CS-449 Project Milestone 2: Optimizing, Scaling, and Economics

Name: Adda Julien
Sciper: 282449
Email: julien.adda@epfl.ch
Name: Epple Pascal
Sciper: 287340
Email: pascal.epple@epfl.ch

February 21, 2023

Our code was executed with the command "sbt "runMain ..."" and after running : export JAVA_OPTS="-Xmx8G"

1 Optimizing with Breeze, a Linear Algebra Library

BR.1 These are the values we computed using $k = 10$:

Table 1:

$s_{1,1}$	0
$s_{1,864}$	0.2423
$s_{1,886}$	0
$p_{1,1}$	4.3190
$p_{327,2}$	2.6994
MAE	0.8287

BR.2 For k = 300 and with 5 measurements, our previous mean time on the data set data/ml-100k/u2.test is 27208.6293ms and the std is 1427.7720ms. With the Breeze library implementation our mean time on the data set data/ml-100k/u2.test is 4519.96ms and the std is 142.61ms.
This gives us a ratio of

$$\frac{\text{old mean time (ms)}}{\text{new mean time (ms)}} = \frac{27208.629}{4519.96} = 5.64$$

The time was calculated on the same device as milestone 1. Technical specifications:

- Model : MacBook Pro (2019)
- CPU Speed : 2.4 GHz
- RAM : 16 Go
- OS : macOS Monterey
- Scala language version : 2.13.8
- JVM version : 1.8.0_272

Our implementation works with data/ml-1m/rb.train and data/ml-1m/rb.test.

2 Parallel k-NN Computations with Replicated Ratings

EK.1 These are the values we computed using $k = 10$. They are the same as Table 1:

Table 2:

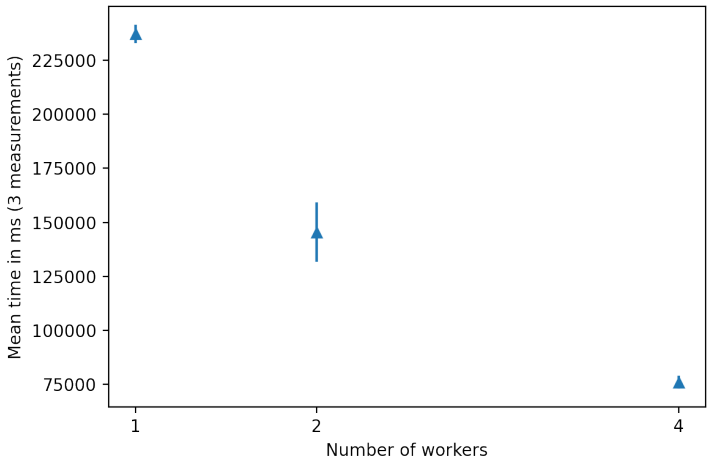
$s_{1,1}$	0
$s_{1,864}$	0.2423
$s_{1,886}$	0
$p_{1,1}$	4.3190
$p_{327,2}$	2.6994
MAE	0.8287

EK.2 The following values were calculated on the cluster. They are the results of 3 measurements on the ml-1m/rb.train for training and ml-1m/rb.test for test, with k = 300 and workers 1, 2 and 4 :

Table 3:

Number of Workers	Average time (ms)	Standard Deviation (ms)
4	75913.10	2942.65
2	145394.21	13810.86
1	237287.22	4260.57

We do observe a speedup. The standard deviation is very low. By plotting the mean time with respect to the number of workers we can observe a quadratic relationship:



3 Distributed Approximate k-NN

AK.1 With 10 partitions and 2 replications, we computed the following values using 10-approximate-NN:

Table 4:

User pair (u,v)	Similarity between user u and user v
(1,1)	0.0
(1,864)	0.0
(1,344)	0.23659364388510976
(1,16)	0.0
(1,334)	0.19282239907090362
(1,2)	0.0

AK.2 Depending on the number of user replications, the MAE varies as follows:

Table 5:

Number of Replications	MAE (using $k = 300$ and 10 partitions)
1	0.8090674652315339
2	0.7587841249643481
3	0.7457705527637317
4	0.7410921634897718
6	0.7391562504199756
8	0.7391562504199756

If we compare this to the baseline predictor of Milestone 1 which yielded

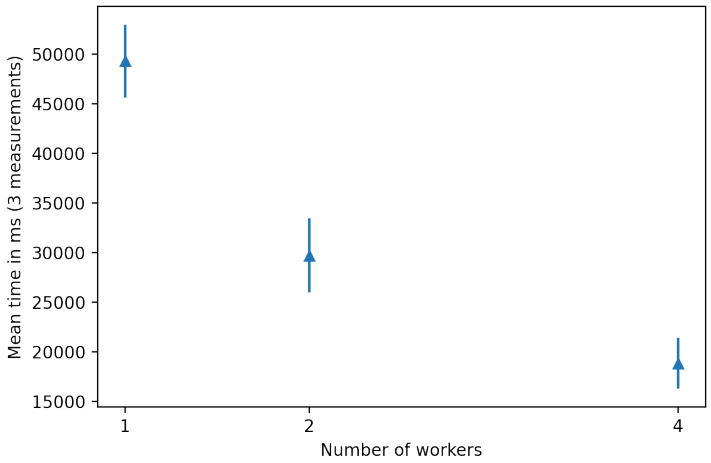
and MAE of 0.7604, we can see that 2 replications suffice for the approximate MAE to perform better than the baseline prediction method.
Given a train set of n users, the exact k-NN method will compute n^2 similarities and only store the k largest. Let us now analyze the number of similarity computations for a partition of the users into 10 different groups, with 2 replicas. For simplicity, we can consider that each worker receives a train set of $\frac{2n}{10}$ users, and will therefore compute $(\frac{2n}{10})^2$ similarities. As there are 10 workers, the total number of similarities computed is $\frac{4n^2}{10}$. The ratio of similarity computations for both methods is therefore $\frac{4}{10}$, which clearly shows how the approximate k-NN method reduces similarity computations. However, the reduction of similarity computations is highly dependent on the number of replicas and partitions you choose. Indeed, if we denote by p the number of partitions, by k the number of replicas and by n the total number of users present in the system, then the number of similarity computations needed for the approximate k-NN method is $\frac{k^2}{p}n^2$, which of of course can be much higher than the usual n^2 similarity computations needed for the exact k-NN method approach, depending on the value chosen for k and p. However, one definitely gains in partition tolerance and in the number of users one can store across clusters.

AK.3 The following table shows how computation time evolves with respect to the number of workers used, using 300-NN. The test was done on the cluster with the 1m dataset, with 8 partitions and a replication factor of 1.

Table 6:

Number of Workers	Average Time (ms)	Standard Deviation (ms)
4	18830.38	2559.66
2	29705.07	3733.07
1	49315.18	3692.27

Comparing these average times to the ones computed using the exact k-NN method (3), we do indeed observe a significant speed-up in computation time (around 75%). This is not surprising: taking the formula of part **AK.2**, we compute $\frac{7}{8}$ less similarities than for the exact k-NN method. The computation times do not directly scale to this factor, this is surely due to more data being moved around in the approximate k-NN method. With the MAEs of the approximate and exact method not being too different (approximately 0.01 difference), we can clearly see that the method is not only fast, but also very reliable.



4 Economics

Implement the computations for the different answers in the Economics.scala file. You don't need to provide unit tests for this question, nor written answers for these questions in your report.