Assignment 2 Report

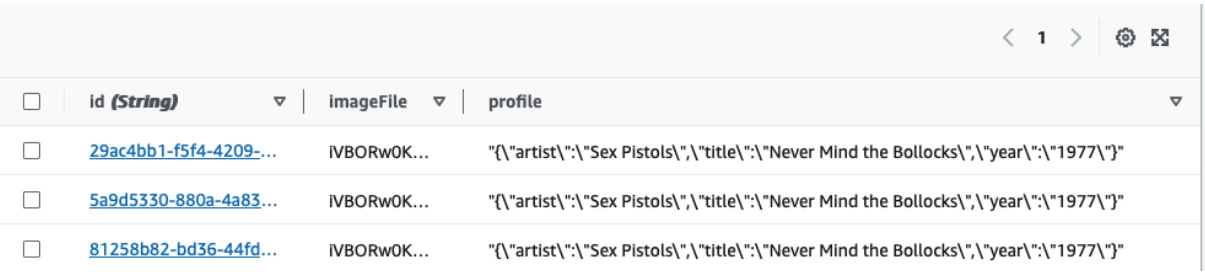
Github Repo:

https://github.com/peihsuan-lin/cs6650/tree/main/assignment2

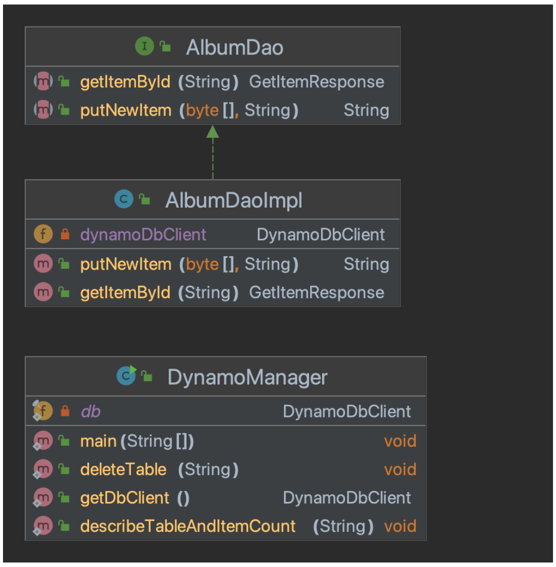
Description of Data Model

The application employ Amazon DynamoDB as its database for storing album information, including metadata and identifiers for album objects.

- **Album Information Storage:** Each album is uniquely identified by a UUID. Associated with this identifier is album metadata, stored in JSON format, and binary data for the image file.
    - id: the primary key, generated by UUID
    - imageFile: the album image converted to byte, image size is downgrade to 500 byte to keep costs down.
    - profile: the artist, title, and year of the album in JSON format
- **Data Retrieval:** The `getItemById` method enables fetching album data by its unique identifier, facilitating quick lookups.
- **Data Insertion:** The `putNewItem` method allows for inserting new albums, automatically generating a unique identifier and storing the image and profile as binary and string data types, respectively.
- **Database Management:** Encapsulate in `DynamoManager` class to provide utility functions, such as connecting and deleting tables within DynamoDB.



□	id (String)	imageFile	profile
□	29ac4bb1-f5f4-4209-...	iVBORw0K...	"{\"artist\":\"Sex Pistols\",\"title\":\"Never Mind the Bollocks\",\"year\":\"1977\"}"
□	5a9d5330-880a-4a83...	iVBORw0K...	"{\"artist\":\"Sex Pistols\",\"title\":\"Never Mind the Bollocks\",\"year\":\"1977\"}"
□	81258b82-bd36-44fd...	iVBORw0K...	"{\"artist\":\"Sex Pistols\",\"title\":\"Never Mind the Bollocks\",\"year\":\"1977\"}"


AlbumDao
getItemById (String) GetItemResponse
putNewItem (byte [], String) String
AlbumDaoImpl
dynamoDbClient DynamoDbClient
putNewItem (byte [], String) String
getItemById (String) GetItemResponse
DynamoManager
db DynamoDbClient
main (String []) void
deleteTable (String) void
getDbClient () DynamoDbClient
describeTableAndItemCount (String) void

