Cost Optimization - Deep Dive

Table of Contents

Pre-Requisites

- Basic AWS know-how and AWS console operating knowledge
- Must have access to Billing console to view cost explorer data , access to Trusted Advisor, Compute Optimiser
- Enable "Hourly and Resource level Data" and "Receive Amazon EC2 resource recommendations" under Cost Management →Preference in Cost Explorer console or click the link here
- Enable compute optimizer

Deep Dive

Cost Visibility:

Tool - Cost Explorer
https://console.aws.amazon.com/cost-management/home#/dashboard

Goals of Cost visibility:-

- Which EC2 instance generation (6th gen, 5th , 4th gen) is being used. To allow you to move older generation to new gen to save on cost and performance.
- Which instance types are being used to understand how we can reserve instances.
- Understanding and scoping costs by Dev/Staging/Production to identify which workload needs optimization.
- Diversify compute across different pricing types to take advantage of reduced costs.
- Diversify the Processor types for workloads to take advantage of reduced costs of Graviton and AMD.

Most Frequently used reports:-

1.) Check last 6 month AWS usage pattern by service
2.) Check last 6 month AWS spend by Linked Account
3.) Last 6 months AWS spend by Usage-Type
4.) Regional Costs for last 6 months.

The first step to cost optimization is cost visibility to identify patterns and anomalies. So, Let's first check the Top 5 services contributing to the cost:- Check last 6 month AWS usage pattern by service

Let's say you have a category showing 'Others' and you want to check their trend over 6 months, then exclude some of the services from the visualization of this report by excluding them using filters – Example by excluding EC2, RDS and Support. You can see the other services and usage in the bottom pane which shows all the services or download the CSV to review the file.

Now, let's say you want to figure out the usage-type or API-usage-type cost for a single Service (based on whether the service is API service or infra-based which you have to manage) then we can include the service in filter and group by either 'Usage-Type' or Group by 'API-Operation'

Also, identify Check how much of the spend is happening for Prod , Dev, Staging environments or different workloads using different tags or accounts based on your setup.

This is going to help in next steps:-

Tagging or Multi-account:-

1. If Multi-Account — Check last 6 month AWS spend by Linked Account (If you do not have organisations enabled and it's a must)
2. If Single account, then use tags as 'groupBy' — Example using 'Name' as tag

Enable and activate your tags to appear in Cost Explorer:-
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/custom-tags.html

Enable AWS generated tags to automatically tag newly created resources: -
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/aws-tags.html

Cost and Usage Reports

The AWS Cost and Usage Reports (AWS CUR) contains the most comprehensive set of cost and usage data available. You can use Cost and Usage Reports to publish your AWS billing reports to an Amazon Simple Storage Service (Amazon S3) bucket that you own

Step by step procedure to setup CUR:-

https://wellarchitectedlabs.com/cost/200_labs/200_4_cost_and_usage_analysis/

CUR Dashboard automation:-

https://wellarchitectedlabs.com/cost/200_labs/200_cloud_intelligence/

EC2 Optimization

Most Frequently used reports:-

1.) EC2 instance usage type report - to check instance type, ebs etc used by EC2 service.
2.) EC2 Spend by Linked Account
3.) EC2 Spend by Instance type
4.) Spend by Region
5.) EC2 cost by Purchase/Pricing model n – to check on-demand, spot and Saving plan distribution.
6.) Savings Plan Recommendations

7.) EC2 Savings Plan Coverage
8.) EC2 Reserved Instances Coverage Report (In case you have EC2 reservations not using Savings plans)

Right sizing

So, the first step is to calculate the percentage of how much we can reduce by using rightsizing of the EC2 instances by removing unused/idle instances and then downsizing or changing the family of under-utilised instances. This report is available under compute-optimizer and your EC2 console as well when you opt in.

☛ Check Rightsizing recommendation – Select the regions where you run your workloads to see the report for utilization of EC2. If you see a non-zero number in Over-provisioned then review the recommendation as the resources are not being used optimally. Ideally target a utilization of 50% for both CPU and Memory.

