Cloud-Based Data Processing
Project - Sorting 1 TB in the cloud

Handout: **23th December 2020**
Due: **08th February 2021 by 6pm**
Discussions: **13th, 20th 27th January and 03rd and 10th February 2021 in the tutorials**

1 Introduction

The last assighment of this course is the project to sort 1 TB of data in the cloud using an external sorting algorithm [1] where the input is read and the output is written from and to persistent storage.

This exercise is designed to be open and allows you to explore what you find most interesting personally, e.g. using existing tools like Spark or Timely Dataflow, try to exploit new hardware like persistent memory or 100GB Ethernet or try to implement a server-less approach based on AWS lambda functions. Your fantasy is only constrained by four simple rules:

1. Sort a fixed number of 1 TB of randomly permuted 100-byte records with 10-byte keys
2. The sort must start with the input on S3 and finish with output on a S3.
3. All operations must be performed on a commercially available public cloud.
4. You must use some form of distributed, external sorting and are not allowed to work on a single machine.

These are the official rules of the CloudSort benchmark, adjusted to a dataset size of 1 TB [4] and to run on S3. You should report two metrics: end-to-end run-time (disk-to-disk) and costs calculated according to the standard AWS pricing for on-demand instances in the region `US East`. Prices should be calculated as if you could book the service **per second**. Reporting additional metrics is allowed and welcome, e.g. if you want to run on Spot instances you can do this and report your savings.

We are aware that this task is very open. This is done intentionally because many tasks in your life after the master have similar characteristics, in particular, for system-design exercises. You do not receive any other grades than pass or fail and any solution that can sort 1 TB of data passes. This exercise is designed to give you a lot of freedom and you should aim to show that you can design a system from scratch and argue your decisions, rather than building the best performing system possible. We strongly believe the lecture and former exercises gave you the background and experience needed!

Below, we give a short recap of the history of external sorting benchmarks, describe the tools used for data generation and output verification, the general schedule for this exercise and some sources of inspiration for a great project.

2 A short history of external sorting

External sorting has a long history as a very challenging benchmark for computer systems. Proposed in 1985 by the Turing Award winner Jim Gray, it has been considered a great tool to measure advances in software and hardware. Back in 1985, it was a huge challenge to sort 100 MB of data and the winning benchmark entry took around 980 seconds. By 2001, the same task took only 0.44 seconds and the benchmark changed to sort 1 TB of data. Less than 10 years later, in 2009, the task became sorting 100 TB which took 72 minutes on 2100 machines. Only 5 years after the same task was performed in 23 minutes on 207 machines in the AWS cloud using Spark.

The GraySort Benchmark is the most renown external sort benchmark and its only metric is time taken to sort 100 TB of data. However, a well designed external sort is nearly linearly scalable, to be the best performer in this benchmark is costly. As an answer to this problem, the CloudSort benchmark features the costs of sorting on publicly available cloud resources as the main metric for success [4].

This change allowed a team from the University of California to win the race with costs of $451 [3]. Two years later, a Spark-based system, called NADSort, completed the same tasks while only spending $144 [5, 6]. In 2019, a system build with server-less resources is as fast and only marginally more expensive than the Spark system using 395 compute nodes [2].

3 Data generation

We use the `gensort` data generator from the CloudSort benchmark. You can download binaries for Linux and Windows from its website. Use the following command line to generate 1 TB of data:

```
gensort 10000000000 <output-file>

// to generate partitions of the input on different machines use
gensort -b0 5000000000 part0
gensort -b5000000000 5000000000 part01
```

Listing 1: Generating input data

We also provide 1 TB of data hosted in a S3 bucket. The link will be provided on Mattermost.

4 Output validation

Check the output of your program with `valsort`. You can download the program from the same website as the dataset generation tool. The necessary commands are shown below:

```
valsort <path/to/sorted-output>

// to validate partitioned output
valsort -o out0.sum out0.dat
valsort -o out1.sum out2.daft
cat out0.sum out1.cum > all.sum
valsort -s all.sum
```

Listing 2: Verifying your output

5 Schedule

As requested by students, we publish the exercise before the Christmas holidays. You do not have to work over the Christmas holidays but you can if you prefer to do so.

You will give two 5-minute-presentations. The first is on the 20th of January where you should present your overall approach. The second is on the 10th of February where you conclude the outcome of your project.

6 Optional background reading

The following documents give a more detailed specification of the metrics to measure and rules to follow. This is **optional reading**. Read it if you want further specification of the task. Otherwise, you can ask me for any kind of clarification needed.

- the official CloudSort document
- the CloudSort website
- the FAQ page of cloud sort
- the `gensort` and `valsort` website

The list below links some interesting papers describing external sorting and sorting in the cloud. They are provided for inspiration and reading them is not necessary to pass this exercise nor for the background! We recommend not reading any if you have a direction for your project already and reading the most interesting otherwise, designing your project with ideas from one paper. Combining ideas from multiple papers is welcome but not necessary.

- NADSort, the winning Spark based implementation from 2016 [5]
- Triton Sort, the second last winning entry using EC2 machines [3]
- server-less external sorting from 2019 [2]

References

[1] Wikipedia Community. Wikipedia: external sort, 20xx.

[2] Pu et al. Shuffling, fast and slow: Scalable analytics on serverless infrastructure. *NSDI*, 2019.

[3] Rasmussen et al. Tritonsort: A balanced large-scale sorting system. *NSDI*, 2011.

[4] Shah et al. Cloudsort: A TCO sort benchmark, 2014.

[5] Wang et al. NADsort. *f*, 2016.

[6] Reynold Xin. $1.44 per terabyte: setting a new world record with apache spark, 2016.