Single Server

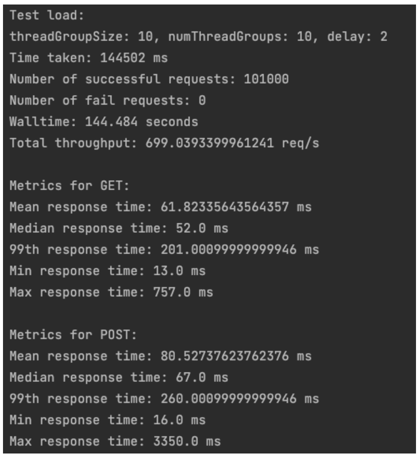
Configuration

1. **Initial Setup with EC2 t2.micro**: The starting configuration with 50 RCU and WCU led to a high CPU usage of 86% and timeouts during peak loads.
2. **First Adjustment to RCU/WCU**: Upon increasing the provisioned RCU and WCU to 500 each, the system achieved full CPU utilization of 100%, handling 597 requests per second under a load of 10 thread groups with 10 threads each.
3. **Modification for Comparable Testing**: To prevent CPU bottlenecks and obtain a fair comparison of system performance, a reduction in CPU usage was considered essential.
4. **Upgrade to t2.medium Instance**: Retaining the same database settings but moving to a t2.medium instance, the system's throughput improved to 881 requests per second, although the CPU peak utilization remained high at 96%.
5. **Switch to t3.small Instance**: Finally, with the transition to a t3.small instance, there was a balance between throughput and CPU efficiency. The system managed 699 requests per second with a peak CPU utilization of 90%.

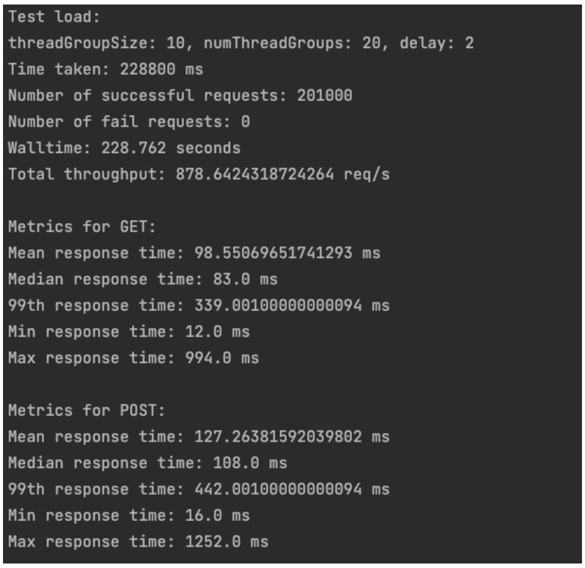
Below is the tabulated performance under varying loads with the final configuration on the t3.small instance, maintaining 500 RCU and WCU:

numThreads	threadGroup	numOfRequest	throughput (req/s)	consumed read (units/s)	consumed write (units/s)	CPU utilization (%)
10	10	101000	699	388	403	90
10	20	201000	878	460	921	89
10	30	301000	999	524	1050	93

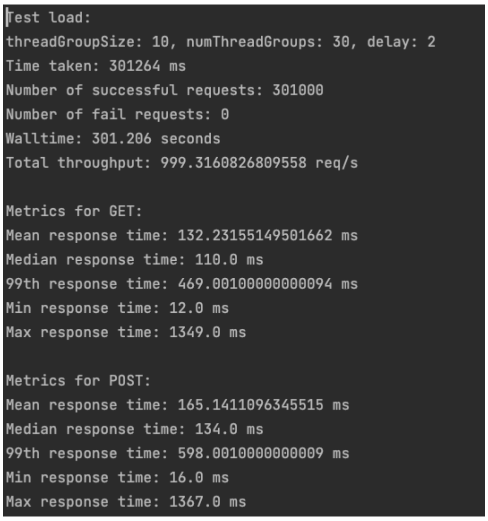
Output windows for a single server



```
Test load:
threadGroupSize: 10, numThreadGroups: 10, delay: 2
Time taken: 144502 ms
Number of successful requests: 101000
Number of fail requests: 0
Walltime: 144.484 seconds
Total throughput: 699.0393399961241 req/s

Metrics for GET:
Mean response time: 61.82335643564357 ms
Median response time: 52.0 ms
99th response time: 201.00099999999946 ms
Min response time: 13.0 ms
Max response time: 757.0 ms

Metrics for POST:
Mean response time: 80.52737623762376 ms
Median response time: 67.0 ms
99th response time: 260.00099999999946 ms
Min response time: 16.0 ms
Max response time: 3350.0 ms
```



```
Test load:
threadGroupSize: 10, numThreadGroups: 20, delay: 2
Time taken: 228800 ms
Number of successful requests: 201000
Number of fail requests: 0
Walltime: 228.762 seconds
Total throughput: 878.6424318724264 req/s

Metrics for GET:
Mean response time: 98.55069651741293 ms
Median response time: 83.0 ms
99th response time: 339.00100000000094 ms
Min response time: 12.0 ms
Max response time: 994.0 ms

Metrics for POST:
Mean response time: 127.26381592039802 ms
Median response time: 108.0 ms
99th response time: 442.00100000000094 ms
Min response time: 16.0 ms
Max response time: 1252.0 ms
```

