KINISI

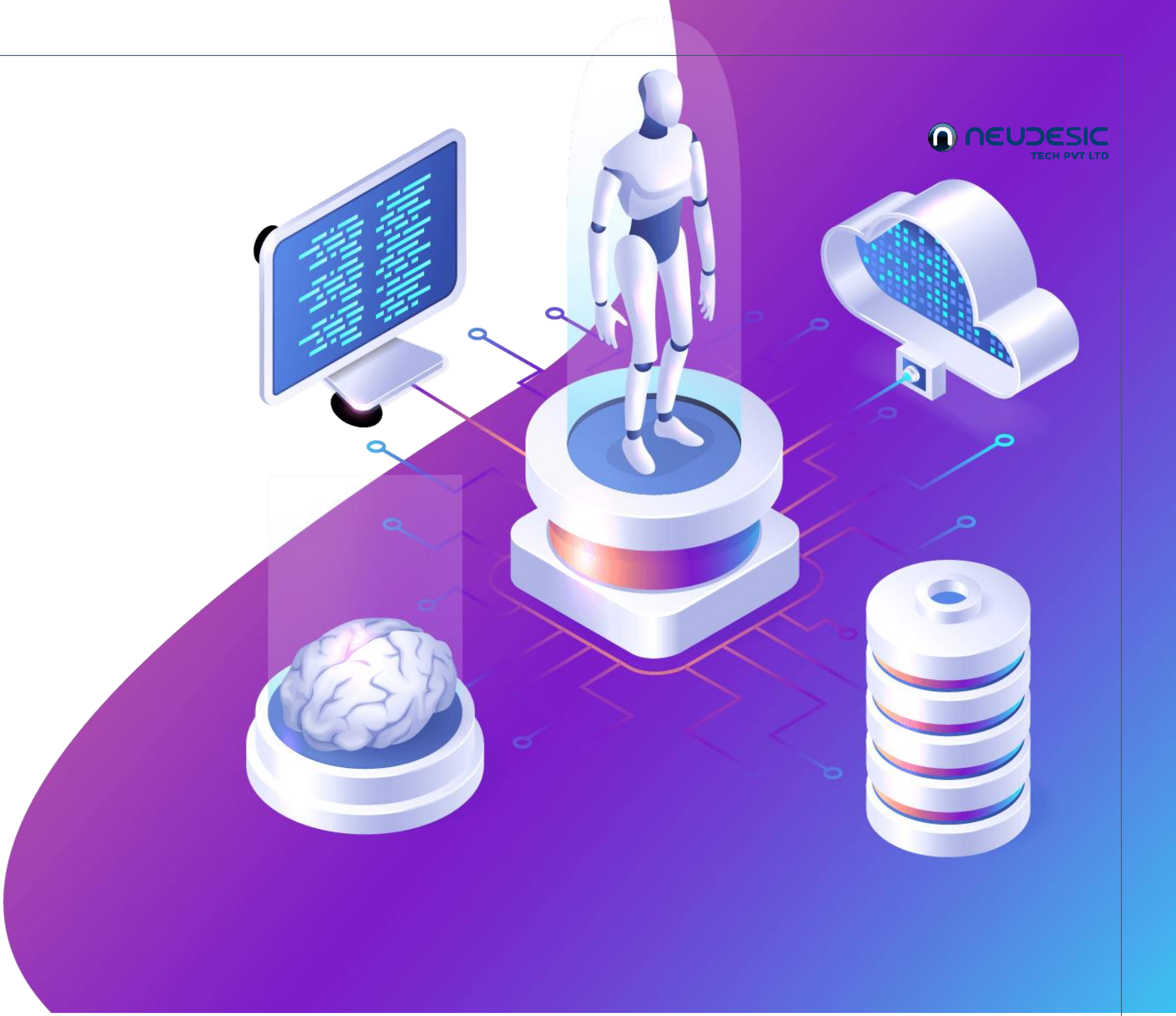
A Data Offload Utility for Cloud White Paper

Neudesic Technologies Pvt Ltd

TABLE OF CONTENTS

DISCLAIMER

This document is provided for informational purposes only. It represents Neudesic's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. The matrices and observations shared in this document are based on the experiments conducted. Based on the different environments and resources available, the performance may vary. Viewers are responsible for making their independent assessment of the information in this document and any use of Neudesic's products or services, each of which is provided "as-is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions, or assurances from Neudesic, its affiliates, suppliers, or licensors. The responsibilities and liabilities of Neudesic to its customers are controlled by Neudesic agreements, and this document is not part of, nor does it modify, any agreement between Neudesic and its customers.



