TU Munich, Department of Informatics
Chair III: Database Systems
Prof. Dr. Jana Giceva
Winter Term 2024/25: Cloud-Based Data Processing

Assignment 5 - Managing Shared State for Distributed Query Execution

2 Points
Handout date: 28.11.2024
Due date: 19.12.2024
Discussion: 09.01.2025

Michalis Georgoulakis, Tobias Götz ({geom,goetzt}@in.tum.de)

Introduction

This assignment includes two tasks. The first one, which is the easiest, involves turning your Assignment 3 solution (or the base solution we provide) into a cloud-based solution. This task aims to familiarise you with the services of a public cloud.

The second task involves implementing a **distributed query execution application with shared state**. We start again with Assignment 3 partitioned data on external storage and similar elasticity goals. However, we now process a query that needs to share state between workers.

As a query, we want to calculate how often each domain appeared (not only a specific domain as in the last assignment) and report the result for the top 25 domains. We now build multiple (partial) aggregates for each input partition, one for each domain, which we then need to merge. Contrary to what we did in the last assignment, merging on the coordinator does not scale well for this query. To scale the merge phase, we partition the aggregates and store them in shared state. After the initial aggregation and partitioning, we distribute the work of merging the partial aggregate partitions and send the merged results to the coordinator.

Feel free to reuse anything from the last assignment. You can find an example in the README.md file of the repository, as well as the expected result. We also provide you with a helper function to extract the domain from the URL, which is commented out in the coordinator file.

Submission guidelines

Deliverables

For this assignment, you are expected to:

- submit code that implements the assignment and produces the correct result
- write a brief report answering the questions of this assignment sheet (**report.pdf**)
- write your group members' names (**groupNames.txt**)

