Amazon RDS vs Redshift vs Redshift Spectrum

Porównanie wydajności wczytywania oraz manipulacji danych na przykładzie zbioru danych OpenAQ

Amazon RDS

Rozproszona relacyjna baza danych w modelu SaaS (Software as a Service). Automatycznie obsługuje aktualizacje, backupy oraz odzyskiwanie danych (Point-In-Time Recovery). Obsługuje większość najpopularniejszych systemów zarządzania (Oracle, Microsoft SQL Server, MySQL, PostgreSQL, MariaDB).

Jest zoptymalizowana pod kątem transakcyjnych obciążeń OLTP (Online Transaction Processing), co oznacza, że idealnie nadaje się do obsługi aplikacji, które wymagają szybkich operacji odczytu i zapisu na poziomie pojedynczych rekordów. Wczytywanie danych odbywa się przez standardowe interfejsy API, narzędzia bazy danych lub pliki, a czas wykonania zapytań zależy od skali i struktury bazy.

Amazon Redshift

Hurtownia danych zaprojektowana do obsługi analitycznych obciążeń OLAP (Online Analytical Processing). Oparta na systemie zarządzania PostgreSQL 8.0.2.

Przetwarzanie odbywa się na dużą skalę przy użyciu kolumnowej architektury bazy danych, co pozwala na szybkie agregacje i analizy dużych zbiorów danych. Wykorzystuje przetwarzanie równoległe i kompresję w celu skrócenia czasu wykonywania — pozwala to na wykonywanie operacji na miliardach wierszy jednocześnie. Wczytywanie danych jest zoptymalizowane za pomocą operacji COPY, które umożliwiają szybki transfer danych z S3 lub innych źródeł.

Redshift Spectrum

Redshift Spectrum to rozszerzenie Redshift, które umożliwia bezpośrednie wykonywanie zapytań SQL na danych przechowywanych w Amazon S3 w formatach takich jak Parquet czy ORC, bez konieczności ich wcześniejszego wczytywania do Redshift. Dzięki temu pozwala analizować dane na większą skalę, bez potrzeby replikowania ich w hurtowni — wystarczy zadeklarować dane z Amazon S3 jako zewnętrzny schemat bazy danych.

OpenAQ

Publicznie dostępny zbiór wyników jakości powietrza z punktów pomiaru na całym świecie. Interfejs API zapewnia otwarty dostęp zgodnie z zasadami REST z adresami URL zorientowanymi na zasoby, standardowymi kodami odpowiedzi HTTP i odpowiedziami w formacie JSON. Koncentruje się na kryterialnych zanieczyszczeniach powietrza, głównie agregując dane pomiarowe PM2.5, PM10, SO2, NO2, CO, O3, BC, wilgotności względnej i temperatury.

Dane historyczne dostępne są w postaci skompresowanej (GZIP) na Amazon S3.

Schemat tabeli (RDS vs Redshift)

```sql
CREATE TABLE air_quality_data (
    location_id INTEGER NOT NULL,
    sensors_id INTEGER NOT NULL,
    location TEXT NOT NULL,
    datetime TIMESTAMP WITH TIME ZONE NOT NULL,
    lat NUMERIC(8, 6) NOT NULL,
    lon NUMERIC(9, 6) NOT NULL,
    parameter VARCHAR(50) NOT NULL,
    units VARCHAR(50) NOT NULL,
    value NUMERIC(10, 2) NOT NULL
);
```

```sql
CREATE TABLE air_quality_data (
    location_id INTEGER NOT NULL,
    sensors_id INTEGER NOT NULL,
    location VARCHAR(255) NOT NULL,
    datetime TIMESTAMP NOT NULL,
    lat NUMERIC(8, 6) NOT NULL,
    lon NUMERIC(9, 6) NOT NULL,
    parameter VARCHAR(50) NOT NULL,
    units VARCHAR(50) NOT NULL,
    value NUMERIC(10, 2) NOT NULL
)
```

Import danych z Amazon S3 do RDS

```
DO $$
DECLARE
    file_endings TEXT[] := ARRAY[...];
    file_ending TEXT;
    s3_path TEXT;
BEGIN
    FOREACH file_ending IN ARRAY file_endings
    LOOP
        s3_path := FORMAT(
            'records/csv.gz/locationid=2178/year=2022/month=05/location-2178-202205%s.csv.gz',
            file_ending
        );

        EXECUTE FORMAT(
            $$SELECT aws_s3.table_import_from_s3(
                'air_quality_data',
                '',
                '(FORMAT csv, HEADER match, DELIMITER '','', QUOTE '''"''', ESCAPE ''\'')',
                '<CUSTOM_BUCKET_NAME>', '%s', 'us-east-1'
            );$$,
            s3_path
        );

        RAISE NOTICE 'Zaimportowano plik: %', s3_path;
    END LOOP;
END $$;
```

Problemy?