INTRODUCTION

Data is at the core of nearly every business decision made. It is becoming increasingly difficult for the organizations to store and process the growing volumes and variety of data with their existing information management infrastructures. Many organizations are looking for a unified data platform to manage data easily. In addition to being person-centric, built for delivery resiliency, and location-agnostic, a unified data platform should support engineer capabilities, automation, and other factors. Cloud based architecture offers organizations the advantages of these features and a compelling reason to shift their on-premises data warehouses and databases to the cloud.

KINISI is a UI based data migration tool built on python with scalable architecture and configurable throughputs for accelerating the migration of multiple relational and non-relational databases like Oracle, PostgreSQL, MongoDB, Flat files, Teradata, and Cosmos DB. **KINISI** works to extract, aggregate, convert and load source databases to one or more destination databases. This accelerates conversion of database objects to any destination syntax decreasing manual efforts and errors. **KINISI** has self-healing schema management capabilities and can provide recommendations on a migration strategy.

FEATURES

Highlights

Solution

- Python Based
- Scalable Architecture
- Configurable Throughputs

Strategize

- Source Assessment
- Volume / Storage Estimates
- Workload Estimates
- Resource Usage Estimates

Offload

- Migration to Cosmos DB
- Migration to Synapse
- Auto-healing Metadata
- Auto DML, Procedure conversion

Monitor

- Dashboard & reports
- Directed Acyclic Graphs (DAGs)
- Job Execution History
- Alerts

Value Proposition

Speed and Scale

- Parallel execution strategy
- Scalable infrastructure

Zero Coding

- Intuitive User Interface
- Configurable connectors & throughputs
- Easy migrate job creation

Dynamic Pipelines

- Auto-generation of data offload pipelines

Auto Validations

- Automated Validations
- Source & Target comparisons
- Failure Alerts

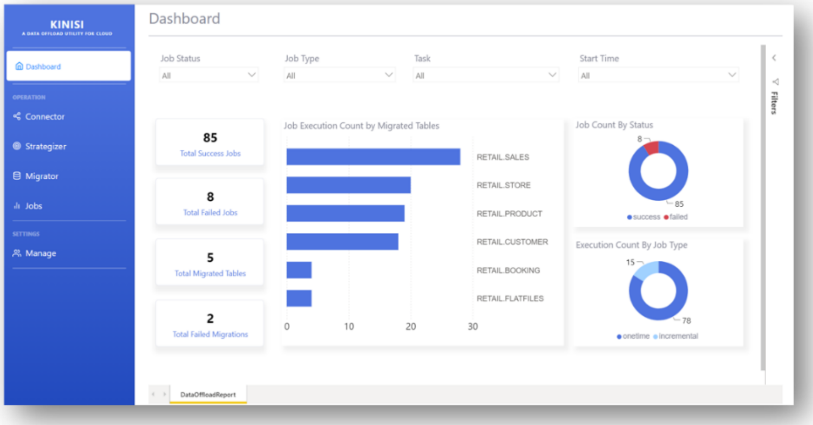
Engineer's Playground

- Migration Assessment
- Test Jobs
- Job definitions
- Scheduling
- Monitoring

COMPONENTS

Dashboard

Dashboard and reports provide a consolidated view of the migration jobs. It provides total visibility over the migration activities.



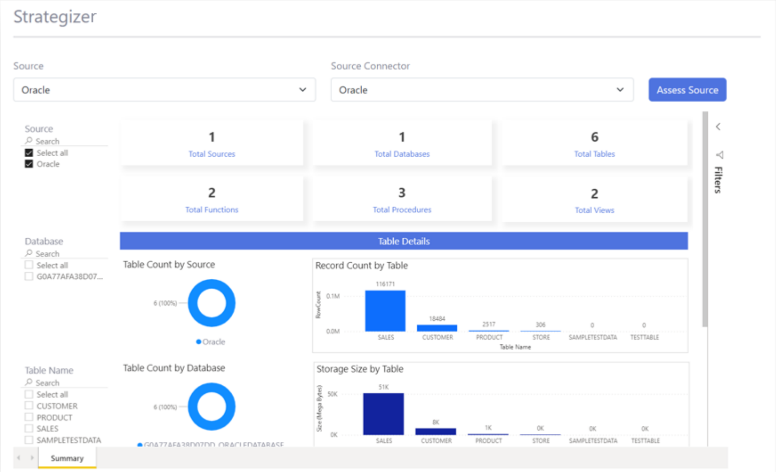
Dashboard

Job Status: All
Job Type: All
Task: All
Start Time: All

85 Total Success Jobs
8 Total Failed Jobs
5 Total Migrated Tables
2 Total Failed Migrations

Strategizer

This module helps to strategize and plan the migration activities. The source assessment module provides information about the source database. Additionally, it performs a test migration, and based on that it predicts the actual time and resources required to complete the entire database migration.



