
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

DAT203-R

Dive deep into Amazon DynamoDB using design puzzles

Jason Hunter (he/him)

Principal Solutions Architect, DynamoDB
AWS

Robert McCauley

Senior Solutions Architect, DynamoDB
AWS

Agenda

We're going to collaboratively explore a set of design puzzlers:

1. "Amuse Bouche" puzzlers to get us warmed up

2. A design to vend redemption codes

3. A design to look up IP addresses

For each puzzler, I think I have a good answer and will guide you there, but maybe you'll surprise me with something better!

"Amuse Bouche" warm-ups

“My on-demand table is very slow to load some initial data. I had to switch to Provisioned and it loaded much faster”

What could explain this?

“I'm looking at a table with lots of write traffic but zero read traffic. Why would anyone have a table they never read from?”

What could explain this?

Vending redemption codes

Codes table

- Imagine you're vending unique redemption codes as a service. Your partition key (PK) is each marketing campaign, and the sort key (SK) holds the codes as a random alpha-numeric. You might have thousands of unique PKs and a million unique SKs. You want to efficiently pull a code for a campaign, after which you will use it and then "burn" it from the table so it's never vended again.

How do you do this?

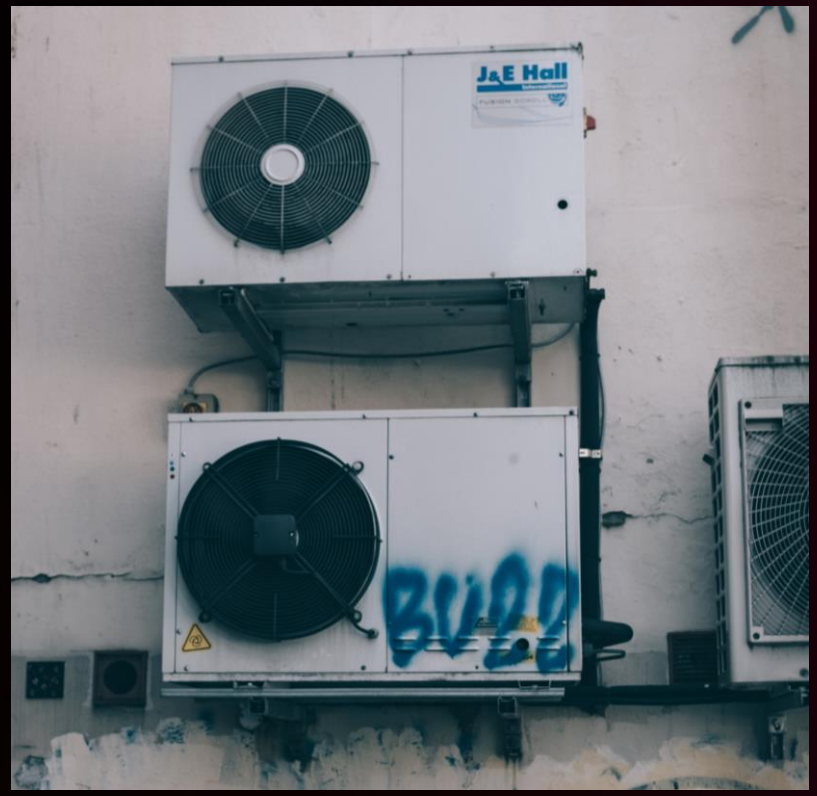
Campaign (Partition key) : String	Code (Sort key) : String
SFDC#725	PJLM7-QG408
SFDC#725	OUZAJ-ULCVZ
SFDC#725	TR4JW-H5VPT
SFDC#910	HTDRL-5C5AF
SFDC#910	I43CX-GRQP9
SFDC#910	Z38B6-KNQ2S

Example RedemptionCode table data

Atomic get and delete

First, how do you get and delete an item safely?

Hint:



Pick an item

What's the best way to pick an item to "burn"?

What happens if you always pick the first?

Can you pull the first N items and select a random one? Pros and cons?

Can you pull a true random code from a specific campaign?