- każdy plik GZIP trzeba importować osobno (dla każdego dnia miesiąca)
- domyślnie ustawiony Content-Encoding to octet-stream, RDS wymaga wartości GZIP
- Terraform ignoruje niektóre metadane przy kopiowaniu danych na S3

Rozwiązanie?

- przerzucenie plików GZIP zbioru danych na własnego bucketa S3 za pomocą skryptu AWS CLI z wymuszeniem Content-Encoding równym GZIP (z pominięciem Terraforma)

Import danych z Amazon S3 do Redshift

```
COPY air_quality_data
FROM 's3://openaq-data-archive/records/csv.gz/locationid=2178/year=2022/month=05/'
IAM_ROLE 'arn:aws:iam::847382997868:role/LabRole'
CSV
GZIP
IGNOREHEADER 1
TIMEFORMAT 'auto';
```

Stworzenie zewnętrznego schematu dla Spectrum

```
create external schema myspectrum_schema
from data catalog
database 'myspectrum_db'
iam_role 'arn:aws:iam::847382997868:role/LabRole'
create external database if not exists;

create external table myspectrum_schema.air_quality_data (
    _location_id INTEGER,
    sensors_id INTEGER,
    location VARCHAR(255),
    datetime TIMESTAMP,
    lat NUMERIC(8, 6),
    lon NUMERIC(9, 6),
    parameter VARCHAR(50),
    units VARCHAR(50),
    value NUMERIC(10, 2)
)
PARTITIONED BY (location_id INTEGER, year INTEGER, month INTEGER)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 's3://openaq-data-archive/records/csv.gz/';


alter table myspectrum_schema.air_quality_data add
partition(location_id=2178, year=2022, month=05)
location 's3://openaq-data-archive/records/csv.gz/locationid=2178/year=2022/month=05/';
```

Infrastruktura rozwiązania

Ze względu na ograniczania konta studenckiego udało nam się uruchomić serwisy z następującymi parametrami:

RDS:

- Engine: PostgreSQL
- InstanceType: t4g.medium
- VolumeSize: 50
- EnhancedMonitoring: Disabled

Redshift:

