoints. With AWS Transit Gateway, you can simplify the connectivity between multiple VPCs and also connect to any VPC attached to AWS Transit Gateway with a single VPN connection. AWS Transit Gateway also enables you to scale beyond the 1.25Gbps IPsec VPN throughput limit by enabling equal cost multi-path (ECMP) routing support over multiple VPN tunnels.

Leverage load-balancing and encryption offloading: Distribute traffic across multiple resources or services to allow your workload to take advantage of the elasticity that the cloud provides. You can also use load balancing for offloading encryption termination to improve performance and to manage and route traffic effectively.

When implementing a scale-out architecture where you want to use multiple instances for service content, you can leverage load balancers inside your Amazon VPC. AWS provides multiple models for your applications in the ELB service. Application Load Balancer is best suited for load balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers.

Network Load Balancer is best suited for load balancing of TCP traffic where extreme performance is required. It is capable of handling millions of requests per second while maintaining ultra-low latencies, and it is optimized to handle sudden and volatile traffic patterns.

Elastic Load Balancing provides integrated certificate management and SSL/TLS decryption, allowing you the flexibility to centrally manage the SSL settings of the load balancer and offload CPU intensive work from your workload.

Choose network protocols to optimize network traffic: Make decisions about protocols for communication between systems and networks based on the impact to the workload's performance.

There is a relationship between latency and bandwidth to achieve throughput. If your file transfer is using TCP, higher latencies will reduce overall throughput. There are approaches to fix this with TCP tuning and optimized transfer protocols, some approaches use UDP.

Choose location based on network requirements: Use the cloud location options available to reduce network latency or improve throughput. Utilize AWS Regions, Availability Zones, placement groups, and edge locations such as Outposts, Local Zones, and Wavelength, to reduce network latency or improve throughput.

The AWS Cloud infrastructure is built around Regions and Availability Zones. A Region is a physical location in the world having multiple Availability Zones.

Availability Zones consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities. These Availability Zones offer you the ability to operate production applications and databases that are more highly available, fault tolerant, and scalable than would be possible from a single data center

Choose the appropriate Region or Regions for your deployment based on the following key elements:

- **Where your users are located**: Choosing a Region close to your workload's users ensures lower latency when they use the workload.
- **Where your data is located**: For data-heavy applications, the major bottleneck in latency is data transfer. Application code should execute as close to the data as possible.
- **Other constraints**: Consider constraints such as security and compliance.

Amazon EC2 provides placement groups for networking. A placement group is a logical grouping of instances within a single Availability Zone. Using placement groups with supported instance types and an Elastic Network Adapter (ENA) enables workloads to participate in a low-latency, 25 Gbps network. Placement groups are recommended for workloads that benefit from low network latency, high network throughput, or both. Using placement groups has the benefit of lowering jitter in network communications.

Latency-sensitive services are delivered at the edge using a global network of edge locations. These edge locations commonly provide services such as content delivery network (CDN) and domain name system (DNS). By having these services at the edge, workloads can respond with low latency to requests for content or DNS resolution. These services also provide geographic services such as geo targeting of content (providing different content based on the end users' location), or latency-based routing to direct end users to the nearest Region (minimum latency).

Amazon CloudFront is a global CDN that can be used to accelerate both static content such as images, scripts, and videos, as well as dynamic content such as APIs or web applications. It relies on a global network of edge locations that will cache the content and provide high-performance network connectivity to your users. CloudFront also accelerates many other features such as content uploading and dynamic applications, making it a performance addition to all applications serving traffic over the internet. Lambda@Edge is a feature of Amazon CloudFront that will let you run code closer to users of your workload, which improves performance and reduces latency.

Amazon Route 53 is a highly available and scalable cloud DNS web service. It's designed to give developers and businesses an extremely reliable and cost-effective way to route end users to internet applications by translating names, like www.example.com, into numeric IP addresses, like 192.168.2.1, that computers use to connect to each other. Route 53 is fully compliant with IPv6.

AWS Outposts is designed for workloads that need to remain on-premises due to latency requirements, where you want that workload to run seamlessly with the rest of your other workloads in AWS. AWS Outposts are fully managed and configurable compute and storage racks built with AWS-designed hardware that allow you to run compute and storage on-premises, while seamlessly connecting to AWS's broad array of services in the cloud.