Keep an audit

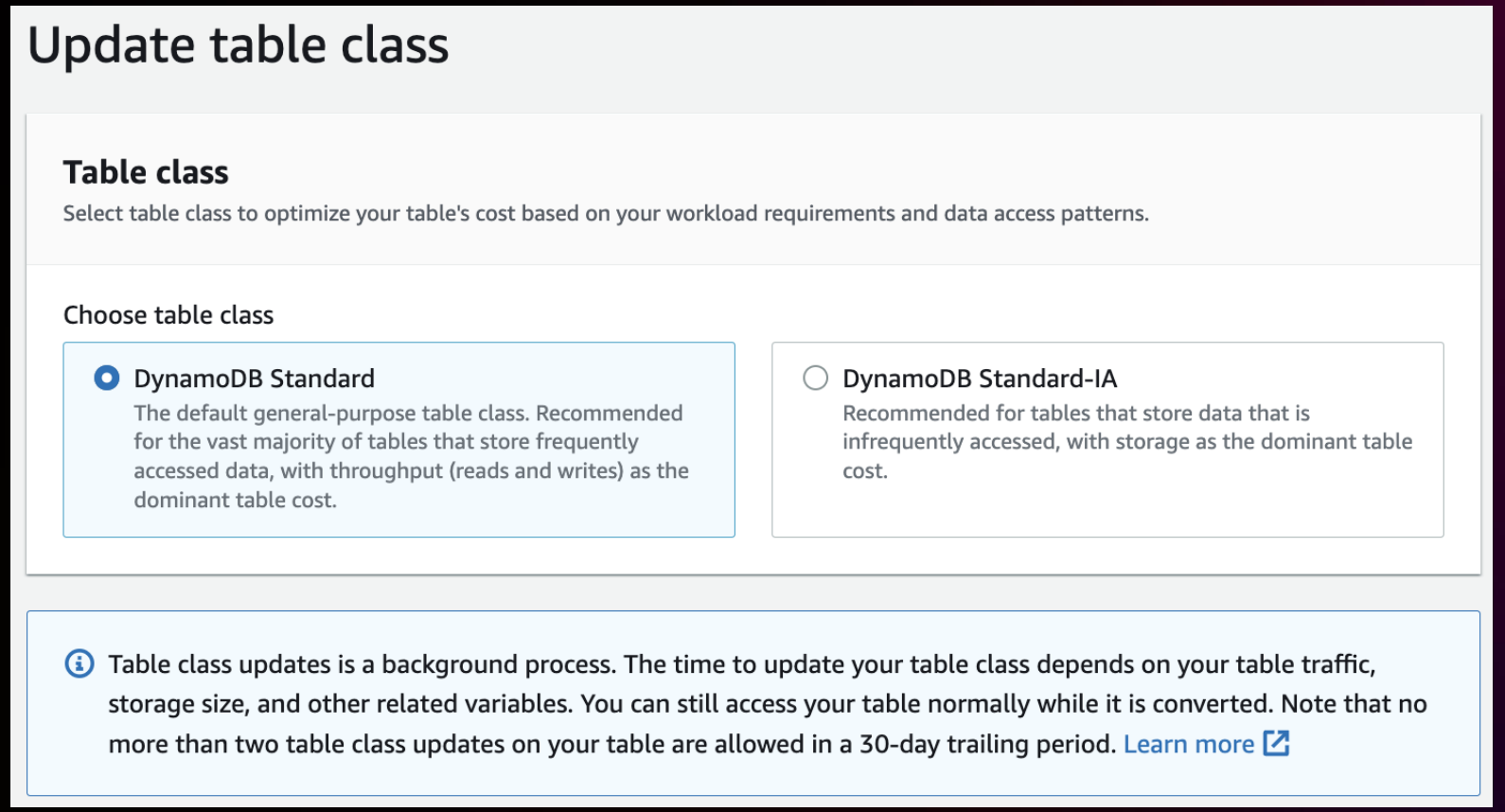
What if you need to keep a record of "burned" codes?

Must you use TransactWriteItems?

Can Amazon DynamoDB Streams be helpful?

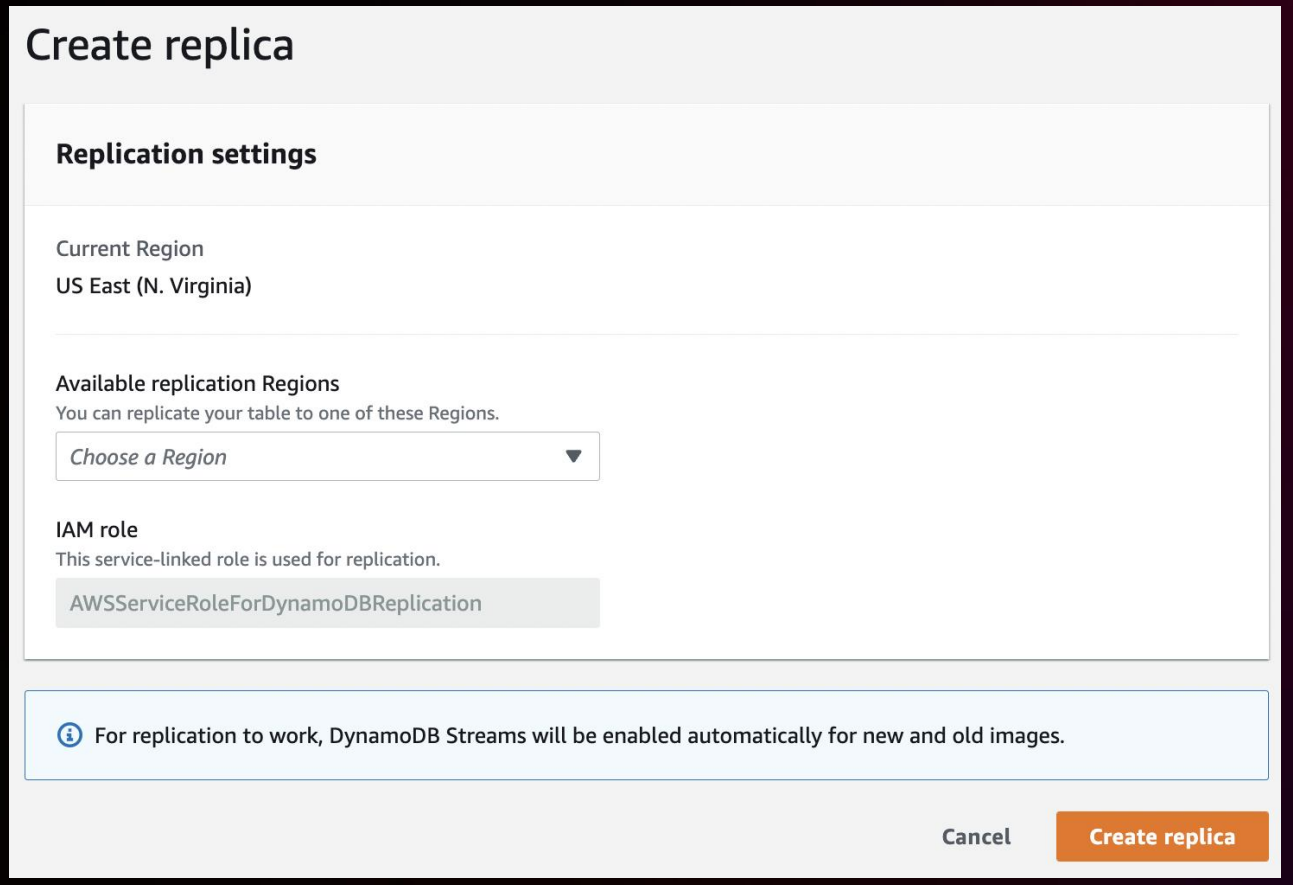
Table class

What table class would be best for an audit table?