Strategizer

Source: Oracle
Source Connector: Oracle

1 Total Sources
1 Total Databases
6 Total Tables
2 Total Functions
3 Total Procedures
2 Total Views

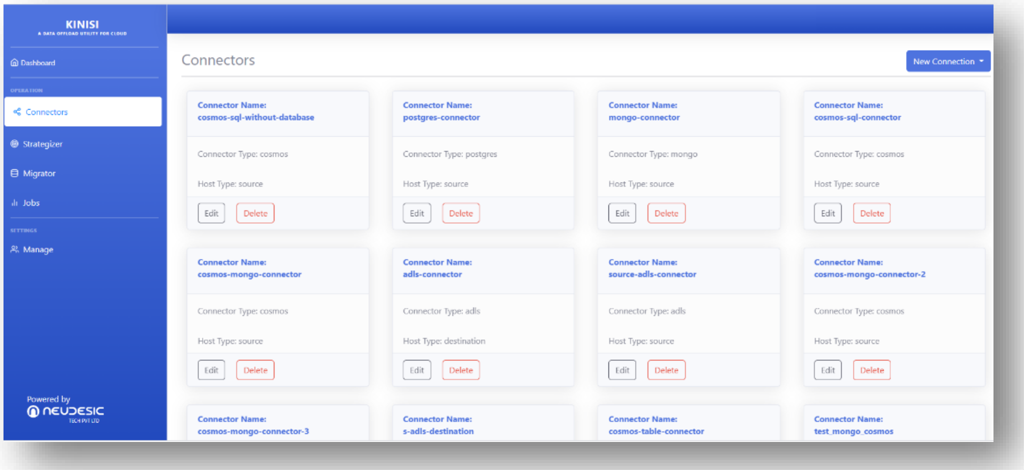
Source
☑ Select all
☑ Oracle

Database
☐ Select all
☐ G0A77AFA38D07...

Table Name
☐ Select all
☐ CUSTOMER
☐ PRODUCT
☐ SALES
☐ SAMPLETESTDATA

Connector

We can use this module to configure and define connectors to source and destination databases. With COSMOS DB as the destination database, it is possible to configure the API that needs be used, choose serverless deployment or provisioned throughput, and the Cosmos DB Request Unit (RU) as well.



Connectors

Connector Name: cosmos-sql-without-database
Connector Type: cosmos
Host Type: source

Connector Name: postgres-connector
Connector Type: postgres
Host Type: source

Connector Name: mongo-connector
Connector Type: mongo
Host Type: source

Connector Name: cosmos-sql-connector
Connector Type: cosmos
Host Type: source

Connector Name: cosmos-mongo-connector
Connector Type: cosmos
Host Type: source

Connector Name: adls-connector
Connector Type: adls
Host Type: destination

Connector Name: source-adls-connector
Connector Type: adls
Host Type: source

Connector Name: cosmos-mongo-connector-2
Connector Type: cosmos
Host Type: source

