# Benchmark calculation using hierarchical partitioning

~~EIOTP-3261~~ - POC on finalizing partition strategy DONE

We need to get the numbers when we are choosing either of the below as hierarchical partition key:

- `tenantId` + `siteId` + `deviceId`
- `deviceId` + `date` ( as recommended here : https://youtu.be/oToAY8NGWA0?t=1425)

For us, deviceId will be `manufacturer name` and `serial number`.

- Each device can exceed **20GB** of data
- No need to implement synthetic keys

Let's consider some numbers to calculate benchmark:

- 1 device generates nearly ~=2 Lac records per month
- We need to consider `1 year` TTL for such records

Let's create collections for `15 tenants` each having `100 sites`.
Each site would have `5 devices` generating a `~5 KB` document every `30th sec`.
Devices will only only be active for 12 hrs (considering only US hours).

---

**#### What will be the number of telemetry generated every day by 100 sites?**

**1 tenant 100 sites 1 day | Records Generated**

```
100 sites * 120 requests per hour * 12 hours
=  100 * 1440 (1 device generating telemetry for 12 hours -US consideration - generating telemetry every 30
secs)
Number of telemetry requests generated by 1 device each , for 100 sites in a day =   144000

For 5 devices = 144000*5 => 720,000
```

**#### 1 tenant 5000 sites telemetry for 1 day**

```
5000 sites * 120 requests per hour * 12 hours
=  5000 * 1440 (1 device generating telemetry for 12 hours -US consideration - generating telemetry every 30
secs)
Number of telemetry requests generated by 1 device each , for 100 sites in a day = 7,200,000

1 device in a month = 216 million  = 21.6 crores (furcating based on partition key)
5 devices in a day= 7,200,000*5 => 36,000,000 (5 devices in 5000 sites generating telemetry for 12 hrs in a day)
5 devices for a month : one billion eighty million (108 crores)
Two tenants with 5 devices each : 216 crores
```

If a device is populating 21.6 crores records of telemetry in a month, what is the data we are planning to pull on the mobile app?

**#### What should be the storage requirements for storing 720,000 records?**
Considering each document stored in cosmos collection has `~5KB` size.

**For 1 tenant 100 sites 1 day | Size Requirements**

```
Total size required for 1 day = (720,000 * 5)/1024*1024 => 3.43GB ~=3.5GB per day

For 1 day , size requirement = 3.5 GB
For 2 days, size requirement = 7 GB
For 3 days, size requirement = 10.5GB
For 4 days, size requirement = 14GB
For 5 days, size requirement = 17.5GB
For 6 days, size requirement = 21GB
For 7 days, size requirement = 24.5GB
For month , size requirement = 105 GB per month
```

**#### In action**

> 1. Take sample collection for 5 devices from dev cosmos db instance.

```
SELECT DISTINCT c.metadata["customerDeviceClass"] FROM c

//returns  `Bakery Oven` , `Fryer` , `Rotisserie`
//Let's assume 2 more `TypeA` , `TypeB`
```

> 2. Use the **Platform.Benchmark.Services** tool for calculating RU/s for write operations. Modify code for parallelizing task where each device writes data every 30 secs.

**Collection throughput of deli-telemetry :** 4000 RU/s
**Environment :** Local

# Testing Infrastructure

## Cosmos account

- Single master account deployed in **West US** region with Private Link enabled.
- **Throughput : (regions: 1, 400 RU/s - 4000 RU/s, $0.00012/RU)**
- **Cost : (Min per month $ 35.04 , max per month $ 350.40)**

## Demo Application

Connection mode to cosmos is direct.
Platform.Benchmark.Services

## Application Hosting

**WCNP host**

```
.Net core web api deployed on wus-dev-a2,scus-dev-a4

Cluster IP : 10.50.136.93
Min RAM in MB : 256Mi
Max RAM in MB: 1024Mi
```

# Observation of test results

- This was data was sampled against 1.45Cr (14680805) records.
- Throughput : 8k RU/s
- Partition Key :
    - dev : Hierarchical , **/deviceId**,**/date**
    - prod : Synthetic , **/siteClassId**

| Source (App Cloud /Region) | Target (Cosmos Write Region) | Operation Type | Payload Size (Dev) | Duration (Dev) | Request Charge (Dev) | Payload Size (Prod) | Duration (Prod) | Request Charge (Prod) |
|---|---|---|---|---|---|---|---|---|
| WMT West US Central | Azure West US | Point Read | 1.89 KB | 7 | 1.05 | 1.78 KB | 0.91 | 1.05 |
| | | ReadingAnItemAsStream | 1.89 KB | 1 | 1.05 | 1.78 KB | 1 | 1.05 |
| | | Read All Items (paged: 1000) | 1890 KB | 8 | 69 | 1428 KB | 2 | 69 |

## Optimizations

- **Resolution to Issue 1: Latency**

  Hosting app and db in the same region **OR** pointing application to nearest cosmos instance.

  While testing,we have have configured my app to use application region as West US.