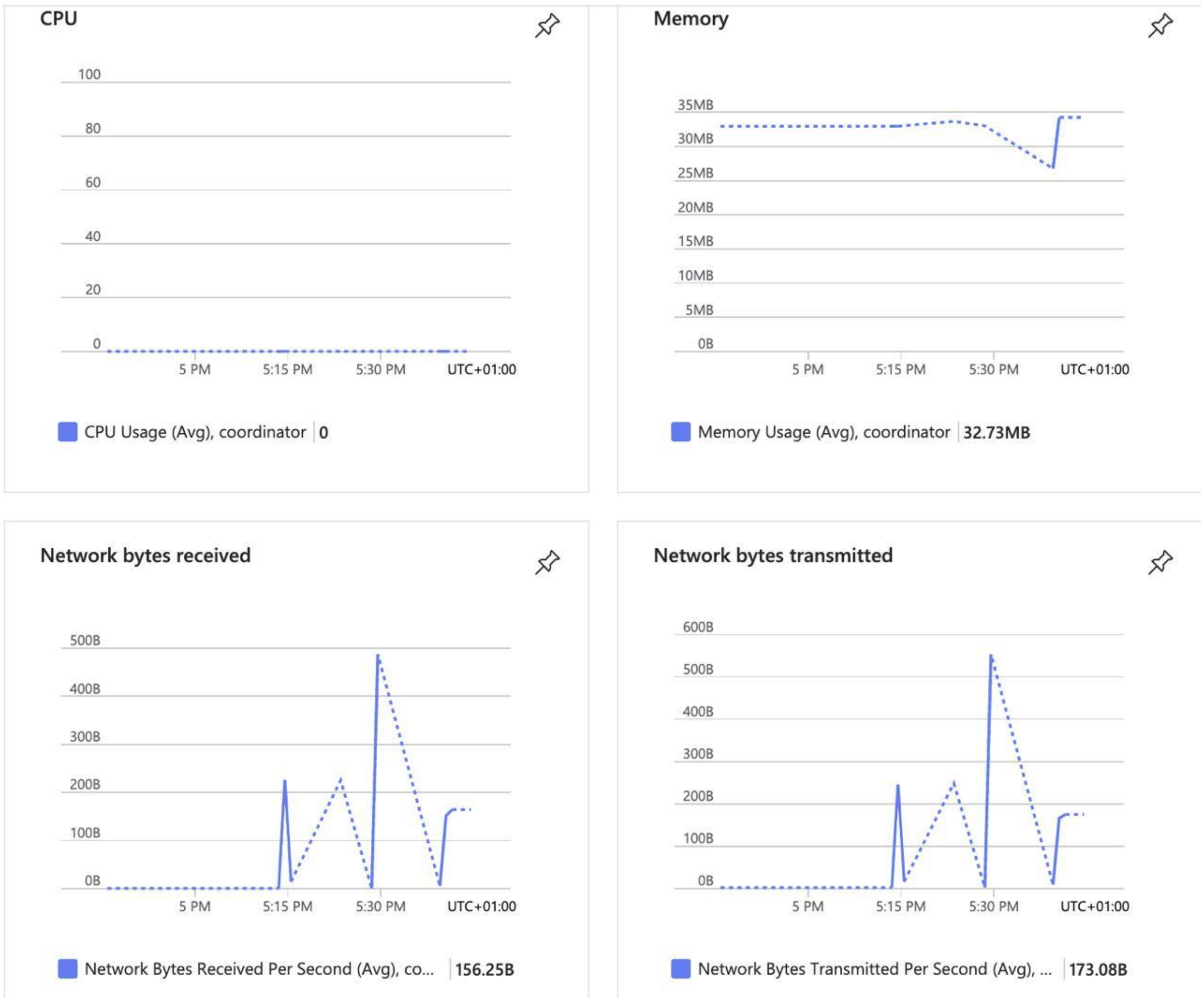
Section 1: Azure Deployment and Testing

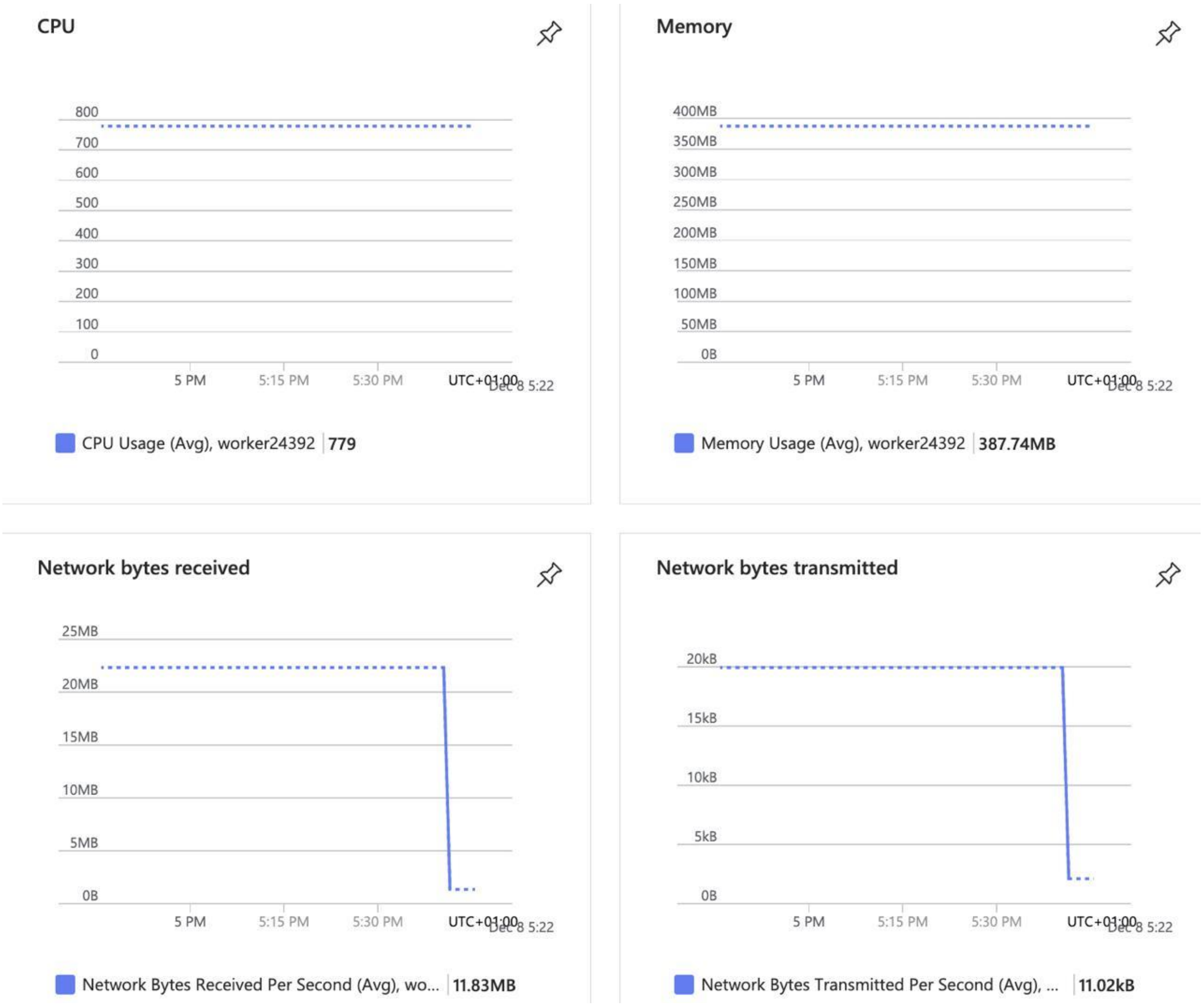
(i) After you have finished with the Azure tutorial, measure the time it takes for the Assignment 3 query to run on Azure. What do you notice?