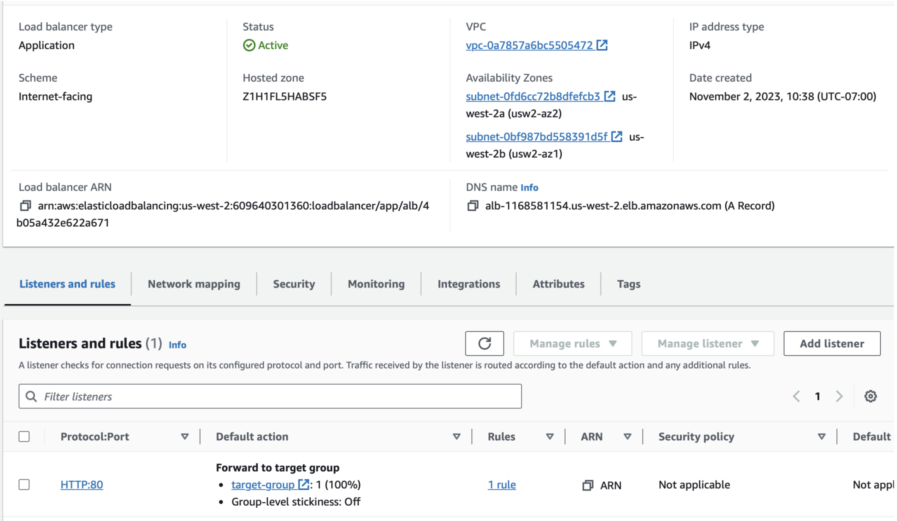


```
Test load:
threadGroupSize: 10, numThreadGroups: 30, delay: 2
Time taken: 301264 ms
Number of successful requests: 301000
Number of fail requests: 0
Walltime: 301.206 seconds
Total throughput: 999.3160826809558 req/s

Metrics for GET:
Mean response time: 132.23155149501662 ms
Median response time: 110.0 ms
99th response time: 469.00100000000094 ms
Min response time: 12.0 ms
Max response time: 1349.0 ms

Metrics for POST:
Mean response time: 165.1411096345515 ms
Median response time: 134.0 ms
99th response time: 598.0010000000009 ms
Min response time: 16.0 ms
Max response time: 1367.0 ms
```

Two Servers with Load Balancer

Load Balance & Target Group Configuration

Employing a pair of t3.small EC2 instances grouped within a target group, where a load balancer directs incoming requests between the two.



Load balancer type	Status	VPC	IP address type
Application	Active	vpc-0a7857a6bc5505472	IPv4

Scheme	Hosted zone	Availability Zones	Date created
Internet-facing	Z1H1FL5HABSF5	subnet-0fd6cc72b8dfefcb3 us-west-2a (usw2-az2) subnet-0bf987bd558391d5f us-west-2b (usw2-az1)	November 2, 2023, 10:38 (UTC-07:00)

Load balancer ARN
arn:aws:elasticloadbalancing:us-west-2:609640301360:loadbalancer/app/alb/4b05a432e622a671

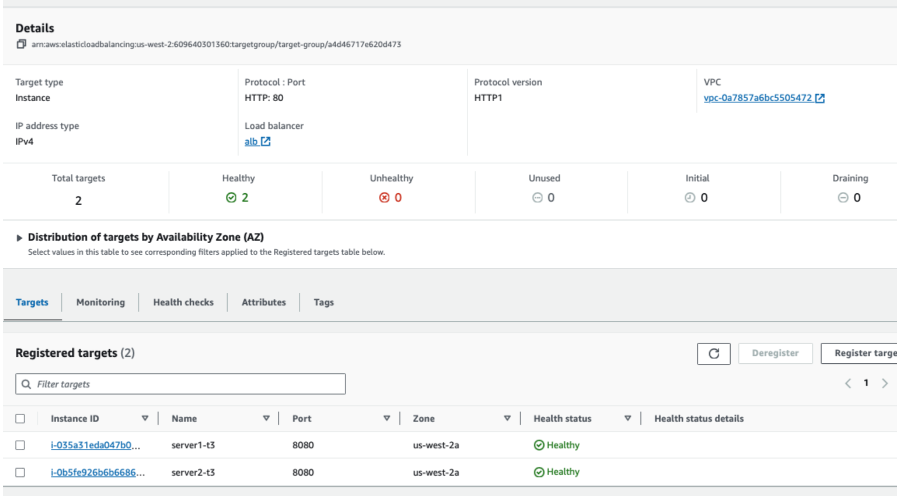
DNS name Info
alb-1168581154.us-west-2.elb.amazonaws.com (A Record)

Listeners and rules (1) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

□	Protocol:Port	Default action	Rules	ARN	Security policy	Default
□	HTTP:80	Forward to target group • target-group: 1 (100%) • Group-level stickiness: Off	1 rule	ARN	Not applicable	Not appl



