Performance Efficiency Pillar

AWS Well-Architected Framework

Performance Efficiency Pillar: AWS Well-Architected Framework

Table of Contents

Performance Efficiency Pillar - AWS Well-Architected Framework

Publication date: **April 10, 2023** (*Document revisions* (p. 90))

Abstract

This whitepaper focuses on the performance efficiency pillar of the 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, cost-effective, and sustainable 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 six pillars:

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

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.

Topics

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 allow 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.

Topics

Performance architecture selection

Often, multiple approaches are required to get optimal performance across a workload. Well-architected systems use multiple solutions and allow 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 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).

Best practices

PERF01-BP01 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.

Common anti-patterns:

- You use the cloud as a collocated data center.
- You use shared storage for all things that need persistent storage.
- You do not use automatic scaling.
- You use instance types that are closest matched, but larger where needed, to your current standards.
- You deploy and manage technologies that are available as managed services.

Benefits of establishing this best practice: By considering services you may be unfamiliar with, you may be able to greatly reduce the cost of infrastructure and the effort required to maintain your services. You may be able to accelerate your time to market by deploying new services and features.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Inventory your workload software and architecture for related services: Gather an inventory of your workload and decide which category of products to learn more about. Identify workload components that can be replaced with managed services to increase performance and reduce operational complexity.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP02 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 run. 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.

Common anti-patterns:

- You assume your current architecture will become static and not be updated over time.
- You introduce architecture changes over time without justification.

Benefits of establishing this best practice: By having a defined process for making architectural changes, you permit using the gathered data to influence your workload design over time.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Select an architectural approach: Identify the kind of architecture that meets your performance requirements. Identify constraints, such as the media for delivery (desktop, web, mobile, IoT), legacy requirements, and integrations. Identify opportunities for reuse, including refactoring. Consult other teams, architecture diagrams, and resources such as AWS Solution Architects, AWS Reference Architectures, and AWS Partners to help you choose an architecture.

Define performance requirements: Use the customer experience to identify the most important metrics. For each metric, identify the target, measurement approach, and priority. Define the customer experience. Document the performance experience required by customers, including how customers will judge the performance of the workload. Prioritize experience concerns for critical user stories. Include performance requirements and implement scripted user journeys to ensure that you know how the stories perform against your requirements.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP03 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 ETL 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.

Common anti-patterns:

- You only use one family of instances.
- You do not evaluate licensed solutions versus open-source solutions
- You only use block storage.
- You deploy common software on EC2 instances and Amazon EBS or ephemeral volumes that are available as a managed service.

Benefits of establishing this best practice: Considering cost when making your selections will allow you to allow other investments.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Optimize workload components to reduce cost: Right size workload components and allow elasticity to reduce cost and maximize component efficiency. Determine which workload components can be replaced with managed services when appropriate, such as managed databases, in-memory caches, and reverse proxies.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- AWS Compute Optimizer

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture
- Optimize performance and cost for your AWS compute (CMP323-R1)

Related examples:

- AWS Samples
- AWS SDK Examples
- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF01-BP04 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.

Common anti-patterns:

- You allow wide use of technology selection that may impact the management overhead of your company.

Benefits of establishing this best practice: Establishing a policy for architecture, technology, and vendor choices will allow decisions to be made quickly.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Deploy your workload using existing policies or reference architectures: Integrate the services into your cloud deployment, then use your performance tests to ensure that you can continue to meet your performance requirements.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP05 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. AWS Partners provide AWS expertise to help you unlock agility and innovation for your business.

Common anti-patterns:

- You use AWS as a common data center provider.
- You use AWS services in a manner that they were not designed for.

Benefits of establishing this best practice: Consulting with your provider or a partner will give you confidence in your decisions.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Reach out to AWS resources for assistance: AWS Solutions Architects and Professional Services provide guidance for solution implementation. APN Partners provide AWS expertise to help you unlock agility and innovation for your business.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP06 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 and real-user monitoring 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.

Common anti-patterns:

- You rely on common benchmarks that are not indicative of your workload characteristics.
- You rely on customer feedback and perceptions as your only benchmark.

Benefits of establishing this best practice: Benchmarking your current implementation allows you to measure the improvement in performance.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Monitor performance during development: Implement processes that provide visibility into performance as your workload evolves.

Integrate into your delivery pipeline: Automatically run load tests in your delivery pipeline. Compare the test results against pre-defined key performance indicators (KPIs) and thresholds to ensure that you continue to meet performance requirements.

Test user journeys: Use synthetic or sanitized versions of production data (remove sensitive or identifying information) for load testing. Exercise your entire architecture by using replayed or pre-programmed user journeys through your application at scale.

Real-user monitoring: Use CloudWatch RUM to help you collect and view client-side data about your application performance. Use this data to help establish your real-user performance benchmarks.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- AWS Samples
- AWS SDK Examples
- Distributed Load Tests

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP07 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 that you can see how your solution performs in a production environment. Load tests must be run 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.

Common anti-patterns:

- You load test individual parts of your workload but not your entire workload.
- You load test on infrastructure that is not the same as your production environment.
- You only conduct load testing to your expected load and not beyond, to help foresee where you may have future problems.
- Performing load testing without informing AWS Support, and having your test defeated as it looks like a denial of service event.

Benefits of establishing this best practice: Measuring your performance under a load test will show you where you will be impacted as load increases. This can provide you with the capability of anticipating needed changes before they impact your workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Validate your approach with load testing: Load test a proof-of-concept to find out if you meet your performance requirements. You can use AWS services to run production-scale environments to test your architecture. Because 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.

Monitor metrics: 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 third-party solutions to set alarms that indicate when thresholds are breached.

Test at scale: Load testing uses your actual workload so you can see how your solution performs in a production environment. You can use AWS services to run production-scale environments to test your architecture. Because you only pay for the test environment when it is needed, you can run full-scale testing at a lower cost than using an on-premises environment. Take advantage of the AWS Cloud to test your workload to discover where it fails to scale, or if it scales in a non-linear way. For example, use Spot Instances to generate loads at low cost and discover bottlenecks before they are experienced in production.

Resources

Related documents:

- AWS CloudFormation

- Building AWS CloudFormation Templates using CloudFormer
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Distributed Load Testing on AWS

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 allow different features to improve performance. Selecting the wrong compute choice for an architecture can lead to lower performance efficiency.

Best practices

- PERF02-BP01 Evaluate the available compute options (p. 11)
- PERF02-BP02 Understand the available compute configuration options (p. 14)
- PERF02-BP03 Collect compute-related metrics (p. 17)
- PERF02-BP04 Determine the required configuration by right-sizing (p. 18)
- PERF02-BP05 Use the available elasticity of resources (p. 20)
- PERF02-BP06 Continually evaluate compute needs based on metrics (p. 22)

PERF02-BP01 Evaluate the available compute options

Understand how your workload can benefit from the use of different compute options, such as instances, containers and functions.

Desired outcome: By understanding all of the compute options available, you will be aware of the opportunities to increase performance, reduce unnecessary infrastructure costs, and lower the operational effort required to maintain your workload. You can also accelerate your time to market when you deploy new services and features.

Common anti-patterns:

- In a post-migration workload, using the same compute solution that was being used on premises.
- Lacking awareness of the cloud compute solutions and how those solutions might improve your compute performance.
- Oversizing an existing compute solution to meet scaling or performance requirements, when an alternative compute solution would align to your workload characteristics more precisely.

Benefits of establishing this best practice: By identifying the compute requirements and evaluating the available compute solutions, business stakeholders and engineering teams will understand the benefits and limitations of using the selected compute solution. The selected compute solution should fit the workload performance criteria. Key criteria include processing needs, traffic patterns, data access patterns, scaling needs, and latency requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understand the virtualization, containerization, and management solutions that can benefit your workload and meet your performance requirements. A workload can contain multiple types of compute solutions. Each compute solution has differing characteristics. Based on your workload scale and compute requirements, a compute solution can be selected and configured to meet your needs. The cloud architect should learn the advantages and disadvantages of instances, containers, and functions. The following steps will help you through how to select your compute solution to match your workload characteristics and performance requirements.

Type	Server	Containers	Function
AWS service	Amazon Elastic Compute Cloud (Amazon EC2)	Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS)	AWS Lambda
Key Characteristics	Has dedicated option for hardware license requirements, Placement Options, and a large selection of different instance families based on compute metrics	Easy deployment, consistent environments, runs on top of EC2 instances, Scalable	Short runtime (15 minutes or less), maximum memory and CPU are not as high as other services, Managed hardware layer, Scales to millions of concurrent requests
Common use-cases	Lift and shift migrations, monolithic application, hybrid environments, enterprise applications	Microservices, hybrid environments,	Microservices, event-driven applications

Implementation steps:

1. Select the location of where the compute solution must reside by evaluating the section called "PERF05-BP06 Choose your workload's location based on network requirements" (p. 60). This location will limit the types of compute solution available to you.
2. Identify the type of compute solution that works with the location requirement and application requirements
   a. Amazon Elastic Compute Cloud (Amazon EC2) virtual server instances come in a wide variety of different families and sizes. They offer a wide variety of capabilities, including solid state drives (SSDs) and graphics processing units (GPUs). EC2 instances offer the greatest flexibility on instance choice. When you launch an EC2 instance, the instance type that you specify determines the hardware of your instance. Each instance type offers different compute, memory, and storage capabilities. Instance types are grouped in instance families based on these capabilities. Typical use

cases include: running enterprise applications, high performance computing (HPC), training and deploying machine learning applications and running cloud native applications.

b. Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that allows you to automatically run and manage containers on a cluster of EC2 instances or serverless instances using AWS Fargate. You can use Amazon ECS with other services such as Amazon Route 53, Secrets Manager, AWS Identity and Access Management (IAM), and Amazon CloudWatch. Amazon ECS is recommended if your application is containerized and your engineering team prefers Docker containers.

c. 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. Managing Amazon EKS is simplified due to integrations with AWS 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 compute metrics to select the optimal type for your workload, similar to how you use compute metrics to select your EC2 or AWS Fargate instance types. Amazon EKS is recommended if your application is containerized and your engineering team prefers Kubernetes over Docker containers.

d. You can use AWS Lambda to run code that supports the allowed runtime, memory, and CPU options. 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 run from other AWS services or call it directly. Lambda is recommended for short running, microservice architectures developed for the cloud.

3. After you have experimented with your new compute solution, plan your migration and validate your performance metrics. This is a continual process, see the section called "PERF02-BP04 Determine the required configuration by right-sizing" (p. 18).

Level of effort for the implementation plan: If a workload is moving from one compute solution to another, there could be a *moderate* level of effort involved in refactoring the application.

Resources

Related documents:

- Cloud Compute with AWS
- EC2 Instance Types
- Processor State Control for Your EC2 Instance
- EKS Containers: EKS Worker Nodes
- Amazon ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration
- Prescriptive Guidance for Containers
- Prescriptive Guidance for Serverless

Related videos:

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

Related examples:

- Migrating the web application to containers
- Run a Serverless Hello World

PERF02-BP02 Understand the available compute configuration options

Each compute solution has options and configurations available to you to support your workload characteristics. Learn how various options complement your workload, and which configuration options are best for your application. Examples of these options include instance family, sizes, features (GPU, I/O), bursting, time-outs, function sizes, container instances, and concurrency.

Desired outcome: The workload characteristics including CPU, memory, network throughput, GPU, IOPS, traffic patterns, and data access patterns are documented and used to configure the compute solution to match the workload characteristics. Each of these metrics plus custom metrics specific to your workload are recorded, monitored, and then used to optimize the compute configuration to best meet the requirements.

Common anti-patterns:

- Using the same compute solution that was being used on premises.
- Not reviewing the compute options or instance family to match workload characteristics.
- Oversizing the compute to ensure bursting capability.
- You use multiple compute management platforms for the same workload.

Benefits of establishing this best practice: Be familiar with the AWS compute offerings so that you can determine the correct solution for each of your workloads. After you have selected the compute offerings for your workload, you can quickly experiment with those compute offerings to determine how well they meet your workload needs. A compute solution that is optimized to meet your workload characteristics will increase your performance, lower your cost and increase your reliability.

Level of risk exposed if this best practice is not established: High

Implementation guidance

If your workload has been using the same compute option for more than four weeks and you anticipate that the characteristics will remain the same in the future, you can use AWS Compute Optimizer to provide a recommendation to you based on your compute characteristics. If AWS Compute Optimizer is not an option due to lack of metrics, a non-supported instance type or a foreseeable change in your characteristics then you must predict your metrics based on load testing and experimentation.

Implementation steps:

1. Are you running on EC2 instances or containers with the EC2 Launch Type?
   a. Can your workload use GPUs to increase performance?
      i. Accelerated Computing instances are GPU-based instances that provide the highest performance for machine learning training, inference and high performance computing.
   b. Does your workload run machine learning inference applications?
      i. 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, such as 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, 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, runtime, and profiling tools that optimize the machine learning inference performance of Inferentia chips.

c. Does your workload integrate with the low-level hardware to improve performance?
    i. Field Programmable Gate Arrays (FPGA) — Using FPGAs, you can optimize your workloads by having custom hardware-accelerated operation 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.

d. Do you have at least four weeks of metrics and can predict that your traffic pattern and metrics will remain about the same in the future?
    i. Use Compute Optimizer to get a machine learning recommendation on which compute configuration best matches your compute characteristics.

e. Is your workload performance constrained by the CPU metrics?
    i. Compute-optimized instances are ideal for the workloads that require high performing processors.

f. Is your workload performance constrained by the memory metrics?
    i. Memory-optimized instances deliver large amounts of memory to support memory intensive workloads.

g. Is your workload performance constrained by IOPS?
    i. Storage-optimized instances are designed for workloads that require high, sequential read and write access (IOPS) to local storage.

h. Do your workload characteristics represent a balanced need across all metrics?
    i. Does your workload CPU need to burst to handle spikes in traffic?
        A. Burstable Performance instances are similar to Compute Optimized instances except they offer the ability to burst past the fixed CPU baseline identified in a compute-optimized instance.
    ii. General Purpose instances provide a balance of all characteristics to support a variety of workloads.

i. Is your compute instance running on Linux and constrained by network throughput on the network interface card?
    i. Review Performance Question 5, Best Practice 2: Evaluate available networking features to find the right instance type and family to meet your performance needs.

j. Does your workload need consistent and predictable instances in a specific Availability Zone that you can commit to for a year?
    i. Reserved Instances confirms capacity reservations in a specific Availability Zone. Reserved Instances are ideal for required compute power in a specific Availability Zone.

k. Does your workload have licenses that require dedicated hardware?
    i. Dedicated Hosts support existing software licenses and help you meet compliance requirements.

l. Does your compute solution burst and require synchronous processing?
    i. On-Demand Instances let you use the compute capacity by the hour or second with no long-term commitment. These instances are good for bursting above performance baseline needs.

m. Is your compute solution stateless, fault-tolerant, and asynchronous?
    i. Spot Instances let you take advantage of unused instance capacity for your stateless, fault-tolerant workloads.

2. Are you running containers on Fargate?
    a. Is your task performance constrained by the memory or CPU?
        i. Use the Task Size to adjust your memory or CPU.
    b. Is your performance being affected by your traffic pattern bursts?
        i. Use the Auto Scaling configuration to match your traffic patterns.