It takes slightly less time on the cloud, likely due to higher network bandwidth. However, the difference is not significant. The time was measured by adding the `time` command at the beginning of the coordinator execution in the Docker file.

(ii) Observe the Azure monitoring panel metrics for your containers: explain what is the bottle-neck that increases query execution time (e.g. network-bound, CPU-bound). Include screenshots if needed.

The network-bound is the bottleneck for now, as we must wait for the coordinator to download data from the internet, probably using blob storage would be much faster.





(iii) Let's get faster: Pre-upload the data partitions and fileList inside Azure blob storage. Adapt your solution to read from there (you can use the **AzureBlobClient** helper functions). What is the speedup you observe? How is it explained?

Pre-uploading data partitions and the fileList to Azure Blob Storage reduces query execution time. Using Azure Blob Storage can reduce latency and network transfer delays because the data resides within the same cloud infrastructure, minimizing external fetches. This likely contributes to improved query execution time.

Section 2: Managing Shared State - Design Questions

Give a brief description of your solution for the programming task, answering the following questions.

Brief description:

Our solution implements a distributed query execution system that processes and aggregates data across multiple workers using a shared state for scalability and fault tolerance. The coordinator is responsible for distributing CSV file URLs to workers, managing their states, and handling fault tolerance by redistributing tasks if a worker fails. It orchestrates two main phases: downloading data and merging partial results. Worker responses are monitored using polling, and tasks are dynamically assigned.

The workers process assigned CSV files by extracting domains from URLs and counting their occurrences. Results are partitioned using a hash function and stored as subpartition files in a blob storage. Workers also merge subpartition files into sorted partitions and upload them back to the storage.

