NewgenONE Content Cloud

Performance Report

Version: 2024.1

Newgen Software Technologies Ltd.

Executive Summary

At Newgen's Enterprise Technology Labs, Newgen benchmarked the performance of NewgenONE Content Cloud on the Microsoft Azure Cloud Platform in April 2024. The results point to NCC's ability to scale up to thousands of users and millions of transactions—all while maintaining a high-quality, quick-response experience for users.

NewgenONE Content Cloud is a REST-based API platform offering, developed on contemporary microservices architecture and design patterns. It provides a cloud-ready backend services suite for business customers to build engaging software solutions.

This performance benchmarking test was performed with the JMeter tool. The Apache JMeter™ application is open-source software, a 100% pure Java application designed to load-test functional behavior and measure performance.

The benchmark testing was successful, with Business Operations **Generate Auth Code, Generate Auth Token, Create Folder with Data Class, Upload Document with Data Class, Search Folder on Data Class, Search Document on Data Class, Get Folder Properties by ID, Download Document, Get Document Properties by ID, Get List of all Documents, Get List of all Folders and Upload to Blob with average time of 0.058 seconds while running with 5,100 concurrent users in 2 hours and 45 minutes run of performance benchmarking with over 2.6 Million Folders and over 136 Million Documents** in the system. During this time, the system performance was not degraded.

Table of contents

1 Introduction 4
1.1 NewgenONE Content Cloud microservices 4
2 Benchmark testing overview 5
2.1 Server throughput 5
2.2 Business operations 5
3 Test environment details 6
4 Business operations 8
4.1 Generate Auth code 8
4.2 Generate Auth token 8
4.3 Create folder with data class 8
4.4 Get folders list 8
4.5 Get folder by ID and fetch folder properties 8
4.6 Search folder on data class field value 8
4.7 Generate Blob URI for document upload 8
4.8 Upload document to Blob 8
4.9 Get documents list 9
4.10 Get document by ID and fetch document properties 9
4.11 Search document on data class field value 9
4.12 Download document 9
5 Server configurations 9
5.1 Microservices server 9
5.2 Database server 9
6 Test data set 10
7 Benchmark testing results 10
7.1 Server capability 10
7.1.1 Number of concurrent users 10
7.1.2 Number of concurrent requests 11
7.1.3 Continuous performance over time 11
7.1.4 High volume support 11
7.2 Business operations 12
8 Resource usage 12
9 Conclusion 18
10 Appendix A 19

1 Introduction

NewgenONE Content Cloud (NCC) is a modern API platform that empowers businesses to build next-generation software solutions. Built on a robust microservices architecture and hosted on Microsoft Azure, NCC offers a suite of cloud-ready backend services. This approach ensures:

- **Security**: NCC leverages best practices for data and application protection, keeping your information and workflows safe.
- **Scalability**: Easily adapt your infrastructure to meet evolving business needs with NCC's flexible and scalable architecture.
- **Performance**: Experience consistently high performance, even under heavy workloads, thanks to NCC's cloud-native design.
- **Availability**: Minimize downtime and ensure continuous operation with NCC's built-in fault tolerance and disaster recovery capabilities.

With NCC, businesses can focus on building engaging software experiences while enjoying the benefits of a secure, scalable, and highly available cloud platform.

1.1 NewgenONE Content Cloud microservices

The following core services are provided in NewgenONE Content Cloud to fulfill the content service platform requirements:

- Annotation Service
- Application Service
- Cabinet Service
- Content Service
- Data Class Service
- Document Download Service
- Folder Service
- Notes Service
- Roles Service
- Search Service
- Secret Key Service
- Security Classification Service
- User Service

2 Benchmark testing overview

The benchmarking tests were designed to evaluate the performance and scalability of the Business Operation of NewgenONE Content Cloud on Microsoft Azure Cloud Platform. Performance benchmarking is an industry-standard way of demonstrating the performance capability of any given software product. However, NewgenONE Content Cloud provides scalability and provides APIs to achieve different purposes for content Management solutions, so certain types of tests are essential for a particular benchmark to properly depict the product's performance. For that, NCC performance benchmarks directly measure performance rates for:

- Demonstrating Server Throughput
- Response Times for Business Operations

2.1 Server throughput

NewgenONE Content Cloud was subjected to various stress conditions to check its ability to scale up to handle increasing loads.

- Number of Concurrent Users
- Number of Concurrent Requests
- Continuous Performance over time
- High Volume support

For more details, see Benchmark testing results.

2.2 Business operations

Response time for various Business Operations with increasing user loads was verified.

- Generate Auth Code
- Generate Auth Token from Code
- Create Folder with Data Class
- Getting Folders List
- Search Document on Data Class Field Value
- Get Folder by ID and Fetch Folder Properties
- Search Folder on Data Class
- Getting Documents List
- Get Document by ID and Fetch Document Properties
- Generate Blob URI for Document Upload
- Upload Document to Blob
- Download Document

For more details, see Benchmark testing results.