Global tables

How would this design work with global tables?



Looking up IP addresses

IP lookup table

- Imagine you want to look up metadata about incoming IPv4 addresses (such as owner, country of origin, security rules to apply, and more). Your metadata is represented by 500,000 IP address ranges, each having a start and end address as well as the metadata payload to be returned for any IP address in that range

How do you data model this? To what query rate can it scale?

The algorithm

First, what's the algorithm?

Let's assume continuous non-overlapping ranges

You can do this with a single query

What data type to use for the sort key?

What value do you pick for the partition key?

Option: IP address as a string

PK	SK	Metadata
0	10.0.0.0	Range 10.0.0.0/8 metadata
0	11.0.0.0	Range 11.0.0.0/16 metadata
0	11.128.0.0	Range 11.128.0.0/16 metadata
0	12.0.0.0	Range 12.0.0.0/8 metadata

Option: IP address as a number

PK	SK	Metadata
0	167772160	Range 10.0.0.0/8 metadata
0	184549376	Range 11.0.0.0/16 metadata
0	192937984	Range 11.128.0.0/16 metadata
0	201326592	Range 12.0.0.0/8 metadata

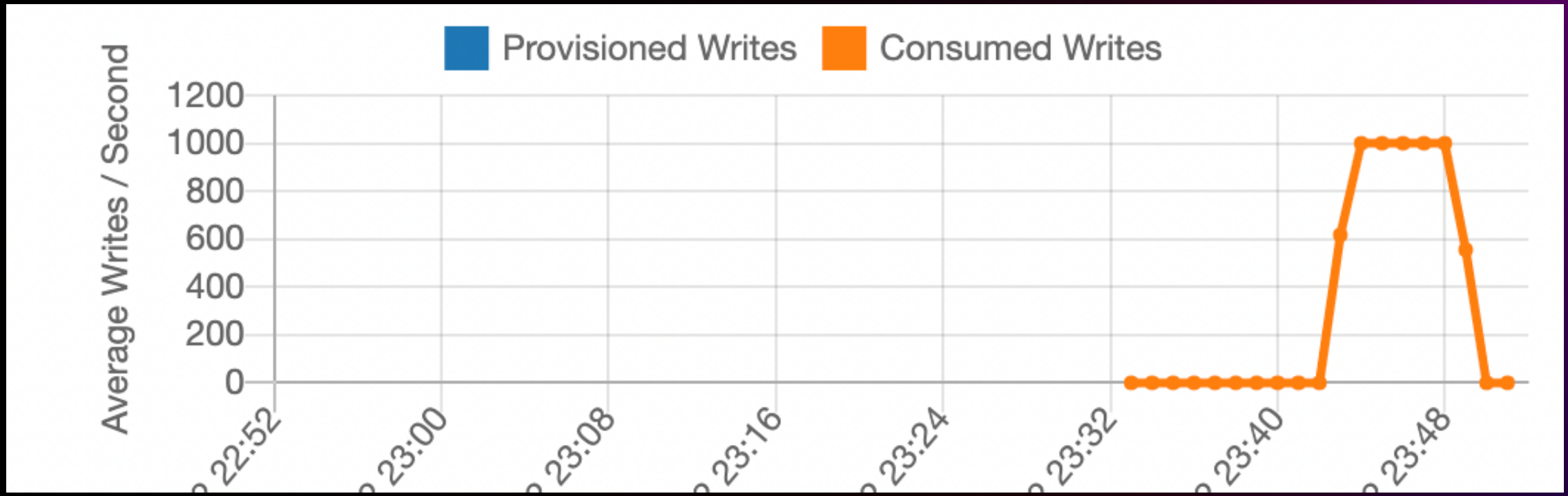
Option: A higher cardinality partition key

PK	SK	Metadata
10	167772160	Range 10.0.0.0/8 metadata
11	184549376	Range 11.0.0.0/16 metadata
11	192937984	Range 11.128.0.0/16 metadata
12	201326592	Range 12.0.0.0/8 metadata