AWS Local Zones are a new type of AWS infrastructure designed to run workloads that require single-digit millisecond latency, like video rendering and graphics intensive, virtual desktop applications. Local Zones allow you to gain all the benefits of having compute and storage resources closer to end-users.

AWS Wavelength is designed to deliver ultra-low latency applications to 5G devices by extending AWS infrastructure, services, APIs, and tools to 5G networks. Wavelength embeds storage and compute inside telco providers 5G networks to help your 5G workload if it requires single-digit millisecond latency, such as IoT devices, game streaming, autonomous vehicles, and live media production.

Use edge services to reduce latency and to enable content caching. Ensure that you have configured cache control correctly for both DNS and HTTP/HTTPS to gain the most benefit from these approaches.

Optimize network configuration based on metrics: Use collected and analyzed data to make informed decisions about optimizing your network configuration. Measure the impact of those changes and use the impact measurements to make future decisions.

Enable VPC Flow logs for all VPC networks that are used by your workload. VPC Flow Logs are a feature that allows you to capture information about the IP traffic going to and from network interfaces in your VPC. VPC Flow Logs help you with a number of tasks, such as troubleshooting why specific traffic is not reaching an instance, which in turn helps you diagnose overly restrictive security group rules. You can use flow logs as a security tool to monitor the traffic that is reaching your instance, to profile your network traffic, and to look for abnormal traffic behaviors.

Use networking metrics to make changes to networking configuration as the workload evolves. Cloud based networks can be quickly re-built, so evolving your network architecture over time is necessary to maintain performance efficiency.

Resources

Refer to the following resources to learn more about AWS best practices for networking.

Videos

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)

Documentation

- Transitioning to Latency-Based Routing in Amazon Route 53
- Networking Products with AWS
- EC2
  - Amazon EBS – Optimized Instances
  - EC2 Enhanced Networking on Linux
  - EC2 Enhanced Networking on Windows
  - EC2 Placement Groups

  - Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- VPC
  - Transit Gateway
  - VPC Endpoints
  - VPC Flow Logs
- Elastic Load Balancers
  - Application Load Balancer
  - Network Load Balancer

Review

When architecting workloads, there are finite options that you can choose from. However, over time, new technologies and approaches become available that could improve the performance of your workload. In the cloud, it's much easier to experiment with new features and services because your infrastructure is code.

To adopt a data-driven approach to architecture you should implement a performance review process that considerers the following:

- **Infrastructure as code:** Define your infrastructure as code using approaches such as AWS CloudFormation templates. The use of templates allows you to place your infrastructure into source control alongside your application code and configurations. This allows you to apply the same practices you use to develop software in your infrastructure so you can iterate rapidly.
- **Deployment pipeline:** Use a continuous integration/continuous deployment (CI/CD) pipeline (for example, source code repository, build systems, deployment, and testing automation) to deploy your infrastructure. This enables you to deploy in a repeatable, consistent, and low-cost fashion as you iterate.
- **Well-defined metrics:** Set up your metrics and monitor to capture key performance indicators (KPIs). We recommend that you use both technical and business metrics. For websites or mobile apps, key metrics are capturing time to first byte or rendering. Other generally applicable metrics include thread count, garbage collection rate, and wait states. Business metrics, such as the aggregate cumulative cost per request, can alert you to ways to drive down costs. Carefully consider how you plan to interpret metrics. For example, you could choose the maximum or 99th percentile instead of the average.
- **Performance test automatically:** As part of your deployment process, automatically trigger performance tests after the quicker running tests have passed successfully. The automation should create a new environment, set up initial conditions such as test data, and then execute a series of benchmarks and load tests. Results from these tests should be tied back to the build so you can track performance changes over time. For long running tests, you can make this part of the pipeline asynchronous from the rest of the build. Alternatively, you could execute performance tests overnight using Amazon EC2 Spot Instances.

- **Load generation:** You should create a series of test scripts that replicate synthetic or prerecorded user journeys. These scripts should be idempotent and not coupled, and you might need to include “pre- warming” scripts to yield valid results. As much as possible, your test scripts should replicate the behavior of usage in production. You can use software or software-as-a-service (SaaS) solutions to generate the load. Consider using AWS Marketplace solutions and Spot Instances — they can be cost-effective ways to generate the load.
- **Performance visibility:** Key metrics should be visible to your team, especially metrics against each build version. This allows you to see any significant positive or negative trend over time. You should also display metrics on the number of errors or exceptions to make sure you are testing a working system.
- **Visualization:** Use visualization techniques that make it clear where performance issues, hot spots, wait states, or low utilization is occurring. Overlay performance metrics over architecture diagrams — call graphs or code can help identify issues quickly.