Connector Name: cosmos-mongo-connector-3

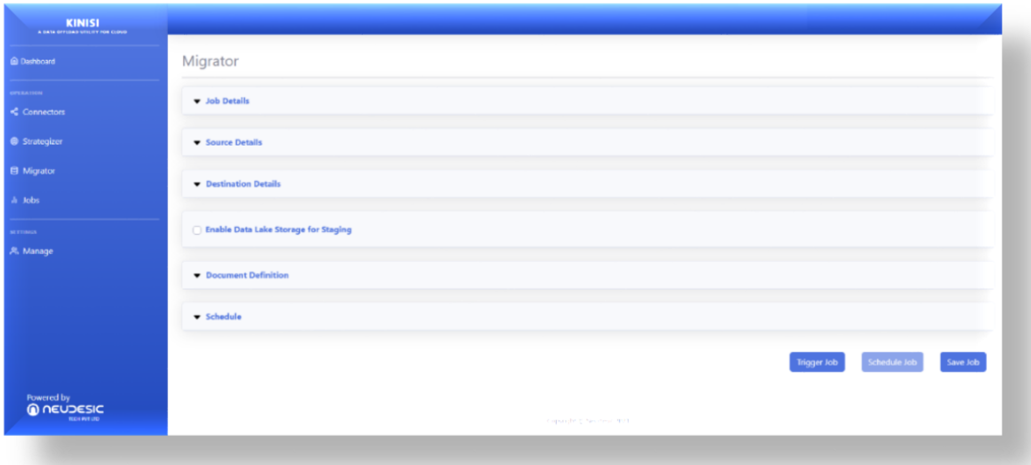
Connector Name: s-adls-destination

Connector Name: cosmos-table-connector

Connector Name: test_mongo_cosmos

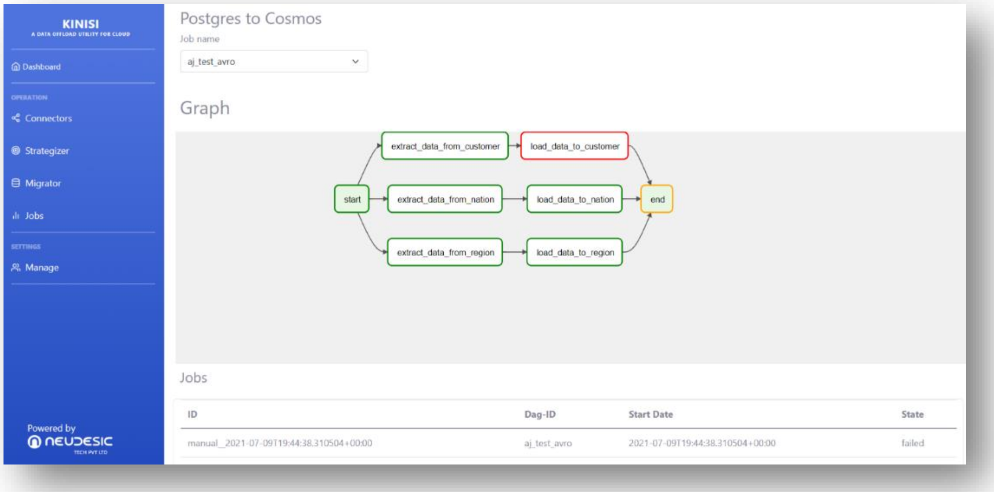
Migrator

The migrate module is used to define migration job details including the source and destination objects that needs to be migrated. In migration jobs with Cosmos DB as a destination, this module allows us to specify partition and indexing strategies. Using this module, you can also validate and schedule the jobs that have been created.



Jobs

This module provides details of the actual execution of the migrated jobs. Besides showing the execution history, it also displays the performance and workflow (DAG).



Postgres to Cosmos

Job name: aj_test_avro

Jobs

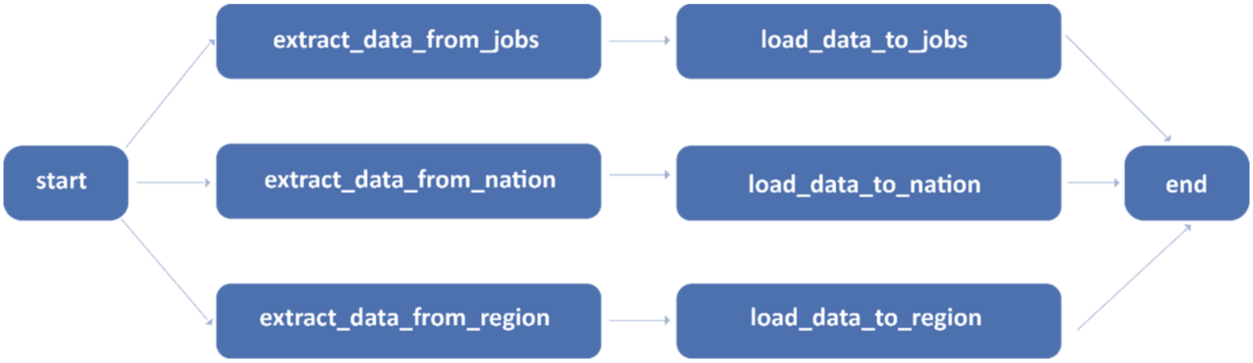
ID	Dag-ID	Start Date	State
manual__2021-07-09T19:44:38.310504+00:00	aj_test_avro	2021-07-09T19:44:38.310504+00:00	failed

KINISI FRAMEWORK CAPABILITY

Parallelization Strategy

KINISI framework can run multiple tables / collections in parallel, based on configurations. The parallelism feature accelerates completion of each stage and reduces runtime for data migration activities.