Loading speed

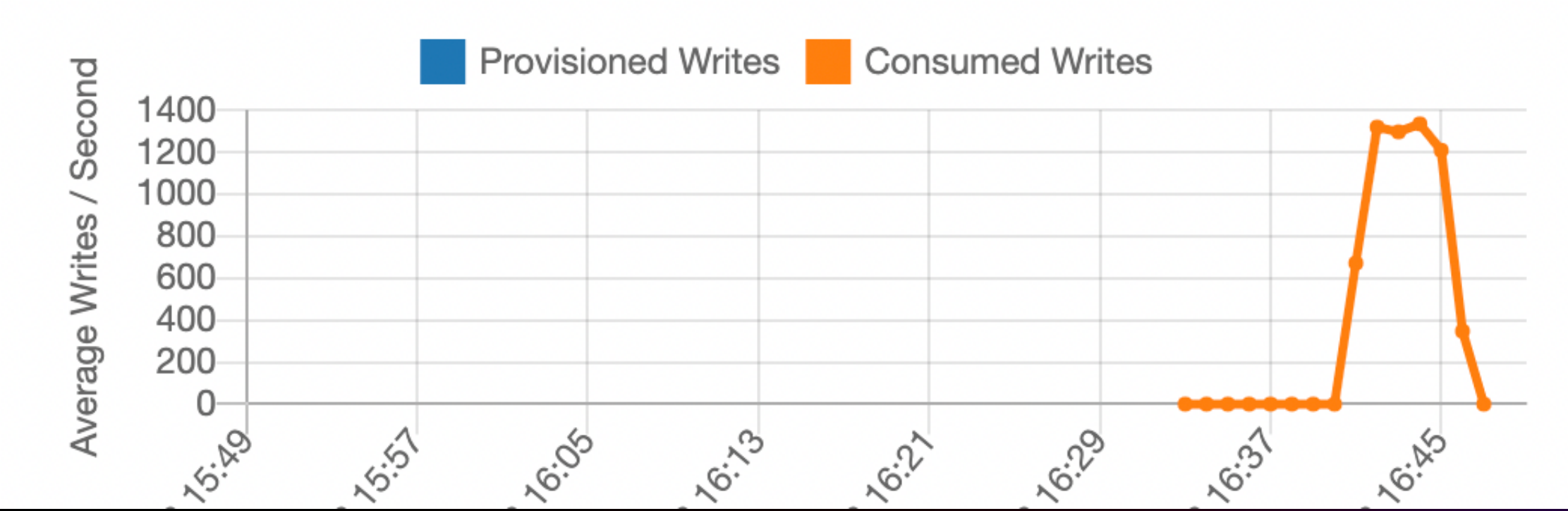
How fast should we expect to load the 500,000 items?

Loading from a CSV, using one partition key



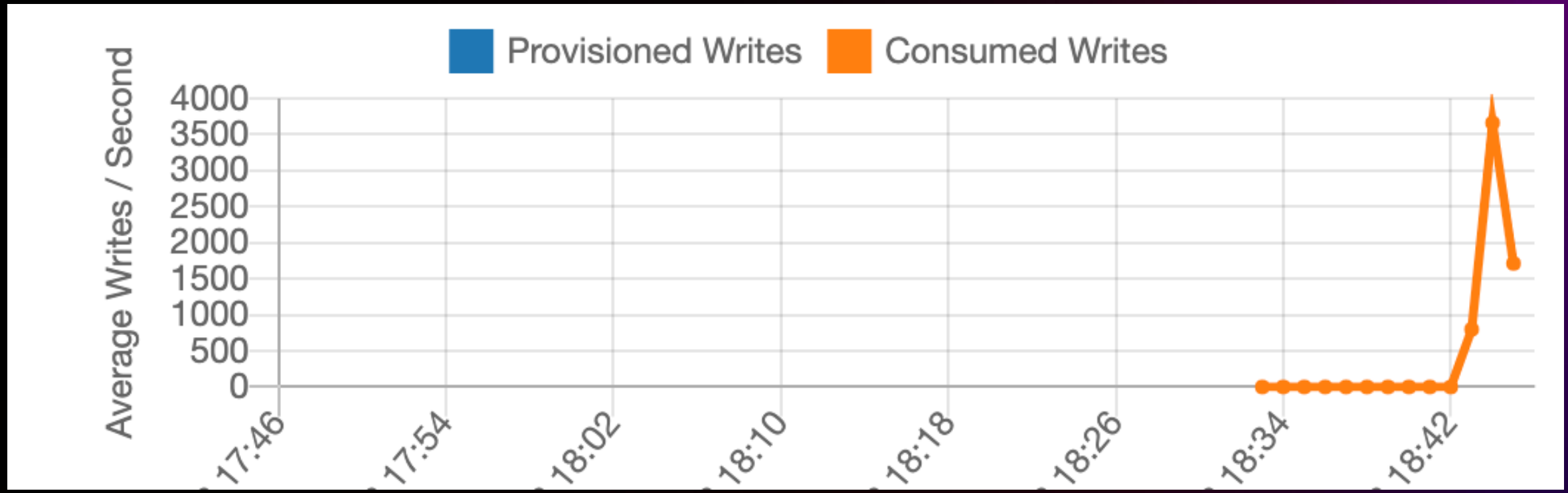
A 6 minute load. Why do we see this performance? Is this good or bad?

Loading again with 200+ partition keys



Why do we see this performance?