This performance review process can be implemented as a simple extension of your existing deployment pipeline and then evolved over time as your testing requirements become more sophisticated. For future architectures, you can generalize your approach and reuse the same process and artifacts.

Architectures performing poorly is usually the result of a non-existent or broken performance review process. If your architecture is performing poorly, implementing a performance review process will allow you to apply Deming’s plan-do-check-act (PDCA) cycle to drive iterative improvement.

Evolve Your Workload to Take Advantage of New Releases

Take advantage of the continual innovation at AWS driven by customer need. We release new Regions, edge locations, services, and features regularly. Any of these releases could positively improve the performance efficiency of your architecture.

Stay up-to-date on new resources and services: Evaluate ways to improve performance as new services, design patterns, and product offerings become available. Determine which of these could improve performance or increase the efficiency of the workload through ad-hoc evaluation, internal discussion, or external analysis.

Define a process to evaluate updates, new features, and services from AWS. For example, building proof-of-concepts that use new technologies or consulting with an internal group. When trying new ideas or services, run performance tests to measure the impact that they have on the efficiency or performance of the workload. Take advantage of the flexibility that you have in AWS to test new ideas or technologies frequently with minimal cost or risk.

Define a process to improve workload performance: Define a process to evaluate new services, design patterns, resource types, and configurations as they become available. For example, run existing performance tests on new instance offerings to determine their potential to improve your workload.

Your workload's performance has a few key constraints. Document these so that you know what kinds of innovation might improve the performance of your workload. Use this information when learning about new services or technology as it becomes available to identify ways to alleviate constraints or bottlenecks.

Evolve workload performance over time: As an organization, use the information gathered through the evaluation process to actively drive adoption of new services or resources when they become available.

Use the information you gather when evaluating new services or technologies to drive change. As your business or workload changes, performance needs also change. Use data gathered from your workload metrics to evaluate areas where you can get the biggest gains in efficiency or performance, and proactively adopt new services and technologies to keep up with demand.

Resources

Refer to the following resources to learn more about AWS best practices for benchmarking.

Videos

- Amazon Web Services YouTube Channel
- AWS Online Tech Talks YouTube Channel
- AWS Events YouTube Channel

Monitoring

After you implement your architecture you must monitor its performance so that you can remediate any issues before they impact your customers. Monitoring metrics should be used to raise alarms when thresholds are breached.

Monitoring at AWS consists of five distinct phases, which are explained in more detail in the Reliability Pillar whitepaper:

1. **Generation** – scope of monitoring, metrics, and thresholds
2. **Aggregation** – creating a complete view from multiple sources
3. **Real-time processing and alarming** – recognizing and responding
4. **Storage** – data management and retention policies
5. **Analytics** – dashboards, reporting, and insights

CloudWatch is a monitoring service for AWS Cloud resources and the workloads that run on AWS. You can use CloudWatch to collect and track metrics, collect and monitor log files, and set alarms. CloudWatch can monitor AWS resources such as EC2 instances and RDS DB instances, as well as custom metrics generated by your workloads and services, and any log files your applications generate. You can use CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health. You can use these insights to react quickly and keep your workload running smoothly.

CloudWatch dashboards enable you to create reusable graphs of AWS resources and custom metrics so you can monitor operational status and identify issues at a glance.

Ensuring that you do not see false positives is key to an effective monitoring solution. Automated triggers avoid human error and can reduce the time it takes to fix problems. Plan for game days, where simulations are conducted in the production environment, to test your alarm solution and ensure that it correctly recognizes issues.

Monitoring solutions fall into two types: active monitoring (AM) and passive monitoring (PM). AM and PM complement each other to give you a full view of how your workload is performing.

Active monitoring simulates user activity in scripted user journeys across critical paths in your product. AM should be continuously performed in order to test the performance and availability of a workload. AM complements PM by being continuous, lightweight,

and predictable. It can be run across all environments (especially pre-production environments) to identify problems or performance issues before they impact end users.