Details
arn:aws:elasticloadbalancing:us-west-2:609640301360:targetgroup/target-group/a4d46717e620d473

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	vpc-0a7857a6bc5505472
IP address type	Load balancer		
IPv4	alb		

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
2	2	0	0	0	0

Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

Registered targets (2)

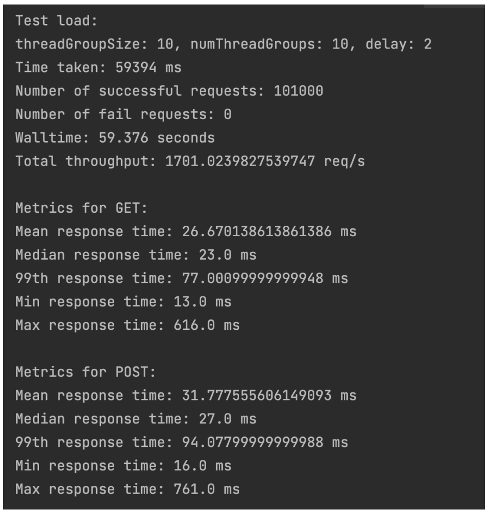
Filter targets

□	Instance ID	Name	Port	Zone	Health status	Health status details
□	i-035a31eda047b0...	server1-t3	8080	us-west-2a	Healthy	
□	i-0b5fe926b6b6686...	server2-t3	8080	us-west-2a	Healthy	

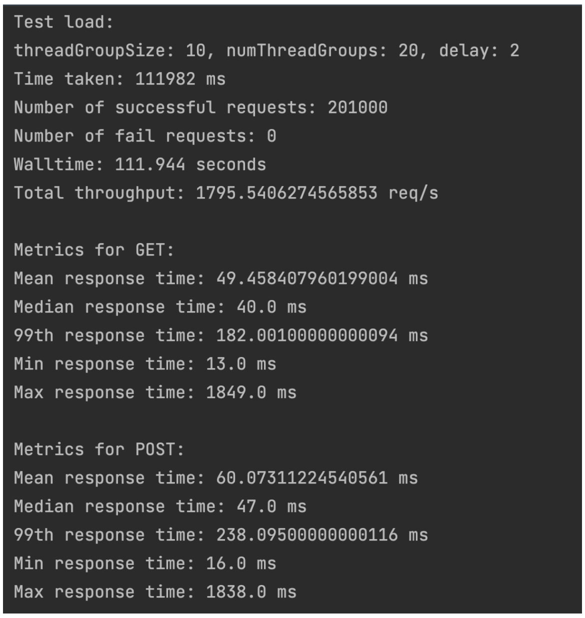
numThreads	threadGroup	numOfRequest	throughput (req/s)	consumed read (units/s)	consumed write (units/s)	CPU utilization (%)
10	10	101000	1701	767	1535	61
10	20	201000	1795	996	1932	79
10	30	301000	974	905	1809	78

- For the same load of 10 threads in 10 thread groups, the load balancer increases the throughput to 1701 requests per second with only 61% CPU utilization. This indicates a significant performance enhancement, as the two servers together can handle more than double the requests of a single server at a lower CPU utilization.
- When increasing to 20 thread groups, the throughput further improves to 1795 requests per second, although CPU utilization climbs to 79%, which is still lower than the single server's peak of 93%.
- At 30 thread groups is a reduction in throughput with a CPU utilization of 78%. This decrease could be due to reaching a bottleneck.

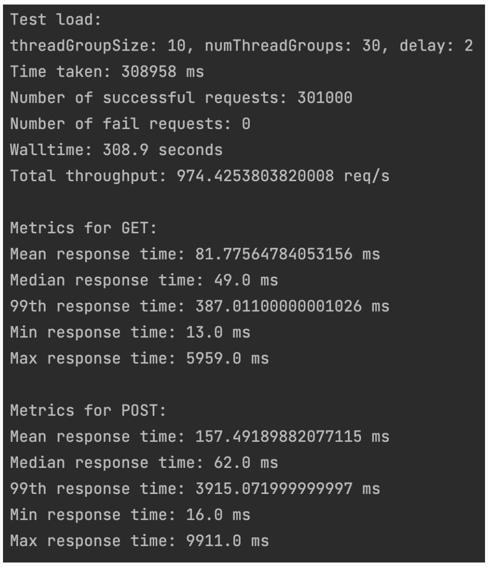
Output windows for a two load balanced servers



```
Test load:
threadGroupSize: 10, numThreadGroups: 10, delay: 2
Time taken: 59394 ms
Number of successful requests: 101000
Number of fail requests: 0
Walltime: 59.376 seconds
Total throughput: 1701.0239827539747 req/s

Metrics for GET:
Mean response time: 26.670138613861386 ms
Median response time: 23.0 ms
99th response time: 77.00099999999948 ms
Min response time: 13.0 ms
Max response time: 616.0 ms

Metrics for POST:
Mean response time: 31.777555606149093 ms
Median response time: 27.0 ms
99th response time: 94.07799999999988 ms
Min response time: 16.0 ms
Max response time: 761.0 ms
```