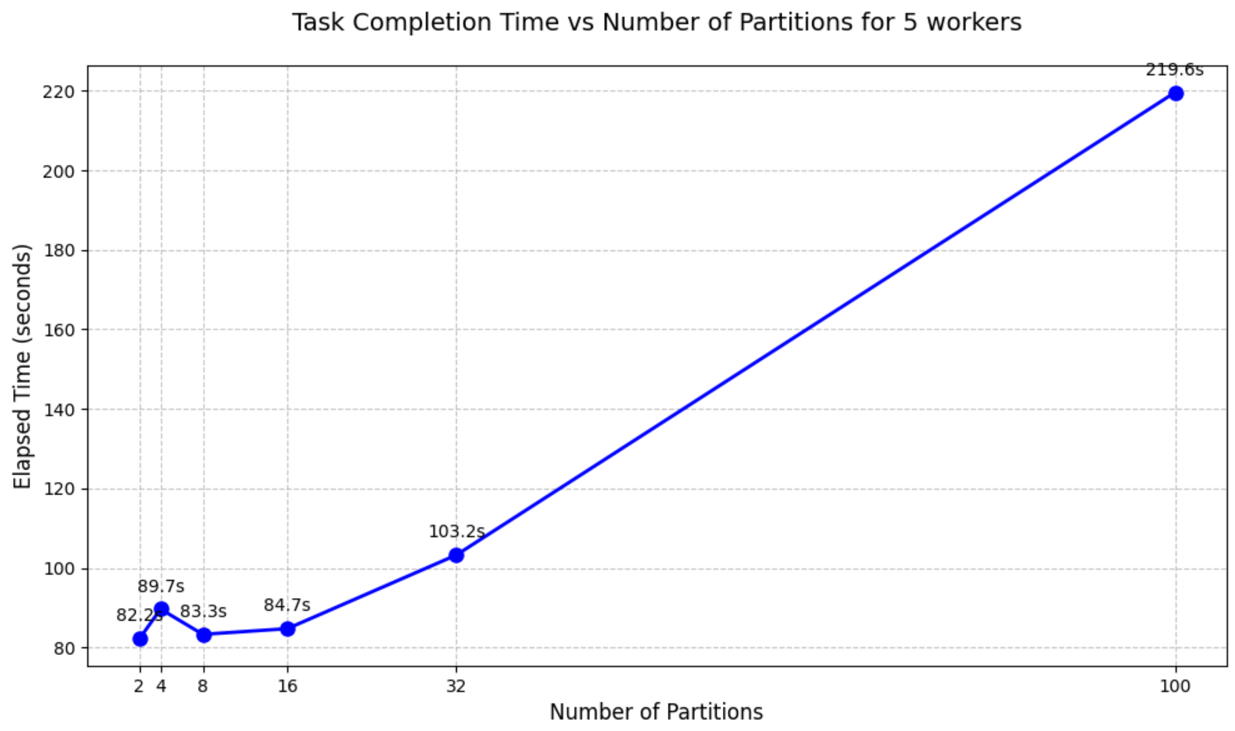
Intermediate results are stored in shared storage to facilitate collaboration between workers. The system ensures scalability by evenly distributing tasks among workers and supports fault tolerance by detecting failures and reassigning incomplete tasks to other workers.

Finally, the coordinator retrieves merged partition results from storage, combines them using a priority sort to identify the most frequent domains, and outputs the top 25 domains to a result file.

(i) Which partitioning method did you choose for the calculation of the partial aggregates?

Hash-based Partitioning. Our solution hashes domain names with hash(domain) % partition_count to divide data into buckets (subpartitions). This method balances the load among workers. In the next step, the subpartitions are merged and sorted. Finally, when merging the sorted partitions, we can assume that no two partitions contain the same url. This allows us to merge them in the final step and compute the top 25 in linear time.

(ii) What number of subpartitions did you choose, and why? If you choose to create 100 subpartitions for each partition, is it a good choice? Is there a scalability limit?



We chose 2 subpartitions because it achieves the lowest task completion time of 82.4 seconds, as shown in the graph. The more partitions we have, the more shuffling overhead occurs (i.e., read/write operations to blob storage), and the more stress is placed on the coordinator. With an increasing number of partitions, the coordinator becomes a bottleneck during the merging phase.

Creating 100 subpartitions is not a good choice. As seen in the graph, task completion time rises significantly to 219.6 seconds with 100 partitions. This indicates that the overhead of managing and merging such many subpartitions outweighs any benefit of finer-grained work distribution. The coordinator becomes overwhelmed with the additional merging load, leading to inefficiencies and scalability issues.

(iii) How does the coordinator know which partitions to send to the workers for the merging phase?

The coordinator assigns subpartitions with a certain partition ID (hash % partition_count) any free worker and communicated the partition ID. Using the ID, the worker can find the subpartitions in the local store by matching the filename.

(iv) How do workers differentiate from the two tasks they have to perform (partial aggregation, merging subpartitions)?