Also review any resources which are in Under-Provisioned section to increase their size as being under-provisioned could limit the performance.

1. Compute optimizer Dashboard: https://console.aws.amazon.com/computeoptimizer/home?#/dashboard

Note:- Compute optimizer would use CPU and Disk and Network metrics by default and would use Memory metric only when you push the memory metrics using Cloudwatch as memory data is only available in the OS.

Additional steps

1. Pushing Memory metrics to Cloudwatch:-
   - https://wellarchitectedlabs.com/cost/200_labs/200_aws_resource_optimization/

2. Instance Scheduler: -
   This involves step if you can shut-down or schedule some instances to only run at specified time of the day by using instance scheduler. This solution can schedule EC2 instances and RDS databases
   https://aws.amazon.com/solutions/implementations/instance-scheduler/

Best practices:-
https://wellarchitectedlabs.com/cost/200_labs/200_aws_resource_optimization/6_ec2_rs_best_practices/

Right Pricing

☛ Check breakup of Pricing model - purchase option of your EC2 instances

(Best case is to see all three - ‘On-demand’, ‘Reservation’/’Savings-Plan’ and ‘Spot’ and try to reduce the on-demand cost as much as possible. On-demand should only be used for New workloads)

☛ Check Savings Plan recommendations

Compare the trends in usage to see if your usage is increasing or decreasing. If usage is decreasing make a smaller initial hourly commitment, then re-analyze in 2-4 weeks. If usage is steady or increasing make a commitment closer to the recommended commitment:

You can also measure the Elasticity of your workload, by using the hourly cost explorer utilization report for last 14 days
Hourly Cost and Elasticity of your EC2 instances
Hourly Cost and Elasticity per instance type

Note the hourly data is only available if you enable the hourly usage in cost explorer in preferences: -
https://console.aws.amazon.com/cost-management/home?#/settings

Additional steps

https://wellarchitectedlabs.com/cost/100_labs/100_3_pricing_models/3_analyze_recommendations/

Spot Instance Strategy

Based on the above reports now you should know which workloads are Test/Dev/Staging and how much cost is accrued from production.

The low hanging fruit available to use spot is on Staging environments integrating the environments with Spot either by Autoscaling Fleets or EC2 Spot fleet for standalone instances. This gives around 50-90% cost savings on these workloads.

☛ Check the Spot Advisor page now
o Use Spot advisor page to understand the savings per EC2 instance type and their rate of interruption for the region and select your instances by first grouping the instances which have least interruption for your region (<5%) and then sort them by price or performance:

Additional steps

Whitepaper on using Spot: -
https://d1.awsstatic.com/whitepapers/cost-optimization-leveraging-ec2-spot-instances.pdf

For a Hands-on workshop and demo on Spot please use the below link which has examples and workshops for all types of workloads which can be used with Spot: -

https://ec2spotworkshops.com/
https://ec2spotworkshops.com/amazon-ec2-spot-cicd-workshop.html

Key here is to understand when to use Spot Fleet, EC2 Fleet or using Autoscaling EC2 fleet and make sure to use "Capacity optimized" option to ensure capacity is always provisioned using spot.

Apart from this you can use Spot instances capacity in multiple other workloads however this would need testing your workloads with Spot.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-best-practices.html

Ideally you can use Spot instances with container based workloads or big-data workloads easily where the EC2 instances are stateless.

https://www.youtube.com/watch?v=7q5AeoKsGJw

Processor Strategy

Use AMD or Graviton (Arm) based instances as alternative processor types. AMD will bring another 45% over Intel price in Mumbai (other regions ~10%) and using Graviton will bring 50% over Intel (in Mumbai region).

- Check out AMD processor instances types
- Check out Arm processor (Graviton) instance types – Use 6th Generation Graviton instances.

