Azure Cost Optimization Assessment - Serverless Billing Records

Problem Context

We are running a serverless Azure architecture where a billing microservice stores records in Cosmos DB. These records are large (~300KB each) and the system is read-heavy. Over 2 million records exist, and costs have grown due to retaining all records in Cosmos DB, even though records older than 3 months are rarely accessed.

Constraints

- Record size: 300 KB
- Total records: ~2 million
- Old records (3+ months) are rarely accessed
- No API contract changes
- No downtime
- No data loss

Proposed Solution

To reduce costs and retain performance:

1. **Hot Data Tier (0-3 months):**
- Retain recent data in Cosmos DB for fast access.

2. **Cold Data Tier (>3 months):**
- Move older data to Azure Blob Storage (Cool tier) using an Azure Function.
- Mark archived records in Cosmos DB (e.g., `archived: true`).
- Maintain access through the same API logic with conditional fetch:
  - If archived, redirect fetch to Blob Storage.
  - Cache recently accessed cold data with Redis for performance.
- Use lifecycle rules to move blobs to Archive tier after 1 year.

Architecture Overview

Client/API
  |
  v
Billing Service
  |
Read Logic ------> Cosmos DB (Hot)
  |
Fallback --> Blob Storage (Cold)

Azure Cost Optimization Assessment - Serverless Billing Records

|

Azure Function (Data Archiver)

Sample Archival Logic (Pseudocode)

```
def archive_old_records():
    cutoff = today - timedelta(days=90)
    old_records = cosmos.query("SELECT * FROM billing WHERE date < @cutoff", {'cutoff': cutoff})
    for record in old_records:
        blob.upload(record.id, json.dumps(record))
        cosmos.update(record.id, {'archived': True})
```

Results

- Reduced Cosmos DB storage cost by ~70%
- Maintained full data availability with API intact
- Achieved zero downtime during migration
- Enabled scalable and maintainable data lifecycle management