GraphQL PoC

Loirto Alves dos Santos

Paulo Bolinhas

MERCEDES-BENZ .iO

Context

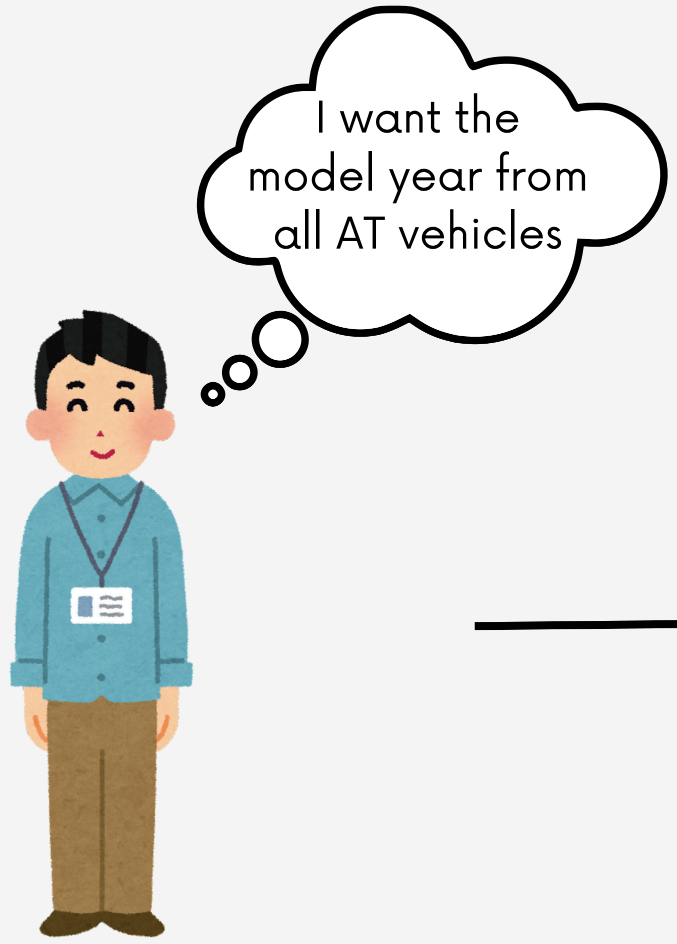
This Proof of Concept (PoC) showcases how GraphQL can be used for the 'os4vs' core search endpoint, providing a flexible and efficient alternative to REST (migration).

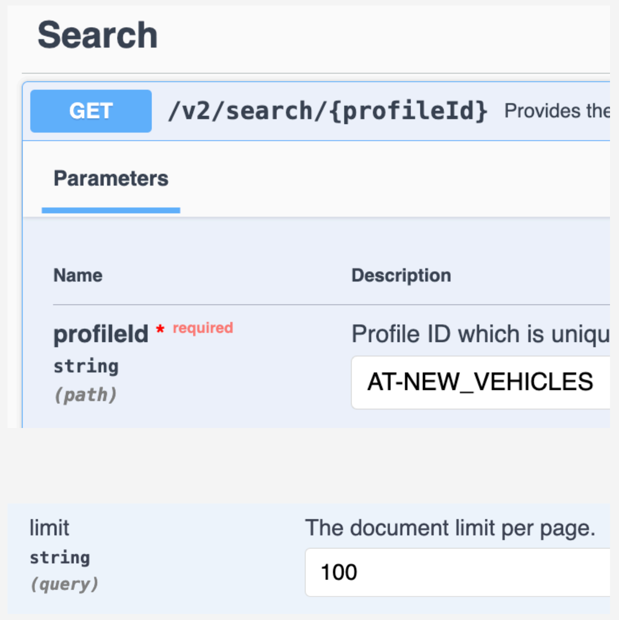
The objective is not only to reduce the overall processing time, increasing the performance, but also to fetch only the specific data fields required.

Problem

Our current search endpoint always retrieves the entire document, regardless of the fields needed.

Therefore, the fewer fields I need, the more inefficient and time-consuming the query becomes.