3. Is your compute solution on Lambda?
    a. Do you have at least four weeks of metrics and can predict that your traffic pattern and metrics will remain about the same in the future?
        i. Use Compute Optimizer to get a machine learning recommendation on which compute configuration best matches your compute characteristics.
    b. Do you not have enough metrics to use AWS Compute Optimizer?
        i. If you do not have metrics available to use Compute Optimizer, use AWS Lambda Power Tuning to help select the best configuration.
    c. Is your function performance constrained by the memory or CPU?
        i. Configure your Lambda memory to meet your performance needs metrics.
    d. Is your function timing out when running?
        i. Change the timeout settings
    e. Is your function performance constrained by bursts of activity and concurrency?
        i. Configure the concurrency settings to meet your performance requirements.
    f. Does your function run asynchronously and is failing on retries?
        i. Configure the maximum age of the event and the maximum retry limit in the asynchronous configuration settings.

Level of effort for the implementation plan:

To establish this best practice, you must be aware of your current compute characteristics and metrics. Gathering those metrics, establishing a baseline and then using those metrics to identify the ideal compute option is a *low* to *moderate* level of effort. This is best validated by load tests and experimentation.

Resources

Related documents:

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

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Optimize performance and cost for your AWS compute (CMP323-R1)

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF02-BP03 Collect compute-related metrics

To understand how your compute resources are performing, you must record and track the utilization of various systems. This data can be used to make more accurate determinations about resource requirements.

Workloads can generate large volumes of data such as metrics, logs, and events. Determine if your existing storage, monitoring, and observability service can manage the data generated. Identify which metrics reflect resource utilization and can be collected, aggregated, and correlated on a single platform across. Those metrics should represent all your workload resources, applications, and services, so you can easily gain system-wide visibility and quickly identify performance improvement opportunities and issues.

Desired outcome: All metrics related to the compute-related resources are identified, collected, aggregated, and correlated on a single platform with retention implemented to support cost and operational goals.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.

Benefits of establishing this best practice: To monitor the performance of your workloads, you must record multiple performance metrics over a period of time. These metrics allow you to detect anomalies in performance. They will also help gauge performance against business metrics to ensure that you are meeting your workload needs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify, collect, aggregate, and correlate compute-related metrics. Using a service such as Amazon CloudWatch, can make the implementation quicker and easier to maintain. In addition to the default metrics recorded, identify and track additional system-level metrics within your workload. Record data such as CPU utilization, memory, disk I/O, and network inbound and outbound metrics to gain insight into utilization levels or bottlenecks. This data is crucial to understand how the workload is performing and how the compute solution is utilized. Use these metrics as part of a data-driven approach to actively tune and optimize your workload's resources.

Implementation steps:

1. Which compute solution metrics are important to track?
   a. EC2 default metrics
   b. Amazon ECS default metrics
   c. EKS default metrics
   d. Lambda default metrics
   e. EC2 memory and disk metrics
2. Do I currently have an approved logging and monitoring solution?
   a. Amazon CloudWatch
   b. AWS Distro for OpenTelemetry
   c. Amazon Managed Service for Prometheus

3. Have I identified and configured my data retention policies to match my security and operational goals?
   a. Default data retention for CloudWatch metrics
   b. Default data retention for CloudWatch Logs
4. How do you deploy your metric and log aggregation agents?
   a. AWS Systems Manager automation
   b. OpenTelemetry Collector

Level of effort for the Implementation Plan: There is a *medium* level of effort to identify, track, collect, aggregate, and correlate metrics from all compute resources.

Resources

Related documents:

- Amazon CloudWatch documentation
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch Agent
- Accessing Amazon CloudWatch Logs for AWS Lambda
- Using CloudWatch Logs with container instances
- Publish custom metrics
- AWS Answers: Centralized Logging
- AWS Services That Publish CloudWatch Metrics
- Monitoring Amazon EKS on AWS Fargate

Related videos:

- Application Performance Management on AWS
- Build a Monitoring Plan

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- Level 100: Monitoring Windows EC2 instance with CloudWatch Dashboards
- Level 100: Monitoring an Amazon Linux EC2 instance with CloudWatch Dashboards

PERF02-BP04 Determine the required configuration by right-sizing

Analyze the various performance characteristics of your workload and how these characteristics relate to memory, network, I/O, and CPU usage. Use this data to choose resources that best match your workload's profile. For example, a memory-intensive workload like a database may benefit from a higher memory per core ratio. However, a compute intensive workload may need a higher core count and frequency, but can be satisfied with a lower amount of memory per core.

Common anti-patterns:

- You choose an instance with the largest values across all performance characteristics available for all workloads.

- You standardize all instances types to one type for ease of management.
- You optimize against standard synthetic benchmarks without validating the actual requirements of a particular workload.
- You keep the same infrastructure for a long period of time without reevaluating and integrating new offerings.

Benefits of establishing this best practice: When you are familiar with the requirements of your workload, you can compare these needs with available compute offerings and quickly experiment to determine which ones meet the needs of your workload most efficiently. This allows for optimal performance without overpaying for resources not required.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Modify your workload configuration by right sizing. To optimize performance, overall efficiency, and cost effectiveness, determine first which resources your workload needs. Choose memory-optimized instances, such as the R-family of instances, for memory-intensive workloads like a database. For workloads that require higher compute capacity, choose the C-family of instances, or choose instances with higher core counts or higher core frequency. Choose I/O performance based on the needs of your workload instead of comparing against standard, synthetic benchmarks. For higher I/O performance, choose instances from the I-family of instances, select I/O optimized Amazon EBS volumes, or choose instances with instance store. For more detail on particular instance types, see Amazon EC2 instance types.

Right sizing verifies that your workloads perform as well as possible while not overpaying on resources not needed.

Implementation steps

- Know your workload or analyze its resource requirements.
- Evaluate workloads separately. The AWS Cloud gives you flexibility and agility to right-size each workload on its own without needing to compromise.
- Create test environments to find the best match of compute offerings against your workload.
- Continually reevaluate new compute offerings, and compare against your workload's needs.
- Routinely review new service offers for better price performance.
- Regularly conduct Well-Architected Framework Reviews.

Resources

Related best practices:

- PERF02-BP03 Collect compute-related metrics (p. 17)
- PERF02-BP06 Continually evaluate compute needs based on metrics (p. 22)

Related documents:

- AWS Compute Optimizer
- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- How to choose compute option for startups
- Optimize performance and cost for your AWS compute (CMP323-R1)

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

PERF02-BP05 Use the available elasticity of resources

The cloud provides the flexibility to expand and reduce your resources dynamically through a variety of mechanisms to meet changes in demand. Combining this elasticity with compute-related metrics, a workload can automatically respond to changes to use the resources it needs and only the resources it needs.

Common anti-patterns:

- You overprovision to cover possible spikes.
- You react to alarms by manually increasing capacity.
- You increase capacity without considering provisioning time.
- You leave increased capacity after a scaling event instead of scaling back down.
- You monitor metrics that don't directly reflect your workloads true requirements.

Benefits of establishing this best practice: Demand can be fixed, variable, follow a pattern or be spiky. Matching supply to demand delivers the lowest cost for a workload. Monitoring, testing, and configuring workload elasticity will optimize performance, save money, and improve reliability as usage demands change. Although a manual approach to this is possible, it is impractical at larger scales. An automated and metrics-based approach assures resources meet demands and any given time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Metric based automation should be used to take advantage of elasticity with the goal that the supply of resources you have matches the demand of the resources your workload requires. For example, you can use Amazon CloudWatch metrics to monitor your resources, or use Amazon CloudWatch metrics for your Auto Scaling groups.

Combined with compute-related metrics, a workload can automatically respond to changes and use the optimal set of resources to achieve its goal. You also must plan for provisioning time and potential resource failures.

Instances, containers, and functions provide mechanisms for elasticity either as a feature of the service, in the form of Application Auto Scaling, or in combination with Amazon EC2 Auto Scaling. Use elasticity in your architecture to verify that you have sufficient capacity to meet performance requirements at a wide variety of scales of use.

Validate your metrics for scaling up or down elastic resources against the type of workload being deployed. As an example, if you are deploying a video transcoding application, 100% CPU utilization is expected and should not be your primary metric. Alternatively, you can measure against the queue depth of transcoding jobs waiting to scale your instance types.

Workload deployments need to handle both scale up and scale down events. Scaling down workload components safely is as critical as scaling up resources when demand dictates.

Create test scenarios for scaling events to verify that the workload behaves as expected.

Implementation steps

- Leverage historical data to analyze your workload's resource demands over time. Ask specific questions like:
  - Is your workload steady and increasing over time at a known rate?
  - Does your workload increase and decrease in seasonal, repeatable patterns?
  - Is your workload spiky? Can the spikes be anticipated or predicted?
- Leverage monitoring services and historical data as much as possible.
- Tagging resources can help with monitoring. When using tags, refer to tagging best practices. Additionally, tags can help you manage, identify, and organize resources.
- With AWS, you can use a number of different approaches to match supply with demand. The cost optimization pillar best practices (COST09-BP01 through COST09-03) describe how to use the following approaches to cost:
  - COST09-BP01 Perform an analysis on the workload demand
  - COST09-BP02 Implement a buffer or throttle to manage demand
  - COST09-BP03 Supply resources dynamically
- Create test scenarios for scale down events to verify that the workload behaves as expected.
- Most non-production instances should be stopped when they are not being used.
- For storage needs when using Amazon Elastic Block Store (Amazon EBS), take advantage of volume-based elasticity.
- For Amazon Elastic Compute Cloud (Amazon EC2), consider using Auto Scaling groups, which allow you to optimize performance and cost by automatically increasing the number of compute instances during demand spikes and decreasing capacity when demand decreases.

Resources

Related best practices:

- PERF02-BP03 Collect compute-related metrics (p. 17)
- PERF02-BP04 Determine the required configuration by right-sizing (p. 18)
- PERF02-BP06 Continually evaluate compute needs based on metrics (p. 22)

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system

Related examples:

- Amazon EC2 Auto Scaling Group Examples
- Amazon EFS Tutorials

PERF02-BP06 Continually evaluate compute needs based on metrics

Use a data-driven approach to continually evaluate and optimize the compute resources for your workload over time.

Desired outcome: Use system-level metrics to actively monitor the behavior and requirements of your workload over time. Evaluate the demands of your workload against available resources based on the collected data, and make changes to your compute environment to best match your workload's profile. For example, a workload might be observed over time to be more memory-intensive than initially specified, so moving to a different instance family or size could improve both performance and efficiency.

Common anti-patterns:

- Monitoring system-level metrics to gain insight into your workload and not re-evaluating compute needs.
- Architecting your compute needs for peak workload requirements.
- Oversizing the existing compute solution to meet scaling or performance requirements when moving to an alternative compute solution would more efficiently match your workload characteristics.

Benefits of establishing this best practice: Optimized compute resources based on real-world data and your desired balance of cost and performance.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use a data-driven approach to optimize compute resources based on observed workload behavior. To achieve maximum performance and efficiency, use the data gathered over time from your workload to continually tune and optimize your resources. Look at the trends in your workload's usage of current resources and determine where you can make changes to better match your workload's needs. When resources are over-committed, system performance degrades, and when resources are not adequately used, the system is operating less efficiently and at a higher cost.

To optimize performance and resource utilization, you need a unified operational view, real-time granular data, and a historical reference. You can create automated dashboards to visualize this data and derive operational and utilization insights.

Implementation steps

1. Collect compute-related metrics over time.

2. Compare workload metrics against available resources in your selected compute solution.
3. Determine any required configuration changes by right-sizing the existing solution or evaluating alternative compute solutions.

Resources

Related best practices:

- PERF02-BP01 Evaluate the available compute options (p. 11)
- PERF02-BP02 Understand the available compute configuration options (p. 14)
- PERF02-BP03 Collect compute-related metrics (p. 17)
- PERF02-BP04 Determine the required configuration by right-sizing (p. 18)

Related documents:

- Cloud Compute with AWS
- AWS Compute Optimizer
- EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Best practices for working with AWS Lambda functions

Related videos:

- Amazon EC2 foundations (CMP211-R2)
- Better, faster, cheaper compute: Cost-optimizing Amazon EC2 (CMP202-R1)
- Deliver high performance ML inference with AWS Inferentia (CMP324-R1)
- Optimize performance and cost for your AWS compute (CMP323-R1)
- Powering next-gen Amazon EC2: Deep dive into the Nitro system
- Selecting and optimizing Amazon EC2 instances

Related examples:

- Rightsizing with Compute Optimizer and Memory utilization enabled
- AWS Compute Optimizer Demo code

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 allow 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 permits high levels of aggregate throughput and operations per second. For Amazon EFS, use a benchmark or load test to select the appropriate performance mode.

Best practices

- PERF03-BP01 Understand storage characteristics and requirements (p. 24)
- PERF03-BP02 Evaluate available configuration options (p. 28)
- PERF03-BP03 Make decisions based on access patterns and metrics (p. 29)

PERF03-BP01 Understand storage characteristics and requirements

Identify and document the workload storage needs and define the storage characteristics of each location. Examples of storage characteristics include: shareable access, file size, growth rate, throughput, IOPS, latency, access patterns, and persistence of data. Use these characteristics to evaluate if block, file, object, or instance storage services are the most efficient solution for your storage needs.

Desired outcome: Identify and document the storage requirements per storage requirement and evaluate the available storage solutions. Based on the key storage characteristics, your team will understand how the selected storage services will benefit your workload performance. Key criteria include data access patterns, growth rate, scaling needs, and latency requirements.

Common anti-patterns:

- You only use one storage type, such as Amazon Elastic Block Store (Amazon EBS), for all workloads.
- You assume that all workloads have similar storage access performance requirements.

Benefits of establishing this best practice: Selecting the storage solution based on the identified and required characteristics will help improve your workloads performance, decrease costs and lower your

operational efforts in maintaining your workload. Your workload performance will benefit from the solution, configuration, and location of the storage service.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify your workload's most important storage performance metrics and implement improvements as part of a data-driven approach, using benchmarking or load testing. Use this data to identify where your storage solution is constrained, and examine configuration options to improve the solution. Determine the expected growth rate for your workload and choose a storage solution that will meet those rates. Research the AWS storage offerings to determine the correct storage solution for your various workload needs. Provisioning storage solutions in AWS increases the opportunity for you to test storage offerings and determine if they are appropriate for your workload needs.

AWS service	Key characteristics	Common use cases
Amazon S3	99.999999999% durability, unlimited growth, accessible from anywhere, several cost models based on access and resiliency	Cloud-native application data, data archiving, and backups, analytics, data lakes, static website hosting, IoT data
Amazon S3 Glacier	Seconds to hours latency, unlimited growth, lowest cost, long-term storage	Data archiving, media archives, long-term backup retention.
Amazon EBS	Storage size requires management and monitoring, low latency, persistent storage, 99.8% to 99.9% durability, most volume types are accessible only from one EC2 instance.	COTS applications, I/O intensive applications, relational and NoSQL databases, backup and recovery
EC2 Instance Store	Pre-determined storage size, lowest latency, not persisted, accessible only from one EC2 instance	COTS applications, I/O intensive applications, in-memory data store
Amazon EFS	99.999999999% durability, unlimited growth, accessible by multiple compute services	Modernized applications sharing files across multiple compute services, file storage for scaling content management systems
Amazon FSx	Supports four file systems (NetApp, OpenZFS, Windows File Server, and Amazon FSx for Lustre), storage available different per file system, accessible by multiple compute services	Cloud native workloads, private cloud bursting, migrated workloads that require a specific file system, VMC, ERP systems, on-premises file storage and backups
Snow family	Portable devices, 256-bit encryption, NFS endpoint, on-board computing, TBs of storage	Migrating data to the cloud, storage, and computing in extreme on-premises conditions, disaster recovery, remote data collection

AWS service	Key characteristics	Common use cases
AWS Storage Gateway	Provides low-latency on-premises access to cloud-backed storage, fully managed on-premises cache	On-premises data to cloud migrations, populate cloud data lakes from on-premises sources, modernized file sharing.