  Using point reads instead of queries allows cosmos SDK to skip the query execution plan.

- **Resolution to Issue 2: Rate Limiting or throttling**

  Choosing a partition key with more cardinality is the best choice.

  Partitioning strategy tells Cosmos DB how you want us to distribute your data among all of the machines that will store the data.

  For read heavy scenarios it tells us exactly which machine your data is on, so that we don't have to check all of them.

  And then we use the partition key in the filter criteria while querying.

  Also , we have used query options like MaxItemCount(allowing pagination)**,**MaxBufferedItemCount (using cosmos feature to buffer the results),MaxConcurrency(to allow client system to automatically decide parallel query execution).

- **Optimizing bulk writes:**

  Let's say we have provisioned a lot of throughput (e.g 10,000 RU/s).

  We use bulk executor library to ingest high data into Cosmos DB.

  While inserting the data we observed that we are not able to use complete effective RUs/s.

  What we found is to investigate again is our partitioning strategy.

  We were using **/date** and while simulating data we inserted data on the same day. So we were all bottlenecked on the same partition.

  Instead we should pick a partition key with high cardinality such as **timestamp**.

---

### /tenantSiteId,/deviceId,/ts as hierarchical partitioning key

Considering support upto 3 levels of hierarchical partitioning , let's assume three levels for this example are : /tenantSiteId,/deviceId,/ts (timestamp)

**Numbers for 7 days 15 tenants**

```
1 day 1 tenant's all sites generate = 7,20,000 records
1 day 15 tenant's all sites generate = 1,08,00,000 (1.08 million) records
7 days total data for 15 tenants = 16,20,00,000 (162 million) records

//Hierarchical Partition Key 1 : tenant+SiteId
If we query for a tenant + siteId over 162,000,000(162 million) records.
Querying with first partition key would require looking up over 1500 partitions/machines
162,000,000/1500 => 108,000
Size occupied by 108,000 docs in each machine: (108,000 * 5)/1024*1024 => 0.514 GB

//Hierarchical Partition Key 2 : deviceId

5 devices per site

15 tenants * 100 sites = 150
162,000,000 /1500 = 108,000 records per first hierarchical partition.
Second partition key would have to lookup over 108,000/5 = 21,600 records per second hierarchical partition.


//Hierarchical Partition Key 3 : timestamp

Telemetry is generated every 30th sec (one timestamp makes up for one partition key)
1500 sites * 5 devices = 7500 records per partition key
Third hierarchical partition key will have to lookup over 7500 records.
```

**Numbers for 30 days 15 tenants**

```
1 day 1 tenant's all sites generate = 7,20,000 records
1 day 15 tenant's all sites generate = 1,08,00,000 (1.08 million) records
30 days total data for 15 tenants =  32,40,00,000 (32.4 million) records

//Hierarchical Partition Key 1 : tenant+SiteId
If we query for a tenant + siteId over 32,40,00,000 records.
Querying with first partition key would require looking up over 1500 partitions/machines

32,40,00,000/1500 => 216,000
Size occupied by 216,000 docs in each machine: (216,000 * 5)/1024*1024 => 1.029 GB

//Hierarchical Partition Key 2 : deviceId

5 devices per site

15 tenants * 100 sites = 1500
32,40,00,000 /1500 =  216,000 records per first hierarchical partition.
Second partition key would have to lookup over  216,000/5 = 43,200 records per second hierarchical partition.

//Hierarchical Partition Key 3 : timestamp

Telemetry is generated every 30th sec (one timestamp makes up for one partition key)
1500 sites * 5 devices = 7500 records per partition key
Third hierarchical partition key will have to lookup over 7500 records.
```

## Observation of test results

- This was data was sampled against 1.09Cr (10908823) records.
- Throughput : 8k RU/s
- Partition Key :
  - dev : Hierarchical , **/tenantSiteId,/deviceId,/ts**
- Initial run is <1 sec , whereas consequent runs are 0.4 sec for all operations

| **Source (App Cloud /Region)** | **Target (Cosmos Write Region)** | **Operation Type** | **Duration (In secs) for Initial run** | **Duration (In secs) for consequent runs** | **Request Charge** |
|---|---|---|---|---|---|
| Azure US West 2 | Azure West US | PointReadAnItemAsync | 1 | 0.4 | 1.05 |
| | | ReadingAnItemAsAStreamAsync | 0.9 | 0.4 | 1.05 |
| | | Read All Items with /deviceId as filter (paged:1000) | 0.9 | 0.4 | 2.8 |
| | | Read All Items with /tenantSiteId as filter (paged:1000) | 1 | 0.4 | 2.8 |
| | | Read All Items with /ts as filter (paged:1000) | 0.9 | 0.4 | 2.8 |
| | | Read All Items with all id,/deviceId,/tenantSiteId,/ts as filter (paged:1000) | 0.9 | 0.4 | 3.17 |

## Notes:

1. If we are expecting payload size to increase , we should do compression using wrappers.Cosmos SDK does not do any payload compression when in transit.