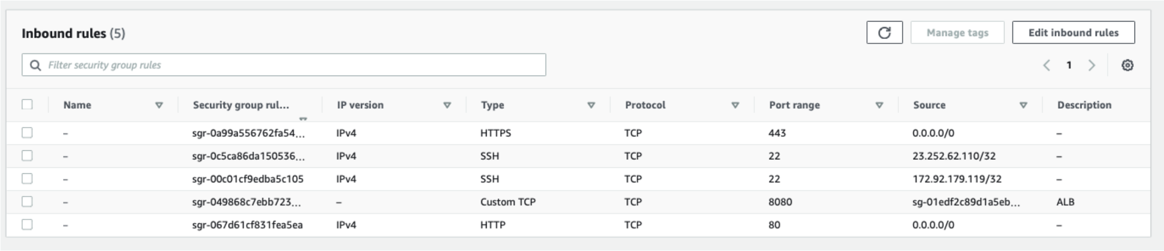
```
Test load:
threadGroupSize: 10, numThreadGroups: 20, delay: 2
Time taken: 111982 ms
Number of successful requests: 201000
Number of fail requests: 0
Walltime: 111.944 seconds
Total throughput: 1795.5406274565853 req/s

Metrics for GET:
Mean response time: 49.458407960199004 ms
Median response time: 40.0 ms
99th response time: 182.00100000000094 ms
Min response time: 13.0 ms
Max response time: 1849.0 ms

Metrics for POST:
Mean response time: 60.07311224540561 ms
Median response time: 47.0 ms
99th response time: 238.09500000000116 ms
Min response time: 16.0 ms
Max response time: 1838.0 ms
```



```
Test load:
threadGroupSize: 10, numThreadGroups: 30, delay: 2
Time taken: 308958 ms
Number of successful requests: 301000
Number of fail requests: 0
Walltime: 308.9 seconds
Total throughput: 974.4253803820008 req/s

Metrics for GET:
Mean response time: 81.77564784053156 ms
Median response time: 49.0 ms
99th response time: 387.01100000001026 ms
Min response time: 13.0 ms
Max response time: 5959.0 ms

Metrics for POST:
Mean response time: 157.49189882077115 ms
Median response time: 62.0 ms
99th response time: 3915.071999999997 ms
Min response time: 16.0 ms
Max response time: 9911.0 ms
```

Set up screenshots

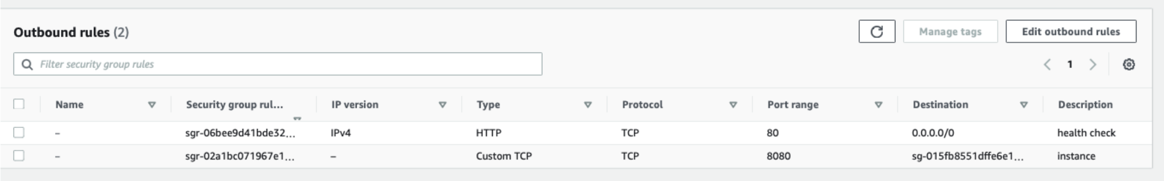
- instance security group



Inbound rules (5)

☐	Name	Security group rul...	IP version	Type	Protocol	Port range	Source	Description
☐	–	sgr-0a99a556762fa54...	IPv4	HTTPS	TCP	443	0.0.0.0/0	–
☐	–	sgr-0c5ca86da150536...	IPv4	SSH	TCP	22	23.252.62.110/32	–
☐	–	sgr-00c01cf9edba5c105	IPv4	SSH	TCP	22	172.92.179.119/32	–
☐	–	sgr-049868c7ebb723...	–	Custom TCP	TCP	8080	sg-01edf2c89d1a5eb...	ALB
☐	–	sgr-067d61cf831fea5ea	IPv4	HTTP	TCP	80	0.0.0.0/0	–

- load balancer security group



Outbound rules (2)

☐	Name	Security group rul...	IP version	Type	Protocol	Port range	Destination	Description
☐	–	sgr-06bee9d41bde32...	IPv4	HTTP	TCP	80	0.0.0.0/0	health check
☐	–	sgr-02a1bc071967e1...	–	Custom TCP	TCP	8080	sg-015fb8551dffe6e1...	instance