3 Test environment details

The test environment was running as Docker containers on Microsoft Azure Kubernetes service. Web components were running on the underlying web server - Spring Boot embedded Tomcat (Spring Boot version 2.7.15). Microsoft Azure Cosmos DB (Mongo API version 4.0) was the database.

Three load generators were used for generating load through JMeter, each running on an Azure VM with 4 vCPU and 16 GB RAM.

Think time of 6 seconds and Ramp-up time of 1.5 seconds were configured to simulate real-world usage of the platform.

Node resource allocation

Machine Class	Nodes Count	CPU (Core)	Memory (GB)
Standard_E8as_v4 8 Core CPU 64 GB RAM, 12800 IOPS	2	8	64

Kubernetes containers resource allocation

Container Name	Total Pods	CPU (Core)	Memory (GB)
Content Service	1	1.75	5
Folder Service	1	1.75	5
Dataclass Service	1	2	5
Logging Service	1	3.5	5
App Service	1	2.25	4
Zuul Service	2	2.5	10
Storage Service	1	1.25	4

Database resource allocation

Database Configuration	
Database	**Provisioned Throughput (RU/s)**
CosmosDB - ecmdata	3,95,000
CosmosDB - ecmdataext	10,000
Total: 4,05,000 RU/s	
Database	**Provisioned DTUs**
MS SQL DB – Standard S1	20

Database collections

Collection	Database	Mode
auditLogs	ecmdata	Manual
content	ecmdata	Shared Auto-scale
contentLocation	ecmdata	Shared Auto-scale
dataclass	ecmdata	Shared Auto-scale
folder	ecmdata	Shared Auto-scale
storageCredentials	ecmdata	Shared Auto-scale
storageLocation	ecmdata	Shared Auto-scale
tenant	ecmdata	Shared Auto-scale
users	ecmdata	Shared Auto-scale
passwordChangeToken	ecmdata	Manual
app	ecmdataext	Shared Auto-scale
globalTag	ecmdataext	Shared Auto-scale
mail	ecmdataext	Manual
roles	ecmdataext	Shared Auto-scale
secretkeys	ecmdataext	Shared Auto-scale
securityClass	ecmdataext	Shared Auto-scale
MicroAPI	ecmdataext	Manual
subscription	ecmdataext	Manual

Server configuration

Component	Software
Microservices Server	Spring boot Embedded tomcat (Spring boot ver. 2.7.15)
Database Service	Azure Cosmos DB (Mongo API 4.0) and Azure MS SQL
Load Generator	Apache JMeter 5.4.3

4 Business operations

This section describes the various business operations that were used for the test.

4.1 Generate Auth code

The Code API allows you to obtain the authorization code that helps you to obtain an authentication token.

4.2 Generate Auth token

The Token API allows you to obtain an Access Token in exchange for the Auth Code.

4.3 Create folder with data class

The Create Folder API allows you to create a folder or sub-folder with a data class associated with it in the folder. repository.

4.4 Get folders list

The Getting list of all folders API allows you to get the list of all folders for a particular tenant.

4.5 Get folder by ID and fetch folder properties

The Get folder by ID API allows you to get the details of a folder using its folder ID.

4.6 Search folder on data class field value

The Search Folder on data class API allows you to search a folder based on the data class fields.

4.7 Generate Blob URI for document upload

The Generate file URI API allows you to upload documents of any size by generating the secure blob URI for direct upload to Azure blob storage.

4.8 Upload document to Blob

The Upload to Blob API directly uploads the file binary content to blob storage.

4.9 Get documents list

The Getting list of all documents API allows you to get the list of all documents for a particular tenant.

4.10 Get document by ID and fetch document properties

The Get document by ID API allows you to get the details of a document using its folder ID.

4.11 Search document on data class field value

The Search Document on data class API allows you to search a document based on the data class fields.

4.12 Download document

The Download document API retrieves content from the blob storage and provides a secure content URL from which the binary data of the document can be downloaded.

5 Server configurations

This section describes in more detail some of the server configurations used in the benchmark testing. The Microservices in the test environment are running as Docker containers on Azure Kubernetes Service with one microservice in each container.

5.1 Microservices server

Microservices are running on Apache Tomcat integrated with Spring Boot 2.7.15. Each service uses a separate Apache Tomcat.
For details on container resource allocation, see the Containers resource allocation table.

5.2 Database server

NewgenONE Content Cloud has a high number of backend transactions, so a high-configuration database is required to fulfill the need for benchmarking tests.
Database Accounts are running as a service on Microsoft Azure, and we have used CosmosDB (Mongo API v4.0) and MS SQL DB. For database service configurations, please refer to the Database resource allocation table.