Implementation steps:

1. Use benchmarking or load tests to collect the key characteristics of your storage needs. Key characteristics include:
   a. Shareable (what components access this storage)
   b. Growth rate
   c. Throughput
   d. Latency
   e. I/O size
   f. Durability
   g. Access patterns (reads vs writes, frequency, spikey, or consistent)
2. Identify the type of storage solution that supports your storage characteristics.
   a. Amazon S3 is an object storage service with unlimited scalability, high availability, and multiple options for accessibility. Transferring and accessing objects in and out of Amazon S3 can use a service, such as Transfer Acceleration or Access Points to support your location, security needs, and access patterns. Use the Amazon S3 performance guidelines to help you optimize your Amazon S3 configuration to meet your workload performance needs.
   b. Amazon S3 Glacier is a storage class of Amazon S3 built for data archiving. You can choose from three archiving solutions ranging from millisecond access to 5-12 hour access with different cost and security options. Amazon S3 Glacier can help you meet performance requirements by implementing a data lifecycle that supports your business requirements and data characteristics.
   c. Amazon Elastic Block Store (Amazon EBS) is a high-performance block storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). You can choose from SSD- or HDD-based solutions with different characteristics that prioritize IOPS or throughput. EBS volumes are well suited for high-performance workloads, primary storage for file systems, databases, or applications that can only access attached stage systems.
   d. Amazon EC2 Instance Store is similar to Amazon EBS as it attaches to an Amazon EC2 instance however, the Instance Store is only temporary storage that should ideally be used as a buffer, cache, or other temporary content. You cannot detach an Instance Store and all data is lost if the instance shuts down. Instance Stores can be used for high I/O performance and low latency use cases where data doesn't need to persist.
   e. Amazon Elastic File System (Amazon EFS) is a mountable file system that can be accessed by multiple types of compute solutions. Amazon EFS automatically grows and shrinks storage and is performance-optimized to deliver consistent low latencies. EFS has two performance configuration modes: General Purpose and Max I/O. General Purpose has a sub-millisecond read latency and a single-digit millisecond write latency. The Max I/O feature can support thousands of compute instance requiring a shared file system. Amazon EFS supports two throughput modes: Bursting and Provisioned. A workload that experiences a spikey access pattern will benefit from the bursting throughput mode while a workload that is consistently high would be performant with a provisioned throughput mode.
   f. Amazon FSx is built on the latest AWS compute solutions to support four commonly used file systems: NetApp ONTAP, OpenZFS, Windows File Server, and Lustre. Amazon FSx latency, throughput, and IOPS vary per file system and should be considered when selecting the right file system for your workload needs.

   g. AWS Snow Family are storage and compute devices that support online and offline data migration to the cloud and data storage and computing on premises. AWS Snow devices support collecting large amounts of on-premises data, processing of that data and moving that data to the cloud. There are several documented performance best practices when it comes to the number of files, file sizes, and compression.
   h. AWS Storage Gateway provides on-premises applications access to cloud-based storage. AWS Storage Gateway supports multiple cloud storage services including Amazon S3, Amazon S3 Glacier, Amazon FSx, and Amazon EBS. It supports a number of protocols such as iSCSI, SMB, and NFS. It provides low-latency performance by caching frequently accessed data on premises and only sends changed data and compressed data to AWS.

3. After you have experimented with your new storage solution and identified the optimal configuration, plan your migration and validate your performance metrics. This is a continual process, and should be reevaluated when key characteristics change or available services or options change.

Level of effort for the implementation plan: If a workload is moving from one storage solution to another, there could be a *moderate* level of effort involved in refactoring the application.

Resources

Related documents:

- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon FSx for NetApp ONTAP performance
- Amazon FSx for OpenZFS performance
- Amazon S3 Glacier: Amazon S3 Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- AWS Snow Family
- EBS I/O Characteristics

Related videos:

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

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples
- Amazon FSx for Lustre Container Storage Interface (CSI) Driver

PERF03-BP02 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 allows 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 four file systems to choose from: Amazon FSx for Windows File Server for enterprise workloads, Amazon FSx for Lustre for high-performance workloads, Amazon FSx for NetApp ONTAP for NetApps popular ONTAP file system, and Amazon FSx for OpenZFS for Linux-based file servers. 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.

Common anti-patterns:

- You only use one storage type, such as Amazon EBS, for all workloads.
- You use Provisioned IOPS for all workloads without real-world testing against all storage tiers.
- You assume that all workloads have similar storage access performance requirements.

Benefits of establishing this best practice: Evaluating all storage service options can reduce the cost of infrastructure and the effort required to maintain your workloads. It can potentially accelerate your time to market for deploying new services and features.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Determine storage characteristics: When you evaluate a storage solution, determine which storage characteristics you require, such as ability to share, file size, cache size, latency, throughput, and persistence of data. Then match your requirements to the AWS service that best fits your needs.

Resources

Related documents:

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

Related videos:

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

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples

PERF03-BP03 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 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 using 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 size, 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

Common anti-patterns:

- You assume that storage performance is adequate if customers are not complaining.
- You only use one tier of storage, assuming all workloads fit within that tier.

Benefits of establishing this best practice: You need a unified operational view, real-time granular data, and historical reference to optimize performance and resource utilization. You can create automatic dashboards and data with one-second granularity to perform metric math on your data and derive operational and utilization insights for your storage needs.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Optimize your storage usage and access patterns: Choose storage systems based on your workload's access patterns and the characteristics of the available storage options. Determine the best place to store data that will allow you to meet your requirements while reducing overhead. Use performance optimizations and access patterns when configuring and interacting with data based on the characteristics of your storage (for example, striping volumes or partitioning data).

Select appropriate metrics for storage options: Ensure that you select the appropriate storage metrics for the workload. Each storage option offers various metrics to track how your workload performs over time. Ensure that you are measuring against any storage burst metrics (for example, monitoring burst credits for Amazon EFS). For storage systems that are fixed sized, such as Amazon Elastic Block Store or Amazon FSx, ensure that you are monitoring the amount of storage used versus the overall storage size. Create automation when possible to increase the storage size when reaching a threshold.

Monitor metrics: 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 third-party solutions to set alarms that indicate when thresholds are breached.

Resources

Related documents:

- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon Glacier: Amazon Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- EBS I/O Characteristics
- Monitoring and understanding Amazon EBS performance using Amazon CloudWatch

Related videos:

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

Related examples:

- Amazon EFS CSI Driver
- Amazon EBS CSI Driver
- Amazon EFS Utilities
- Amazon EBS Autoscale
- Amazon S3 Examples

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 allow different features to improve performance. Selecting the wrong database solution and features for a system can lead to lower performance efficiency.

Best practices

- PERF04-BP01 Understand data characteristics (p. 31)
- PERF04-BP02 Evaluate the available options (p. 36)
- PERF04-BP03 Collect and record database performance metrics (p. 41)
- PERF04-BP04 Choose data storage based on access patterns (p. 43)
- PERF04-BP05 Optimize data storage based on access patterns and metrics (p. 46)

PERF04-BP01 Understand data characteristics

Choose your data management solutions to optimally match the characteristics, access patterns, and requirements of your workload datasets. When selecting and implementing a data management solution, you must ensure that the querying, scaling, and storage characteristics support the workload data requirements. Learn how various database options match your data models, and which configuration options are best for your use-case.

AWS provides numerous database engines including relational, key-value, document, in-memory, graph, time series, and ledger databases. Each data management solution has options and configurations available to you to support your use-cases and data models. Your workload might be able to use several different database solutions, based on the data characteristics. By selecting the best database solutions to a specific problem, you can break away from monolithic databases, with the one-size-fits-all approach that is restrictive and focus on managing data to meet your customer's need.

Desired outcome: The workload data characteristics are documented with enough detail to facilitate selection and configuration of supporting database solutions, and provide insight into potential alternatives.

Common anti-patterns:

- Not considering ways to segment large datasets into smaller collections of data that have similar characteristics, resulting in missing opportunities to use more purpose-built databases that better match data and growth characteristics.

- Not identifying the data access patterns up front, which leads to costly and complex rework later.
- Limiting growth by using data storage strategies that don't scale as quickly as is needed
- Choosing one database type and vendor for all workloads.
- Sticking to one database solution because there is internal experience and knowledge of one particular type of database solution.
- Keeping a database solution because it worked well in an on-premises environment.

Benefits of establishing this best practice: Be familiar with all of the AWS database solutions so that you can determine the correct database solution for your various workloads. After you select the appropriate database solution for your workload, you can quickly experiment on each of those database offerings to determine if they continue to meet your workload needs.

Level of risk exposed if this best practice is not established: High

- Potential cost savings may not be identified.
- Data may not be secured to the level required.
- Data access and storage performance may not be optimal.

Implementation guidance

Define the data characteristics and access patterns of your workload. Review all available database solutions to identify which solution supports your data requirements. Within a given workload, multiple databases may be selected. Evaluate each service or group of services and assess them individually. If potential alternative data management solutions are identified for part or all of the data, experiment with alternative implementations that might unlock cost, security, performance, and reliability benefits. Update existing documentation, should a new data management approach be adopted.

Type	AWS Services	Key Characteristics	Common use-cases
Relational	Amazon RDS, Amazon Aurora	Referential integrity, ACID transactions, schema on write	ERP, CRM, Commercial off-the-shelf software
Key Value	Amazon DynamoDB	High throughput, low latency, near-infinite scalability	Shopping carts (ecommerce), product catalogs, chat applications
Document	Amazon DocumentDB	Store JSON documents and query on any attribute	Content Management (CMS), customer profiles, mobile applications
In Memory	Amazon ElastiCache, Amazon MemoryDB	Microsecond latency	Caching, game leaderboards
Graph	Amazon Neptune	Highly relational data where the relationships between data have meaning	Social networks, personalization engines, fraud detection
Time Series	Amazon Timestream	Data where the primary dimension is time	DevOps, IoT, Monitoring

Type	AWS Services	Key Characteristics	Common use-cases
Wide column	Amazon Keyspaces	Cassandra workloads.	Industrial equipment maintenance, route optimization
Ledger	Amazon QLDB	Immutable and cryptographically verifiable ledger of changes	Systems of record, healthcare, supply chains, financial institutions

Implementation steps

1. How is the data structured? (for example, unstructured, key-value, semi-structured, relational)
   a. If the data is unstructured, consider an object-store such as Amazon S3 or a NoSQL database such as Amazon DocumentDB.
   b. For key-value data, consider DynamoDB, ElastiCache for Redis or MemoryDB.
   c. If the data has a relational structure, what level of referential integrity is required?
      i. For foreign key constraints, relational databases such as Amazon RDS and Aurora can provide this level of integrity.
      ii. Typically, within a NoSQL data-model, you would de-normalize your data into a single document or collection of documents to be retrieved in a single request rather than joining across documents or tables.
2. Is ACID (atomicity, consistency, isolation, durability) compliance required?
   a. If the ACID properties associated with relational databases are required, consider a relational database such as Amazon RDS and Aurora.
3. What consistency model is required?
   a. If your application can tolerate eventual consistency, consider a NoSQL implementation. Review the other characteristics to help choose which NoSQL database is most appropriate.
   b. If strong consistency is required, you can use strongly consistent reads with DynamoDB or a relational database such as Amazon RDS.
4. What query and result formats must be supported? (for example, SQL, CSV, Parque, Avro, JSON, etc.)
5. What data types, field sizes and overall quantities are present? (for example, text, numeric, spatial, time-series calculated, binary or blob, document)
6. How will the storage requirements change over time? How does this impact scalability?
   a. Serverless databases such as DynamoDB and Amazon Quantum Ledger Database will scale dynamically up to near-unlimited storage.
   b. Relational databases have upper bounds on provisioned storage, and often must be horizontally partitioned via mechanisms such as sharding once they reach these limits.
7. What is the proportion of read queries in relation to write queries? Would caching be likely to improve performance?
   a. Read-heavy workloads can benefit from a caching layer, this could be ElastiCache or DAX if the database is DynamoDB.
   b. Reads can also be offloaded to read replicas with relational databases such as Amazon RDS.
8. Does storage and modification (OLTP - Online Transaction Processing) or retrieval and reporting (OLAP - Online Analytical Processing) have a higher priority?
   a. For high-throughput transactional processing, consider a NoSQL database such as DynamoDB or Amazon DocumentDB.
   b. For analytical queries, consider a columnar database such as Amazon Redshift or exporting the data to Amazon S3 and performing analytics using Athena or QuickSight.
9. How sensitive is this data and what level of protection and encryption does it require?

   a. All Amazon RDS and Aurora engines support data encryption at rest using AWS KMS. Microsoft SQL Server and Oracle also support native Transparent Data Encryption (TDE) when using Amazon RDS.
   b. For DynamoDB, you can use fine-grained access control with IAM to control who has access to what data at the key level.

10. What level of durability does the data require?
   a. Aurora automatically replicates your data across three Availability Zones within a Region, meaning your data is highly durable with less chance of data loss.
   b. DynamoDB is automatically replicated across multiple Availability Zones, providing high availability and data durability.
   c. Amazon S3 provides 11 9s of durability. Many database services such as Amazon RDS and DynamoDB support exporting data to Amazon S3 for long-term retention and archival.

11. Do Recovery Time Objective (RTO) or Recovery Point Objectives (RPO) requirements influence the solution?
   a. Amazon RDS, Aurora, DynamoDB, Amazon DocumentDB, and Neptune all support point in time recovery and on-demand backup and restore.
   b. For high availability requirements, DynamoDB tables can be replicated globally using the Global Tables feature and Aurora clusters can be replicated across multiple Regions using the Global database feature. Additionally, S3 buckets can be replicated across AWS Regions using cross-region replication.

12. Is there a desire to move away from commercial database engines / licensing costs?
   a. Consider open-source engines such as PostgreSQL and MySQL on Amazon RDS or Aurora
   b. Leverage AWS DMS and AWS SCT to perform migrations from commercial database engines to open-source

13. What is the operational expectation for the database? Is moving to managed services a primary concern?
   a. Leveraging Amazon RDS instead of Amazon EC2, and DynamoDB or Amazon DocumentDB instead of self-hosting a NoSQL database can reduce operational overhead.

14. How is the database currently accessed? Is it only application access, or are there Business Intelligence (BI) users and other connected off-the-shelf applications?
   a. If you have dependencies on external tooling then you may have to maintain compatibility with the databases they support. Amazon RDS is fully compatible with the difference engine versions that it supports including Microsoft SQL Server, Oracle, MySQL, and PostgreSQL.

15. The following is a list of potential data management services, and where these can best be used:
   a. 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 ecommerce 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.
   b. 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, ecommerce 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.
   c. In-memory databases are used for applications that require real-time access to data, lowest latency and highest throughput. By storing data directly in memory, these databases deliver microsecond latency to applications where 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. In case the applications also higher durability requirements, Amazon

MemoryDB for Redis offers this in combination being a durable, in-memory database service for ultra-fast performance.

d. A document database is designed to store semistructured 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.
e. 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 Keyspaces (for Apache Cassandra) is a wide column scalable, highly available, and managed Apache Cassandra–compatible database service.
f. 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.
g. 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.
h. 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 (Amazon 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.

Level of effort for the implementation plan: If a workload is moving from one database solution to another, there could be a *high* level of effort involved in refactoring the data and application.