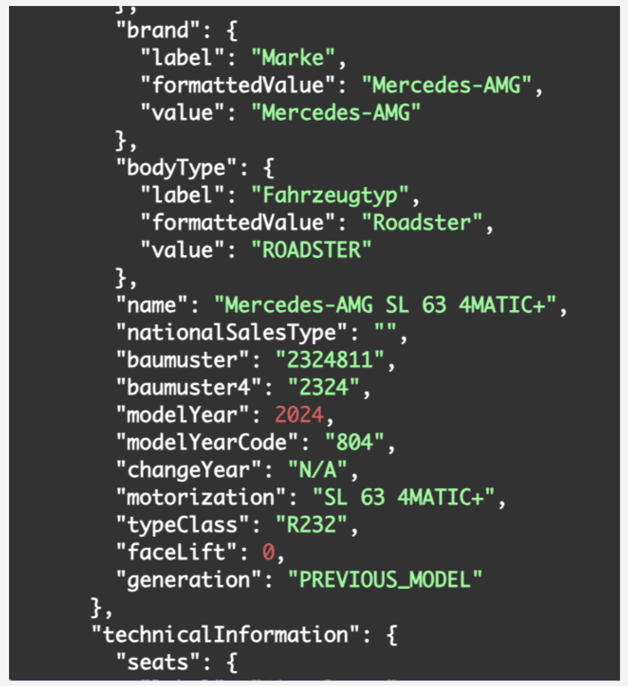


Search

GET /v2/search/{profileId} Provides the

Parameters

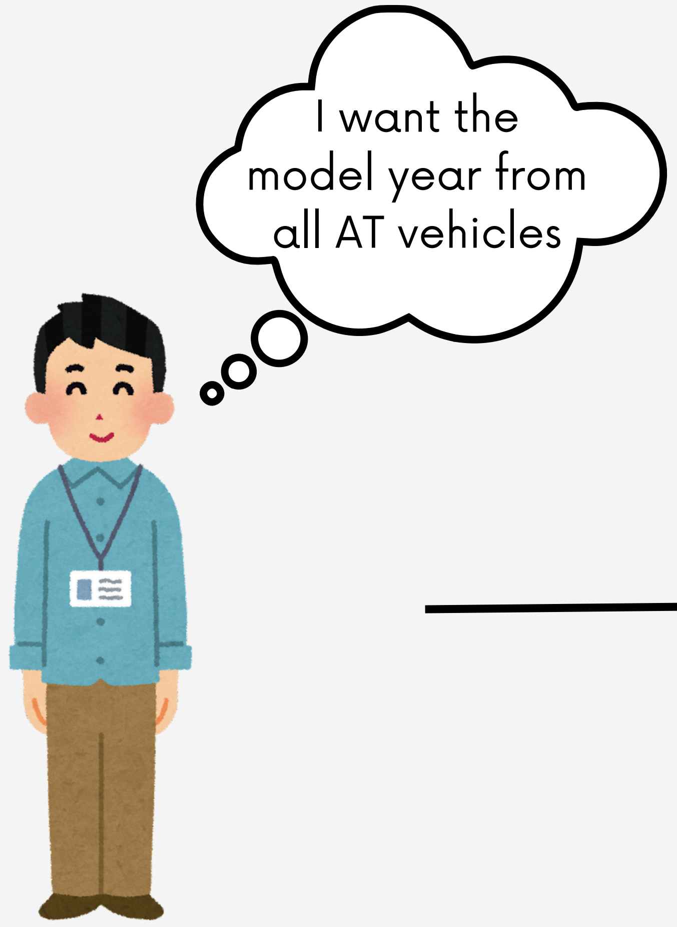
Name	Description
profileId * required string (path)	Profile ID which is uniqu AT-NEW_VEHICLES
limit string (query)	The document limit per page. 100

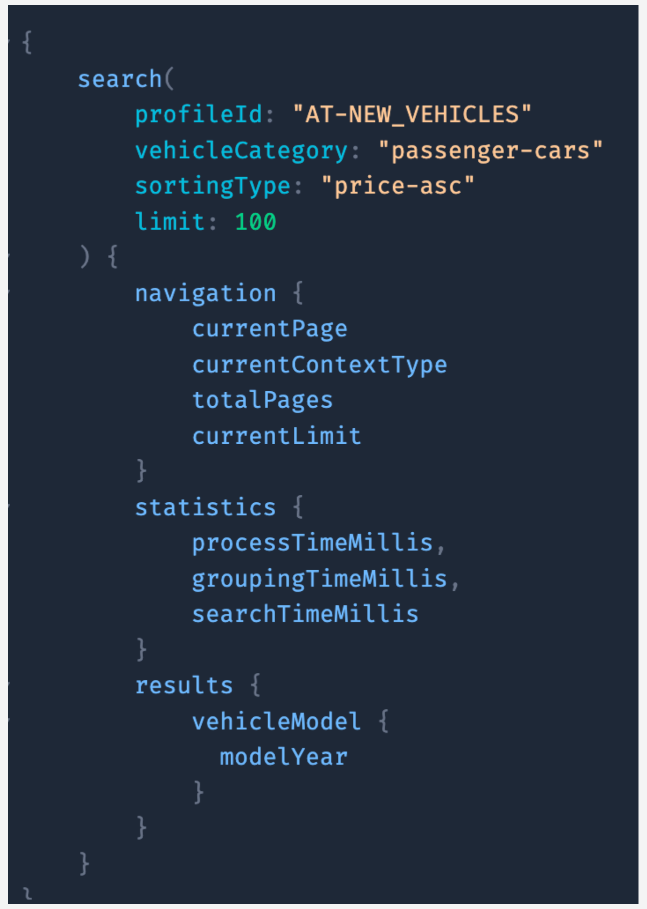


```
  "brand": {
    "label": "Marke",
    "formattedValue": "Mercedes-AMG",
    "value": "Mercedes-AMG"
  },
  "bodyType": {
    "label": "Fahrzeugtyp",
    "formattedValue": "Roadster",
    "value": "ROADSTER"
  },
  "name": "Mercedes-AMG SL 63 4MATIC+",
  "nationalSalesType": "",
  "baumuster": "2324811",
  "baumuster4": "2324",
  "modelYear": 2024,
  "modelYearCode": "804",
  "changeYear": "N/A",
  "motorization": "SL 63 4MATIC+",
  "typeClass": "R232",
  "faceLift": 0,
  "generation": "PREVIOUS_MODEL"
},
"technicalInformation": {
  "seats": {
```