Each job may have multiple tasks configured and tasks running in parallel - unless a dependency is set. Furthermore, data for each table is divided into batches (as per batch size) and processed parallelly to achieve maximum extraction and loading speed. A sample DAG is shown below:

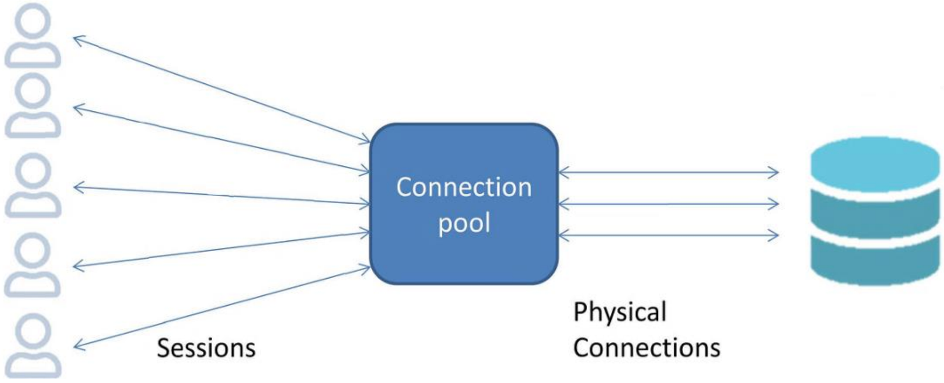


Aggregated Documents

For relational sources when this option is turned 'on' from UI, it combines multiple tables into a single aggregated document flattening all the source attributes. This de-normalizes the normalized relational data and improves the overall speed of the workflow.

Connection Pools

KINISI utilizes connection pools, creating a pool of connections and caching them so that they can be reused. By using Connection Pooling, whenever there is a request from the front-end application to create a database connection, a connection is created from the pool. Upon completion of a session or transaction, the connection is returned to the pool.



PERFORMANCE BENCHMARK

KINISI UI provides a Performance Benchmark dashboard, with benchmarking metrics collected according to workload sizes. Performance metrics are collected across different stages: extract, transform, upload to blob & write. It is possible to set batch sizes for extraction and compare them across batches to determine the optimal batch size for each workload.

Objective of Performance Benchmark

The objective of Performance Benchmark is to showcase the benchmarking capabilities of migration workloads running in **KINISI**. It depicts the appropriate configuration; ideal batch size for source and Request Units (RUs) for Cosmos DB based on the different workloads migrated for varieties of SQL and NoSQL data formats.

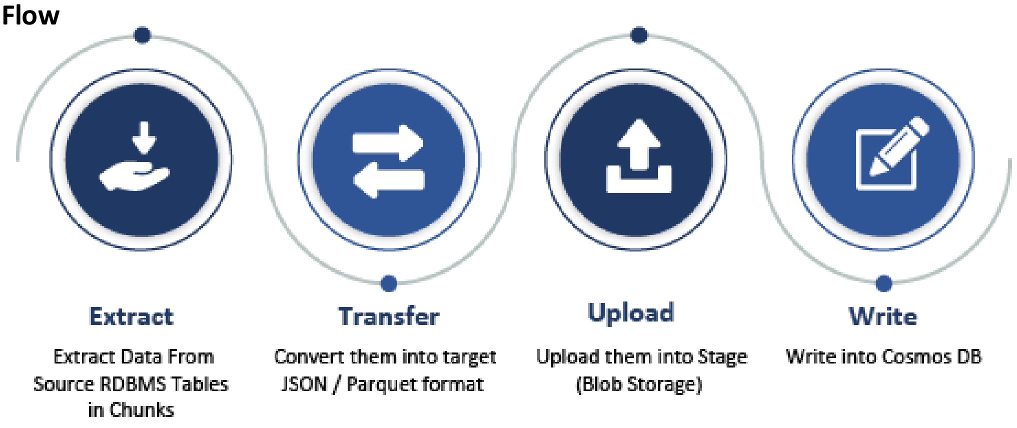
The benchmarking metrics are collected for multiple RDBMS to Cosmos DB and NoSQL to Cosmos DB data migrations.

Key Observations

- To improve performance on larger workloads, batch size should be set larger, but should not exceed **200 MB** per batch (~200000 records).
- Write speed to Cosmos DB varies based on the RUs available.
- More the RUs, better the write speed to Cosmos DB.