Consider moving to Graviton2 based processors specially for Managed services like RDS, Elasticache , EMR , Elasticsearch to reduce cost with equal or improved performance as it's very easy to move these managed services to Graviton2.

Use the below AWS workshop to get a hands on experience of moving workloads to Graviton2:-

https://graviton2-workshop.workshop.aws/en/

https://github.com/aws/aws-graviton-getting-started

Below is an open Global Competition known as Graviton Challenge to win prizes, if you are interested: -
https://aws.amazon.com/blogs/aws/migrate-your-workloads-with-the-graviton-challenge/

Additional steps

Use AMD and ARM instances after benchmarking your workloads with these CPU options.
https://www.anandtech.com/show/15578/cloud-clash-amazon-graviton2-arm-against-intel-and-amd
https://www.anandtech.com/show/15578/cloud-clash-amazon-graviton2-arm-against-intel-and-amd/9
https://www.percona.com/blog/2021/01/22/postgresql-on-arm-based-aws-ec2-instances-is-it-any-good/

Best practices for other services which use EC2:-

- Use Autoscaling with EC2 or container workloads specially if you are running containers on ECS or EKS.
- Use Spot for Container workloads to cater to the burst and spiky traffic with a right balance with on-demand instances for consistent.
- For Batch workloads use Spot instances or Spot Blocks to get a confirmed Spot instances for a dedicated duration using Spot fleet.
- For EMR Spark workloads use the EMR Instance fleet to schedule the Task nodes with Spot and define the timeout to fallback to on-demand in case Spot capacity is not available.
- Use EMR managed Autoscaling to automatically scale the Task nodes when the jobs need more resources. This can allow you to reduce the number of running instances in the cluster during off-peak hours and automatically scale when needed.
- For Machine learning training jobs use checkpointing and use spot instance fleets to reduce costs for trainings.
- If you want to do this in a managed way then use Sagemaker Managed Spot to do this for you and ensure your training jobs does not get interrupted. Sagemaker can do checkpointing for your models and start again automatically when spot is available again from the checkpoint.
- Sagemaker also has Savings plan with which you can reserve your capacity if you run ML learning jobs 24x7.
- Lastly, calculate your Savings plan commitment you should do and below steps help you do that:-

Right Architecture:-

- Use Serverless architecture for workloads that are bursty in nature and do not require 24x7 utilization.
- Use containers to efficiently use EC2 resources for multiple applications by packing applications efficiently in a cluster using ECS or EKS services and autoscaling.

EC2-Other optimization: -

The EC2-Other category includes multiple service-related usage types, tracking costs associated Amazon EBS volumes and snapshots, elastic IP addresses, NAT gateways, data transfer, and more.

- Check the break-up of EC2-Other costs (EBS , NAT, ELB)
- Checkout EBS Volume recommendations (Select your regions)

EBS Volume Optimization:-

If EBS is a considerable cost then Use EC2-Compute optimizer to check Idle EBS volumes and remove or try to downsize over-provisioned volumes which are not utilised.
https://console.aws.amazon.com/compute-optimizer/home#/resources-lists/ebs

Move GP2 volumes to GP3 to reduce the cost and at the same time get sustained performance and reduced cost with gp3 volumes as compared to burstable performance on gp2.

You cannot optimize on NAT gateway running hours except otherwise by removing it, which may not be possible due to NAT requirements. So, here you take a trade-off between Cost and Resilience.

Data Transfer Out (DTO) optimization:-

The below reports will help you understand Data Transfer per service:-

- DataTransfer cost by service – Additionally to this report add a filter(Right Pane) for Usage-Type → Search for "Bytes" in the Usage Type → Select All
- Check now for Data Transfer cost by API Operation - additionally add a filter(Right Pane) for Usage-Type → Search for "Bytes" in the Usage Type → Select All

Additional steps

1.) For Any EC2-Other Data Transfer Cost Use Cloudfront and Private Pricing on Cloudfront to Reduce the Data-Transfer out.
2.) Regional-Data Transfer on Public Ip's can be reduced by using Private IP's between Applications.
3.) InterZone-In and Interzone-Out are charges due to inter-AZ traffic.
4.) PublicIPIn/PublicIP-Out are charges due to transfer over Public IP's in the same region, charged in each direction.