6 Test data set

This benchmark test was performed on a large number of datasets, where we used **2.6 million folders** and **136 million documents** in the system.
The document size was **500 KB**. All documents used in performance benchmarking were PDF-type.
The total number of users was **5100**. All the users performed different operations concurrently.
A total of **10 tenants** were used in performance benchmarking.

User distribution on different operations is tabled below:

Thread Groups	Concurrent Users	Distribution %
Thread Group 0 - Generate Auth Code	5100	100
Thread Group 0 - Generate Auth Token from Code	5100	100
Thread Group 1 - Create Folder with Data class	510	10
Thread Group 2 - Upload File with Data class	765	15
Thread Group 3 - Search folder on Data class	765	15
Thread Group 4 - Search Document on Data class	1020	20
Thread Group 5 - Get folder by ID and fetch folder properties	510	10
Thread Group 6 - Download file	510	10
Thread Group 7 - Get Document by ID and fetch document properties	510	10
Thread Group 8 - Get list of all documents	255	5
Thread Group 9 - Get list of all folders	255	5
Total	**5100**	**100**

7 Benchmark testing results

This section describes the benchmark testing results.

7.1 Server capability

This section describes the server's capability.

7.1.1 Number of concurrent users

Description

The Concurrent user test simulates multiple users simultaneously performing different types of operations randomly in NewgenONE Content Cloud. This test determines the average number of users handled by the CPU with an acceptable response time.

Results

- The system was benchmarked with 5100 concurrent users with 23 containers running on a cluster of 2 nodes with the configuration of 8 core CPU and 64 GB RAM per node with an average response time of 0.058 seconds.

7.1.2 Number of concurrent requests

Description

The Concurrent request test simulates multiple users simultaneously performing different types of operations randomly in NewgenONE Content Cloud. This test determines the number of transactions served by the NCC server in a minute.

Results

- The system served 534 concurrent requests in a second.

7.1.3 Continuous performance over time

Description

The Continuous Performance test simulates the simultaneous requests of users continuously over an extended period of time. This test determines the response time of the NewgenONE Content Cloud server when it is used continuously over an extended period of time and also measures any degradation in server performance. For this test, the performance of the server was benchmarked over a period of 2 hours and 45 minutes.

Results:

- Handled 51,72,422 requests in 2 hours and 45 minutes with no degradation in server performance. Despite the heavy load, response times remained constant and low throughout.

7.1.4 High volume support

Description

The High-Volume test simulates the simultaneous requests of users on a huge number of documents and folders simultaneous requests.

Results

- The system was benchmarked with 2.6 million Folders and 136 million documents with no degradation in performance.

7.2 Business operations

Business Operation	Total Concurrent Users	Average Response Time in Seconds
Generate Auth Code	5100	0.313
Generate Auth Token from Code	5100	0.111
Create Folder with Data Class	510	0.068
Getting Folders List	255	0.039
Get Folder by ID and Fetch Folder Properties	510	0.047
Search Folder on Data Class Field Value	765	0.062
Generate Blob URI for Document Upload	765	0.131
Upload Document to Blob	765	0.067
Getting Documents List	255	0.042
Get Document by ID and Fetch Document Properties	510	0.043
Search Document on Data Class Field Value	1020	0.040
Download Document	510	0.060

For a complete JMeter summary report, see Appendix A.

8 Resource usage

Resource-usage results were obtained by Container Insights in Microsoft Azure Kubernetes Service. Overall, the resource usage on the servers shows that hardware was not a significant bottleneck during the benchmarking tests; the results of the tests, therefore, demonstrate the maximum throughput of the software.

Resource usage of all microservices

Services	CPU Utilization (Max %)	Memory Utilization (Max %)
App Service	51%	36%
Content Service	40%	9%
Folder Service	54%	16%
Data Class Service	52%	9%
Storage Service	35%	48%
Logging Service	11%	11%
Zuul Service	33%	52%

Resource usage of database services

Service	Metrics	Resource Utilization %
Azure Cosmos DB	Normalized RU Consumption (Max) Normalized RU Consumption (Avg)	100% 47%
Azure MS SQL DB	DTU Percentage (Max) DTU Percentage (Avg)	29% 25%

Below you can find CPU and Memory utilization metrics of the Services and Database. Images below demonstrate server health at peak load.