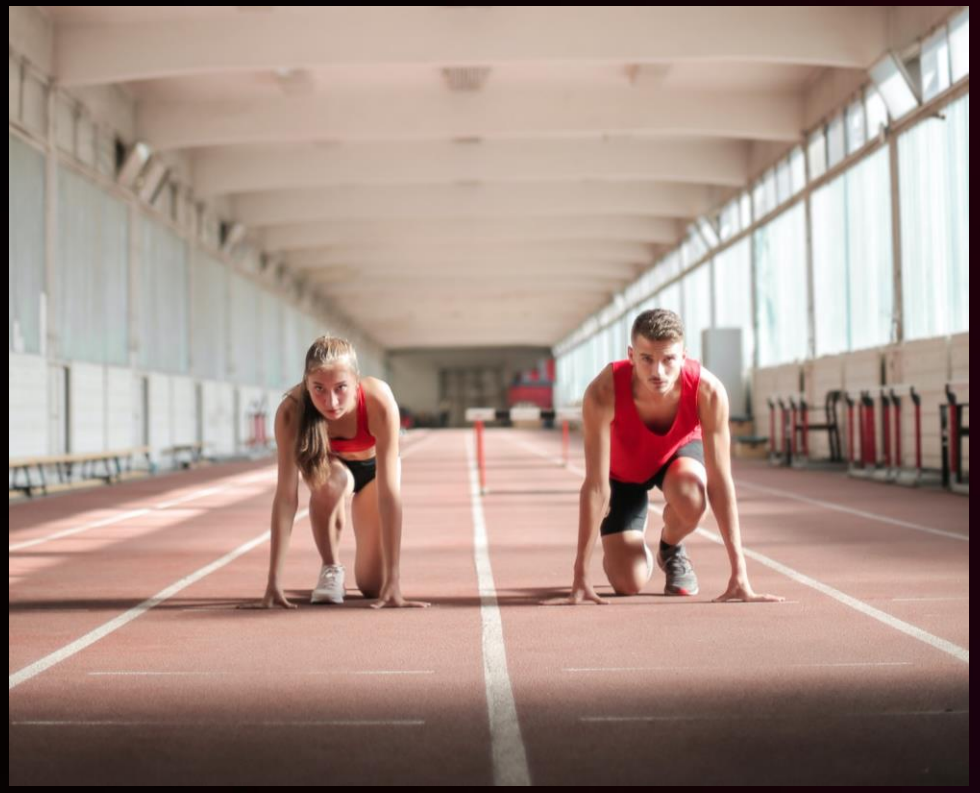
Loading from a tweaked CSV, still 200+ PKs



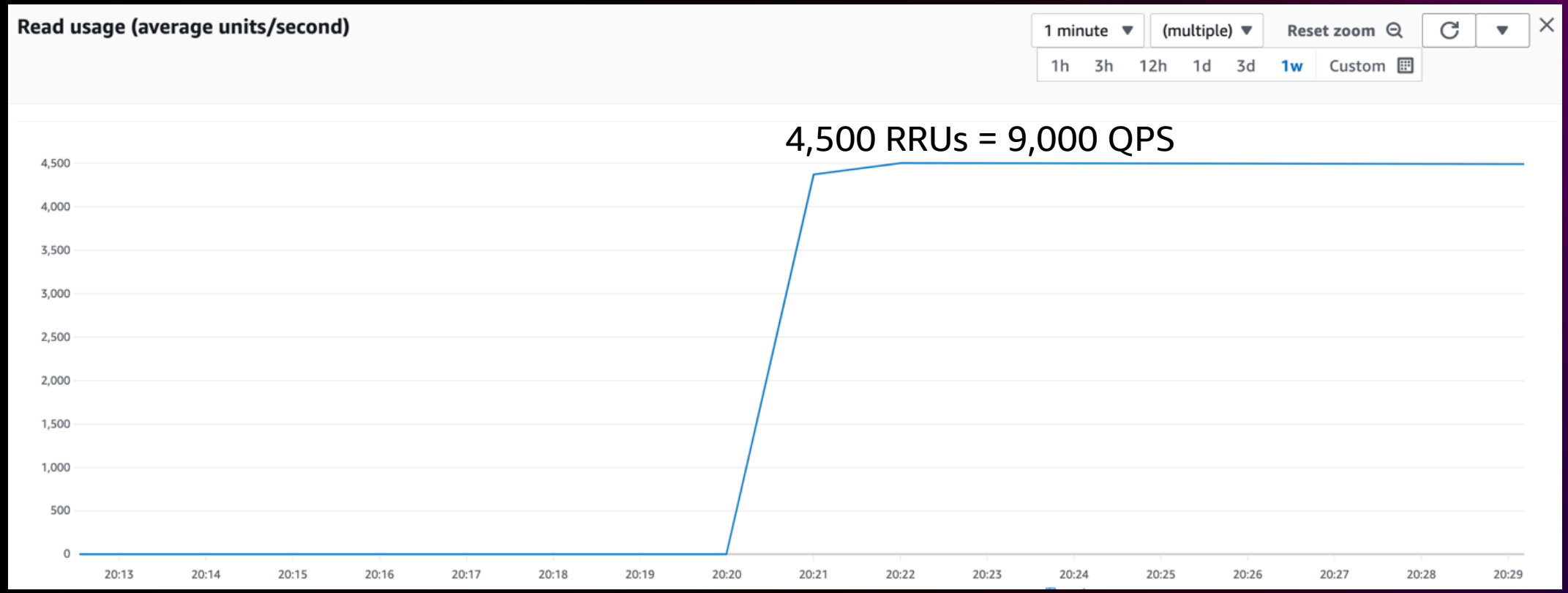
What change was made to the CSV to improve performance like this?

Querying rate

If we use one PK, how many queries per second should we expect?
(assume eventually consistent reads)