5.) Move chatty servers close to each other and in the Same AZ if possible, but take a tradeoff on High Availability of Business requires it.
6.) *Reduce NAT Gateway Data Transfer Costs using VPC Endpoints for S3 or DynamoDB.*

S3-optimization: -

Report - S3 costs by usage type

Optimizing Costs for workloads with predictable access patterns:-

- Use Storage Lens advanced reports to get a detailed understanding of the bucket usage across all regions and accounts.
- Identify your largest S3 buckets.
- Locate incomplete multipart uploads.
- Reduce the number of noncurrent versions retained.
- Uncover cold Amazon S3 buckets.
- Create Lifecycle rules to transition data to infrequent-access or archive layers.

https://docs.aws.amazon.com/AmazonS3/latest/userguide/storage-lens-optimize-storage.html

https://aws.amazon.com/blogs/aws/s3-storage-lens/

https://aws.amazon.com/blogs/storage/5-ways-to-reduce-costs-using-amazon-s3-storage-lens/

Optimizing Storage Costs for workloads with changing access patterns:-

Use S3 intelligent tiering for new S3 buckets when you do not know the data access patterns.

S3 Intelligent-Tiering storage class gives you a way to save money even under changing access patterns, with no performance impact, no operational overhead, and no retrieval fees.

It works by storing objects in four access tiers: two low latency access tiers optimized for frequent and infrequent access, and two optional archive access tiers designed for asynchronous access.

S3 Intelligent-Tiering works by monitoring access patterns and then moving the objects that have not been accessed in 30 consecutive days to the Infrequent Access tier. You can also choose to activate one or both of the optional archive access tiers.

If you activate the archive access tiers, S3 Intelligent-Tiering automatically moves objects that have not been accessed for 90 consecutive days to the Archive Access tier. After 180 consecutive days of no access, S3 Intelligent-Tiering moves objects to the Deep Archive Access tier. If the objects are accessed again later, they are moved back to the Frequent Access Tier.

Note:- We recommend that you keep objects smaller than 128KB in S3 Standard because they are not eligible for auto-tiering

Further best practices for S3 optimization: -

- Use Cloudfront to cache and reduce the retrieval requests to reduce s3 API usage costs.
- If you are using S3 as a Data Lake, then ensure that queries running against S3 objects from Athena or any other ETL tools should use larger object size, otherwise you will end up with too many S3 GET requests for each query. Also, use compressed data formats like Parquet or ORC to optimize the query performance and also reduce the data GET requests.
- *Reduce NAT Data Transfer Costs in VPC using VPC Endpoints for S3.*

CloudWatch Optimization

Best practices:-

- If you are using Cloudwatch Agent to send custom metrics and if using it to just get memory and disk metrics then use the 'Basic' configuration to only push these metrics.

- If you are using Lambda functions then do not log all the console output in Lambda for logging purpose but use a logger library and reduce the payload of logs as larger the payload, more data you are ingesting into Cloudwatch logs.

- Define a lifecycle for Cloudwatch logs to expire older logs to S3 or purge older logs if not required.
- Do not let VPC flow logs run indefinitely if you do not need it.

https://aws.amazon.com/premiumsupport/knowledge-center/cloudwatch-understand-and-reduce-charges/

RDS-optimization: -

Right sizing

For production and pre-prod downsize the RDS instances based on the CPU/memory utilisation if the Database is not going to scale anytime soon. Use EBS autoscaling on RDS to elastically increase the volume size automatically. If you have provisioned Iops enabled then check if the Iops are being used appropriately.