The Command enum defines specific values for the tasks. The workers differentiate tasks based on enum values, which the coordinator sends as part of its communication protocol.

0: Initialization (INIT).

1: Partial Aggregation / Partitioning (DOWNLOAD).

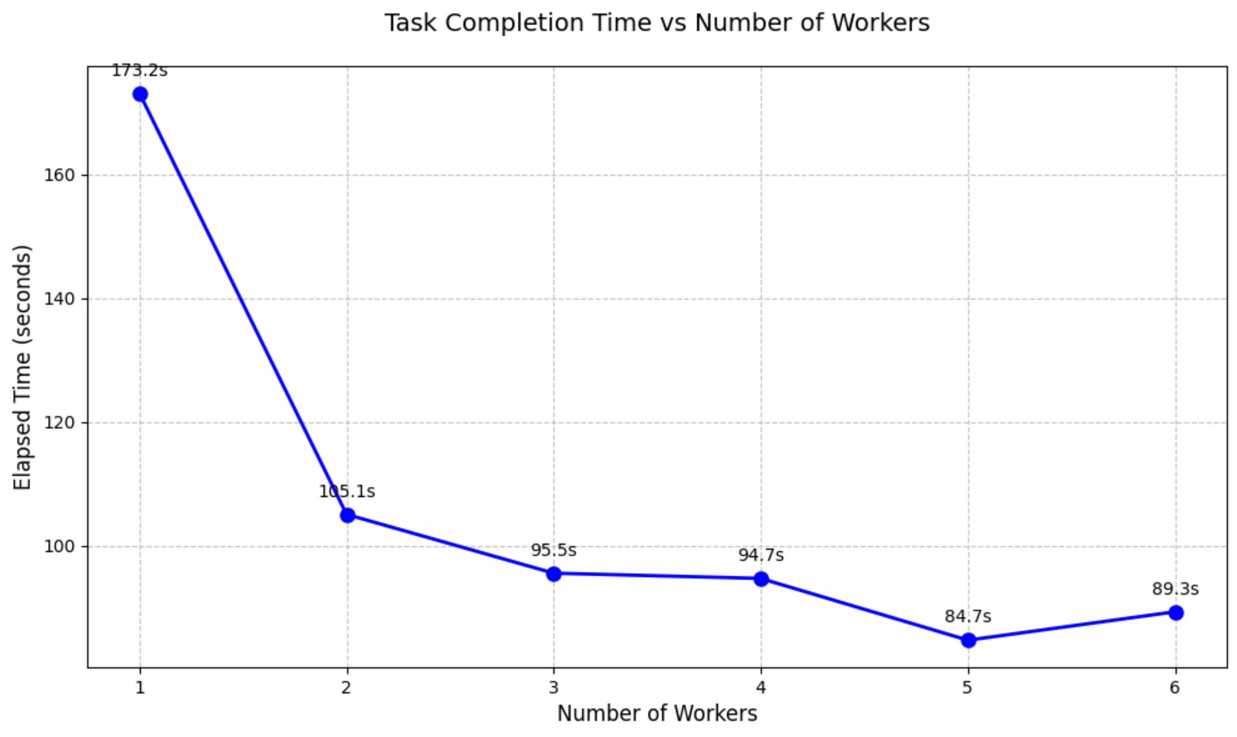
2: Merging Subpartitions (MERGE).