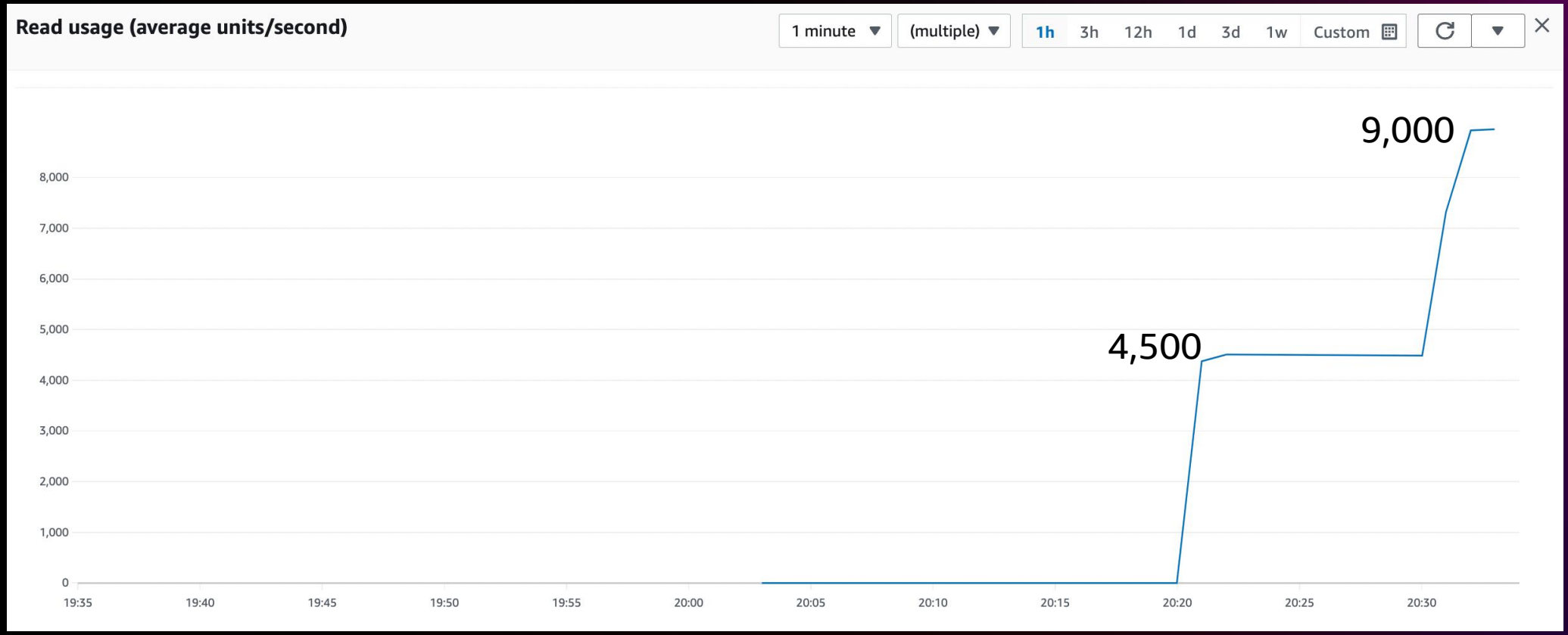


New on-demand table, using one partition key



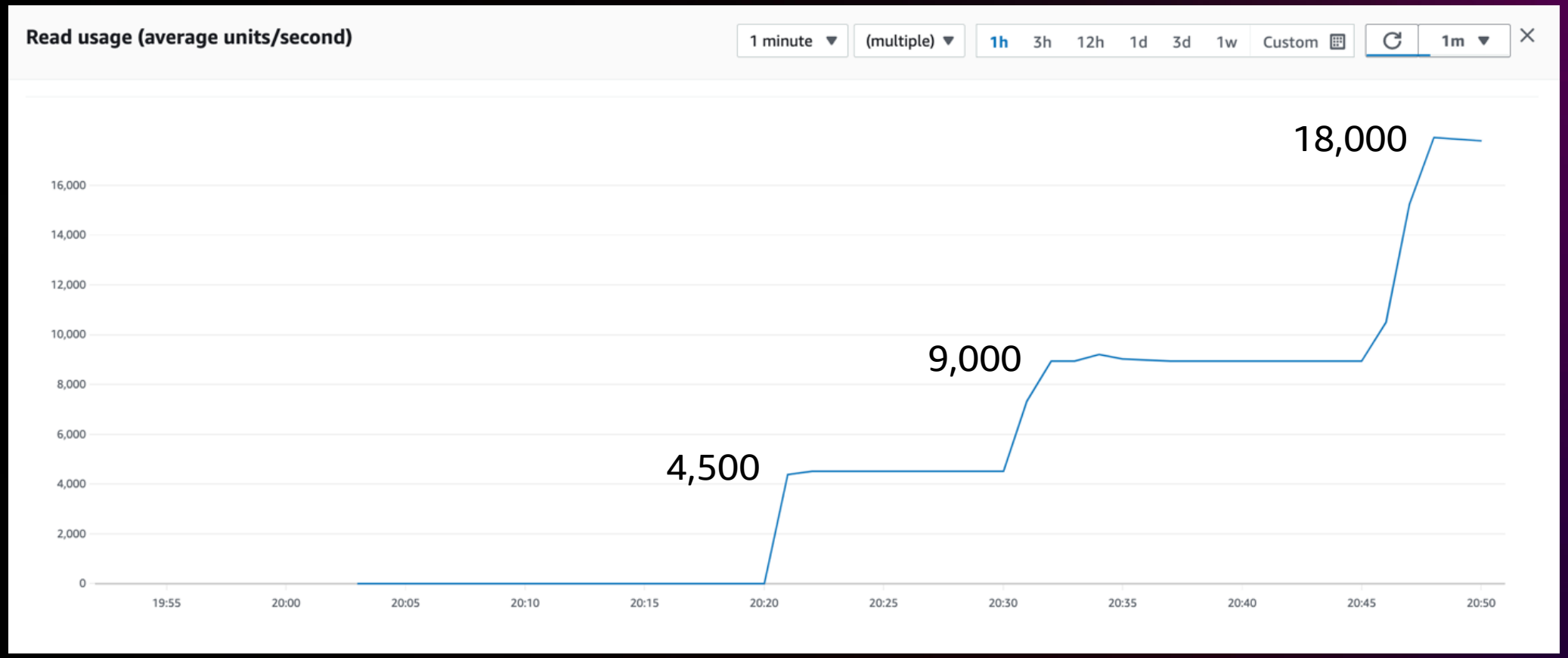
5 minutes of traffic. Why do we see this performance?