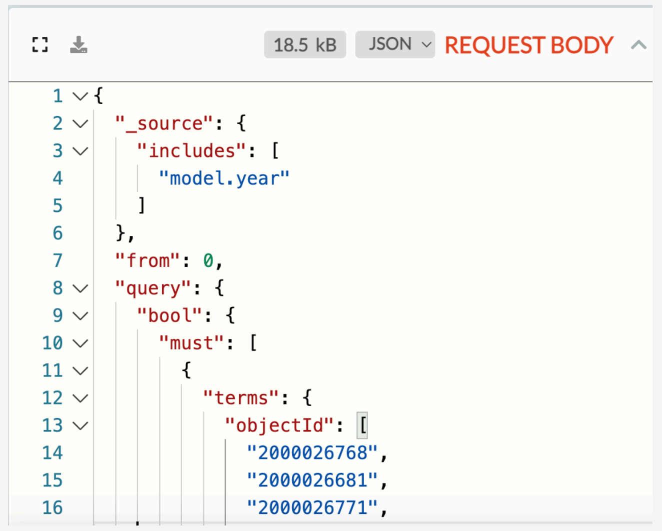
Solution

Using GraphQL query functionalities our endpoint allows us to decide which fields we want it to fetch, reducing the results (all data) to the required results, increasing the performance and erasing the unnecessary retrieved data.



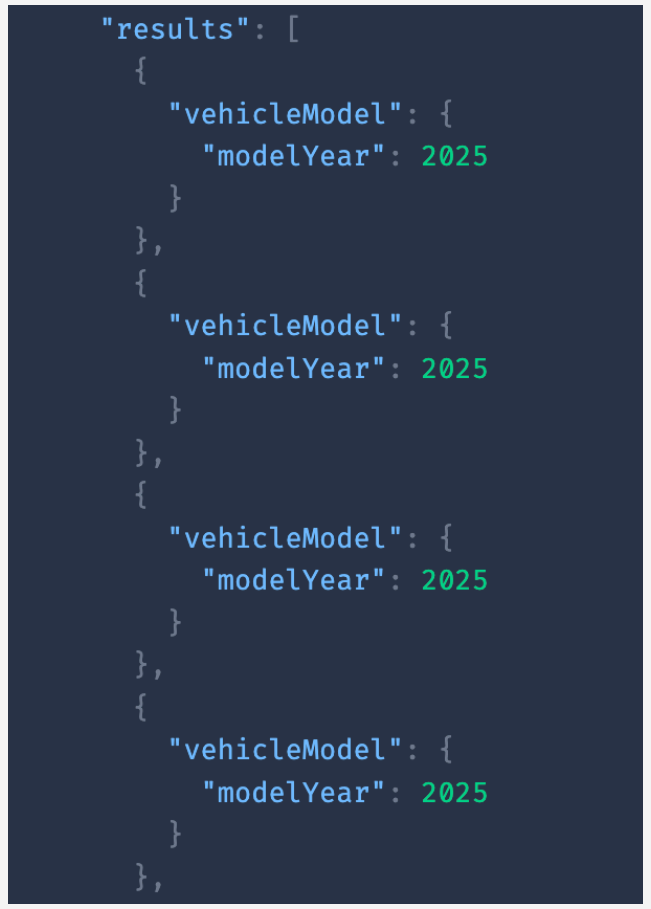


```
{
  search(
      profileId: "AT-NEW_VEHICLES"
      vehicleCategory: "passenger-cars"
      sortingType: "price-asc"
      limit: 100
  ) {
      navigation {
          currentPage
          currentContextType
          totalPages
          currentLimit
      }
      statistics {
          processTimeMillis,
          groupingTimeMillis,
          searchTimeMillis
      }
      results {
          vehicleModel {
            modelYear
          }
      }
  }
}
```



18.5 kB JSON REQUEST BODY

```
{
  "_source": {
    "includes": [
      "model.year"
    ]
  },
  "from": 0,
  "query": {
    "bool": {
      "must": [
        {
          "terms": {
            "objectId": [
              "200026768",
              "200026681",
              "200026771",
```