Passive monitoring is commonly used with web-based workloads. PM collects performance metrics from the browser (non-web-based workloads can use a similar approach). You can collect metrics across all users (or a subset of users), geographies, browsers, and device types. Use PM to understand the following issues:

- **User experience performance**: PM provides you with metrics on what your users are experiencing, which gives you a continuous view into how production is working, as well as a view into the impact of changes over time.
- **Geographic performance variability**: If a workload has a global footprint and users access the workload from all around the world, using PM can enable you to spot a performance problem impacting users in a specific geography.
- **The impact of API use**: Modern workloads use internal APIs and third-party APIs. PM provides the visibility into the use of APIs so you can identify performance bottlenecks that originate not only from internal APIs, but also from third-party API providers.

CloudWatch provides the ability to monitor and send notification alarms. You can use automation to work around performance issues by triggering actions through Amazon Kinesis, Amazon Simple Queue Service (Amazon SQS), and AWS Lambda.

Monitor Your Resources to Ensure That They Are Performing as Expected

System performance can degrade over time. Monitor system performance to identify degradation and remediate internal or external factors, such as the operating system or application load .

Record performance-related metrics: Use a monitoring and observability service to record performance-related metrics. For example, record database transactions, slow queries, I/O latency, HTTP request throughput, service latency, or other key data.

Identify the performance metrics that matter for your workload and record them. This data is an important part of being able to identify which components are impacting overall performance or efficiency of the workload.

Working back from the customer experience, identify metrics that matter. For each metric, identify the target, measurement approach, and priority. Use these to build alarms and notifications to proactively address performance-related issues.

Analyze metrics when events or incidents occur: In response to (or during) an event or incident, use monitoring dashboards or reports to understand and diagnose the impact. These views provide insight into which portions of the workload are not performing as expected.

When you write critical user stories for your architecture, include performance requirements, such as specifying how quickly each critical story should execute. For these critical stories, implement additional scripted user journeys to ensure that you know how these stories perform against your requirement

Establish Key Performance Indicators (KPIs) to measure workload performance: Identify the KPIs that indicate whether the workload is performing as intended. For example, an API-based workload might use overall response latency as an indication of overall performance, and an e-commerce site might choose to use the number of purchases as its KPI.

Document the performance experience required by customers, including how customers will judge the performance of the workload. Use these requirements to establish your key performance indicators (KPIs), which will indicate how the system is performing overall.

Use monitoring to generate alarm-based notifications: Using the performance-related key performance indicators (KPIs) that you defined, use a monitoring system that generates alarms automatically when these measurements are outside expected boundaries .

Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or a 3rd party monitoring service to set alarms that indicate when thresholds are breached; the alarms signal that a metric is outside of the expected boundaries.

Review metrics at regular intervals: As routine maintenance, or in response to events or incidents, review which metrics are collected. Use these reviews to identify which metrics were key in addressing issues and which additional metrics, if they were being tracked, would help to identify, address, or prevent issues.

As part of responding to incidents or events, evaluate which metrics were helpful in addressing the issue and which metrics could have helped that are not currently being tracked. Use this to improve the quality of metrics you collect so that you can prevent or more quickly resolve future incidents.

Monitor and alarm proactively: Use key performance indicators (KPIs), combined with monitoring and alerting systems, to proactively address performance-related issues. Use alarms to trigger automated actions to remediate issues where possible. Escalate the alarm to those able to respond if automated response is not possible. For example, you may have a system that can predict expected key performance indicators (KPI) values and alarm when they breach certain thresholds, or a tool that can automatically halt or roll back deployments if KPIs are outside of expected values.

Implement processes that provide visibility into performance as your workload is running. Build monitoring dashboards and establish baseline norms for performance expectations to determine if the workload is performing optimally.

Resources

Refer to the following resources to learn more about AWS best practices for monitoring to promote performance efficiency.

Videos

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)

Documentation

- X-Ray Documentation
- CloudWatch Documentation

Trade-offs

When you architect solutions, think about trade-offs to ensure an optimal approach. Depending on your situation, you could trade consistency, durability, and space for time or latency, to deliver higher performance.

Using AWS, you can go global in minutes and deploy resources in multiple locations across the globe to be closer to your end users. You can also dynamically add read-only replicas to information stores (such as database systems) to reduce the load on the primary database.

AWS offers caching solutions such as Amazon ElastiCache, which provides an in-memory data store or cache, and Amazon CloudFront, which caches copies of your static content closer to end users. Amazon DynamoDB Accelerator (DAX) provides a read-through/write-through distributed caching tier in front of Amazon DynamoDB, supporting the same API, but providing sub-millisecond latency for entities that are in the cache.