Recommendations

- ☛ Check this → Use Instance Scheduler solution to schedule RDS stop start for Dev/Test environments, if possible, for efficient use of capacity https://aws.amazon.com/solutions/implementations/instancescheduler/

- ☛ Use Trusted Advisor to find Idle RDS instances

Right Pricing

Recommendations

- ☛ Check this → Reserve price recommendations for RDS

Additional steps

1. RDS monthly cost by instance type : Total RDS cost.
2. RDS Reserved Instance Coverage : Check Reservation coverage.
3. RDS monthly cost for on-demand instances (excluding reservation) :- Find the RDS instances running which are not reserved.

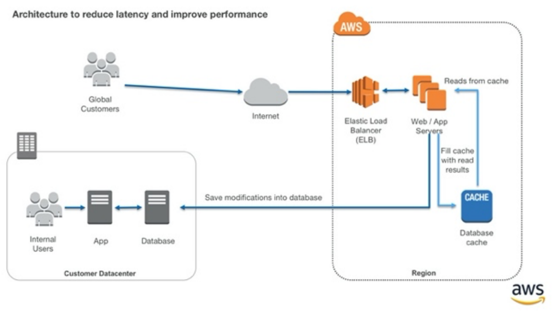
Right Processor Strategy:-

Graviton2 instances family includes several new performance optimizations such as larger L1 and L2 caches per core, higher Amazon Elastic Block Store (EBS) throughput than comparable x86 instances, fully encrypted RAM, and many others as detailed on this page. You can benefit from these optimizations with minimal effort, by provisioning or migrating your RDS instances today.

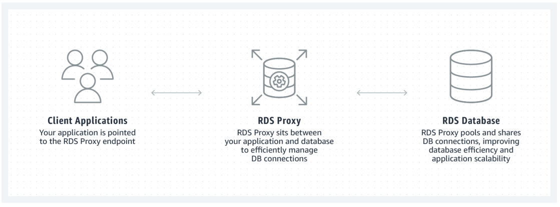
https://aws.amazon.com/blogs/database/key-considerations-in-moving-to-graviton2-for-amazon-rds-and-amazon-aurora-databases/

Right Architecture

Combine RDS database with Elasticache for caching /serving requests to database. This reduces the burden on RDS db and therefore better performance and on increase in cost


Architecture to reduce latency and improve performance
AWS
Global Customers
Internet
Elastic Load Balancer (ELB)
Web / App Servers
Reads from cache
Fill cache with read results
CACHE
Database cache
Save modifications into database
Internal Users
App
Database
Customer Datacenter
Region
aws

Combine RDS database with RDS Proxy for efficient connection management and scaling of RDS DB


Client Applications
Your application is pointed to the RDS Proxy endpoint
RDS Proxy
RDS Proxy sits between your application and database to efficiently manage DB connections
RDS Database
RDS Proxy pools and shares DB connections, improving database efficiency and application scalability

Amazon Elasticache and Amazon Elasticsearch optimization: -

Check the Usage type for each of these services in Cost-explorer as shown in examples in above services:-

Right processor strategy:-

Amazon Elasticache and Amazon ElasticSearch cost and performance optimization using Graviton:

- https://aws.amazon.com/blogs/database/migrate-amazon-elasticache-to-graviton2-processors/

- https://aws.amazon.com/blogs/big-data/increase-amazon-elasticsearch-service-performance-by-upgrading-to-graviton2/

Right sizing:-

Elasticache:-

- Run benchmarks against your Redis or Memcached cluster to understand what should be the ideal size of the Elasticache cluster:-
- https://aws.amazon.com/blogs/database/five-workload-characteristics-to-consider-when-right-sizing-amazon-elasticache-redis-clusters/
- https://d0.awsstatic.com/whitepapers/performance-at-scale-with-amazon-elasticache.pdf

Elastisearch:-

- https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/sizing-domains.html
- TShirt size reference
- Ultrawarm storage

Right Pricing:-

Elasticache:-
- https://aws.amazon.com/elasticache/reserved-cache-nodes/

Elasticsearch:-
- https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/aes-ri.html