Resources

Related documents:

- Cloud Databases with AWS
- AWS Database Caching
- Amazon DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices
- Choose between EC2 and Amazon RDS
- Best Practices for Implementing Amazon ElastiCache

Related 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)

Related examples:

- Optimize Data Pattern using Amazon Redshift Data Sharing
- Database Migrations
- MS SQL Server - AWS Database Migration Service (DMS) Replication Demo
- Database Modernization Hands On Workshop
- Amazon Neptune Samples

PERF04-BP02 Evaluate the available options

Understand the available database options and how it can optimize your performance before you select your data management solution. Use load testing to identify database metrics that matter for your workload. While you explore the database options, take into consideration various aspects such as the parameter groups, storage options, memory, compute, read replica, eventual consistency, connection pooling, and caching options. Experiment with these various configuration options to improve the metrics.

Desired outcome: A workload could have one or more database solutions used based on data types. The database functionality and benefits optimally match the data characteristics, access patterns, and workload requirements. To optimize your database performance and cost, you must evaluate the data access patterns to determine the appropriate database options. Evaluate the acceptable query times to ensure that the selected database options can meet the requirements.

Common anti-patterns:

- Not identifying the data access patterns.
- Not being aware of the configuration options of your chosen data management solution.
- Relying solely on increasing the instance size without looking at other available configuration options.
- Not testing the scaling characteristics of the chosen solution.

Benefits of establishing this best practice: By exploring and experimenting with the database options you may be able to reduce the cost of infrastructure, improve performance and scalability and lower the effort required to maintain your workloads.

Level of risk exposed if this best practice is not established: High

- Having to optimize for a *one size fits all* database means making unnecessary compromises.
- Higher costs as a result of not configuring the database solution to match the traffic patterns.
- Operational issues may emerge from scaling issues.
- Data may not be secured to the level required.

Implementation guidance

Understand your workload data characteristics so that you can configure your database options. Run load tests to identify your key performance metrics and bottlenecks. Use these characteristics and metrics to evaluate database options and experiment with different configurations.

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoDI	Amazon Document	Amazon ElastiCach	Amazon Neptune	Amazon Timestrea	Amazon Keyspaces	Amazon QLDB
Scaling Compute	Increase instance size, Aurora Serverless instances autoscale in response to changes in load	Automatic read/ write scaling with on-demand capacity mode or automatic scaling of provisioned read/ write capacity in provisioned capacity mode	Increase instance size	Increase instance size, add nodes to cluster	Increase instance size	Automatically scales to adjust capacity	Automatic read/ write scaling with on-demand capacity mode or automatic scaling of provisioned read/ write capacity in provisioned capacity mode	Automatically scales to adjust capacity
Scaling-out reads	All engines support read replicas. Aurora supports automatic scaling of read replica instances	Increase provisioned read capacity units	Read replicas	Read replicas	Read replicas. Supports automatic scaling of read replica instances	Automatically scales	Increase provisioned read capacity units	Automatically scales up to documented concurrency limits
Scaling-out writes	Increasing instance size, batching writes in the application or adding a queue in front of the database. Horizontal scaling via application-level sharding across	Increase provisioned write capacity units. Ensuring optimal partition key to prevent partition level write throttling	Increasing primary instance size	Using Redis in cluster mode to distribute writes across shards	Increasing instance size	Write requests may be throttled while scaling. If you encounter throttling exceptions, continue to send data at the same (or higher) throughput to automatically scale.	Increase provisioned write capacity units. Ensuring optimal partition key to prevent partition level write throttling	Automatically scales up to documented concurrency limits

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoDI	Amazon Document	Amazon ElastiCach	Amazon Neptune	Amazon Timestrea	Amazon Keyspaces	Amazon QLDB
	multiple instances					Batch writes to reduce concurrent write requests		
Engine configuratio	Parameter groups	Not applicable	Parameter groups	Parameter groups	Parameter groups	Not applicable	Not applicable	Not applicable
Caching	In-memory caching, configurable via parameter groups. Pair with a dedicated cache such as ElastiCache for Redis to offload requests for commonly accessed items	DAX (DAX) fully managed cache available	In-memory caching. Optionally, pair with a dedicated cache such as ElastiCache for Redis to offload requests for commonly accessed items	Primary function is caching	Use the query results cache to cache the result of a read-only query	Timestream has two storage tiers; one of these is a high-performance in-memory tier	Deploy a separate dedicated cache such as ElastiCache for Redis to offload requests for commonly accessed items	Not applicable

AWS Services	Amazon RDS, Amazon Aurora	Amazon DynamoDB	Amazon DocumentDB	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon Keyspaces	Amazon QLDB
High availability / disaster recovery	Recommended configuration for production workloads is to run a standby instance in a second Availability Zone to provide resiliency within a Region. For resiliency across Regions, Aurora Global Database can be used	Highly available within a Region. Tables can be replicated across Regions using DynamoDB global tables	Create multiple instances across Availability Zones for availability. Snapshots can be shared across Regions and clusters can be replicated using DMS to provide Cross-Region Replication / disaster recovery	Recommended configuration for production clusters is to create at least one node in a secondary Availability Zone. ElastiCache Global Datastore can be used to replicate clusters across Regions. /	Read replicas in other Availability Zones serve as failover targets. Snapshots can be shared across Region and clusters can be replicated using Neptune streams to replicate data between two clusters in two different Regions.	Highly available within a Region. cross-Region replication requires custom application development using the Timestream SDK	Highly available within a Region. Cross-Region Replication requires custom application logic or third-party tools	Highly available within a Region. To replicate across Regions, export the contents of the Amazon QLDB journal to a S3 bucket and configure the bucket for Cross-Region Replication.

Implementation steps

1. What configuration options are available for the selected databases?
   a. Parameter Groups for Amazon RDS and Aurora allow you to adjust common database engine level settings such as the memory allocated for the cache or adjusting the time zone of the database
   b. For provisioned database services such as Amazon RDS, Aurora, Neptune, Amazon DocumentDB and those deployed on Amazon EC2 you can change the instance type, provisioned storage and add read replicas.
   c. DynamoDB allows you to specify two capacity modes: on-demand and provisioned. To account for differing workloads, you can change between these modes and increase the allocated capacity in provisioned mode at any time.
2. Is the workload read or write heavy?
   a. What solutions are available for offloading reads (read replicas, caching, etc.)?
      i. For DynamoDB tables, you can offload reads using DAX for caching.
      ii. For relational databases, you can create an ElastiCache for Redis cluster and configure your application to read from the cache first, falling back to the database if the requested item is not present.

iii. Relational databases such as Amazon RDS and Aurora, and provisioned NoSQL databases such as Neptune and Amazon DocumentDB all support adding read replicas to offload the read portions of the workload.

iv. Serverless databases such as DynamoDB will scale automatically. Ensure that you have enough read capacity units (RCU) provisioned to handle the workload.

b. What solutions are available for scaling writes (partition key sharding, introducing a queue, etc.)?

i. For relational databases, you can increase the size of the instance to accommodate an increased workload or increase the provisioned IOPs to allow for an increased throughput to the underlying storage.

- You can also introduce a queue in front of your database rather than writing directly to the database. This pattern allows you to decouple the ingestion from the database and control the flow-rate so the database does not get overwhelmed.
- Batching your write requests rather than creating many short-lived transactions can help improve throughput in high-write volume relational databases.

ii. Serverless databases like DynamoDB can scale the write throughput automatically or by adjusting the provisioned write capacity units (WCU) depending on the capacity mode.

- You can still run into issues with *hot* partitions though, when you reach the throughput limits for a given partition key. This can be mitigated by choosing a more evenly distributed partition key or by write-sharding the partition key.

3. What are the current or expected peak transactions per second (TPS)? Test using this volume of traffic and this volume +X% to understand the scaling characteristics.
   a. Native tools such as pg_bench for PostgreSQL can be used to stress-test the database and understand the bottlenecks and scaling characteristics.
   b. Production-like traffic should be captured so that it can be replayed to simulate real-world conditions in addition to synthetic workloads.
4. If using serverless or elastically scalable compute, test the impact of scaling this on the database. If appropriate, introduce connection management or pooling to lower impact on the database.
   a. RDS Proxy can be used with Amazon RDS and Aurora to manage connections to the database.
   b. Serverless databases such as DynamoDB do not have connections associated with them, but consider the provisioned capacity and automatic scaling policies to deal with spikes in load.
5. Is the load predictable, are there spikes in load and periods of inactivity?
   a. If there are periods of inactivity consider scaling down the provisioned capacity or instance size during these times. Aurora Serverless V2 will automatically scale up and down based on load.
   b. For non-production instances, consider pausing or stopping these during non-work hours.
6. Do you need to segment and break apart your data models based on access patterns and data characteristics?
   a. Consider using AWS DMS or AWS SCT to move your data to other services.

Level of effort for the implementation plan:

To establish this best practice, you must be aware of your current data characteristics and metrics. Gathering those metrics, establishing a baseline and then using those metrics to identify the ideal database configuration options is a *low* to *moderate* level of effort. This is best validated by load tests and experimentation.

Resources

Related documents:

- Cloud Databases with AWS
- AWS Database Caching

- Amazon DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices

Related 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)

Related examples:

- Amazon DynamoDB Examples
- AWS Database migration samples
- Database Modernization Workshop
- Working with parameters on your Amazon RDS for Postgress DB

PERF04-BP03 Collect and record database performance metrics

To understand how your data management systems are performing, it is important to track relevant metrics. These metrics will help you to optimize your data management resources, to ensure that your workload requirements are met, and that you have a clear overview on how the workload performs. Use tools, libraries, and systems that record performance measurements related to database performance.

There are metrics that are related to the system on which the database is being hosted (for example, CPU, storage, memory, IOPS), and there are metrics for accessing the data itself (for example, transactions per second, queries rates, response times, errors). These metrics should be readily accessible for any support or operational staff, and have sufficient historical record to be able to identify trends, anomalies, and bottlenecks.

Desired outcome: To monitor the performance of your database workloads, you must record multiple performance metrics over a period of time. This allows you to detect anomalies as well as measure performance against business metrics to ensure you are meeting your workload needs.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools used by your team and don't have a comprehensive picture of your workload.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.
- You only monitor system level metrics, not capturing data access or usage metrics.

Benefits of establishing this best practice: Establishing a performance baseline helps in understanding normal behavior and requirements of workloads. Abnormal patterns can be identified and debugged faster improving performance and reliability of the database. Database capacity can be configured to ensure optimal cost without compromising performance.

Level of risk exposed if this best practice is not established: High

- Inability to differentiate out of normal vs. normal performance level will create difficulties in issue identification, and decision making.
- Potential cost savings may not be identified.
- Growth patterns will not be identified which might result in reliability or performance degradation.

Implementation guidance

Identify, collect, aggregate, and correlate database-related metrics. Metrics should include both the underlying system that is supporting the database and the database metrics. The underlying system metrics might include CPU utilization, memory, available disk storage, disk I/O, and network inbound and outbound metrics while the database metrics might include transactions per second, top queries, average queries rates, response times, index usage, table locks, query timeouts, and number of connections open. This data is crucial to understand how the workload is performing and how the database solution is used. Use these metrics as part of a data-driven approach to tune and optimize your workload's resources.

Implementation steps:

1. Which database metrics are important to track?
    a. Monitoring metrics for Amazon RDS
    b. Monitoring with Performance Insights
    c. Enhanced monitoring
    d. DynamoDB metrics
    e. Monitoring DynamoDB DAX
    f. Monitoring MemoryDB
    g. Monitoring Amazon Redshift
    h. Timeseries metrics and dimensions
    i. Cluster level metrics for Aurora
    j. Monitoring Amazon Keyspaces
    k. Monitoring Amazon Neptune
2. Would the database monitoring benefit from a machine learning solution that detects operational anomalies performance issues?
    a. Amazon DevOps Guru for Amazon RDS provides visibility into performance issues and makes recommendations for corrective actions.
3. Do you need application level details about SQL usage?
    a. AWS X-Ray can be instrumented into the application to gain insights and encapsulate all the data points for single query.
4. Do you currently have an approved logging and monitoring solution?
    a. 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 third-party solutions to set alarms that indicate when thresholds are breached.
5. You identified and configured your data retention policies to match my security and operational goals?
    a. Default data retention for CloudWatch metrics

b. Default data retention for CloudWatch Logs

Level of effort for the implementation plan: There is a *medium* level of effort to identify, track, collect, aggregate, and correlate metrics from all database resources.

Resources

Related documents:

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

Related 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)

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS Monitoring Workshop

PERF04-BP04 Choose data storage based on access patterns

Use the access patterns of the workload and requirements of the applications to decide on optimal data services and technologies to use.

Desired outcome: Data storage has been selected based on identified and documented data access patterns. This might include the most common read, write, and delete queries, the need for as necessary calculations and aggregations, complexity of the data, data interdependency, and the required consistency needs.

Common anti-patterns:

- You only select one database engine to simplify operations management.
- You assume that data access patterns will stay consistent over time.
- You implement complex transactions, rollback, and consistency logic in the application.
- The database is configured to support a potential high traffic burst, which results in the database resources remaining idle most of the time.
- Using a shared database for transactional and analytical uses.

Benefits of establishing this best practice: Selecting and optimizing your data storage based on access patterns will help decrease development complexity and optimize your performance opportunities. Understanding when to use read replicas, global tables, data partitioning, and caching will help you decrease operational overhead and scale based on your workload needs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Identify and evaluate your data access pattern to select the correct storage configuration. Each database solution has options to configure and optimize your storage solution. Use the collected metrics and logs and experiment with options to find the optimal configuration. Use the following table to review storage options per database service.

AWS Services	Amazon RDS	Amazon Aurora	Amazon DynamoD	Amazon Documen	Amazon ElastiCac	Amazon Neptune	Amazon Timestrea	Amazon Keyspace	Amazon QLDB
Scaling Storage	Storage can be scaled up manually or configured to scale automatically to a maximum of 64 TiB based on engine types. Provisioned storage cannot be decreased.	Storage scales automatically up to maximum of 128 TiB and decreases when data is removed. Maximum storage size also depends upon specific Aurora MySQL or Aurora Postgres engine versions.	Storage automatically scales. Tables are unconstrained in terms of size.	Storage scales automatically up to maximum of 64 TiB. Starting Amazon DocumentDB 4.0 storage can decrease by comparable amounts for data removal through dropping a collection or index. With Amazon DocumentDB 3.6 allocated space remains same and free space is reused when data volume increases.	Storage is in-memory, tied to instance type or count.	Storage scales automatically can grow up to 128 TiB (or 64 TiB in few Regions). Upon data removal from, total allocated space remains same and is reused in the future.	Organizes your time series data to optimize query processing and reduce storage costs. Retention period can be configured through in-memory and magnetic tiers.	Scales table storage up and down automatically as your application writes, updates, and deletes data.	Storage automatically scales. Tables are unconstrained in terms of size.

Implementation steps:

1. Understand the requirement of transactions, atomicity, consistency, isolation, and durability (ACID) compliance, and consistent reads. Not every database supports these and most of the NoSQL databases provide an eventual consistency model.
2. Consider the traffic patterns, latency, and access requirements for a globally distributed application in order to identify the optimal storage solution.
3. Analyze query patterns, random access patterns and one-time queries. Considerations around highly specialized query functionality for text and natural language processing, time series, and graphs must also be taken into account.
4. Identify and document the anticipated growth of the data and traffic.
    a. Amazon RDS and Aurora support storage automatic scaling up to documented limits. Beyond this, consider transitioning older data to Amazon S3 for archival, aggregating historical data for analytics or scaling horizontally using sharding.
    b. DynamoDB and Amazon S3 will scale to near limitless storage volume automatically.
    c. Amazon RDS instances and databases running on EC2 can be manually resized and EC2 instances can have new EBS volumes added at a later date for additional storage.
    d. Instance types can be changed based on changes in activity. For example, you can start with a smaller instance while you are testing, then scale the instance as you begin to receive production traffic to the service. Aurora Serverless V2 automatically scales in response to changes in load.
5. Baseline requirements around normal and peak performance (transactions per second TPS and queries per second QPS) and consistency (ACID and eventual consistency).
6. Document solution deployment aspects and the database access requirements (like global replication, Multi-AZ, read replication, and multiple write nodes).