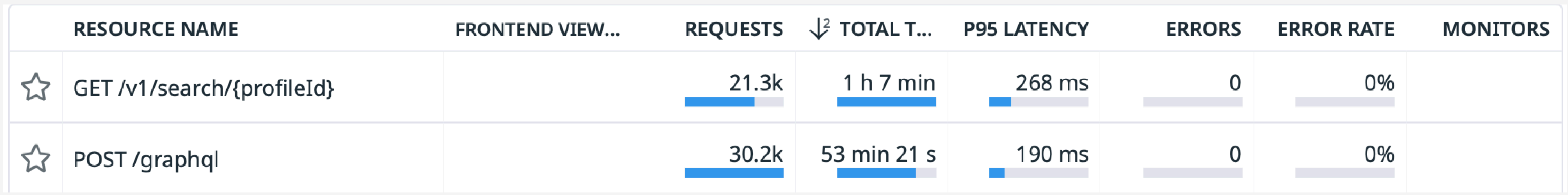


```
"results": [
  {
    "vehicleModel": {
      "modelYear": 2025
    }
  },
  {
    "vehicleModel": {
      "modelYear": 2025
    }
  },
  {
    "vehicleModel": {
      "modelYear": 2025
    }
  },
  {
    "vehicleModel": {
      "modelYear": 2025
    }
  },
```

Testing

Original REST search endpoint
x
GraphQL query that replicates original

(scalability)

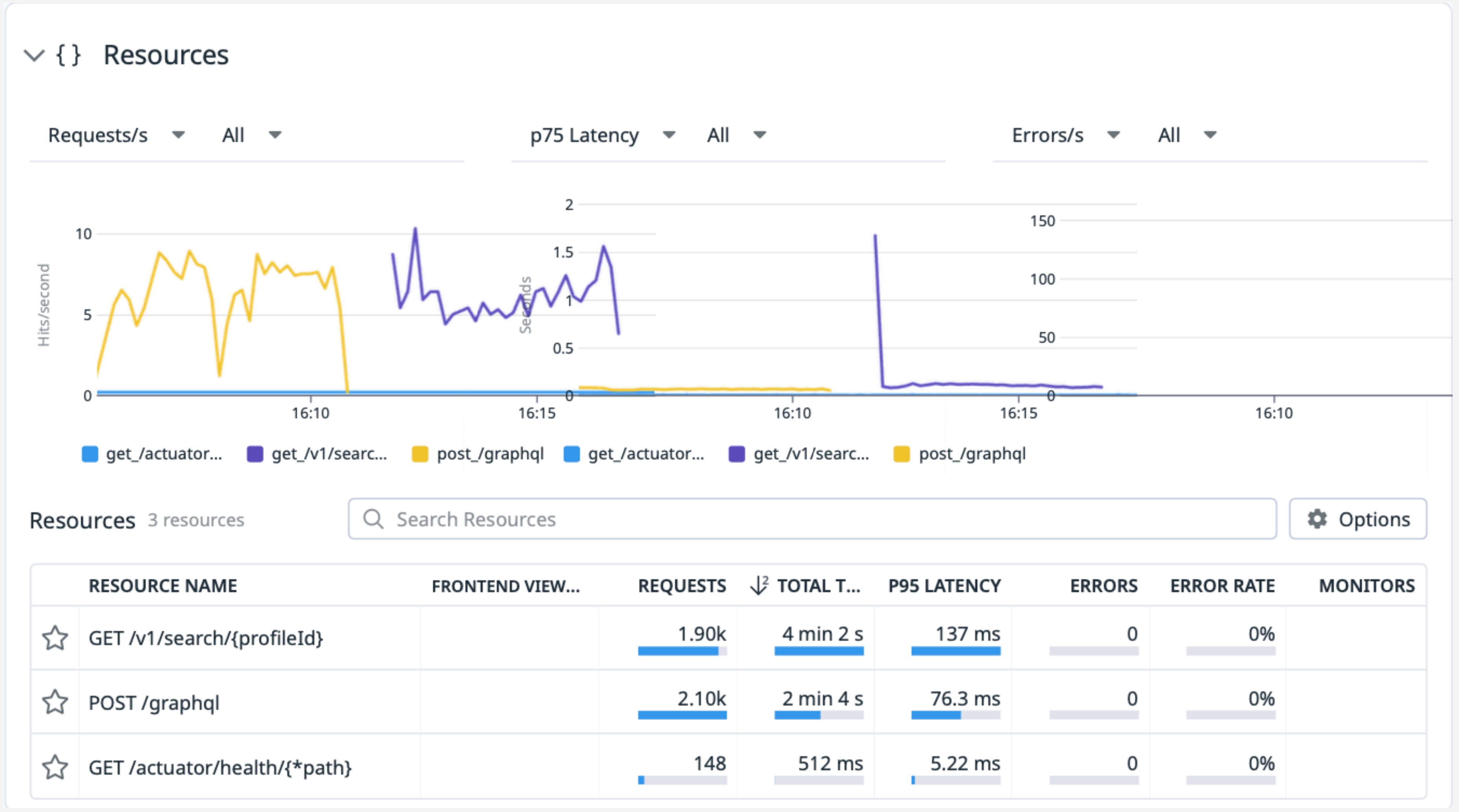


	RESOURCE NAME	FRONTEND VIEW...	REQUESTS	TOTAL T...	P95 LATENCY	ERRORS	ERROR RATE	MONITORS
☆	GET /v1/search/{profileId}		21.3k	1 h 7 min	268 ms	0	0%	
☆	POST /graphql		30.2k	53 min 21 s	190 ms	0	0%	