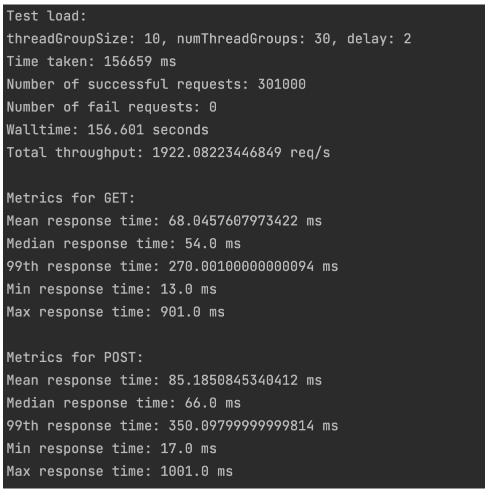
Tune the System

Adjusted based on a load of 10 threads in 30 thread groups.

DataBase Improvements

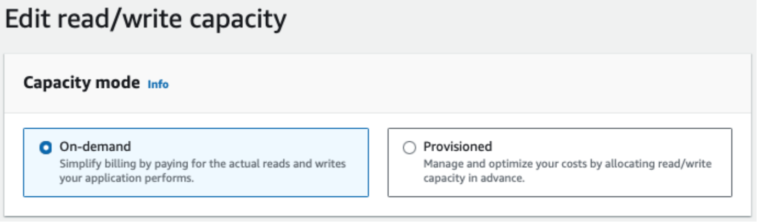
The transition to on-demand capacity yielded a marked improvement in system throughput, increased from 974 to 1922 requests per second. The adoption of on-demand capacity enabled the system to better harness DynamoDB's automatic scaling capabilities, ensuring robust performance and resource optimization in response to high loads.

Configuration	Throughput (req/s)	Consumed Read (units/s)	Consumed Write (units/s)	CPU Utilization (%)
provisioned	974	905	1809	78
on demand	1922	973	1948	81



```
Test load:
threadGroupSize: 10, numThreadGroups: 30, delay: 2
Time taken: 156659 ms
Number of successful requests: 301000
Number of fail requests: 0
Walltime: 156.601 seconds
Total throughput: 1922.08223446849 req/s

Metrics for GET:
Mean response time: 68.0457607973422 ms
Median response time: 54.0 ms
99th response time: 270.00100000000094 ms
Min response time: 13.0 ms
Max response time: 901.0 ms

Metrics for POST:
Mean response time: 85.1850845340412 ms
Median response time: 66.0 ms
99th response time: 350.09799999999814 ms
Min response time: 17.0 ms
Max response time: 1001.0 ms
```

Set up screenshots



Edit read/write capacity

Capacity mode

● On-demand
Simplify billing by paying for the actual reads and writes your application performs.

○ Provisioned
Manage and optimize your costs by allocating read/write capacity in advance.

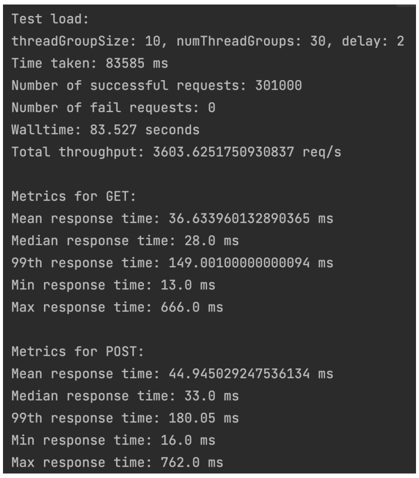
Server Improvements

Increase Instance number

The adoption of on-demand capacity was sustained as I increased the number of t3.small servers.

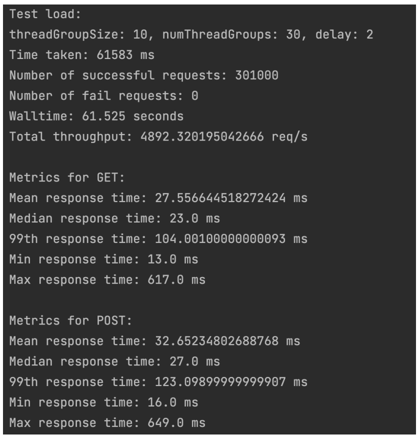
Configuration	Throughput (req/s)	Consumed Read (units/s)	Consumed Write (units/s)	CPU Utilization (%)
2 servers	1922	973	1948	81
5 servers	3603	1333	2663	56
9 servers	4892	1805	3609	45

- **With 5 t3.small Servers**: The system's throughput escalated to 3603 requests per second, with a marked decrease in CPU utilization to 56%. This indicates that spreading the load across more servers leads to more efficient request handling and less CPU strain.