Level of effort for the implementation plan: Low. If you do not have logs or metrics for your data management solution, you will need to complete that before identifying and documenting your data access patterns. Once your data access pattern is understood, selecting and configuring your data storage is a low level of effort.

Resources

Related documents:

- Cloud Databases with AWS
- Working with storage for Amazon RDS DB instances
- Amazon DocumentDB Storage
- AWS Database Caching
- Amazon Timestream Storage
- Storage in Amazon Keyspaces
- Amazon ElastiCache FAQs
- Amazon Neptune storage, reliability, and availability
- Amazon Aurora best practices
- Amazon DynamoDB Accelerator
- Amazon DynamoDB best practices
- Amazon RDS Storage Types
- Hardware specifications for Amazon RDS instance classes
- Aurora Storage limits

Related 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)

Related examples:

- Experiment and test with Distributed Load Testing on AWS

PERF04-BP05 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.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools.

Benefits of establishing this best practice: In order to ensure you are meeting the metrics required for the workload, you must monitor database performance metrics related to both reads and writes. You can use this data to add new optimizations for both reads and writes to the data storage layer.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Optimize data storage based on metrics and patterns: Use reported metrics to identify any underperforming areas in your workload and optimize your database components. Each database system has different performance related characteristics to evaluate, such as how data is indexed, cached, or distributed among multiple systems. Measure the impact of your optimizations.

Resources

Related documents:

- AWS Database Caching
- Amazon Athena top 10 performance tips
- Amazon Aurora best practices
- Amazon DynamoDB Accelerator
- Amazon DynamoDB best practices
- Amazon Redshift Spectrum best practices
- Amazon Redshift performance
- Cloud Databases with AWS
- Analyzing performance anomalies with DevOps Guru for RDS
- Read/Write Capacity Mode for DynamoDB

Related 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)

Related examples:

- Hands-on Labs for Amazon DynamoDB

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.

Best practices

- PERF05-BP01 Understand how networking impacts performance (p. 47)
- PERF05-BP02 Evaluate available networking features (p. 49)
- PERF05-BP03 Choose appropriately sized dedicated connectivity or VPN for hybrid workloads (p. 53)
- PERF05-BP04 Leverage load-balancing and encryption offloading (p. 55)
- PERF05-BP05 Choose network protocols to improve performance (p. 58)
- PERF05-BP06 Choose your workload's location based on network requirements (p. 60)
- PERF05-BP07 Optimize network configuration based on metrics (p. 63)

PERF05-BP01 Understand how networking impacts performance

Analyze and understand how network-related decisions impact workload performance. The network is responsible for the connectivity between application components, cloud services, edge networks and on-premises data and therefor it can highly impact workload performance. In addition to workload performance, user experience is also impacted by network latency, bandwidth, protocols, location, network congestion, jitter, throughput, and routing rules.

Desired outcome: Have a documented list of networking requirements from the workload including latency, packet size, routing rules, protocols, and supporting traffic patterns. Review the available networking solutions and identify which service meets your workload networking characteristics. Cloud-based networks can be quickly rebuilt, so evolving your network architecture over time is necessary to improve performance efficiency.

Common anti-patterns:

- All traffic flows through your existing data centers.
- You overbuild Direct Connect sessions without understanding the actual usage requirements.
- You don't consider workload characteristics and encryption overhead when defining your networking solutions.

- You use on-premises concepts and strategies for networking solutions in the cloud.

Benefits of establishing this best practice: Understanding how networking impacts workload performance will help you identify potential bottlenecks, improve user experience, increase reliability, and lower operational maintenance as the workload changes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify important network performance metrics of your workload and capture its networking characteristics. Define and document requirements as part of a data-driven approach, using benchmarking or load testing. Use this data to identify where your network solution is constrained, and examine configuration options that could improve the workload. Understand the cloud-native networking features and options available and how they can impact your workload performance based on the requirements. Each networking feature has advantages and disadvantages and can be configured to meet your workload characteristics and scale based on your needs.

Implementation steps:

1. Define and document networking performance requirements:
   a. Include metrics such as network latency, bandwidth, protocols, locations, traffic patterns (spikes and frequency), throughput, encryption, inspection, and routing rules
2. Capture your foundational networking characteristics:
   a. VPC Flow Logs
   b. AWS Transit Gateway metrics
   c. AWS PrivateLink metrics
3. Capture your application networking characteristics:
   a. Elastic Network Adaptor
   b. AWS App Mesh metrics
   c. Amazon API Gateway metrics
4. Capture your edge networking characteristics:
   a. Amazon CloudFront metrics
   b. Amazon Route 53 metrics
   c. AWS Global Accelerator metrics
5. Capture your hybrid networking characteristics:
   a. Direct Connect metrics
   b. AWS Site-to-Site VPN metrics
   c. AWS Client VPN metrics
   d. AWS Cloud WAN metrics
6. Capture your security networking characteristics:
   a. AWS Shield, WAF, and Network Firewall metrics
7. Capture end-to-end performance metrics with tracing tools:
   a. AWS X-Ray
   b. Amazon CloudWatch RUM
8. Benchmark and test network performance:
   a. Benchmark network throughput: Some factors that can affect EC2 network performance when the instances are in the same VPC. Measure the network bandwidth between EC2 Linux instances in the same VPC.
   b. Perform load tests to experiment with networking solutions and options

Level of effort for the implementation plan: There is a *medium* level of effort to document workload networking requirements, options, and available solutions.

Resources

Related documents:

- Application Load Balancer
- 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
- Network Load Balancer
- Networking Products with AWS
- Transit Gateway
- Transitioning to latency-based routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- Improve Global Network Performance for Applications
- EC2 Instances and Performance Optimization Best Practices
- Optimizing Network Performance for Amazon EC2 Instances
- Networking best practices and tips with the Well-Architected Framework
- AWS networking best practices in large-scale migrations

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP02 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, packet loss, or jitter.

Desired outcome: You have documented the inventory of components within your workload and have identified which networking configurations per component will help you meet your performance requirements. After evaluating the networking features, you have experimented and measured the performance metrics to identify how to use the features available to you.

Common anti-patterns:

- You put all your workloads into an AWS Region closest to your headquarters instead of an AWS Region close to your users.
- You fail to benchmark your workload performance and do not continually evaluate your workload performance against that benchmark.

- You do not review service configurations for performance improving options.

Benefits of establishing this best practice: Evaluating all service features and options can increase your workload performance, reduce the cost of infrastructure, decrease the effort required to maintain your workload, and increase your overall security posture. You can use the global AWS backbone to ensure that you provide the optimal networking experience for your customers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Review which network-related configuration options are available to you, and review how they could impact your workload. Performance optimization depends on understanding how these options interact with your architecture and the impact that they will have on both measured performance and user experience.

Many services are created to improve performance and others commonly offer features to optimize network performance. Services such as AWS Global Accelerator and Amazon CloudFront exist to improve performance while most other services have product features to optimize network traffic. Review service features, such as Amazon EC2 instance network capability, enhanced networking instance types, Amazon EBS-optimized instances, Amazon S3 transfer acceleration, and CloudFront, to improve your workload performance.

Implementation steps:

1. Create a list of workload components.
   a. Build, manage and monitor your organizations network using AWS Cloud WAN when building a unified global network.
   b. Monitor your global and core networks with Amazon CloudWatch metrics. Leverage CloudWatch Real-User Monitoring (RUM), which provides insights to help to identify, understand, and enhance users' digital experience.
   c. View aggregate network latency between AWS Regions and Availability Zones, as well as within each Availability Zone, using AWS Network Manager to gain insight into how your application performance relates to the performance of the underlying AWS network.
   d. Use an existing configuration management database (CMDB) tool or a service such as AWS Config to create an inventory of your workload and how it's configured.
2. If this is an existing workload, identify and document the benchmark for your performance metrics, focusing on the bottlenecks and areas to improve. Performance-related networking metrics will differ per workload based on business requirements and workload characteristics. As a start, these metrics might be important to review for your workload: bandwidth, latency, packet loss, jitter, and retransmits.
3. If this is a new workload, perform load tests to identify performance bottlenecks.
4. For the performance bottlenecks you identify, review the configuration options for your solutions to identify performance improvement opportunities.
5. If you don't know your network path or routes, use Network Access Analyzer to identify them.
6. Review your network protocols to further reduce your latency (see PERF05-BP05 Choose network protocols to improve performance (p. 58)).
7. When connection from on-premises environments to AWS is required, review available configuration options for connectivity and estimate the bandwidth and latency requirements for your hybrid workload (see PERF05-BP03 Choose appropriately sized dedicated connectivity or VPN for hybrid workloads (p. 53)).
   - If you are using an AWS Site-to-Site VPN across multiple locations to connect to an AWS Region, then use an accelerated Site-to-Site VPN connection for the opportunity to improve network performance.

- If your hybrid network design consists of IPSec VPN connection over AWS Direct Connect, consider using Private IP VPN to improve security and achieve segmentation. AWS Site-to-Site Private IP VPN is deployed on top of Transit virtual Interface (VIF).
- AWS Direct Connect SiteLink allows creating low-latency and redundant connections between your data centers worldwide by sending data over the fastest path between AWS Direct Connect locations, bypassing AWS Regions.

8. When your workload traffic is spread across multiple accounts, evaluate your network topology and services to reduce latency.
   - Evaluate your operational and performance tradeoffs between VPC Peering and AWS Transit Gateway when connecting multiple accounts. AWS Transit Gateway supports Equal Cost Multipath (ECMP) across multiple AWS Site-to-Site VPN connections in order to deliver additional bandwidth. Traffic between an Amazon VPC and AWS Transit Gateway remains on the private AWS network and is not exposed to the internet. AWS Transit Gateway simplifies how you interconnect all of your VPCs, which can span across thousands of AWS accounts and into on-premises networks. Share your AWS Transit Gateway between multiple accounts using Resource Access Manager.
9. Review your user locations and minimize the distance between your users and the workload.
   a. AWS Global Accelerator is a networking service that improves the performance of your users' traffic by up to 60% using the AWS global network infrastructure. When the internet is congested, AWS Global Accelerator optimizes the path to your application to keep packet loss, jitter, and latency consistently low. It also provides static IP addresses that simplify moving endpoints between Availability Zones or AWS Regions without needing to update your DNS configuration or change client-facing applications. Add an accelerator when creating a load balancer to improve the performance and availability of your workload taking advantages of AWS backbone.
   b. Amazon CloudFront can improve the performance of your workload content delivery and latency globally. CloudFront has over 410 globally dispersed points of presence that can cache your content and lower the latency to the end user. Using Lambda@edge to run functions that customize the content that CloudFront delivers closer to the users, reduces latency and Improves performance.
   c. Amazon Route 53 offers latency-based routing, geolocation routing, geoproximity routing, and IP-based routing options to help you improve your workload's performance for a global audience. Identify which routing option would optimize your workload performance by reviewing your workload traffic and user location when your workload is distributed globally.
10. Evaluate additional Amazon S3 features to improve storage IOPs.
   a. Amazon S3 Transfer acceleration is a feature that lets external users benefit from the networking optimizations of CloudFront to upload data to Amazon S3. This improves the ability to transfer large amounts of data from remote locations that don't have dedicated connectivity to the AWS Cloud.
   b. Amazon S3 Multi-Region Access Points replicates content to multiple Regions and simplifies the workload by providing one access point. When a Multi-Region Access Point is used, you can request or write data to Amazon S3 with the service identifying the lowest latency bucket.
11. Review your compute resource network bandwidth.
   a. Elastic Network Interfaces (ENA) used by EC2 instances, containers, and Lambda functions are limited on a per-flow basis. Review your placement groups to optimize your EC2 networking throughput. To avoid the bottleneck at the per flow-basis, design your application to use multiple flows. To monitor and get visibility into your compute related networking metrics, use CloudWatch Metrics and `ethtool`. `ethtool` is included in the ENA driver and exposes additional network-related metrics that can be published as a custom metric to CloudWatch.
   b. Newer Amazon EC2 instances can leverage enhanced networking. C7gn instances featuring new AWS Nitro Cards (Nitro V5) with enhanced networking offer the highest network bandwidth and packet rate performance across Amazon EC2 network-optimized instances.
   c. Amazon Elastic Network Adapters (ENA) provide further optimization by delivering better throughput for your instances within a cluster placement group. Elastic Network Adapter Express (ENA-X) is a new ENA feature using scalable reliable datagram (SRD) protocol that improves single flow bandwidth and lower tail latency by increasing maximum single flow bandwidth of Amazon

EC2 instances from 5 Gbps up to 25 Gbps, and it can provide up to 85% improvement in P99.9 latency for high throughput workloads. ENA-X is currently available for C6gn.16xl.

d. Elastic Fabric Adapter (EFA) is a network interface for Amazon EC2 instances that allows you to run workloads requiring high levels of internode 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.

e. Amazon EBS-optimized instances use an optimized configuration stack and provide additional, dedicated capacity to increase the 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.

Level of effort for the implementation plan:

Low to Medium. To establish this best practice, you must be aware of your current workload component options that impact network performance. Gathering the components, evaluating network improvement options, experimenting, implementing, and documenting those improvements is a *low* to *moderate* level of effort.

Resources

Related documents:

- Amazon EBS - Optimized Instances
- Application Load Balancer
- Amazon EC2 instance network bandwidth
- 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
- Network Load Balancer
- Networking Products with AWS
- AWS Transit Gateway
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs
- Building a cloud CMDB
- Scaling VPN throughput using AWS Transit Gateway

Related videos:

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

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP03 Choose appropriately sized dedicated connectivity or VPN for hybrid workloads

When a common network is required to connect on-premises and cloud resources in AWS, verify that you have adequate bandwidth to meet your performance requirements. Estimate the bandwidth and latency requirements for your hybrid workload. These numbers will drive the sizing requirements for your connectivity options.

Desired outcome: When deploying a workload that will need hybrid networking, you have multiple configuration options for connectivity, such as a dedicated connection or virtual private network (VPN). Select the appropriate connection type for each workload while verifying that you have adequate bandwidth and encryption requirements between your location and the cloud.

Common anti-patterns:

- You fail to understand or identify all workload requirements (bandwidth, latency, jitter, encryption and traffic needs).
- You don't evaluate backup or parallel connectivity options.

Benefits of establishing this best practice: Selecting and configuring appropriately sized hybrid network solutions will increase the reliability of your workload and maximize performance opportunities. By identifying workload requirements, planning ahead, and evaluating hybrid solutions you will minimize expensive physical network changes and operational overhead while increasing your time to market.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Develop a hybrid networking architecture based on your bandwidth requirements. Estimate the bandwidth and latency requirements of your hybrid applications. Consider appropriate connectivity option between using a dedicated network connection or internet-based VPN.

Dedicated connection establishes network connection over private lines. It is suitable when you need high-bandwidth, low-latency while achieving consistent performance. VPN connection establishes secure connection over the internet. It is suitable when you need encrypted connection using an existing internet connection.