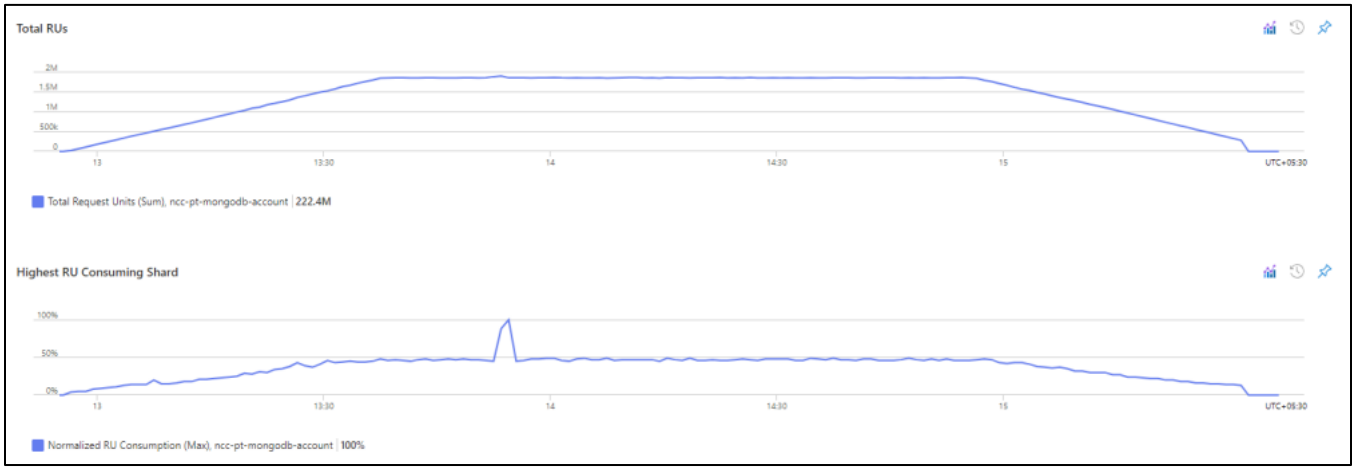

Figure 8.1: CosmosDB - RUs utilization

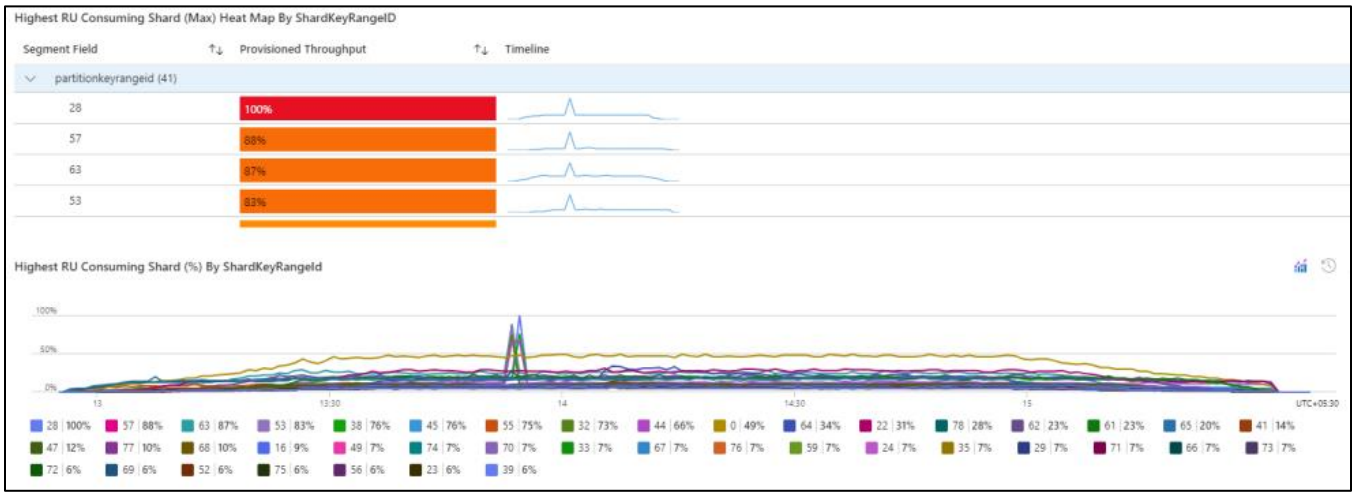

Figure 8.2: CosmosDB - highest RU consuming shards

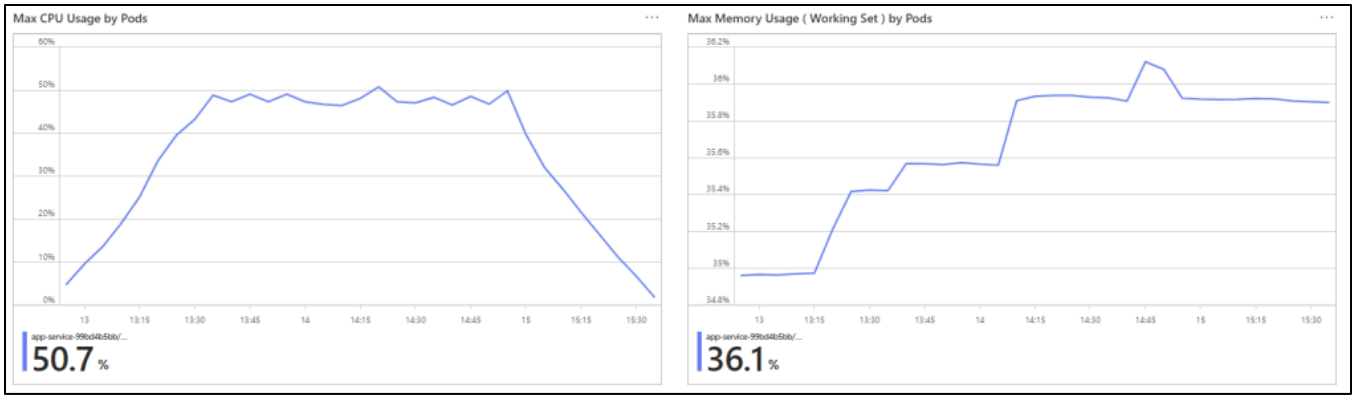