Using Trade-offs to Improve Performance

When architecting solutions, actively considering trade-offs enables you to select an optimal approach. Often you can improve performance by trading consistency, durability, and space for time and latency. Trade-offs can increase the complexity of your architecture and require load testing to ensure that a measurable benefit is obtained.

Understand the areas where performance is most critical: Understand and identify areas where increasing the performance of your workload will have a positive impact on efficiency or customer experience. For example, a website that has a large amount of customer interaction can benefit from using edge services to move content delivery closer to customers.

Learn about design patterns and services: Research and understand the various design patterns and services that help improve workload performance. As part of the analysis, identify what you could trade to achieve higher performance. For example, using a cache service can help to reduce the load placed on database systems; however, it requires some engineering to implement safe caching or possible introduction of eventual consistency in some areas.

Learn which performance configuration options are available to you and how they could impact the workload. Optimizing the performance of your workload depends on understanding how these options interact with your architecture and the impact they will have on both measured performance and the performance perceived by users.

The Amazon Builders' Library provides readers with a detailed description of how Amazon builds and operates technology. These free articles are written by Amazon's senior engineers and cover topics across architecture, software delivery, and operations. For example, you can see how Amazon automates software delivery to achieve over 150 million deployments a year, or how Amazon's engineers implement principles such as shuffle sharding to build resilient systems that are highly available and fault tolerant.

Identify how trade-offs impact customers and efficiency: When evaluating performance-related improvements, determine which choices will impact your customers and workload efficiency. For example, if using a key-value data store increases system performance, it is important to evaluate how the eventually consistent nature of it will impact customers.

Identify areas of poor performance in your system through metrics and monitoring. Determine how you can make improvements, what trade-offs those improvements bring, and how they impact the system and the user experience. For example, implementing caching data can help dramatically improve performance but requires a clear strategy for how and when to update or invalidate cached data to prevent incorrect system behavior.

Measure the impact of performance improvements: As changes are made to improve performance, evaluate the collected metrics and data. Use this information to determine impact that the performance improvement had on the workload, the workload's components, and your customers. This measurement helps you understand the improvements that result from the tradeoff, and helps you determine if any negative side-effects were introduced.

A well-architected system uses a combination of performance related strategies. Determine which strategy will have the largest positive impact on a given hotspot or bottleneck. For example, sharding data across multiple relational database systems could improve overall throughput while retaining support for transactions and, within each shard, caching can help to reduce the load.

Use various performance-related strategies: Where applicable, utilize multiple strategies to improve performance. For example, using strategies like caching data to prevent excessive network or database calls, using read-replicas for database engines to improve read rates, sharding or compressing data where possible to reduce data volumes, and buffering and streaming of results as they are available to avoid blocking.

As you make changes to the workload, collect and evaluate metrics to determine the impact of those changes. Measure the impacts to the system and to the end-user to understand how your trade-offs impact your workload. Use a systematic approach, such as load testing, to explore whether the tradeoff improves performance.

Resources

Refer to the following resources to learn more about AWS best practices for caching.

Video

- Introducing The Amazon Builders' Library (DOP328)

Documentation

- Amazon Builders' Library
- Best Practices for Implementing Amazon ElastiCache

Conclusion

Achieving and maintaining performance efficiency requires a data-driven approach. You should actively consider access patterns and trade-offs that will allow you to optimize for higher performance. Using a review process based on benchmarks and load tests allows you to select the appropriate resource types and configurations. Treating your infrastructure as code enables you to rapidly and safely evolve your architecture, while you use data to make fact-based decisions about your architecture. Putting in place a combination of active and passive monitoring ensures that the performance of your architecture does not degrade over time.

AWS strives to help you build architectures that perform efficiently while delivering business value. Use the tools and techniques discussed in this paper to ensure success.

Contributors

The following individuals and organizations contributed to this document:

- Eric Pullen, Performance Efficiency Lead Well-Architected, Amazon Web Services
- Philip Fitzsimons, Sr Manager Well-Architected, Amazon Web Services
- Julien Lépine, Specialist SA Manager, Amazon Web Services
- Ronnen Slasky, Solutions Architect, Amazon Web Services

Further Reading

For additional help, consult the following sources:

- AWS Well-Architected Framework

Document Revisions

Date	Description
July 2020	Major review and update of content
July 2018	Minor update for grammatical issues
November 2017	Refreshed the whitepaper to reflect changes in AWS
November 2016	First publication