RDBMS to Cosmo DB Migration

Process Flow



Extract

Extract Data From Source RDBMS Tables in Chunks

Transfer

Convert them into target JSON / Parquet format

Upload

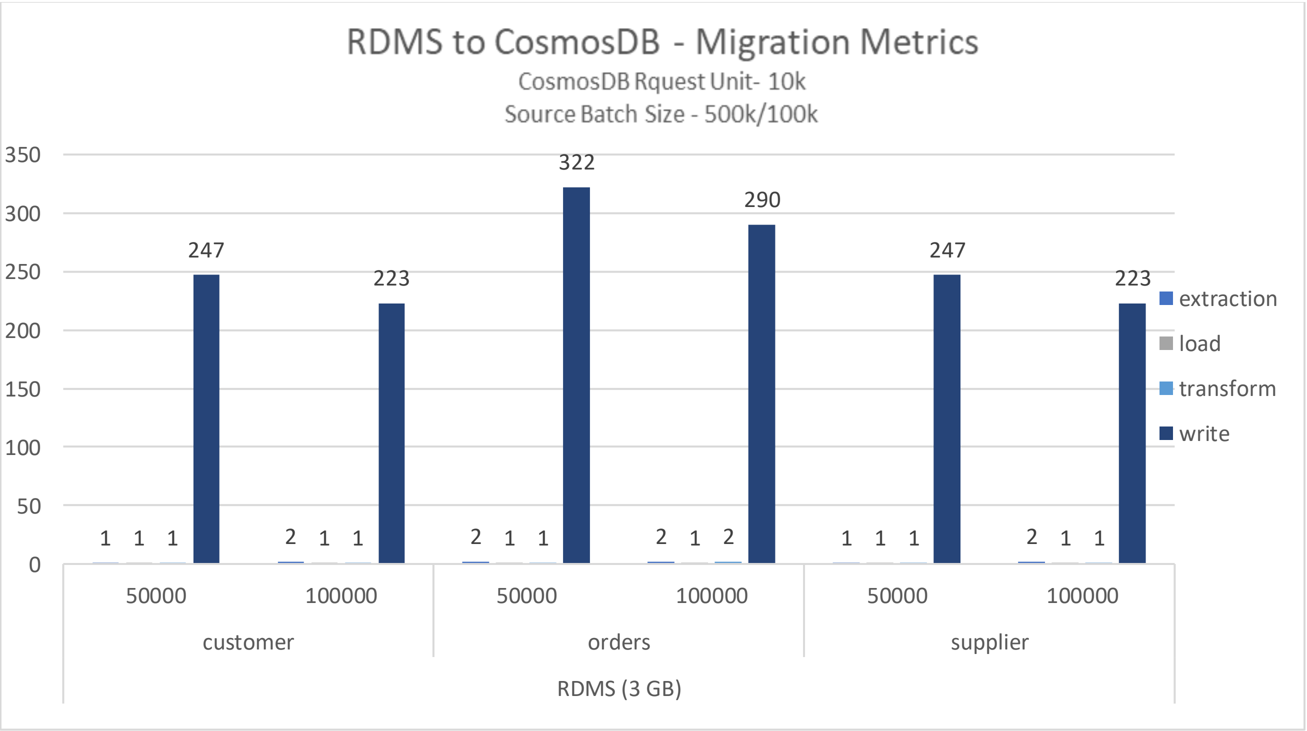
Upload them into Stage (Blob Storage)

Write

Write into Cosmos DB

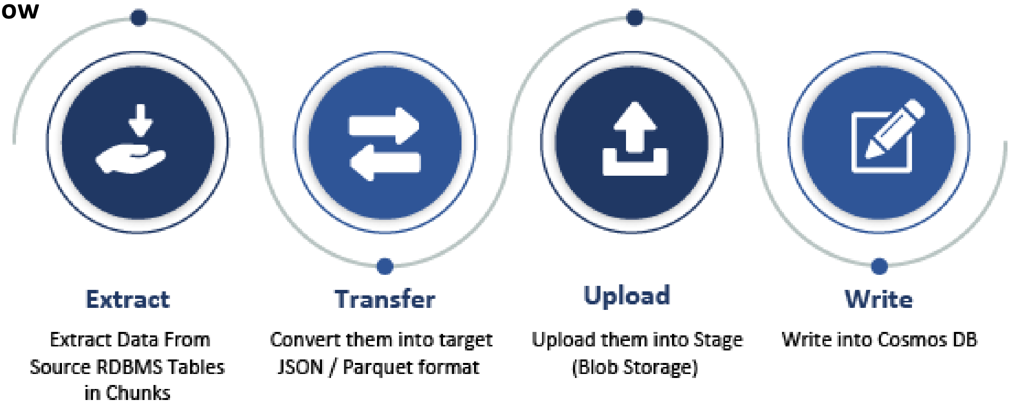
Key Observations

- Using SQL API with RU#10000, the RDBMS source **(10 GB)** took 10 minutes for source extraction and 700 minutes for writing to Cosmos DB.
- The performance of source data "extract" operation depends on availability of resources like CPU, memory, and network bandwidth.
- Availability of connection pool and network bandwidth can execute 'n' number of extract workload in parallel.
- RDBMS Migration works better with **Cosmos Mongo API** destination when compared with SQL/Table API.
- To improve performance on larger workloads, batch size should be set larger, but should not exceed **200 MB** per batch (~200000 records).
- Write speed to Cosmos DB varies based on the RUs available.