Testing

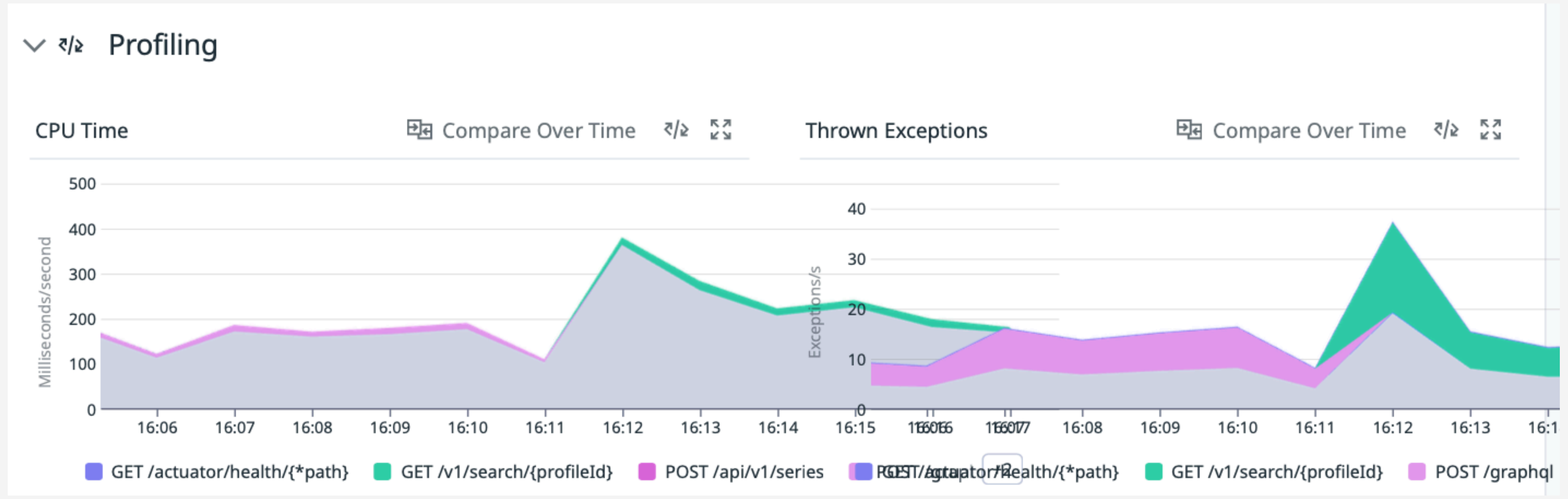
Original REST search endpoint
x
GraphQL query that replicates original

(performance)



