NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Project SCC 2024

Report 1

Work done by:

João Lima nº 60350

Diogo Nunes nº 70502

Motivation

It was given to us the task to develop TuKano, an app inspired by TikTok, to leverage Microsoft Azure Cloud platform, in order to achieve better throughput on a regional scale. This report's objective is to identify in what ways the goal was achieved, and how managing Microsoft Azure Cloud platform has changed the base app given.

What was achieved

In our project, we implemented:

- Blob Storage, to save the blobs regarding the shorts posted;
- NoSQL Cosmos DB, to manage the users, shorts, likes and follows;
- PostgreSQL Cosmos DB, for an alternative storage (*) ;
- Redis Cache, for faster access to objects required.

To understand how these implementations impacted TuKano, we will look at different combinations of the previous and compare its metrics.

(*) - PostgreSQL Cosmos DB was fully coded in this project, but there was a connection error that was impossible for us to debug in the given project time, even with help from the practical classes teacher, Kevin Gallagher. We've decided to leave the metrics regarding the PostgreSQL Cosmos DB out of the report, but mention it in the implementations.

Impact on TuKano

We will be looking at these combinations:

1. Blob Storage + NoSQL Cosmos DB + Redis Cache
2. Blob Storage + NoSQL Cosmos DB

User_register

- With Redis Cache

```
http.codes.400: ..... 400
http.codes.404: ..... 200
http.downloaded_bytes: ..... 15200
http.request_rate: ..... 6/sec
http.requests: ..... 600
http.response_time:
  min: ..... 2
```

  max: 3446
  mean: 94
  median: 3
  p95: 210.6
  p99: 820.7
http.response_time.4xx:
  min: 2
  max: 3446
  mean: 94
  median: 3
  p95: 210.6
  p99: 820.7
http.responses: 600
plugins.metrics-by-endpoint./tukano/rest/users/.codes.400: 200
plugins.metrics-by-endpoint./tukano/rest/users/{{ userId }}?pwd={{ pwd }}.co... 200
plugins.metrics-by-endpoint./tukano/rest/users/{{ userId}}?pwd={{ pwd }}.cod... 200
plugins.metrics-by-endpoint.response_time./tukano/rest/users/:
  min: 2
  max: 356
  mean: 5
  median: 3
  p95: 4
  p99: 7
plugins.metrics-by-endpoint.response_time./tukano/rest/users/{{ userId }}?pwd={{ pwd }}:
  min: 2
  max: 10
  mean: 2.8
  median: 3
  p95: 4
  p99: 6
plugins.metrics-by-endpoint.response_time./tukano/rest/users/{{ userId}}?pwd={{ pwd }}:
  min: 181
  max: 3446
  mean: 274.3
  median: 198.4
  p95: 278.7
  p99: 2836.2
vusers.completed: 200
vusers.created: 200
vusers.created_by_name.TuKanoWholeUserFlow: 200
vusers.failed: 0
vusers.session_length:
  min: 192.8
  max: 3843.4
  mean: 287.9
  median: 206.5
  p95: 295.9
  p99: 2836.2

- Without Redis Cache

```
http.codes.400: ..... 400
http.codes.404: ..... 200
http.downloaded_bytes: ..... 15200
http.request_rate: ..... 6/sec
http.requests: ..... 600
http.response_time:
  min: ..... 1
  max: ..... 2627
  mean: ..... 82.3
  median: ..... 3
  p95: ..... 202.4
  p99: ..... 399.5
http.response_time.4xx:
  min: ..... 1
  max: ..... 2627
  mean: ..... 82.3
  median: ..... 3
  p95: ..... 202.4
  p99: ..... 399.5
http.responses: ..... 600
plugins.metrics-by-endpoint./tukano/rest/users/.codes.400: ..... 200
plugins.metrics-by-endpoint./tukano/rest/users/{{ userId }}?pwd={{ pwd }}.co...  200
plugins.metrics-by-endpoint./tukano/rest/users/{{ userId}}?pwd={{ pwd }}.cod...  200
plugins.metrics-by-endpoint.response_time./tukano/rest/users/:
  min: ..... 2
  max: ..... 302
  mean: ..... 4.7
  median: ..... 3
  p95: ..... 4
  p99: ..... 5
plugins.metrics-by-endpoint.response_time./tukano/rest/users/{{ userId }}?pwd={{ pwd }}:
  min: ..... 1
  max: ..... 15
  mean: ..... 2.6
  median: ..... 2
  p95: ..... 4
  p99: ..... 8.9
plugins.metrics-by-endpoint.response_time./tukano/rest/users/{{ userId}}?pwd={{ pwd }}:
  min: ..... 171
  max: ..... 2627
  mean: ..... 239.5
  median: ..... 194.4
  p95: ..... 252.2
  p99: ..... 1939.5
vusers.completed: ..... 200
```

vusers.created: 200
vusers.created_by_name.TuKanoWholeUserFlow: 200
vusers.failed: 0
vusers.session_length:
  min: 183.4
  max: 2960.3
  mean: 252.1
  median: 202.4
  p95: 262.5
  p99: 1939.5

- Conclusion

In this case, Redis Cache does not improve response times and in some cases the latency increases, for example for the endpoints. This can be explained by the fact that in this case cache is only used for adding new members, not fulfilling its goal.