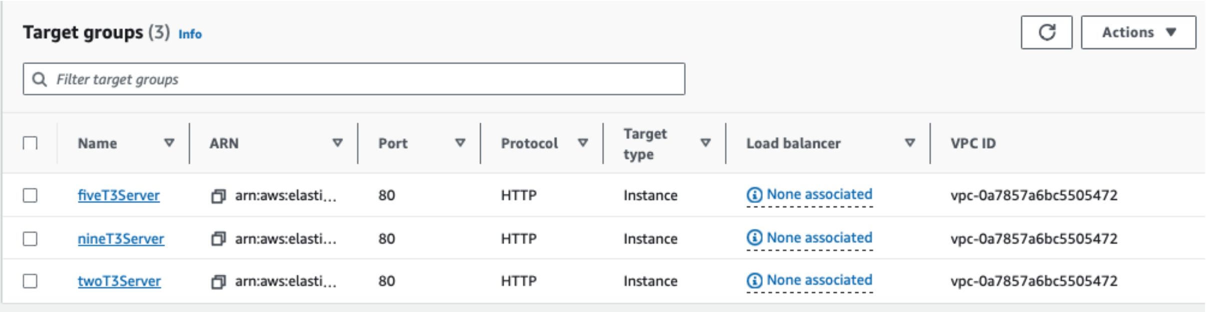
```
Test load:
threadGroupSize: 10, numThreadGroups: 30, delay: 2
Time taken: 83585 ms
Number of successful requests: 301000
Number of fail requests: 0
Walltime: 83.527 seconds
Total throughput: 3603.6251750930837 req/s

Metrics for GET:
Mean response time: 36.633960132890365 ms
Median response time: 28.0 ms
99th response time: 149.00100000000094 ms
Min response time: 13.0 ms
Max response time: 666.0 ms

Metrics for POST:
Mean response time: 44.945029247536134 ms
Median response time: 33.0 ms
99th response time: 180.05 ms
Min response time: 16.0 ms
Max response time: 762.0 ms
```

- **With 9 t3.small Servers**: The benefits of scaling out became even more evident. Throughput reached 4892 requests per second, CPU utilization further decreased to 45%, reinforcing the effectiveness of this scaling strategy.



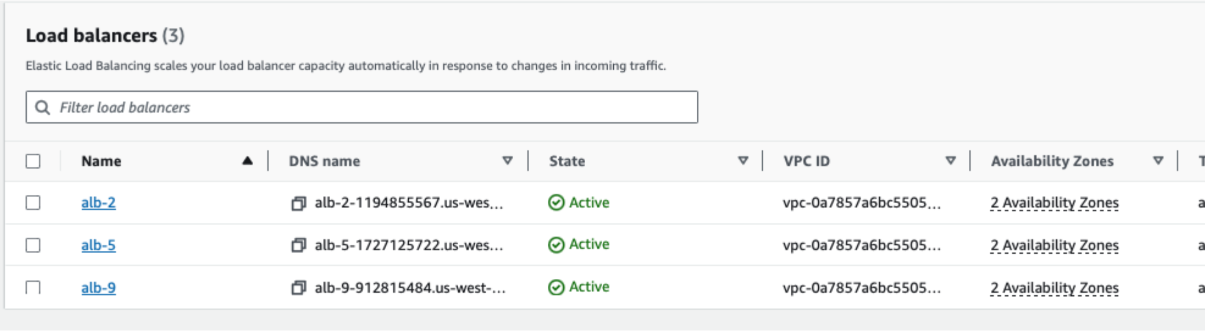
```
Test load:
threadGroupSize: 10, numThreadGroups: 30, delay: 2
Time taken: 61583 ms
Number of successful requests: 301000
Number of fail requests: 0
Walltime: 61.525 seconds
Total throughput: 4892.320195042666 req/s

Metrics for GET:
Mean response time: 27.556644518272424 ms
Median response time: 23.0 ms
99th response time: 104.00100000000093 ms
Min response time: 13.0 ms
Max response time: 617.0 ms

Metrics for POST:
Mean response time: 32.65234802688768 ms
Median response time: 27.0 ms
99th response time: 123.09899999999907 ms
Min response time: 16.0 ms
Max response time: 649.0 ms
```

Set up screenshots



Target groups (3)