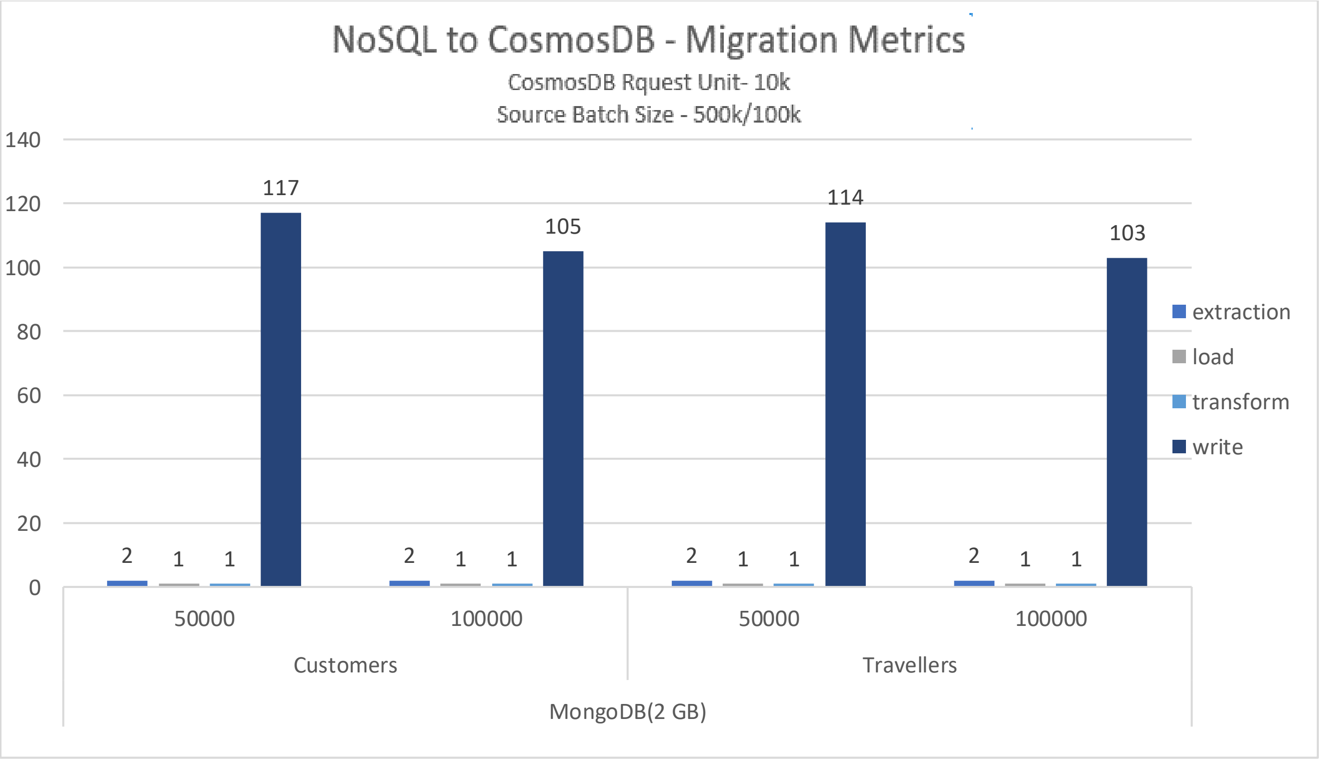
NoSQL DB to Cosmos DB Migration

Process Flow


Extract
Extract Data From Source RDBMS Tables in Chunks
Transfer
Convert them into target JSON / Parquet format
Upload
Upload them into Stage (Blob Storage)
Write
Write into Cosmos DB