Figure 8.3: App Service - CPU and memory utilization (max)

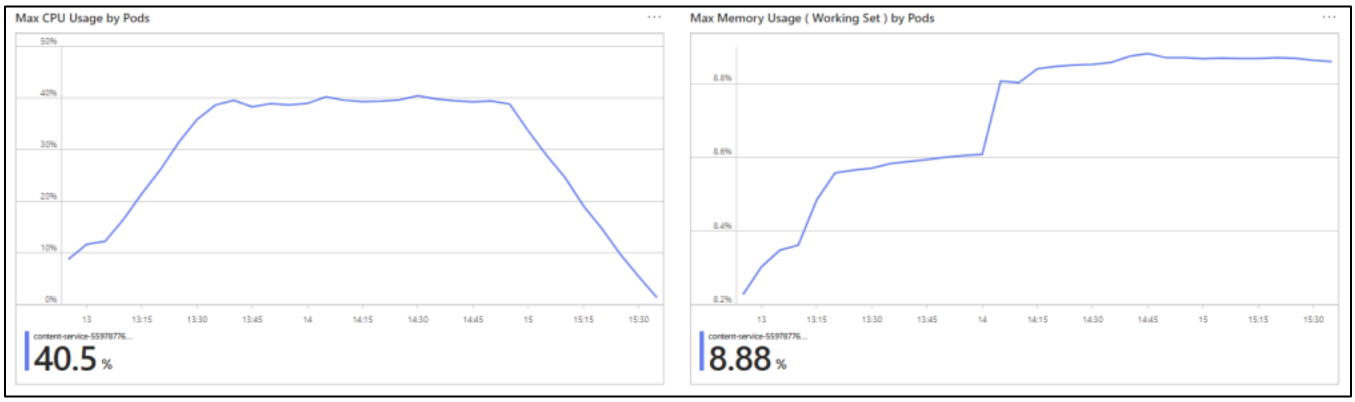

Figure 8.4: Content Service - CPU and memory utilization (max)

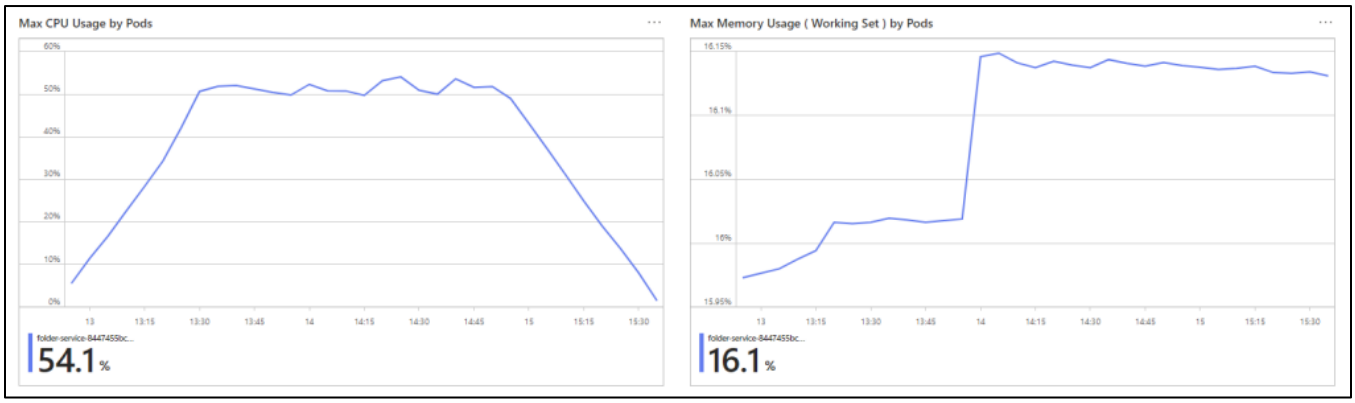

Figure 8.5: Folder Service - CPU and Memory Utilization (max)