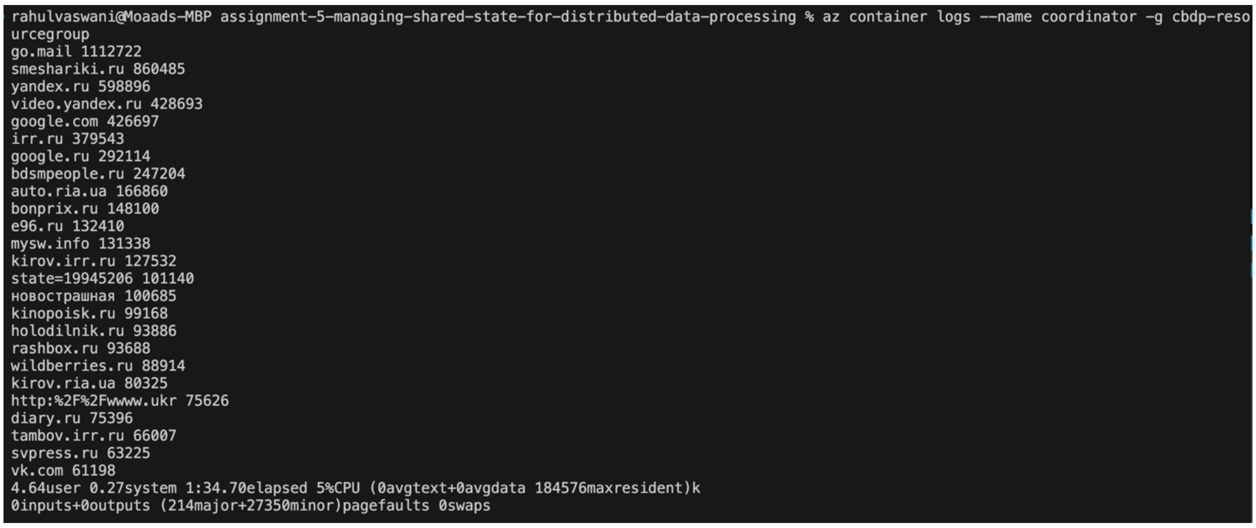
Workers return "Success" if the given task has been completed successfully.

To ensure fault tolerance and prevent inconsistencies / race conditions, all DOWNLOAD Tasks must be completed before the MERGE tasks.

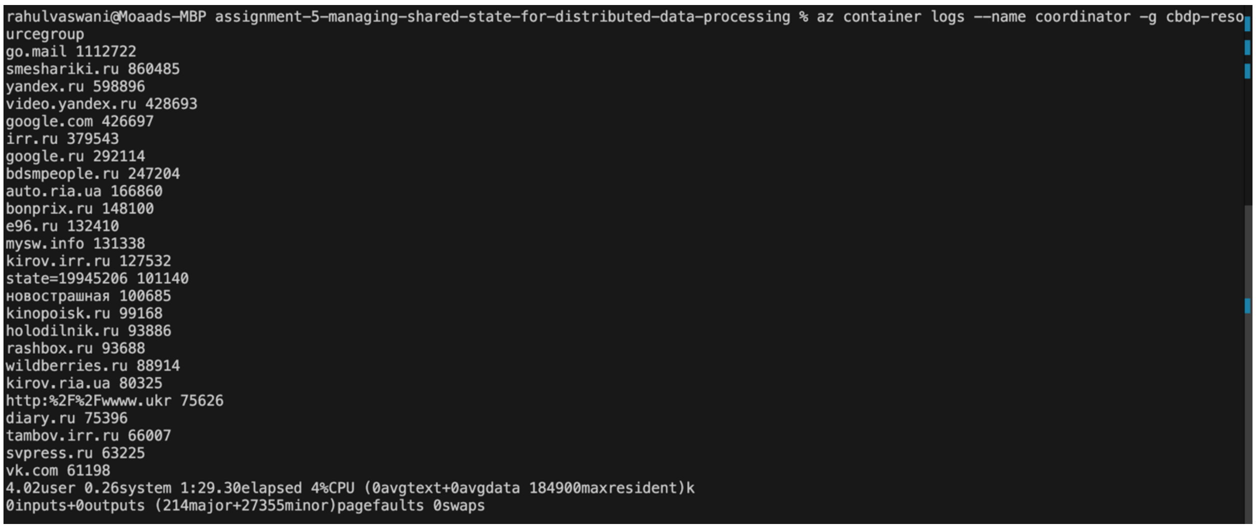
(v) What was the query execution time in Azure? Include screenshots, and logs if needed.

The execution time decreases as the number of workers increases, but the improvement slows down after 4 workers. The optimal number of workers in this setup appears to be 5 for achieving the lowest execution time.