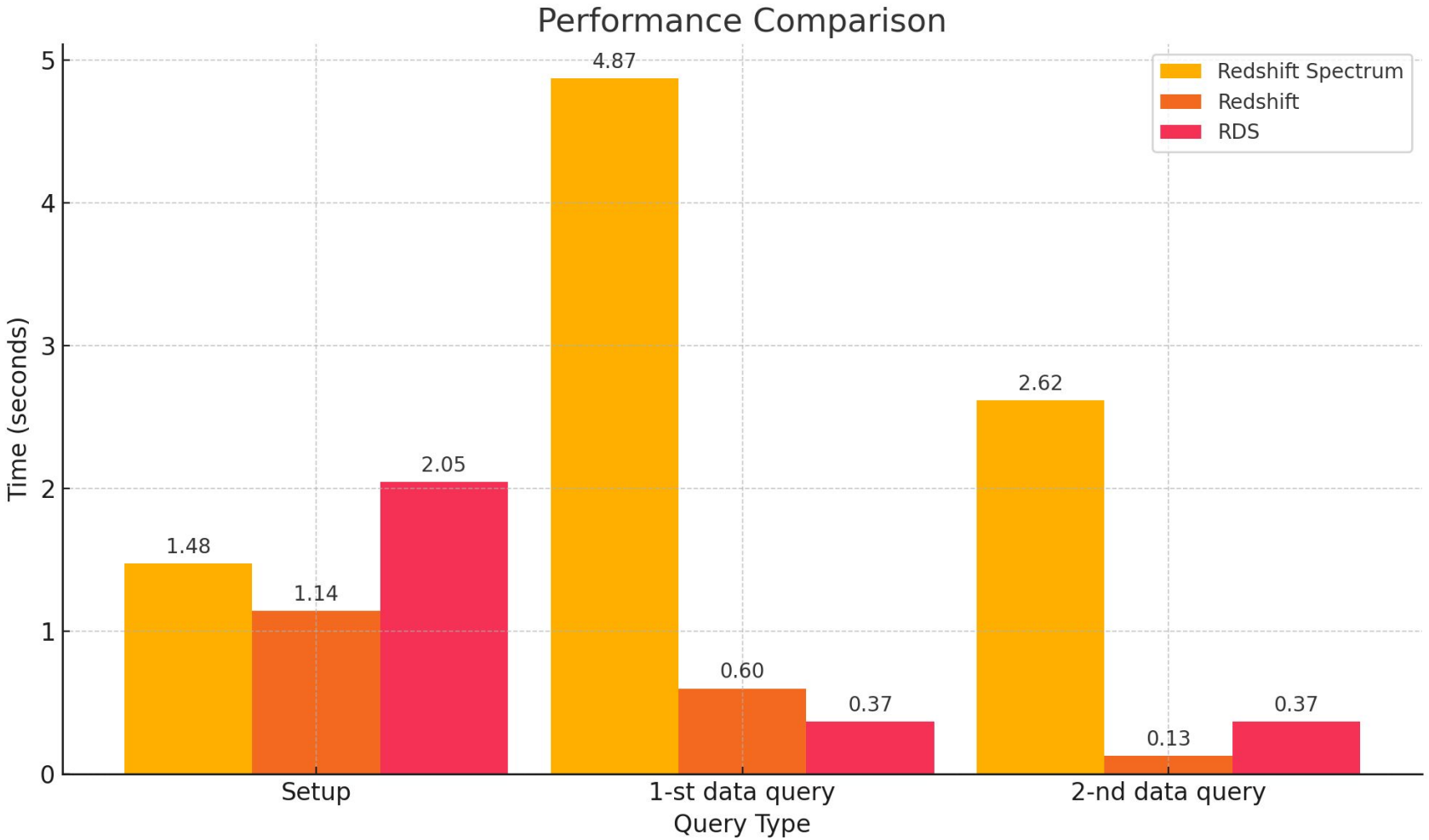
- InstanceType: dc2.large

W celu prostej reprodukcji rozwiązania użyliśmy narzędzia Terraform, które po podłączenia do konta AWS uruchamia wszystkie potrzebne usługi.

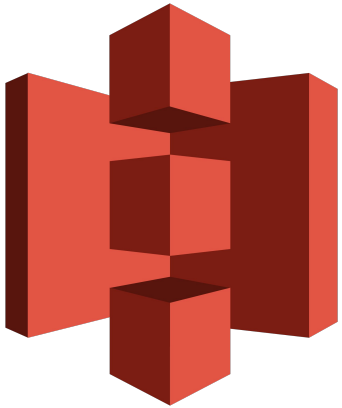
Benchmark query

```sql
WITH aggregated_data AS (
    SELECT
        location,
        parameter,
        date_trunc('day', datetime) AS day,
        AVG(value) AS avg_value,
        MAX(value) AS max_value,
        MIN(value) AS min_value
    FROM
        air_quality_data
    GROUP BY
        location, parameter, date_trunc('day', datetime)
),
location_ranking AS (
    SELECT
        location,
        parameter,
        AVG(avg_value) AS yearly_avg,
        RANK() OVER (PARTITION BY parameter
        ORDER BY AVG(avg_value) DESC) AS rank
    FROM
        aggregated_data
    GROUP BY
        location, parameter
)
```

```sql
SELECT
    lr.location,
    lr.parameter,
    lr.yearly_avg,
    lr.rank,
    ad.day,
    ad.avg_value,
    ad.max_value,
    ad.min_value
FROM
    location_ranking lr
JOIN
    aggregated_data ad
ON
    lr.location = ad.location
    AND lr.parameter = ad.parameter
WHERE
    lr.rank <= 10
ORDER BY
    lr.parameter,
    lr.rank,
    ad.day;
```


Performance Comparison
Redshift Spectrum
Redshift
RDS
Time (seconds)
0
1
2
3
4
5
1.48
1.14
2.05
4.87
0.60
0.37
2.62
0.13
0.37
Setup
1-st data query
2-nd data query
Query Type

Dziękujemy za uwagę



Tobiasz Szulc, Michał Wójcik