Resources 3 resources

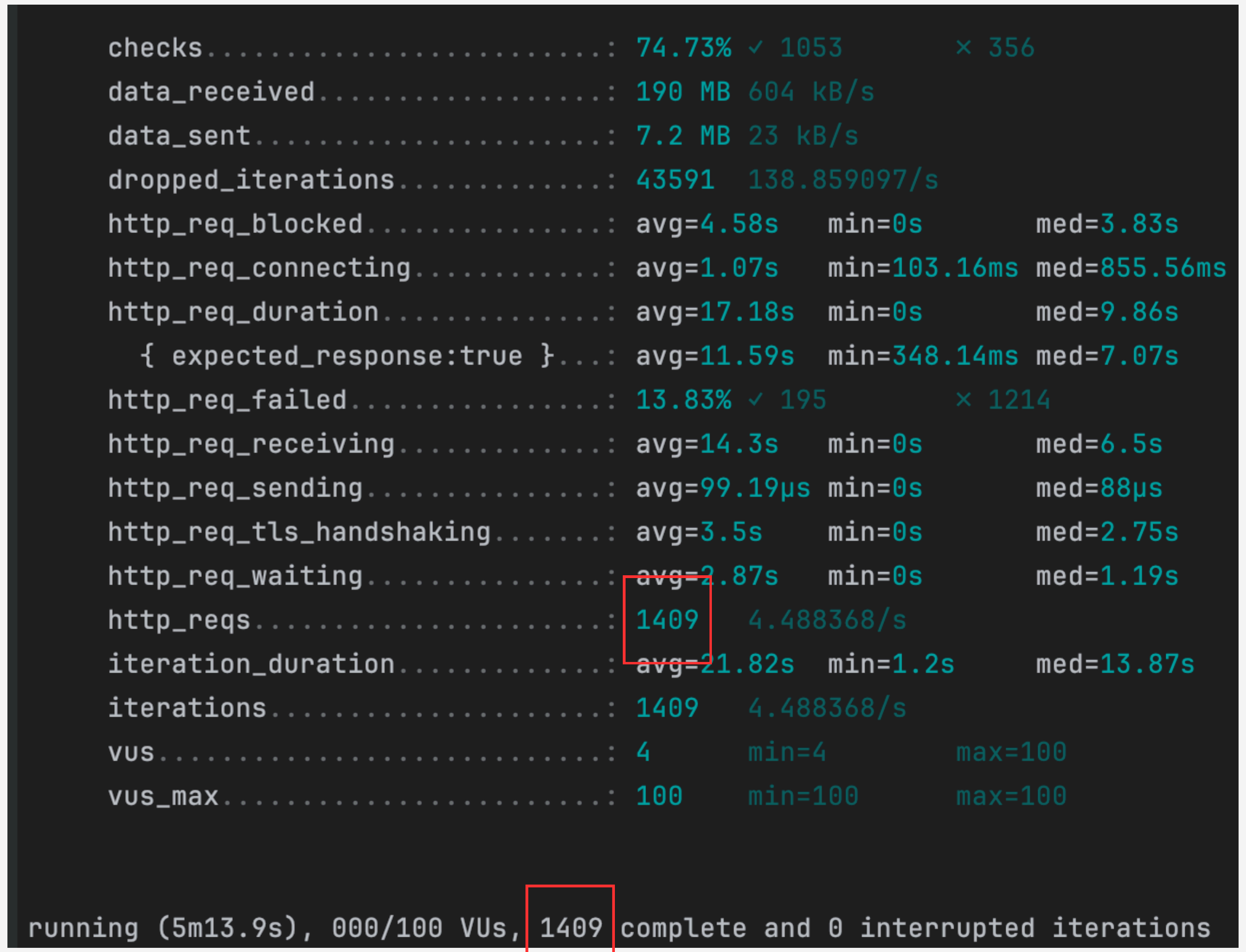
RESOURCE NAME	FRONTEND VIEW...	REQUESTS	TOTAL T...	P95 LATENCY	ERRORS	ERROR RATE	MONITORS
☆ GET /v1/search/{profileId}		1.90k	4 min 2 s	137 ms	0	0%	
☆ POST /graphql		2.10k	2 min 4 s	76.3 ms	0	0%	
☆ GET /actuator/health/{*path}		148	512 ms	5.22 ms	0	0%	



Testing

In 5 minutes (.js datadog script):