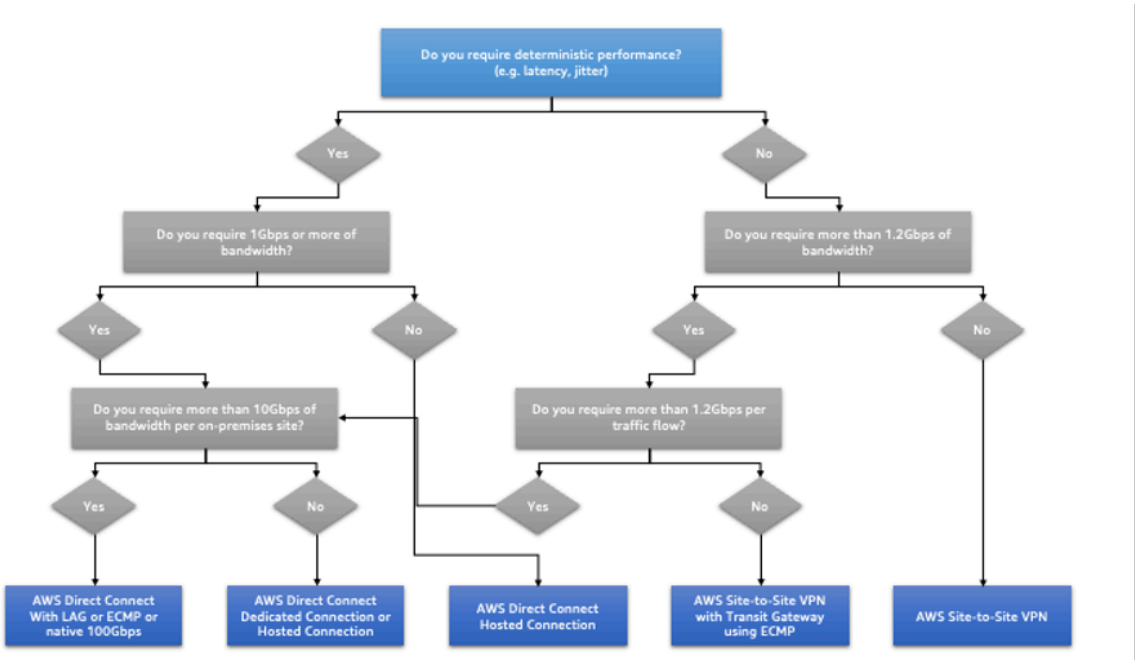
Based on your bandwidth requirements, a single VPN or dedicated connection might not be enough, and you must architect a hybrid setup to permit traffic load balancing across multiple connections.

Implementation steps

1. Estimate the bandwidth and latency requirements of your hybrid applications.
   a. For existing apps that are moving to AWS, leverage the data from your internal network monitoring systems.
   b. For new apps or existing apps for which you don't have monitoring data, consult with the product owners to derive adequate performance metrics and provide a good user experience.
2. Select dedicated connection or VPN as your connectivity option. Based on all workload requirements (encryption, bandwidth and traffic needs), you can either choose AWS Direct Connect or AWS Site-to-Site VPN (or both). The following diagram will help you choose the appropriate connection type.
   a. If you consider dedicated connection, AWS Direct Connect may be required, which offers more predictable and consistent performance due to its private network connectivity. AWS Direct Connect provides dedicated connectivity to the AWS environment, from 50 Mbps up to 100 Gbps, using either dedicated connection or hosted connection. This gives you managed and controlled latency and provisioned bandwidth so your workload can connect efficiently to other environments.

Using an AWS Direct Connect partners, you can have end-to-end connectivity from multiple environments, providing an extended network with consistent performance. AWS offers scaling direct connect connection bandwidth using either native 100 Gbps, Link Aggregation Group (LAG), or BGP Equal-cost multipath (ECMP).

b. If you consider VPN connection, an AWS managed VPN is the recommended option. The AWS Site-to-Site VPN provides a managed VPN service supporting Internet Protocol security (IPsec) protocol. When a VPN connection is created, each VPN connection includes two tunnels for high availability. 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 allows you to scale beyond the 1.25Gbps IPsec VPN throughput limit by allowing equal cost multi-path (ECMP) routing support over multiple VPN tunnels.



Deterministic performance flowchart.

Level of effort for the implementation plan: High. There is significant effort in evaluating workload needs for hybrid networks and implementing hybrid networking solutions.

Resources

Related documents:

- Network Load Balancer
- Networking Products with AWS
- AWS Transit Gateway
- Transitioning to latency-based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs
- AWS Site-to-Site VPN
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- AWS Direct Connect

- Client VPN

Related videos:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- AWS Global Accelerator
- AWS Direct Connect
- Transit Gateway Connect
- VPN Solutions
- Security with VPN Solutions

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP04 Leverage load-balancing and encryption offloading

Use load balancers to achieve optimal performance efficiency of your target resources and improve the responsiveness of your system.

Desired outcome: Reduce the number of computing resources to serve your traffic. Avoid resource consumption imbalance in your targets. Offload compute-intensive tasks to the Load Balancer. Leverage cloud elasticity and flexibility to improve performance and optimize your architecture.

Common anti-patterns:

- You don't consider your workload requirements when choosing the load balancer type.
- You don't leverage the load balancer features for performance optimization.
- The workload is exposed directly to the Internet without a load balancer.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Load balancers act as the entry point for your workload and from there they distribute the traffic to your back-end targets, such as compute instances or containers. Choosing the right load balancer type is the first step to optimize your architecture.

Start by listing your workload characteristics, such as protocol (like TCP, HTTP, TLS, or WebSockets), target type (like instances, containers, or serverless), application requirements (like long running connections, user authentication, or stickiness), and placement (like Region, Local Zone, Outpost, or zonal isolation).

After choosing the right load balancer, you can start leveraging its features to reduce the amount of effort your back-end has to do to serve the traffic.

For example, using both Application Load Balancer (ALB) and Network Load Balancer (NLB), you can perform SSL/TLS encryption offloading, which is an opportunity to avoid the CPU-intensive TLS handshake from being completed by your targets and also to improve certificate management.

When you configure SSL/TLS offloading in your load balancer, it becomes responsible for the encryption of the traffic from and to clients while delivering the traffic unencrypted to your back-ends, freeing up your back-end resources and improving the response time for the clients.

Application Load Balancer can also serve HTTP2 traffic without needing to support it on your targets. This simple decision can improve your application response time, as HTTP2 uses TCP connections more efficiently.

Load balancers can also be used to make your architecture more flexible by distributing traffic across different back-end types such as containers and serverless. For example, Application Load Balancer can be configured with listener rules that forward traffic to different target groups based on the request parameters such as header, method or pattern.

Your workload latency requirements should also be considered when defining the architecture. As an example, if you have a latency-sensitive application, you may decide to use Network Load Balancer, which offers extremely low latencies. Alternatively, you may decide to bring your workload closer to your customers by leveraging Application Load Balancer in AWS Local Zones or even AWS Outposts.

Another consideration for latency-sensitive workloads is cross-zone load balancing. With cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all allowed Availability Zones. This improves availability, although it can add a single digit millisecond to the roundtrip latency.

Lastly, both ALB and NLB offer monitoring resources such as logs and metrics. Properly setting up monitoring can help with gathering performance insights of your application. For example, you can use ALB access logs to find which requests are taking longer to be answered or which back-end targets are causing performance issues.

Implementation steps

1. Choose the right load balancer for your workload.
   a. Use Application Load Balancer for HTTP/HTTPS workloads.
   b. Use Network Load Balancer for non-HTTP workloads that run on TCP or UDP.
   c. Use a combination of both (ALB as a target of NLB) if you want to leverage features of both products. For example, you can do this if you want to use the static IPs of NLB together with HTTP header based routing from ALB, or if you want to expose your HTTP workload to an AWS PrivateLink.
   d. For a full comparison of load balancers, see ELB product comparison.
2. Use SSL/TLS offloading.
   a. Configure HTTPS/TLS listeners with both Application Load Balancer and Network Load Balancer integrated with AWS Certificate Manager.
   b. Note that some workloads may require end-to-end encryption for compliance reasons. In this case, it is a requirement to allow encryption at the targets.
   c. For security best practices, see SEC09-BP02 Enforce encryption in transit.
3. Select the right routing algorithm.
   a. The routing algorithm can make a difference in how well-used your back-end targets are and therefore how they impact performance. For example, ALB provides two options for routing algorithms:
   b. **Least outstanding requests:** Use to achieve a better load distribution to your back-end targets for cases when the requests for your application vary in complexity or your targets vary in processing capability.
   c. **Round robin:** Use when the requests and targets are similar, or if you need to distribute requests equally among targets.
4. Consider cross-zone or zonal isolation.

   a. Use cross-zone turned off (zonal isolation) for latency improvements and zonal failure domains. It is turned off by default in NLB and in ALB you can turn it off per target group.
   b. Use cross-zone turned on for increased availability and flexibility. By default, cross-zone is turned on for ALB and in NLB you can turn it on per target group.
5. Turn on HTTP keep-alives for your HTTP workloads.
   a. For HTTP workloads, turn on HTTP keep-alive in the web server settings for your back-end targets. With this feature, the load balancer can reuse backend connections until the keep-alive timeout expires, improving your HTTP request and response time and also reducing resource utilization on your back-end targets. For detail on how to do this for Apache and Nginx, see What are the optimal settings for using Apache or NGINX as a backend server for ELB?
6. Use Elastic Load Balancing integrations for better orchestration of compute resources.
   a. Use Auto Scaling integrated with your load balancer. One of the key aspects of a performance efficient system has to do with right-sizing your back-end resources. To do this, you can leverage load balancer integrations for back-end target resources. Using the load balancer integration with Auto Scaling groups, targets will be added or removed from the load balancer as required in response to incoming traffic.
   b. Load balancers can also integrate with Amazon ECS and Amazon EKS for containerised workloads.
      - Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group
      - Amazon ECS - Service load balancing
      - Application load balancing on Amazon EKS
      - Network load balancing on Amazon EKS
7. Monitor your load balancer to find performance bottlenecks.
   a. Turn on access logs for your Application Load Balancer and Network Load Balancer.
   b. The main fields to consider for ALB are `request_processing_time`, `request_processing_time`, and `response_processing_time`.
   c. The main fields to consider for NLB are `connection_time` and `tls_handshake_time`.
   d. Be ready to query the logs when you need them. You can use Amazon Athena to query both ALB logs and NLB Logs.
   e. Create alarms for performance related metrics such as `TargetResponseTime` for ALB.

Resources

Related best practices:

- SEC09-BP02 Enforce encryption in transit

Related documents:

- ELB product comparison
- AWS Global Infrastructure
- Improving Performance and Reducing Cost Using Availability Zone Affinity
- Step by step for Log Analysis with Amazon Athena
- Querying Application Load Balancer logs
- Monitor your Application Load Balancers
- Monitor your Network Load Balancers

Related videos:

- AWS re:Invent 2018: [REPEAT 1] Elastic Load Balancing: Deep Dive and Best Practices (NET404-R1)

- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads
- AWS re:Inforce 2022 - How to use Elastic Load Balancing to enhance your security posture at scale (NIS203)
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads (NET407-R2)

Related examples:

- CDK and CloudFormation samples for Log Analysis with Amazon Athena

PERF05-BP05 Choose network protocols to improve performance

Assess the performance requirements for your workload, and choose the network protocols that optimize your workload's overall performance.

There is a relationship between latency and bandwidth to achieve throughput. For instance, if your file transfer is using Transmission Control Protocol (TCP), higher latencies will reduce overall throughput. There are approaches to fix this with TCP tuning and optimized transfer protocols (some approaches use User Datagram Protocol (UDP)).

The scalable reliable datagram (SRD) protocol is a network transport protocol built by AWS for Elastic Fabric Adapters that provides reliable datagram delivery. Unlike the TCP protocol, SRD can reorder packets and deliver them out of order. This out of order delivery mechanism of SRD sends packets in parallel over alternate paths, increasing throughput.

Common anti-patterns:

- Using TCP for all workloads regardless of performance requirements.

Benefits of establishing this best practice:

- Selecting the proper protocol for communication between workload components ensures that you are getting the best performance for that workload.
- Verifying that an appropriate protocol is used for communication between users and workload components helps improve overall user experience for your applications. For instance, by using both TCP and UDP together, VDI workloads can take advantage of the reliability of TCP for critical data and the speed of UDP for real-time data.

Level of risk exposed if this best practice is not established: Medium (Using an inappropriate network protocol can lead to poor performance, such as slow response times, high latency and poor scalability)

Implementation guidance

A primary consideration for improving your workload's performance is to understand the latency and throughput requirements, and then choose network protocols that optimize performance.

When to consider using TCP

TCP provides reliable data delivery, and can be used for communication between workload components where reliability and guaranteed delivery of data is important. Many web-based applications rely on TCP-based protocols, such as HTTP and HTTPS, to open TCP sockets for communication with servers on AWS. Email and file data transfer are common applications that also make use of TCP due to TCP's ability to control the rate of data exchange and network congestion. Using TLS with TCP can add some overhead to the communication, which can result in increased latency and reduced throughput. The overhead

comes mainly from the added overhead of the handshake process, which can take several round-trips to complete. Once the handshake is complete, the overhead of encrypting and decrypting data is relatively small.

When to consider using UDP

UDP is a connectionless-oriented protocol and is therefore suitable for applications that need fast, efficient transmission, such as log, monitoring, and VoIP data. Also, consider using UDP if you have workload components that respond to small queries from large numbers of clients to ensure optimal performance of the workload. Datagram Transport Layer Security (DTLS) is the UDP equivalent of TLS. When using DTLS with UDP, the overhead comes from encrypting and decrypting the data, as the handshake process is simplified. DTLS also adds a small amount of overhead to the UDP packets, as it includes additional fields to indicate the security parameters and to detect tampering.

When to consider using SRD

Scalable reliable datagram (SRD) is a network transport protocol optimized for high-throughput workloads due to its ability to load-balancer traffic across multiple paths and quickly recover from packet drops or link failures. SRD is therefore best used for high performance computing (HPC) workloads that require high throughput and low latency communication between compute nodes. This might include parallel processing tasks such as simulation, modelling, and data analysis that involve a large amount of data transfer between nodes.

Implementation steps

1. Use the AWS Global Accelerator and AWS Transfer Family services to improve the throughput of your online file transfer applications. The AWS Global Accelerator service helps you achieve lower latency between your client devices and your workload on AWS. With AWS Transfer Family, you can use TCP-based protocols such as Secure Shell File Transfer Protocol (SFTP) and File Transfer Protocol over SSL (FTPS) to securely scale and manage your file transfers to AWS storage services.
2. Use network latency to determine if TCP is appropriate for communication between workload components. If the network latency between your client application and server is high, then the TCP three-way handshake can take some time, thereby impacting on the responsiveness of your application. Metrics such as Time to First Byte (TTFB) and Round-Trip Time (RTT) can be used to measure network latency. If your workload serves dynamic content to users, consider using Amazon CloudFront, which establishes a persistent connection to each origin for dynamic content to eliminate the connection setup time that would otherwise slow down each client request.
3. Using TLS with TCP or UDP can result in increased latency and reduced throughput for your workload due to the impact of encryption and decryption. For such workloads, consider SSL/TLS offloading on Elastic Load Balancing to improve workload performance by allowing the load balancer to handle SSL/TLS encryption and decryption process instead of having backend instances do it. This can help reduce the CPU utilization on the backend instances, which can improve performance and increase capacity.
4. Use the Network Load Balancer (NLB) to deploy services that rely on the UDP protocol, such as authentication and authorization, logging, DNS, IoT, and streaming media, to improve the performance and reliability of your workload. The NLB distributes incoming UDP traffic across multiple targets, allowing you to scale your workload horizontally, increase capacity, and reduce the overhead of a single target.
5. For your High Performance Computing (HPC) workloads, consider using the Elastic Network Adapter (ENA) Express functionality that uses the SRD protocol to improve network performance by providing a higher single flow bandwidth (25Gbps) and lower tail latency (99.9 percentile) for network traffic between EC2 instances.
6. Use the Application Load Balancer (ALB) to route and load balance your gRPC (Remote Procedure Calls) traffic between workload components or between gRPC clients and services. gRPC uses the TCP-based HTTP/2 protocol for transport and it provides performance benefits such as lighter network footprint, compression, efficient binary serialization, support for numerous languages, and bi-directional streaming.