10 minutes later...



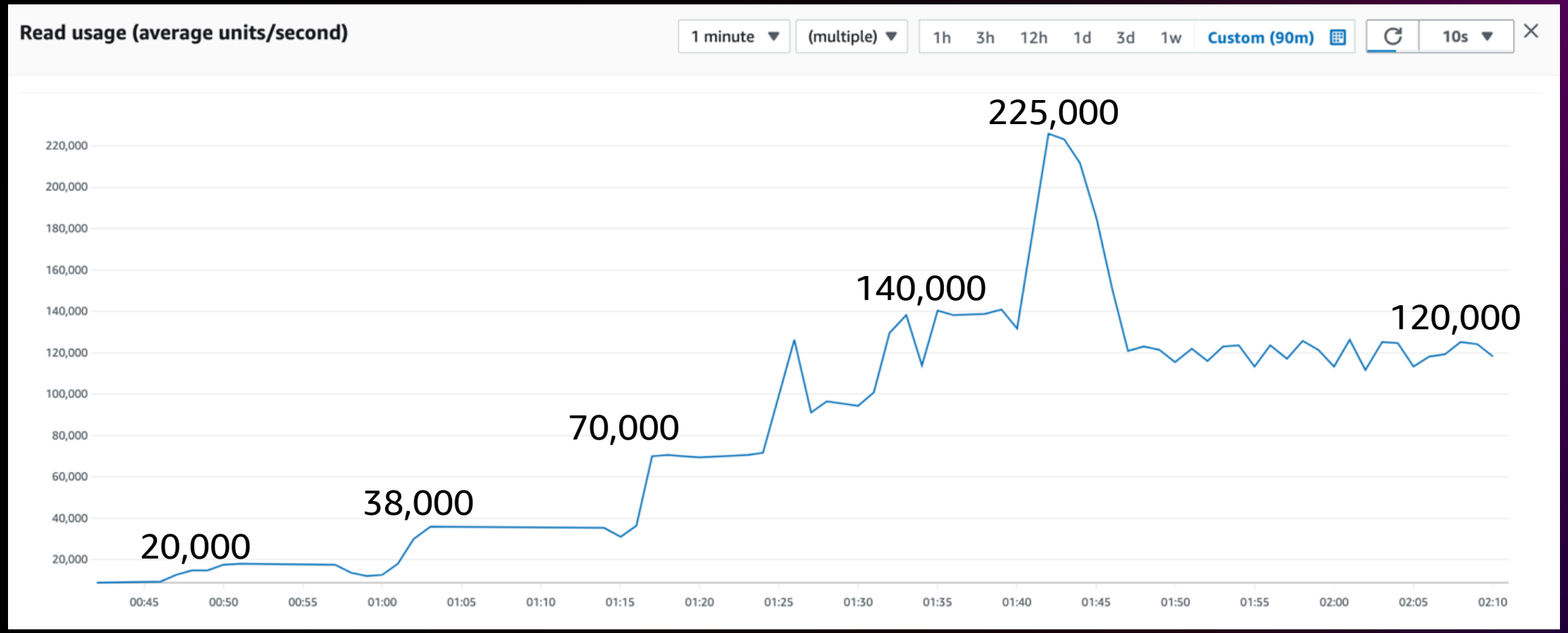
Why do we see this performance?

15 minutes later...



Why do we see this performance?

After a 90 minute run...



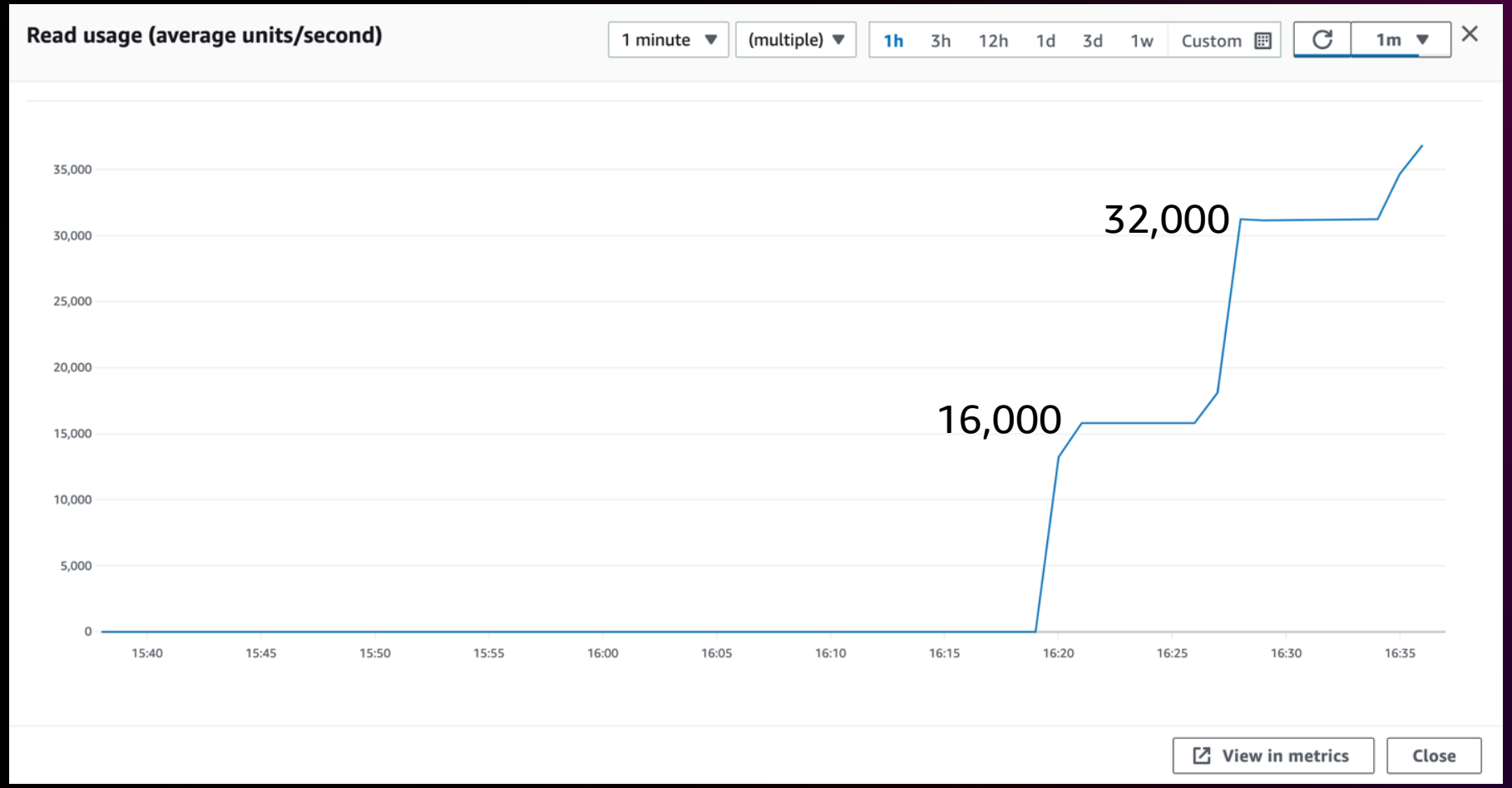
Why do we see this performance?