GraphQL query that replicates original



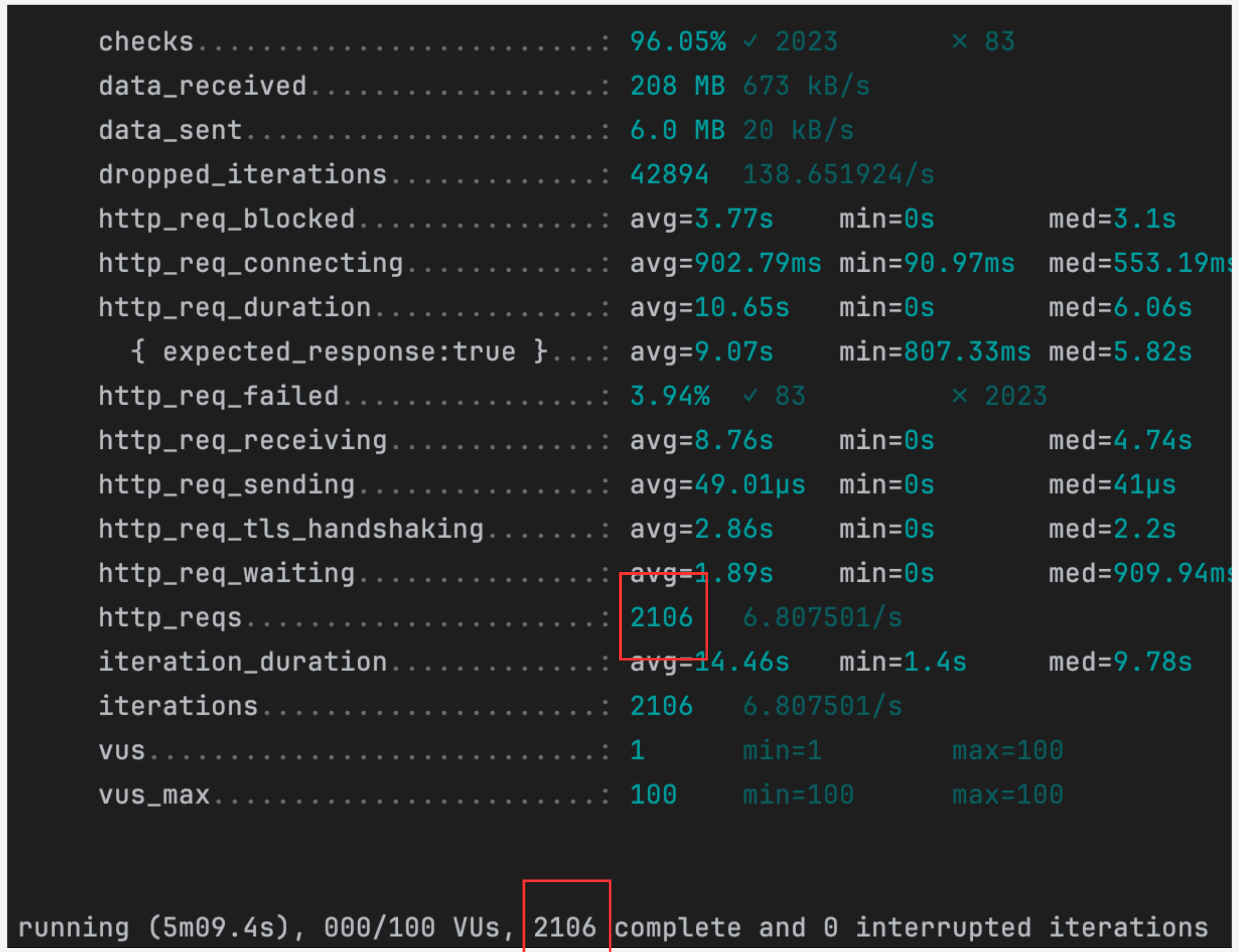
```
     checks.........................: 74.73% ✓ 1053        ✗ 356
     data_received..................: 190 MB 604 kB/s
     data_sent......................: 7.2 MB 23 kB/s
     dropped_iterations.............: 43591  138.859097/s
     http_req_blocked...............: avg=4.58s    min=0s       med=3.83s
     http_req_connecting............: avg=1.07s    min=103.16ms med=855.56ms
     http_req_duration..............: avg=17.18s   min=0s       med=9.86s
       { expected_response:true }...: avg=11.59s   min=348.14ms med=7.07s
     http_req_failed................: 13.83% ✓ 195         ✗ 1214
     http_req_receiving.............: avg=14.3s    min=0s       med=6.5s
     http_req_sending...............: avg=99.19µs  min=0s       med=88µs
     http_req_tls_handshaking.......: avg=3.5s     min=0s       med=2.75s
     http_req_waiting...............: avg=2.87s    min=0s       med=1.19s
     http_reqs......................: 1409   4.488368/s
     iteration_duration.............: avg=21.82s   min=1.2s     med=13.87s
     iterations.....................: 1409   4.488368/s
     vus............................: 4      min=4         max=100
     vus_max........................: 100    min=100       max=100


running (5m13.9s), 000/100 VUs, 1409 complete and 0 interrupted iterations
```

1409 requests

Testing

In 5 minutes (.js datadog script):