Resources

Related documents:

- Amazon EBS - Optimized Instances
- Application Load Balancer
- 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
- Network Load Balancer
- Networking Products with AWS
- Transit Gateway
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- Optimizing Network Performance for Amazon EC2 Instances (CMP308-R1)
- Tuning Your Cloud: Improve Global Network Performance for Application
- Application Scaling with EFA and SRD

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF05-BP06 Choose your workload's location based on network requirements

Evaluate options for resource placement to reduce network latency and improve throughput, providing an optimal user experience by reducing page load and data transfer times.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Resources, such as Amazon EC2 instances, are placed into availability zones within AWS Regions, AWS Local Zones, AWS Outposts, or AWS Wavelength zones. Selection of this location influences network latency and throughput from a given user location. Edge services such as Amazon CloudFront and AWS Global Accelerator can also be used to improve network performance by either caching content at edge locations or providing users with an optimal path to the workload through the AWS global network.

Implementation steps

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

   a. **Where your users are located:** choosing a Region close to your workload's users to ensure low latency when they use the workload.
   b. **Where your data is located:** for data-heavy applications, the major bottleneck in data transfer is latency. Application code should run as close to the data as possible.
   c. **Other constraints:** consider constraints such as security and compliance (for example, data residency requirements).
2. For a given workload, if a component consists of a group of interdependent Amazon EC2 instances requiring low-latency, consider using cluster placement groups to influence placement of those instances to meet the requirements of the workload. Instances in the same cluster placement group enjoy a higher per-flow throughput limit for TCP/IP traffic and are placed in the same high-bisection bandwidth segment of the network. Cluster placement groups are recommended for applications that benefit from low network latency, high network throughput, or both.
3. For a workload that is location-sensitive, for example with low-latency or data residency requirements, review AWS Local Zones or AWS Outposts.
   a. AWS Local Zones are a type of infrastructure deployment that places compute, storage, database, and other select AWS services close to large population and industry centers.
   b. AWS Outposts is a family of fully managed solutions delivering AWS infrastructure and services to virtually any on-premises or edge location for a truly consistent hybrid experience.
4. Applications such as high-resolution live video streaming, high-fidelity audio, and augmented reality/virtual reality (AR/VR) require ultra-low-latency for 5G devices. For such applications, consider AWS Wavelength. AWS Wavelength embeds AWS compute and storage services within 5G networks, providing mobile edge computing infrastructure for developing, deploying, and scaling ultra-low-latency applications.
5. If you have geographically distributed users, a content distribution network (CDN) may be used to accelerate distribution of static and dynamic web content by delivering data through globally dispersed points of presence (PoPs). CDNs typically also provide edge computing capabilities, performing latency sensitive operations such as HTTP header manipulations and URL rewrites and redirects at large scale at the edge. Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content. Use cases for CloudFront include accelerating static website content delivery and serving video on demand or live streaming video. CloudFront can also be used to customize the content and experience for viewers, at reduced latency.
6. Some applications require fixed entry points or higher performance by reducing first byte latency and jitter, and increasing throughput. These applications can benefit from networking services that provide static anycast IP addresses and TCP termination at edge locations. AWS Global Accelerator can improve performance for your applications by up to 60% and provide quick failover for multi-region architectures. AWS Global Accelerator provides you with static anycast IP addresses that serve as a fixed entry point for your applications hosted in one or more AWS Regions. These IP addresses permit traffic to ingress onto the AWS global network as close to your users as possible. AWS Global Accelerator reduces the initial connection setup time by establishing a TCP connection between the client and the AWS edge location closest to the client. Review the use of AWS Global Accelerator to improve the performance of your TCP/UDP workloads and provide quick failover for multi-region architectures.
7. If you have applications or users on-premises, you may benefit from having a dedication network connection between your network and the cloud. A dedicated network connection can reduce the chance of encountering congestion or unexpected increases in latency. AWS Direct Connect can improve application performance by connecting your network directly to AWS and bypassing the public internet. When creating a new connection, you can choose a hosted connection provided by an AWS Direct Connect Delivery Partner, or choose a dedicated connection from AWS and deploy at over 100 AWS Direct Connect locations around the globe. You can also reduce your networking costs with low data transfer rates out of AWS, and optionally configure a Site-to-Site VPN for failover.
8. If you configure a Site-to-Site VPN to connect to your resources within AWS, you can optionally turn on acceleration. An accelerated Site-to-Site VPN connection uses AWS Global Accelerator to route traffic from your on-premises network to an AWS edge location that is closest to your customer gateway device.

9. Identify which DNS routing option would optimize your workload performance by reviewing your workload traffic and user location. Amazon Route 53 offers latency-based routing, geolocation routing, geoproximity routing, and IP-based routing options to help you improve your workload's performance for a global audience.
    a. Route 53 also offers low query latency for your end users. Using a global anycast network of DNS servers around the world, Route 53 is designed to automatically answer queries from the optimal location depending on network conditions.

Resources

Related best practices:

- COST07-BP02 Implement Regions based on cost
- COST08-BP03 Implement services to reduce data transfer costs
- REL10-BP01 Deploy the workload to multiple locations
- REL10-BP02 Select the appropriate locations for your multi-location deployment
- SUS01-BP01 Choose Regions near Amazon renewable energy projects and Regions where the grid has a published carbon intensity that is lower than other locations (or Regions)
- SUS02-BP04 Optimize geographic placement of workloads for user locations
- SUS04-BP07 Minimize data movement across networks

Related documents:

- AWS Global Infrastructure
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- Placement groups
- AWS Local Zones
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- Site-to-Site VPN
- Amazon Route 53

Related videos:

- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How It Works
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

Related examples:

- AWS Global Accelerator Workshop
- Handling Rewrites and Redirects using Edge Functions

PERF05-BP07 Optimize network configuration based on metrics

Improper network configuration often affects network performance, efficiency, and cost. In common network environments, in order to quickly complete the deployment in the early stage, the proper network configuration is not fully considered in terms of network performance. To optimize your network configuration, you must first have visibility and data about your network environment.

To understand how your network resources are performing, collect and analyze 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.

Desired outcome: Use metrics and network monitoring tools to optimize network configuration as workloads evolve. Cloud-based networks can be optimized quickly, so evolving your network architecture over time is necessary to maintain performance efficiency.

Common anti-patterns:

- You assume that all performance-related issues are application-related.
- You only test your network performance from a location close to where you have deployed the workload.
- You use default configurations for all network services.
- You overprovision the network resource to provide sufficient capacity.

Benefits of establishing this best practice: Collecting necessary metrics of your AWS network and implementing network monitoring tools allows you to understand network performance and optimize network configurations.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Monitoring traffic to and from VPCs, subnets, or network interfaces is crucial to understanding how to utilize AWS network resources and how you can optimize network configurations. By using the following tools, you can further inspect information about the traffic usage, network access and logs.

Implementation steps

1. Use Amazon VPC IP Address Manager. You can use IPAM to plan, track, and monitor IP addresses for your AWS and on-premises workloads. This is the best practice for you for to optimize IP address usage and allocation.
2. Turn on VPC Flow logs. Use VPC Flow Logs to capture detailed information about traffic to and from network interfaces in your VPCs. With VPC Flow Logs, you can diagnose overly restrictive or permissive security group rules and determine the direction of the traffic to and from the network interfaces. Data ingestion and archival charges for vended logs apply when you publish flow logs.
3. Turn on DNS query logging. You can configure Amazon Route 53 to log information about public or private DNS queries Route 53 receives. With DNS logs, you can optimize DNS configurations by understanding the domain or subdomain that was requested or Route 53 EDGE locations that responded to DNS queries.
4. Use Reachability Analyzer to analyze and debug network reachability. Reachability Analyzer is a configuration analysis tool that allows you to perform connectivity testing between a source resource

and a destination resource in your VPCs. This tool helps you verify that your network configuration matches your intended connectivity.

5. Use Network Access Analyzer to understand network access to your resources. You can use Network Access Analyzer to specify your network access requirements and identify potential network paths that do not meet your specified requirements. By optimizing your corresponding network configuration, you can understand and verify the state of your network and demonstrate if your network on AWS meets your compliance requirements.
6. Use Amazon CloudWatch and turn on the appropriate metrics for network options. Make sure to choose the right network metric for your workload. For example, you can turn on metrics for VPC Network Address Usage, VPC NAT Gateway, AWS Transit Gateway, VPN tunnel, AWS Network Firewall, Elastic Load Balancing, and AWS Direct Connect. Continually monitoring metrics is a good practice to observe and understand your network status and usage, and helps you optimize network configuration based on your observations.

Level of effort for the implementation plan: Medium

Resources

Related documents:

- VPC Flow Logs
- Public DNS query logging
- What is IPAM?
- What is Reachability Analyzer?
- What is Network Access Analyzer?
- CloudWatch metrics for your VPCs
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format
- Monitoring your global and core networks with Amazon Cloudwatch metrics
- Continuously monitor network traffic and resources

Related videos:

- Networking best practices and tips with the Well-Architected Framework
- Monitoring and troubleshooting network traffic

Related examples:

- AWS Networking Workshops
- AWS Network Monitoring

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 considers 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 allows 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 start 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 run 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 run 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.

Best practices

- PERF06-BP01 Stay up-to-date on new resources and services (p. 66)
- PERF06-BP02 Define a process to improve workload performance (p. 67)
- PERF06-BP03 Evolve workload performance over time (p. 68)

PERF06-BP01 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 evaluation, internal discussion, or external analysis.

Define a process to evaluate updates, new features, and services relevant to your workload. For example, building a proof of concept that uses 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 performance of the workload. Using infrastructure as code (IaC) and a DevOps culture to take advantage of the ability to test new ideas or technologies frequently with minimal cost or risk.

Desired outcome: You have documented the inventory of components, your design pattern, and your workload characteristics. You use that documentation to create a list of subscriptions to notify your team on service updates, features, and new products. You have identified component stakeholders that will evaluate the new releases and provide a recommendation for business impact and priority.

Common anti-patterns:

- You only review new options and services when your workload is not meeting performance requirements.
- You assume all new product offerings will not be useful to your workload.
- You always choose to build as opposed to buy when improving your workload.

Benefits of establishing this best practice: By considering new services or product offerings, you can improve the performance and efficiency of your workload, lower the cost of the infrastructure, and reduce the effort required to maintain your services.

Level of risk exposed if this best practice is not established: High

Implementation guidance

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

Implementation steps

1. Document your workload solutions. Use your configuration management database (CMDB) solution to document your inventory and categorize your services and dependencies. Use tools like AWS Config to get a list of all services in AWS being used by your workload.
2. Use a tagging strategy to document owners for each workload component and category. For example, if you are currently using Amazon RDS as your database solution, have your database administrator (DBA) assigned and documented as the owner for evaluating and researching new services and updates.
3. Identify news and update sources related to your workload components. In the Amazon RDS example previously mentioned, the category owner should subscribe to the What's New at AWS blog for the products that match their workload component. You can subscribe to the RSS feed or manage your email subscriptions. Monitor upgrades to the Amazon RDS database you use, features introduced, instances released and new products like Amazon Aurora Serverless. Monitor industry blogs, products, and vendors that the component relies on.
4. Document your process for evaluating updates and new services. Provide your category owners the time and space needed to research, test, experiment, and validate updates and new services. Refer back to the documented business requirements and KPIs to help prioritize which update will make a positive business impact.

Level of effort for the implementation plan: To establish this best practice, you must be aware of your current workload components, identify category owners and identify sources for service updates. This is a low level of effort to start but is an ongoing process that could evolve and improve over time.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

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

Related examples:

- AWS Github
- AWS Skill Builder

PERF06-BP02 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.

Common anti-patterns:

- You assume your current architecture will become static and never update over time.
- You introduce architecture changes over time with no metric justification.

Benefits of establishing this best practice: By defining your process for making architectural changes, you allow gathered data to influence your workload design over time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Identify the key performance constraints for your workload: Document your workload's performance constraints so that you know what kinds of innovation might improve the performance of your workload.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

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

Related examples:

- AWS Github
- AWS Skill Builder

PERF06-BP03 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.

Common anti-patterns:

- You assume that your current architecture will become static and never update over time.
- You introduce architecture changes over time with no metric justification.
- You change architecture just because everyone else in the industry is using it.

Benefits of establishing this best practice: To optimize your workload performance and cost, you must evaluate all software and services available to determine the appropriate ones for your workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Evolve your workload over time: 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 achieve the biggest gains in efficiency or performance, and proactively adopt new services and technologies to keep up with demand.

Resources

Related documents:

- AWS Blog
- What's New with AWS

Related videos:

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

Related examples:

- AWS Github
- AWS Skill Builder

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:

- **Generation** – scope of monitoring, metrics, and thresholds
- **Aggregation** – creating a complete view from multiple sources
- **Real-time processing and alarming** – recognizing and responding
- **Storage** – data management and retention policies
- **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 allow 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 invocations 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 allow 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 invoking 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.

Best practices

PERF07-BP01 Record performance-related metrics

Use a monitoring and observability service to record performance-related metrics. Examples of metrics include 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.

Common anti-patterns:

- You only monitor operating system level metrics to gain insight into your workload.
- You architect your compute needs for peak workload requirements.

Benefits of establishing this best practice: To optimize performance and resource utilization, you need a unified operational view of your key performance indicators. You can create dashboards and perform metric math on your data to derive operational and utilization insights.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify the relevant performance metrics for your workload and record them. This data helps identify which components are impacting overall performance or efficiency of your workload.

Identify performance metrics: Use the customer experience to identify the most important metrics. For each metric, identify the target, measurement approach, and priority. Use these data points to build alarms and notifications to proactively address performance-related issues.

Resources

Related documents:

- CloudWatch Documentation
- Collect metrics and logs from Amazon EC2 Instances and on-premises servers with the CloudWatch Agent
- Publish custom metrics
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation
- Amazon CloudWatch RUM

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- Build a Monitoring Plan

Related examples:

- Level 100: Monitoring with CloudWatch Dashboards
- Level 100: Monitoring Windows EC2 instance with CloudWatch Dashboards
- Level 100: Monitoring an Amazon Linux EC2 instance with CloudWatch Dashboards

PERF07-BP02 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 run. For these critical stories, implement additional scripted user journeys to ensure that you know how these stories perform against your requirement.

Common anti-patterns:

- You assume that performance events are one-time issues and only related to anomalies.
- You only evaluate existing performance metrics when responding to performance events.

Benefits of establishing this best practice: In determine whether your workload is operating at expected levels, you must respond to performance events by gathering additional metric data for analysis. This data is used to understand the impact of the performance event and suggest changes to improve workload performance.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Prioritize experience concerns for critical user stories: When you write critical user stories for your architecture, include performance requirements, such as specifying how quickly each critical story should

run. For these critical stories, implement additional scripted user journeys to ensure that you know how the user stories perform against your requirements.

Resources

Related documents:

- CloudWatch Documentation
- Amazon CloudWatch Synthetics
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF07-BP03 Establish key performance indicators (KPIs) to measure workload performance

Identify the KPIs that quantitatively and qualitatively measures workload performance. KPIs help to measure the health of a workload as it relates to a business goal. KPIs allow business and engineering teams to align on the measurement of goals and strategies and how this combines to produce business outcomes. KPIs should be revisited when business goals, strategies, or end-user requirements change.