Upload_shorts

- With Redis Cache

http.codes.403: 30
http.codes.404: 30
http.downloaded_bytes: 0
http.request_rate: 6/sec
http.requests: 60
http.response_time:
  min: 3
  max: 429
  mean: 105.3
  median: 175.9
  p95: 202.4
  p99: 210.6
http.response_time.4xx:
  min: 3
  max: 429
  mean: 105.3
  median: 175.9
  p95: 202.4
  p99: 210.6
http.responses: 60
plugins.metrics-by-endpoint./tukano/rest/blobs/{{ blobUrl }}.codes.403: 30
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}?pwd={{ pwd }}.c... 30
plugins.metrics-by-endpoint.response_time./tukano/rest/blobs/{{ blobUrl }}:
  min: 3

  max: 429
  mean: 18
  median: 4
  p95: 5
  p99: 5
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}?pwd={{ pwd }}:
  min: 176
  max: 211
  mean: 192.5
  median: 190.6
  p95: 202.4
  p99: 206.5
vusers.completed: 30
vusers.created: 30
vusers.created_by_name.Upload short: 30
vusers.failed: 0
vusers.session_length:
  min: 188.4
  max: 671.7
  mean: 220
  median: 202.4
  p95: 228.2
  p99: 232.8

- Without Redis Cache

http.codes.403: 30
http.codes.404: 30
http.downloaded_bytes: 0
http.request_rate: 7/sec
http.requests: 60
http.response_time:
  min: 2
  max: 266
  mean: 97.2
  median: 169
  p95: 198.4
  p99: 262.5
http.response_time.4xx:
  min: 2
  max: 266
  mean: 97.2
  median: 169
  p95: 198.4
  p99: 262.5
http.responses: 60
plugins.metrics-by-endpoint./tukano/rest/blobs/{{ blobUrl }}.codes.403: 30

plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}?pwd={{ pwd }}.c... 30
plugins.metrics-by-endpoint.response_time./tukano/rest/blobs/{{ blobUrl }}:
  min: 2
  max: 260
  mean: 11.5
  median: 3
  p95: 4
  p99: 4
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}?pwd={{ pwd }}:
  min: 170
  max: 266
  mean: 183
  median: 179.5
  p95: 198.4
  p99: 198.4
vusers.completed: 30
vusers.created: 30
vusers.created_by_name.Upload short: 30
vusers.failed: 0
vusers.session_length:
  min: 179.2
  max: 547.3
  mean: 202.8
  median: 186.8
  p95: 210.6
  p99: 214.9

- Conclusion

Redis Cache does not have a significant role in Upload_shorts, even getting worse response times in some cases. Same as User_register, this is because a cache purpose is to retrieve wanted objects or information quicker, not create.

*This test had an error that it didn't recognize the path towards shorts.map and shorts.list that we could not solve, so it didn't upload the photos so it probably uploaded an empty file.

Realistic_flow

- With Redis Cache

errors.No shorts exist yet.: 10
http.codes.404: 20
http.downloaded_bytes: 0
http.request_rate: 4/sec
http.requests: 20

http.response_time:
  min: 170
  max: 983
  mean: 284
  median: 190.6
  p95: 757.6
  p99: 757.6
http.response_time.4xx:
  min: 170
  max: 983
  mean: 284
  median: 190.6
  p95: 757.6
  p99: 757.6
http.responses: 20
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ shortId }}/{{ userId }}/l... 1
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}/feed?pwd={{ pwd... 10
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}/followers?pwd={... 1
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}/shorts.codes.404: . 6
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId1 }}/{{ userId2 }}/... 2
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ shortId }}/{{ userId }}/likes?pwd={{ pwd }}:
  min: 175
  max: 175
  mean: 175
  median: 175.9
  p95: 175.9
  p99: 175.9
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}/feed?pwd={{ pwd }}:
  min: 170
  max: 241
  mean: 190.2
  median: 186.8
  p95: 198.4
  p99: 198.4
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}/followers?pwd={{ pwd }}:
  min: 983
  max: 983
  mean: 983
  median: 982.6
  p95: 982.6
  p99: 982.6
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}/shorts:
  min: 287
  max: 762
  mean: 376.2