GraphQL query that doesn't return facets



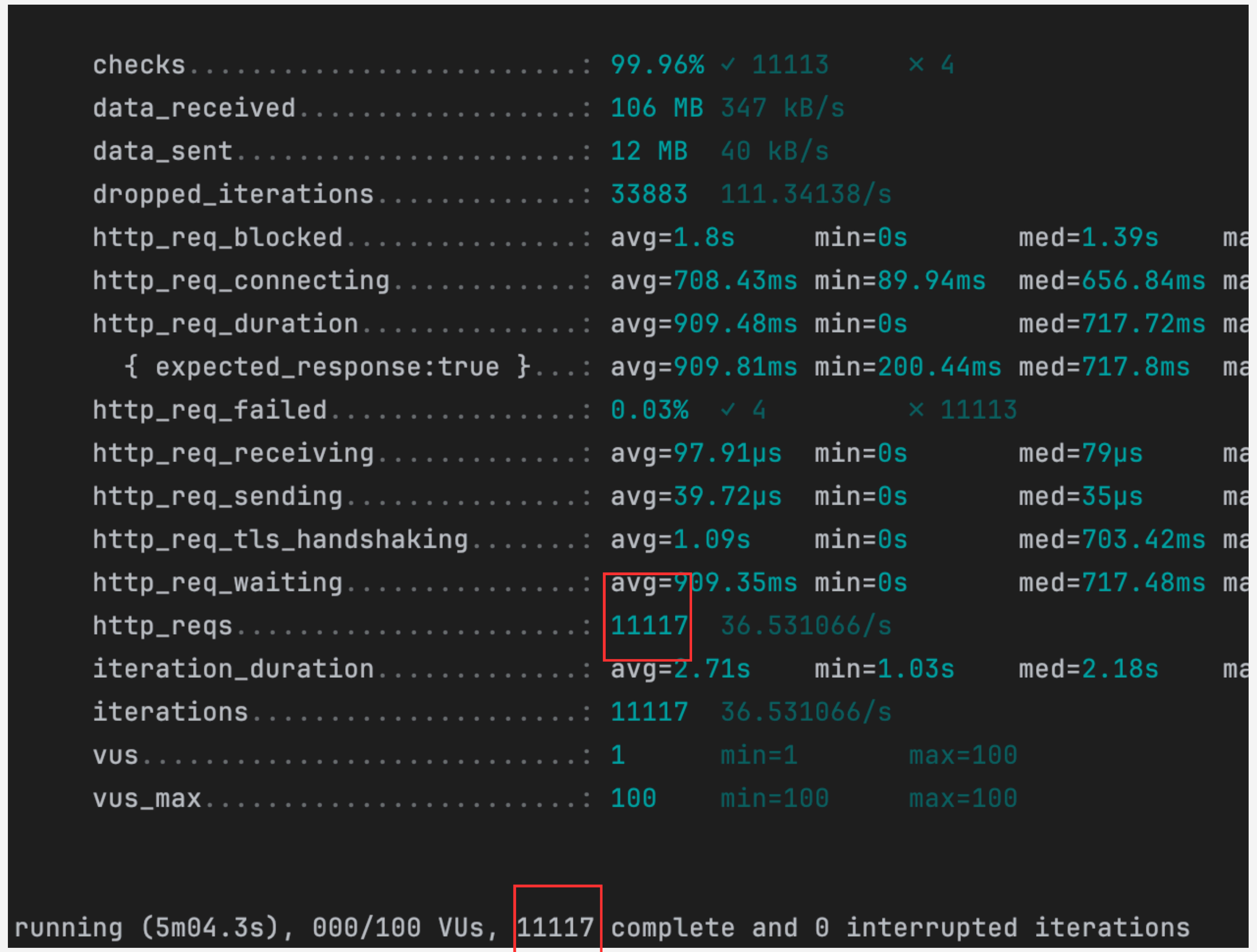
```
     checks.........................: 96.05% ✓ 2023       ✗ 83
     data_received..................: 208 MB 673 kB/s
     data_sent......................: 6.0 MB 20 kB/s
     dropped_iterations.............: 42894  138.651924/s
     http_req_blocked...............: avg=3.77s    min=0s       med=3.1s
     http_req_connecting............: avg=902.79ms min=90.97ms  med=553.19m
     http_req_duration..............: avg=10.65s   min=0s       med=6.06s
       { expected_response:true }...: avg=9.07s    min=807.33ms med=5.82s
     http_req_failed................: 3.94%  ✓ 83         ✗ 2023
     http_req_receiving.............: avg=8.76s    min=0s       med=4.74s
     http_req_sending...............: avg=49.01µs  min=0s       med=41µs
     http_req_tls_handshaking.......: avg=2.86s    min=0s       med=2.2s
     http_req_waiting...............: avg=1.89s    min=0s       med=909.94m
     http_reqs......................: 2106   6.807501/s
     iteration_duration.............: avg=14.46s   min=1.4s     med=9.78s
     iterations.....................: 2106   6.807501/s
     vus............................: 1      min=1        max=100
     vus_max........................: 100    min=100      max=100


running (5m09.4s), 000/100 VUs, 2106 complete and 0 interrupted iterations
```

2106 requests

Testing

In 5 minutes (.js datadog script):

GraphQL query that doesn't return facets and returns only the identification as results



```
     checks.........................: 99.96% ✓ 11113      × 4
     data_received..................: 106 MB 347 kB/s
     data_sent......................: 12 MB  40 kB/s
     dropped_iterations.............: 33883  111.34138/s
     http_req_blocked...............: avg=1.8s     min=0s       med=1.39s    ma
     http_req_connecting............: avg=708.43ms min=89.94ms  med=656.84ms ma
     http_req_duration..............: avg=909.48ms min=0s       med=717.72ms ma
       { expected_response:true }...: avg=909.81ms min=200.44ms med=717.8ms  ma
     http_req_failed................: 0.03%  ✓ 4          × 11113
     http_req_receiving.............: avg=97.91µs  min=0s       med=79µs     ma
     http_req_sending...............: avg=39.72µs  min=0s       med=35µs     ma
     http_req_tls_handshaking.......: avg=1.09s    min=0s       med=703.42ms ma
     http_req_waiting...............: avg=909.35ms min=0s       med=717.48ms ma
     http_reqs......................: 11117  36.531066/s
     iteration_duration.............: avg=2.71s    min=1.03s    med=2.18s    ma
     iterations.....................: 11117  36.531066/s
     vus............................: 1      min=1        max=100
     vus_max........................: 100    min=100      max=100


running (5m04.3s), 000/100 VUs, 11117 complete and 0 interrupted iterations
```

11117 requests