For example, a website workload might use the page load time as an indication of overall performance. This metric would be one of the multiple data points which measure an end user experience. In addition to identifying the page load time thresholds, you should document the expected outcome or business risk if the performance is not met. A long page load time would affect your end users directly, decrease their user experience rating and might lead to a loss of customers. When you define your KPI thresholds, combine both industry benchmarks and your end user expectations. For example, if the current industry benchmark is a webpage loading within a two second time period, but your end users expect a webpage to load within a one second time period, then you should take both of these data points into consideration when establishing the KPI. Another example of a KPI might focus on meeting internal performance needs. A KPI threshold might be established on generating sales reports within one business day after production data has been generated. These reports might directly affect daily decisions and business outcomes.

Desired outcome: Establishing KPIs involve different departments and stakeholders. Your team must evaluate your workload KPIs using real-time granular data and historical data for reference and create dashboards that perform metric math on your KPI data to derive operational and utilization insights. KPIs should be documented which explains the agreed upon KPIs and thresholds that support business goals and strategies as well as mapped to metrics being monitored. The KPIs are identifying performance requirements, reviewed intentionally and are frequently shared and understood with all teams. Risks and tradeoffs are clearly identified and understood how business is impact within KPI thresholds are not met.

Common anti-patterns:

- You only monitor system level metrics to gain insight into your workload and don't understand business impacts to those metrics.
- You assume that your KPIs are already being published and shared as standard metric data.
- Defining KPIs but not sharing them with all the teams.
- Not defining a quantitative, measurable KPI.
- Not aligning KPIs with business goals or strategies.

Benefits of establishing this best practice: Identifying specific metrics which represent workload health help to align teams on their priorities and defining successful business outcomes. Sharing those metrics with all departments provides visibility and alignment on thresholds, expectations, and business impact.

Level of risk exposed if this best practice is not established: High

Implementation guidance

All departments and business teams impacted by the health of the workload should contribute to defining KPIs. A single person should drive the collaboration, timelines, documentation, and information related to an organization's KPIs. This single threaded owner will often share the business goals and strategies and assign business stakeholders tasks to create KPIs in their respective departments. Once KPIs are defined, the operations team will often help define the metrics that will support and inform the success of the different KPIs. KPIs are only effective if all team members supporting a workload are aware of the KPIs.

Implementation steps

1. Identify and document business stakeholders.
2. Identify company goals and strategies.
3. Review common industry KPIs that align with your company goals and strategies.
4. Review end user expectations of your workload.
5. Define and document KPIs that support company goals and strategies.
6. Identify and document approved tradeoff strategies to meet the KPIs.
7. Identify and document metrics that will inform the KPIs.
8. Identify and document KPI thresholds for severity or alarm level.
9. Identify and document the risk and impact if the KPI is not met.
10. Identify the frequency of review per KPI.
11. Communicate KPI documentation with all teams supporting the workload.

Level of effort for the implementation guidance: Defining and communicating the KPIs is a *low* amount of work. This can typically be done over a few weeks meeting with business stakeholders, reviewing goals, strategies, and workload metrics.

Resources

Related documents:

- CloudWatch documentation
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation
- Using Amazon CloudWatch dashboards
- Amazon QuickSight KPIs

Related videos:

- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)
- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Build a Monitoring Plan

Related examples:

- Creating a dashboard with Amazon QuickSight

PERF07-BP04 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 third-party monitoring service to set alarms that indicate when thresholds are breached — alarms signal that a metric is outside of the expected boundaries.

Common anti-patterns:

- You rely on staff to watch metrics and react when they see an issue.
- You rely solely on operational runbooks, when serverless workflows could be started to accomplish the same task.

Benefits of establishing this best practice: You can set alarms and automate actions based on either predefined thresholds, or on machine learning algorithms that identify anomalous behavior in your metrics. These same alarms can also start serverless workflows, which can modify performance characteristics of your workload (for example, increasing compute capacity, altering database configuration).

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Monitor metrics: Amazon CloudWatch can collect metrics across the resources in your architecture. You can collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or a third-party monitoring service to set alarms that indicate when thresholds are exceeded.

Resources

Related documents:

- CloudWatch Documentation
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation
- Using Alarms and Alarm Actions in CloudWatch

Related videos:

- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)
- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Build a Monitoring Plan
- Using AWS Lambda with Amazon CloudWatch Events

Related examples:

- Cloudwatch Logs Customize Alarms

PERF07-BP05 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 essential 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.

Common anti-patterns:

- You allow metrics to stay in an alarm state for an extended period of time.
- You create alarms that are not actionable by an automation system.

Benefits of establishing this best practice: Continually review metrics that are being collected to ensure that they properly identify, address, or prevent issues. Metrics can also become stale if you let them stay in an alarm state for an extended period of time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Constantly improve metric collection and monitoring: 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 method to improve the quality of metrics you collect so that you can prevent or more quickly resolve future incidents.

Resources

Related documents:

- CloudWatch Documentation
- Collect metrics and logs from Amazon EC2 Instances and on-premises servers with the CloudWatch Agent
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- Build a Monitoring Plan

Related examples:

- Creating a dashboard with Amazon QuickSight
- Level 100: Monitoring with CloudWatch Dashboards

PERF07-BP06 Monitor and alarm proactively

Use key performance indicators (KPIs), combined with monitoring and alerting systems, to proactively address performance-related issues. Use alarms to start 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.

Common anti-patterns:

- You only allow operations staff the ability to make operational changes to the workload.
- You let all alarms filter to the operations team with no proactive remediation.

Benefits of establishing this best practice: Proactive remediation of alarm actions allows support staff to concentrate on those items that are not automatically actionable. This ensures that operations staff are not overwhelmed by all alarms and instead focus only on critical alarms.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Monitor performance during operations: Implement processes that provide visibility into performance as your workload is running. Build monitoring dashboards and establish a baseline for performance expectations.

Resources

Related documents:

- CloudWatch Documentation
- Monitoring, Logging, and Performance APN Partners
- X-Ray Documentation
- Using Alarms and Alarm Actions in CloudWatch

Related videos:

- Cut through the chaos: Gain operational visibility and insight (MGT301-R1)
- Application Performance Management on AWS
- Build a Monitoring Plan
- Using AWS Lambda with Amazon CloudWatch Events

Related examples:

- Cloudwatch Logs Customize Alarms

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 allows 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.

Best practices

- PERF08-BP01 Understand the areas where performance is most critical (p. 79)
- PERF08-BP02 Learn about design patterns and services (p. 81)
- PERF08-BP03 Identify how tradeoffs impact customers and efficiency (p. 83)
- PERF08-BP04 Measure the impact of performance improvements (p. 84)
- PERF08-BP05 Use various performance-related strategies (p. 85)

PERF08-BP01 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.

Desired outcome: Increase performance efficiency by understanding your architecture, traffic patterns, and data access patterns, and identify your latency and processing times. Identify the potential bottlenecks that might affect the customer experience as the workload grows. When you identify those areas, look at which solution you could deploy to remove those performance concerns.

Common anti-patterns:

- You assume that standard compute metrics such as `CPUUtilization` or memory pressure are enough to catch performance issues.

- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.

Benefits of establishing this best practice: Understanding critical areas of performance helps workload owners monitor KPIs and prioritize high-impact improvements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Set up end-to-end tracing to identify traffic patterns, latency, and critical performance areas. Monitor your data access patterns for slow queries or poorly fragmented and partitioned data. Identify the constrained areas of the workload using load testing or monitoring.

Implementation steps

1. Set up end-to-end monitoring to capture all workload components and metrics.
   - Use Amazon CloudWatch Real-User Monitoring (RUM) to capture application performance metrics from real user client-side and frontend sessions.
   - Set up AWS X-Ray to trace traffic through the application layers and identify latency between components and dependencies. Use the X-Ray service maps to see relationships and latency between workload components.
   - Use Amazon Relational Database Service Performance Insights to view database performance metrics and identify performance improvements.
   - Use Amazon RDS Enhanced Monitoring to view database OS performance metrics.
   - Collect CloudWatch metrics per workload component and service and identify which metrics impact performance efficiency.
   - Set up Amazon DevOps Guru for additional performance insights and recommendations
2. Perform tests to generate metrics, identify traffic patterns, bottlenecks, and critical performance areas.
   - Set up CloudWatch Synthetic Canaries to mimic browser-based user activities programmatically using `cron` jobs or rate expressions to generate consistent metrics over time.
   - Use the AWS Distributed Load Testing solution to generate peak traffic or test the workload at the expected growth rate.
3. Evaluate the metrics and telemetry to identify your critical performance areas. Review these areas with your team to discuss monitoring and solutions to avoid bottlenecks.
4. Experiment with performance improvements and measure those changes with data.
   - Use CloudWatch Evidently to test new improvements and the performance impact to the workload.

Level of effort for the implementation plan: To establish this best practice, you must review your end-to-end metrics and be aware of your current workload performance. This is a moderate level of effort to set up end to end monitoring and identify your critical performance areas.

Resources

Related documents:

- Amazon Builders' Library
- X-Ray Documentation
- Amazon CloudWatch RUM

- Amazon DevOps Guru
- CloudWatch RUM and X-Ray

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Node.js
- X-Ray SDK for Python
- X-Ray SDK for Java
- X-Ray SDK for .Net
- X-Ray SDK for Ruby
- X-Ray Daemon
- Distributed Load Testing on AWS

PERF08-BP02 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, caching can introduce eventual consistency and requires engineering effort to implement within business requirements and customer expectations.

Desired outcome: Researching design patterns will lead you to choosing an architecture design that will support the best performing system. 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 end users.

Common anti-patterns:

- You assume that all traditional IT workload performance strategies are best suited for cloud workloads.
- You build and manage caching solutions instead of using managed services.
- You use the same design pattern for all your workloads without evaluating which pattern would improve the workload performance.

Benefits of establishing this best practice: By selecting the right design pattern and services for your workload you will be optimizing your performance, improving operational excellence and increasing reliability. The right design pattern will meet your current workload characteristics and help you scale for future growth or changes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Learn which performance configuration options are available 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 have on measured performance and user-perceived performance.

Implementation steps:

1. Evaluate and review design patterns that would improve your workload performance.
   a. The Amazon Builders' Library provides you with a detailed description of how Amazon builds and operates technology. These articles are written by senior engineers at Amazon and cover topics across architecture, software delivery, and operations.
   b. AWS Solutions Library is a collection of ready-to-deploy solutions that assemble services, code, and configurations. These solutions have been created by AWS and AWS Partners based on common use cases and design patterns grouped by industry or workload type. For example, you can set up a distributed load testing solution for your workload.
   c. AWS Architecture Center provides reference architecture diagrams grouped by design pattern, content type, and technology.
   d. AWS samples is a GitHub repository full of hands-on examples to help you explore common architecture patterns, solutions, and services. It is updated frequently with the newest services and examples.
2. Improve your workload to model the selected design patterns and use services and the service configuration options to improve your workload performance.
   a. Train your internal team with resources available at AWS Skills Guild.
   b. Use the AWS Partner Network to provide expertise quickly and to scale your ability to make improvements.

Level of effort for the implementation plan: To establish this best practice, you must be aware of the design patterns and services that could help improve your workload performance. After evaluating the design patterns, implementing the design patterns is a *high* level of effort.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Amazon Builders' Library
- Using load shedding to avoid overload
- Caching challenges and strategies

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- This is My Architecture

Related examples:

- AWS Samples

- AWS SDK Examples

PERF08-BP03 Identify how tradeoffs 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.

Common anti-patterns:

- You assume that all performance gains should be implemented, even if there are tradeoffs for implementation such as eventual consistency.
- You only evaluate changes to workloads when a performance issue has reached a critical point.

Benefits of establishing this best practice: When you are evaluating potential performance-related improvements, you must decide if the tradeoffs for the changes are consistent with the workload requirements. In some cases, you may have to implement additional controls to compensate for the tradeoffs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify tradeoffs: Use metrics and monitoring to identify areas of poor performance in your system. Determine how to make improvements, and how tradeoffs will impact the system and the user experience. For example, implementing caching data can help dramatically improve performance, but it requires a clear strategy for how and when to update or invalidate cached data to prevent incorrect system behavior.

Resources

Related documents:

- Amazon Builders' Library
- Amazon QuickSight KPIs
- Amazon CloudWatch RUM
- X-Ray Documentation

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- Build a Monitoring Plan
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF08-BP04 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.

Common anti-patterns:

- You deploy and manage technologies manually that are available as managed services.
- You focus on just one component, such as networking, when multiple components could be used to increase performance of the workload.
- You rely on customer feedback and perceptions as your only benchmark.

Benefits of establishing this best practice: For implementing performance strategies, you must select multiple services and features that, taken together, will allow you to meet your workload requirements for performance.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

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.

Resources

Related documents:

- Amazon Builders' Library
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- Optimize applications through Amazon CloudWatch RUM

- Demo of Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- Distributed Load Testing on AWS

PERF08-BP05 Use various performance-related strategies

Where applicable, use 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.

Common anti-patterns:

- You assume that workload performance is adequate if customers are not complaining.
- You only collect data on performance after you have made performance-related changes.

Benefits of establishing this best practice: To optimize performance and resource utilization, you need a unified operational view, real-time granular data, and historical reference. You can create dashboards and perform metric math on your data to derive operational and utilization insights for your workloads as they change over time.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use a data-driven approach to evolve your architecture: 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 tradeoffs impact your workload. Use a systematic approach, such as load testing, to explore whether the tradeoff improves performance.

Resources

Related documents:

- Amazon Builders' Library
- Best Practices for Implementing Amazon ElastiCache
- AWS Database Caching
- Amazon CloudWatch RUM
- Distributed Load Testing on AWS

Related videos:

- Introducing The Amazon Builders' Library (DOP328)
- AWS purpose-built databases (DAT209-L)
- Optimize applications through Amazon CloudWatch RUM

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- Distributed Load Testing on AWS

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 helps 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:

- Ryan Comingdeer, Performance Solutions Architect Well-Architected, Amazon Web Services
- Josh Hart, Solutions Architect, Amazon Web Services
- Richard Trabing, Solutions Architect, Amazon Web Services
- Eric Pullen, Solutions Architect, 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
- AWS Architecture Center

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Minor update (p. 90)	Remove non-inclusive language.	April 13, 2023
Updates for new Framework (p. 90)	Best practices updated with prescriptive guidance and new best practices added.	April 10, 2023
Whitepaper updated (p. 90)	Best practices updated with new implementation guidance.	December 15, 2022
Whitepaper updated (p. 90)	Best practices expanded and improvement plans added.	October 20, 2022
Minor update (p. 90)	Removed non-inclusive language.	April 22, 2022
Minor update (p. 1)	Added Sustainability Pillar to introduction.	December 2, 2021
Minor updates (p. 90)	Updated links.	March 10, 2021
Minor updates (p. 90)	Changed AWS Lambda timeout to 900 seconds and corrected name of Amazon Keyspaces (for Apache Cassandra).	October 5, 2020
Minor update (p. 90)	Fixed broken link.	July 15, 2020
Updates for new Framework (p. 90)	Major review and update of content	July 8, 2020
Whitepaper updated (p. 90)	Minor update for grammatical issues	July 1, 2018
Whitepaper updated (p. 90)	Refreshed the whitepaper to reflect changes in AWS	November 1, 2017
Initial publication (p. 90)	Performance Efficiency Pillar - AWS Well-Architected Framework published.	November 1, 2016

Notices

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.