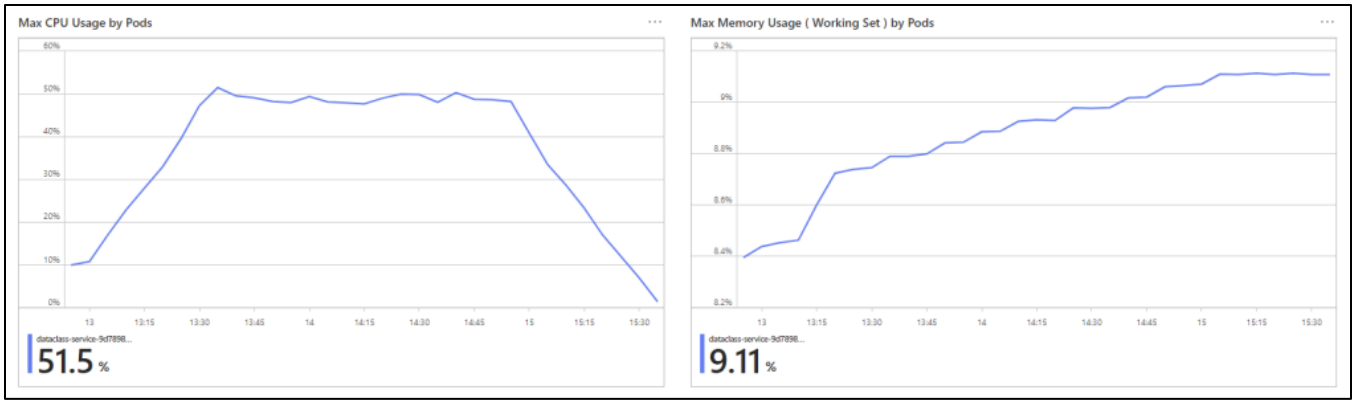

Figure 8.6: Data Class Service - CPU and memory utilization (max)

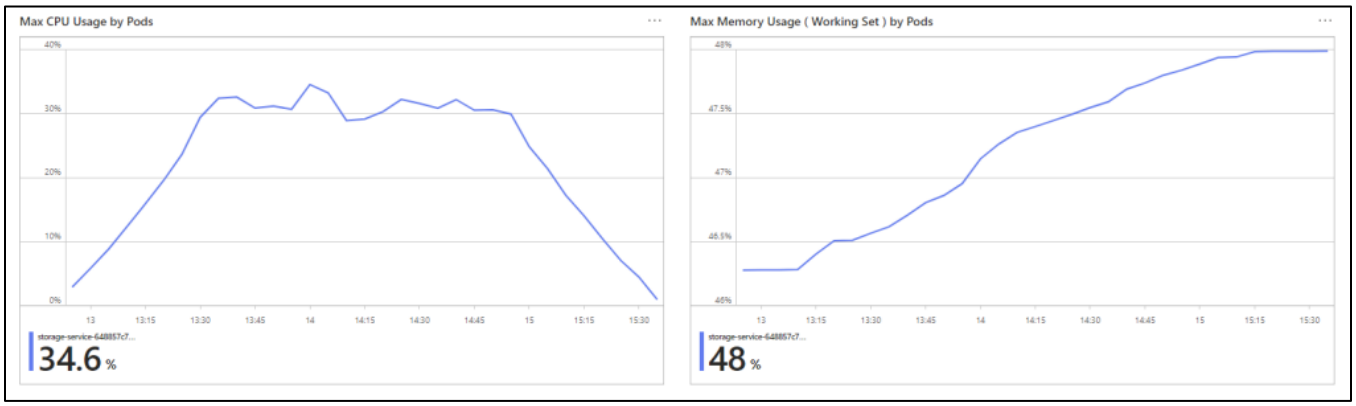

Figure 8.7: Storage Service - CPU and memory utilization (max)

Figure 8.8: Logging Service - CPU and memory utilization (max)