Partition key cardinality

Is there any advantage to high cardinality partition keys?

Hint:

New on-demand table, now 200+ partition keys



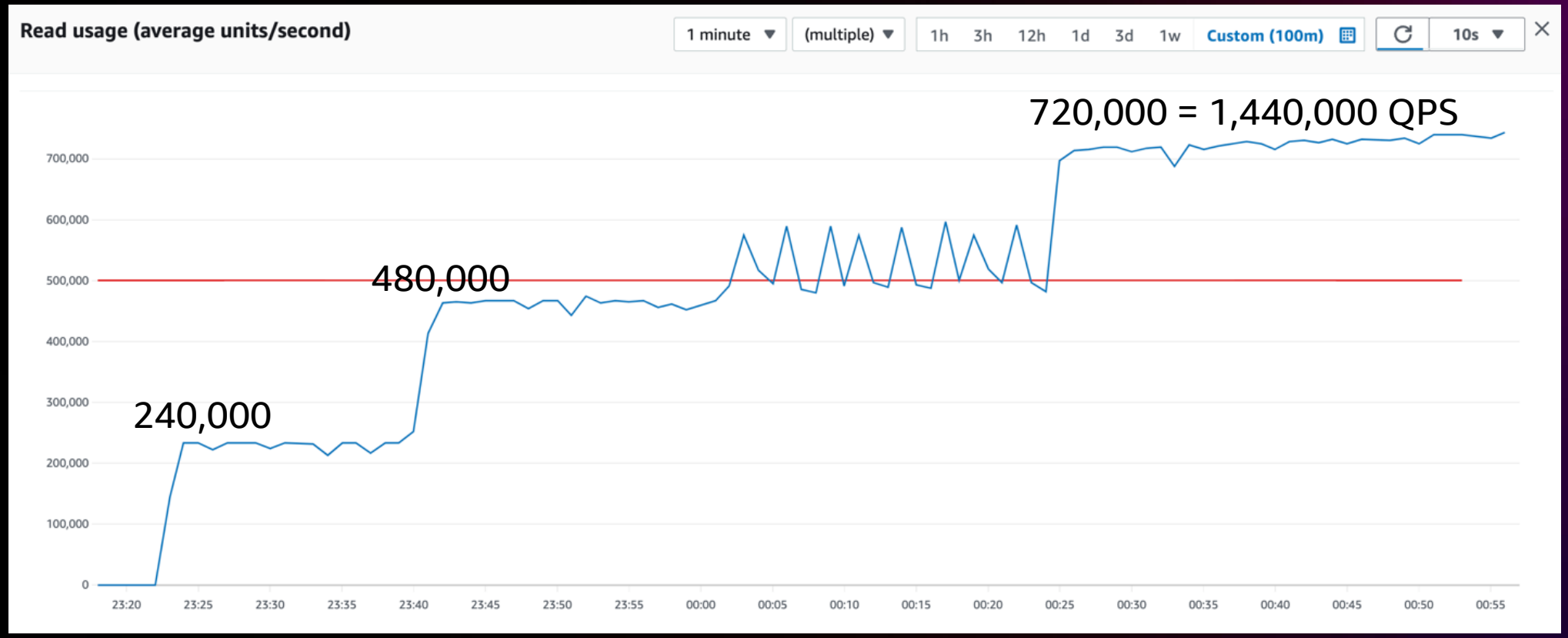
Why do we see this performance?

Scaling up

What max query rate we should expect?

Hint:

Provisioned at 500,000 RCU, using 200+ PKs



90 minute run. Why do we see this performance?

Amazon CloudWatch “query latency”

Average query request latency as measured during the 90 minute run

Thank you!

Please complete the session survey in the **mobile app**