  median: 295.9
  p95: 314.2
  p99: 314.2
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId1 }}/{{ userId2 }}/followers?pwd={{ pwd }}:
  min: 175
  max: 187
  mean: 181
  median: 175.9
  p95: 175.9
  p99: 175.9
vusers.completed: 20
vusers.created: 30
vusers.created_by_name.Download short: 6
vusers.created_by_name.Follow user: 2
vusers.created_by_name.Get Short Likes: 4
vusers.created_by_name.Get User Follows: 1
vusers.created_by_name.Get User's Shorts: 6
vusers.created_by_name.Like short: 1
vusers.created_by_name.View feed: 10
vusers.failed: 10
vusers.session_length:
  min: 178.5
  max: 1003.4
  mean: 291.6
  median: 194.4
  p95: 772.9
  p99: 772.9

- Without Redis Cache

errors.No shorts exist yet.: 12
http.codes.404: 18
http.downloaded_bytes: 0
http.request_rate: 4/sec
http.requests: 18
http.response_time:
  min: 166
  max: 3177
  mean: 653.2
  median: 368.8
  p95: 2101.1
  p99: 2101.1
http.response_time.4xx:
  min: 166
  max: 3177

  mean: 653.2
  median: 368.8
  p95: 2101.1
  p99: 2101.1
http.responses: 18
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ shortId }}/{{ userId }}/l... 5
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}/feed?pwd={{ pwd... 9
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}/followers?pwd={... 1
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId }}/shorts.codes.404: . 1
plugins.metrics-by-endpoint./tukano/rest/shorts/{{ userId1 }}/{{ userId2 }}/... 2
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ shortId }}/{{ userId }}/likes?pwd={{ pwd }}:
  min: 166
  max: 408
  mean: 275.4
  median: 247.2
  p95: 383.8
  p99: 383.8
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}/feed?pwd={{ pwd }}:
  min: 179
  max: 3177
  mean: 971.4
  median: 407.5
  p95: 2101.1
  p99: 2101.1
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}/followers?pwd={{ pwd }}:
  min: 687
  max: 687
  mean: 687
  median: 685.5
  p95: 685.5
  p99: 685.5
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId }}/shorts:
  min: 371
  max: 371
  mean: 371
  median: 368.8
  p95: 368.8
  p99: 368.8
plugins.metrics-by-endpoint.response_time./tukano/rest/shorts/{{ userId1 }}/{{ userId2 }}/followers?pwd={{ pwd }}:
  min: 187
  max: 392
  mean: 289.5
  median: 186.8
  p95: 186.8

```
  p99: ..... 186.8
vusers.completed: ..... 18
vusers.created: ..... 30
vusers.created_by_name.Download short: ..... 9
vusers.created_by_name.Follow user: ..... 2
vusers.created_by_name.Get Short Likes: ..... 3
vusers.created_by_name.Get User Follows: ..... 1
vusers.created_by_name.Get User's Shorts: ..... 1
vusers.created_by_name.Like short: ..... 5
vusers.created_by_name.View feed: ..... 9
vusers.failed: ..... 12
vusers.session_length:
  min: ..... 169.5
  max: ..... 3189.6
  mean: ..... 662.6
  median: ..... 391.6
  p95: ..... 2143.5
  p99: ..... 2143.5
```

- Conclusion

In this case, the use of Redis Cache was very valuable:

- Response Times and Consistency improved:

  With Redis Cache, response times were significantly lower (653.2 ms to 284 ms) and latency was better (P95, P99). For this topic, Redis Cache helped by retrieving user data more quickly, reducing the database workload and improving latency.

- There were less 404 Errors:

  This can be an error in the code, maybe the shorts weren't being correctly deleted from the cache so sometimes they had been already eliminated before and it still had access to them, minimizing some of the 404 Errors (Not Found). Nonetheless, consistency improved.

Overall, there is a significant improvement in the metrics by using Redis Cache, this being the best example from the three to show what changed.

Final Conclusion

Same as mentioned in the beginning, the objective was to compare the results of using PostgreSQL Cosmos DB (which is fully implemented) with NoSQL Cosmos DB but unfortunately we had an error regarding the PostgreSQL Cosmos DB and this was not achieved.

As we expected, the cache increased the time of POST operations but decreased the time of GET operations, which in the context of Tukano is better since in a real life scenario, GET operations are more used. Concluding, **Redis Cache had a positive effect on TuKano**.