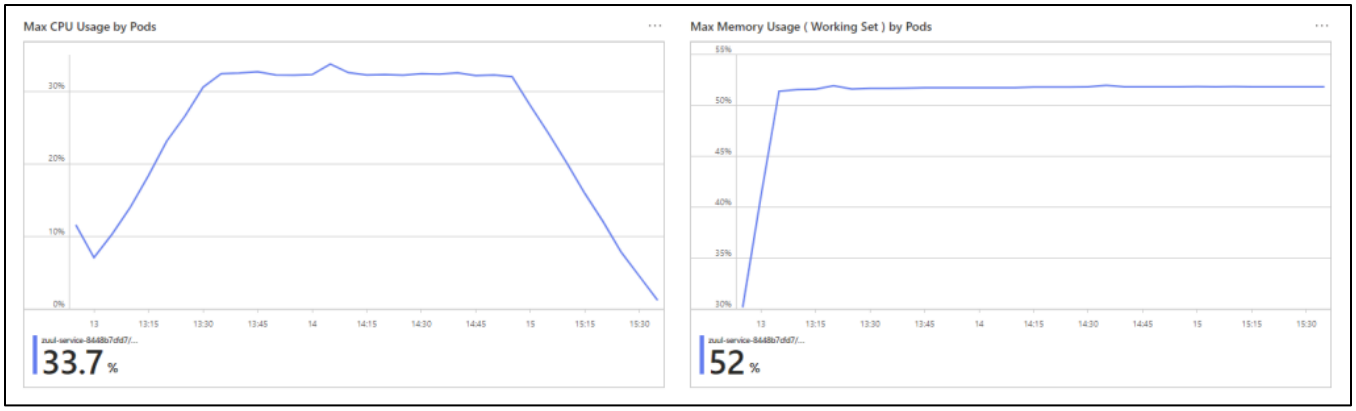

Figure 8.9: Zuul Service - CPU and memory utilization (max)

Name	Status	Max % ↓	Max	Containers	Restarts	UpTime	Node	Trend Max % (1 bar = 5m)
folder-service-8447455bc6 (ReplicaSet)	1	54%	946 mc	1	0	20 days	-	
dataclass-service-9d7898577 (ReplicaSet)	1	52%	1288 mc	1	0	2 hours	-	
app-service-99bd4b5bb (ReplicaSet)	1	51%	1142 mc	1	0	20 days	-	
content-service-559787765 (ReplicaSet)	1	40%	810 mc	1	0	2 hours	-	
storage-service-648857c7b5 (ReplicaSet)	1	35%	432 mc	1	0	22 days	-	
zuul-service-8448b7cfd7 (ReplicaSet)	2	33%	1661 mc	2	0	2 hours	-	
logging-service-689bc4c8d7 (ReplicaSet)	1	11%	393 mc	1	0	22 hours	-	

Figure 8.10: Pods status and CPU usage

Name	Status	Max % ↓	Max	Containers	Restarts	UpTime	Node	Trend Max % (1 bar = 5m)
zuul-service-8448b7cfd7 (ReplicaSet)	2	52%	10.4 GB	2	0	2 hours	-	
zuul-service-8448b7cfd7-mt6gk	Unk	52%	5.2 GB	1	0	2 hours	aks-agentpool-40694225	
zuul-service-8448b7cfd7-gszth	Unk	52%	5.2 GB	1	0	2 hours	aks-agentpool-40694225	
storage-service-648857c7b5 (ReplicaSet)	1	48%	1.92 GB	1	0	22 days	-	
app-service-99bd4b5bb (ReplicaSet)	1	36%	1.44 GB	1	0	20 days	-	
folder-service-8447455bc6 (ReplicaSet)	1	16%	826.82 MB	1	0	20 days	-	
logging-service-689bc4c8d7 (ReplicaSet)	1	11%	544.84 MB	1	0	22 hours	-	
dataclass-service-9d7898577 (ReplicaSet)	1	9%	466.66 MB	1	0	2 hours	-	
content-service-559787765 (ReplicaSet)	1	9%	454.73 MB	1	0	2 hours	-	

Figure 8.11: Pods status and memory usage

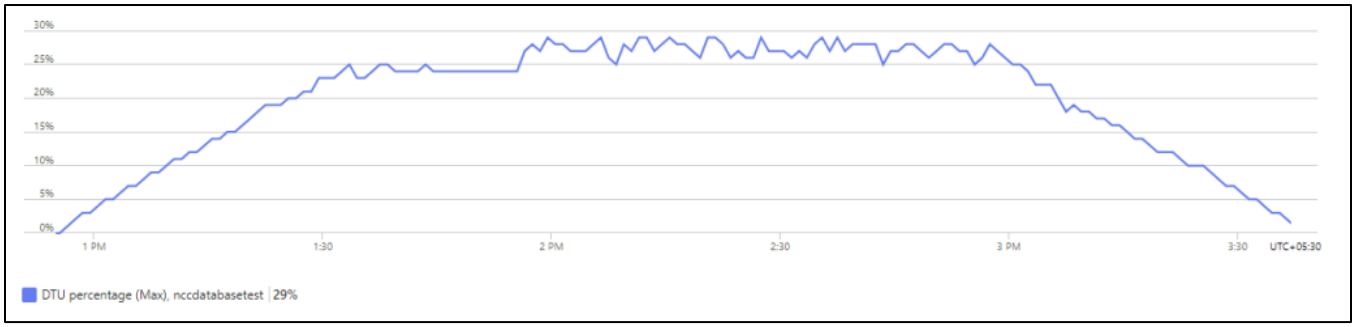

Figure 8.12: Azure MS SQL DB – Resource utilization (max)