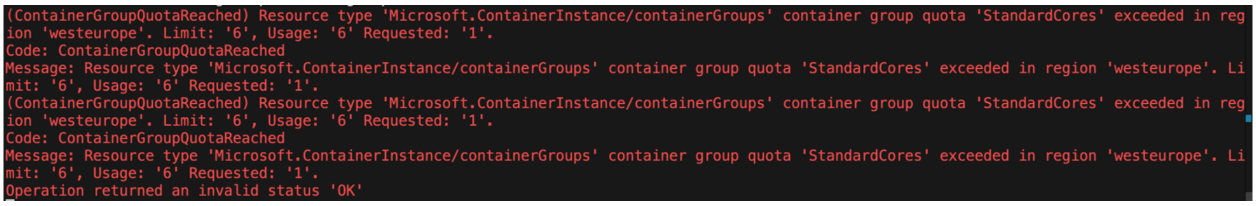




```
rahulvaswani@Moaads-MBP assignment-5-managing-shared-state-for-distributed-data-processing % az container logs --name coordinator -g cbdp-reso
urcegroup
go.mail 1112722
smeshariki.ru 860485
yandex.ru 598896
video.yandex.ru 428693
google.com 426697
irr.ru 379543
google.ru 292114
bdsmpeople.ru 247204
auto.ria.ua 166860
bonprix.ru 148100
e96.ru 132410
mysw.info 131338
kirov.irr.ru 127532
state=19945206 101140
новострашная 100685
kinopoisk.ru 99168
holodilnik.ru 93886
rashbox.ru 93688
wildberries.ru 88914
kirov.ria.ua 80325
http:%2F%2Fwwww.ukr 75626
diary.ru 75396
tambov.irr.ru 66007
svpress.ru 63225
vk.com 61198
4.64user 0.27system 1:34.70elapsed 5%CPU (0avgtext+0avgdata 184576maxresident)k
0inputs+0outputs (214major+27350minor)pagefaults 0swaps
```



```
rahulvaswani@Moaads-MBP assignment-5-managing-shared-state-for-distributed-data-processing % az container logs --name coordinator -g cbdp-reso
urcegroup
go.mail 1112722
smeshariki.ru 860485
yandex.ru 598896
video.yandex.ru 428693
google.com 426697
irr.ru 379543
google.ru 292114
bdsmpeople.ru 247204
auto.ria.ua 166860
bonprix.ru 148100
e96.ru 132410
mysw.info 131338
kirov.irr.ru 127532
state=19945206 101140
новострашная 100685
kinopoisk.ru 99168
holodilnik.ru 93886
rashbox.ru 93688
wildberries.ru 88914
kirov.ria.ua 80325
http:%2F%2Fwwww.ukr 75626
diary.ru 75396
tambov.irr.ru 66007
svpress.ru 63225
vk.com 61198
4.02user 0.26system 1:29.30elapsed 4%CPU (0avgtext+0avgdata 184900maxresident)k
0inputs+0outputs (214major+27355minor)pagefaults 0swaps
```



```
(ContainerGroupQuotaReached) Resource type 'Microsoft.ContainerInstance/containerGroups' container group quota 'StandardCores' exceeded in reg
ion 'westeurope'. Limit: '6', Usage: '6' Requested: '1'.
Code: ContainerGroupQuotaReached
Message: Resource type 'Microsoft.ContainerInstance/containerGroups' container group quota 'StandardCores' exceeded in region 'westeurope'. Li
mit: '6', Usage: '6' Requested: '1'.
(ContainerGroupQuotaReached) Resource type 'Microsoft.ContainerInstance/containerGroups' container group quota 'StandardCores' exceeded in reg
ion 'westeurope'. Limit: '6', Usage: '6' Requested: '1'.
Code: ContainerGroupQuotaReached
Message: Resource type 'Microsoft.ContainerInstance/containerGroups' container group quota 'StandardCores' exceeded in region 'westeurope'. Li
mit: '6', Usage: '6' Requested: '1'.
Operation returned an invalid status 'OK'
```

Section 3: Notes

If we need some prerequisites to run your solution, please include them here. Also, include a brief description of your unit tests.