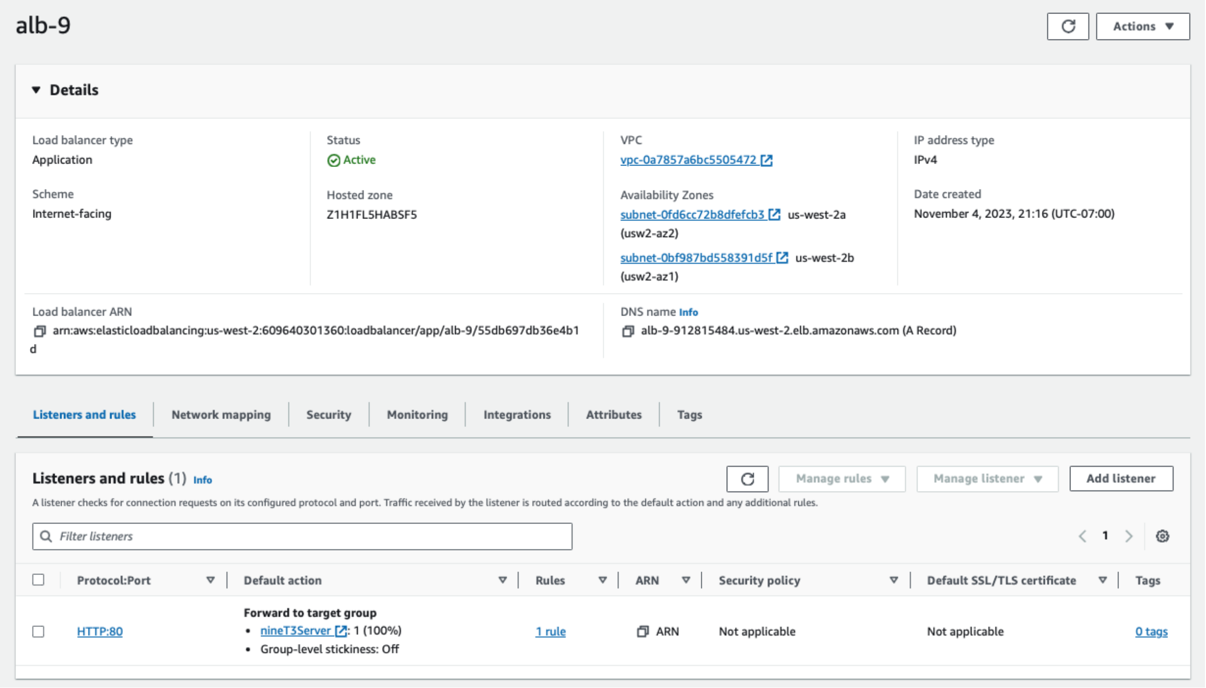
□	Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
□	fiveT3Server	arn:aws:elasti...	80	HTTP	Instance	None associated	vpc-0a7857a6bc5505472
□	nineT3Server	arn:aws:elasti...	80	HTTP	Instance	None associated	vpc-0a7857a6bc5505472
□	twoT3Server	arn:aws:elasti...	80	HTTP	Instance	None associated	vpc-0a7857a6bc5505472



Load balancers (3)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

□	Name	DNS name	State	VPC ID	Availability Zones	T
□	alb-2	alb-2-1194855567.us-wes...	Active	vpc-0a7857a6bc5505...	2 Availability Zones	a
□	alb-5	alb-5-1727125722.us-wes...	Active	vpc-0a7857a6bc5505...	2 Availability Zones	a
□	alb-9	alb-9-912815484.us-west-...	Active	vpc-0a7857a6bc5505...	2 Availability Zones	a



alb-9

Details

Load balancer type
Application

Scheme
Internet-facing

Status
Active

Hosted zone
Z1H1FL5HABSF5

VPC
vpc-0a7857a6bc5505472

Availability Zones
subnet-0fd6cc72b8dfefcb3 us-west-2a (usw2-az2)
subnet-0bf987bd558391d5f us-west-2b (usw2-az1)

IP address type
IPv4

Date created
November 4, 2023, 21:16 (UTC-07:00)

Load balancer ARN
arn:aws:elasticloadbalancing:us-west-2:609640301360:loadbalancer/app/alb-9/55db697db36e4b1d

DNS name Info
alb-9-912815484.us-west-2.elb.amazonaws.com (A Record)

Listeners and rules

Listeners and rules (1) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

□	Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	Tags
□	HTTP:80	**Forward to target group** • nineT3Server: 1 (100%) • Group-level stickiness: Off	1 rule	ARN	Not applicable	Not applicable	0 tags

Overall Throughput Improvement

- Moving from a single server to two load-balanced servers, then optimizing to on-demand capacity and scaling to 9 servers, the system's throughput improved significantly:
  - From 999 req/s (single server) to 4892 req/s (9 load balanced servers), resulting in a **390% improvement** in throughput.
  - CPU utilization was optimized from a peak of 93% (single server) to 45% (9 servers), showing a more efficient system.

DB Config	Server Config	Throughput (req/s)	Consumed Read (units/s)	Consumed Write (units/s)	CPU Utilization (%)
Provisioned	single server	999	524	1050	93
Provisioned	2 load balanced server	974	905	1809	78
On-demand	2 load balanced server	1922	973	1948	81
On-demand	5 load balanced server	3603	1333	2663	56
On-demand	9 load balanced server	4892	1805	3609	45