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
content-service	Deployment/content-service	60%/60%	1	2	1	14d
dataclass-service	Deployment/dataclass-service	49%/60%	1	2	1	14d
folder-service	Deployment/folder-service	54%/60%	1	2	1	14d

Figure 8.13: Pods in the normal state during load less than auto-scale threshold limits

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
content-service	Deployment/content-service	70%/60%	1	2	2	14d
dataclass-service	Deployment/dataclass-service	49%/60%	1	2	1	14d
folder-service	Deployment/folder-service	66%/60%	1	2	2	14d

Figure 8.14: Pods auto-scaled when the load was more than the auto-scale threshold limits

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
content-service	Deployment/content-service	4%/60%	1	2	1	14d
dataclass-service	Deployment/dataclass-service	2%/60%	1	2	1	14d
folder-service	Deployment/folder-service	4%/60%	1	2	1	14d

Figure 8.15: Pods back to normal state when load dropped below auto-scale threshold limits

9 Conclusion

Newgen performed ‘real-world’ benchmarking tests to evaluate the performance and scalability of the NewgenONE Content Cloud running on Microsoft Azure Cloud. The tests measured system performance to benchmark server throughput and response times for user operations amongst various other metrics.

The results showed that NewgenONE Content Cloud, with Microsoft Azure CosmosDB as the database service, can support an extremely high number of concurrent transactions and user loads at rates that satisfy the most demanding environments. The test showed that:

- As the number of users increases, multi-server deployment gives proportionate performance gains.
- Typically, the NewgenONE Content Cloud server can handle 5100 users with an average response time of around 0.058 seconds, depending on the server configuration.
- NewgenONE Content Cloud server can handle 534 requests per second depending on the CosmosDB Provisioned Throughput and Kubernetes Containers configuration without any degradation.
- There is no performance degradation in search speed as the number of requests grows. There is no bottleneck at the server
- Overall throughput was around **534 requests/second**.
- There is no performance degradation in the speed of any Business Operation as the number of requests grows. There is no bottleneck at the server
- The system can handle millions of transactions per day without any degradation in server performance.
- Pods can auto-scale when the load on the server goes beyond the threshold limit set for pods auto-scaling. Also, when the load on the server drops below the threshold limit, pods can scale back to the normal state.

The benchmark tests confirm that the NewgenONE Content Cloud server on Microsoft Azure Cloud in Newgen’s cloud-native, multi-tenant based, SaaS application in the Content Service Platform offering running on Microsoft Azure CosmosDB can provide performance and scalability to meet the needs of even the largest enterprise customers.

10 Appendix A

The following table describes the NewgenONE Content Cloud performance summary report generated by the performance testing tool JMeter. The table shows the minimum, maximum, and average response times of transactions, and how much data was sent and received by the server.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
Generate Auth Code	5100	313	319	547	663	866	33	1425	0.00%	2.04311	1.21	0.89
Generate Auth Token from Code	5100	111	71	243	351	526	45	812	0.00%	2.04315	3.27	0.95
Create Folder with dataclass	515884	68	61	85	105	215	43	2575	0.00%	53.26743	57	75.79
Generate File URI Request	385099	131	121	157	178	318	87	60001	0.00%	39.75087	59.48	62.61
Get folder by ID and fetch folder properties	516994	47	39	62	82	195	26	60039	0.00%	53.37329	55.89	51.55
Retrieve Content from BLOB	516013	60	51	76	99	225	36	60062	0.00%	53.28611	57.95	54.95
Getting folder list	258795	39	30	53	74	187	20	3563	0.00%	26.76716	23.84	28.36
Search Folder on Dataclass	773862	62	54	77	97	212	36	59620	0.00%	79.87872	514.69	92.05
Search Content on Dataclass field value	1034904	40	32	55	76	190	20	3081	0.00%	106.78415	912.59	120.03
Get Content by ID and fetch content properties	517300	43	35	57	78	193	22	2982	0.00%	53.40529	68.77	53.98
Getting content list	258652	42	34	57	77	190	23	10389	0.00%	26.74819	226.62	28.79
Upload To Blob	384719	67	63	102	116	154	21	3138	0.00%	39.7719	17.87	19921.35
TOTAL	5172422	58	48	106	128	222	20	60062	0.00%	533.4395	1993.89	20441.58

Summary Report

Where,

- **Label**: The label of the sample.
- **Samples**: The number of samples with the same label.
- **Average**: The Average elapsed Time of a set of results.
- **Min**: The lowest elapsed time for the samples with the sample label.
- **Max**: The longest elapsed time for the samples with the same label.
- **Error %**: Percent of requests with errors.
- **Throughput**: The throughput is measured in requests per second/minute/hour.
- **KB/SEC**: The throughput is measured in kilobytes per second.
- **Avg. Bytes**: Average size of the sample response in bytes