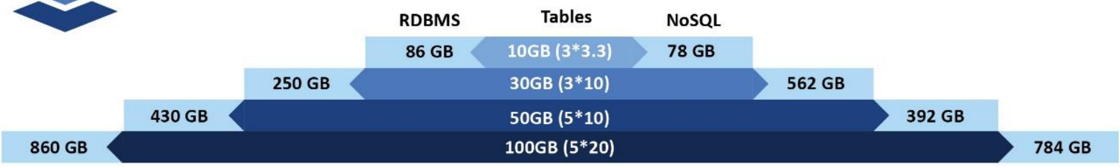
Key Observations

- NoSQL source (**10 GB)** took 20 minutes for source extraction and 760 mins to write to Cosmos DB.
- The performance of source data “extract” operation depends on availability of resources like CPU, memory, and network bandwidth.
- Availability of connection pool and network bandwidth can execute ‘n’ number of extract workload in parallel.
- NoSQL Migration works better with **Cosmos Mongo API** destination when compared with SQL/Table API.
- To improve performance on larger workloads, batch size should be set larger, but should not exceed **200 MB** per batch (~200000 records).
- Write speed to Cosmos DB varies based on the RUs available.



PERFORMANCE SUMMARY

Data



RDBMS	Tables	NoSQL
86 GB	10GB (3*3.3)	78 GB
250 GB	30GB (3*10)	562 GB
430 GB	50GB (5*10)	392 GB
860 GB	100GB (5*20)	784 GB

RDBMS

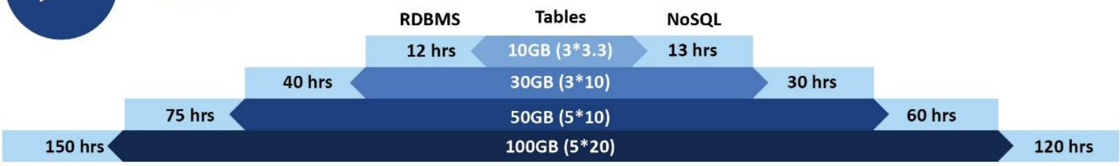
Data Size	Destination & Source files count in million	Destination size in GB
10 GB	74 million	86 GB
30 GB	224 million	250 GB
50 GB	373 million	430 GB
100 GB	747 million	860 GB

NoSQL

Data Size	Destination & Source files count in million	Destination size in GB
10 GB	74 million	78 GB
30 GB	224 million	562 GB
50 GB	373 million	392 GB
100 GB	747 million	784 GB

#Source Connections: 2 #Destination RU: 10000 Mongo API #Batch: 50000

Time



RDBMS	Tables	NoSQL
12 hrs	10GB (3*3.3)	13 hrs
40 hrs	30GB (3*10)	30 hrs
75 hrs	50GB (5*10)	60 hrs
150 hrs	100GB (5*20)	120 hrs

RDBMS

Extract Time (in min)	Convert to JSON Time (in min)	Upload to Blob Time (in min)	Write to Cosmos (in min)	Total time in hrs
4 MIN	3 MIN	3 MIN	700 MIN	12 HRS
14 MIN	10 MIN	10 MIN	2300 MIN	40 HRS
39 MIN	26 MIN	26 MIN	4400 MIN	75 HRS
79 MIN	53 MIN	53 MIN	8800 MIN	150 HRS

NoSQL

Extract Time (in min)	Convert to JSON Time (in min)	Upload to Blob Time (in min)	Write to Cosmos (in min)	Total time in hrs
8 MIN	6 MIN	7 MIN	760 MIN	13 HRS
23 MIN	20 MIN	20 MIN	1750 MIN	30 HRS
65 MIN	65 MIN	55 MIN	3500 MIN	60 HRS
130 MIN	110 MIN	110 MIN	7000 MIN	120 HRS

#Source Connections: 3 #Destination RU: 10000 SQL API #Batch: 50000

CONCLUSION

The **KINISI** product portfolio includes everything you will need to migrate enterprise data. It has an intuitive UI that allows you to configure connections, strategize, create, and automate migration jobs to execute data migration configurations for both relational data and non-relational source data. Data migration is easier with only a few clicks using the tool.

Dashboards give us an overview of completed migrations and other performance metrics. Furthermore, it predicts the optimal configuration for your workloads.

Conventional data migration activities usually take days and weeks to be completed however, based on the parallelization strategy defined in **KINISI**, your workloads will be executed in parallel and thus, **increase Return of Investment (ROI) for your business**. With this tool, you will be able to quickly analyze workloads to migrate based on the variety & volume of data, schedule components to automate workflows & jobs, and create rich dashboards to visualize and analyze all the tasks, thereby improving the ROI of your data migration.

CONTRIBUTORS

The following individuals and organizations contributed to this document:

Abhilash Thomas (Director of Technology, Data and AI)

- Data & AI Business Unit, Neudesic Global Services
- Innovations & Solutions Group, Neudesic Global Services