Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. This paper was not previously presented to another examination board and has not been published.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted.

Meschede, 27th September 2023.

Casimir Giesler
MatNr: 123454678
Email: curie.marie@fh-swf.de
Corresponding Author

This Declaration must be signed by all listed Authors to be valid.

Math stuff for pyspark

1 Performance measurement

To evaluate the scalability of the different implementations, both in rows and columns as well as in cluster size, the DUS Airports Hadoop cluster is used. The Spark native implementation `pyspark.ml.regression.LinearRegression` is used as baseline. Each permutation of number of rows, number of columns, number of nodes and applied algorithm is measured at least five times.

1.1 Data scalability

The test shows, that the custom implementations perform significantly worse than the PySpark implementation, except the map-reduce LU implementation which is the only method which performs better than the PySpark implementation. QR consistently performs worst, as visualized in figures 1 and 2. Besides the worse performance, the QR and SVD implementations still show a linear algorithmic complexity and are somewhat scaleable.

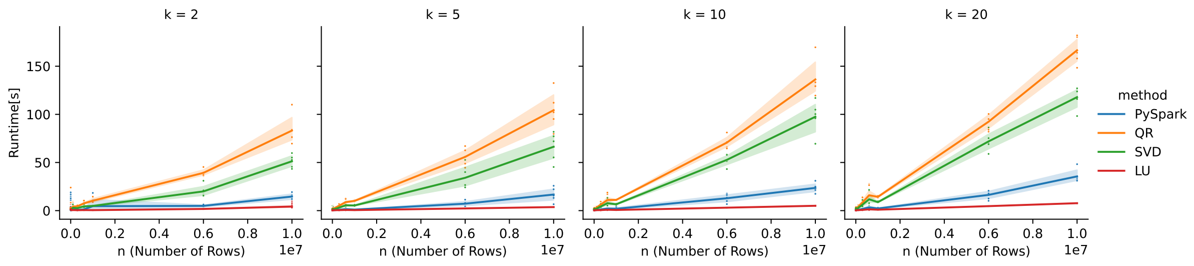

Figure 1: *Runtime comparision for the different implementations with linear scale*

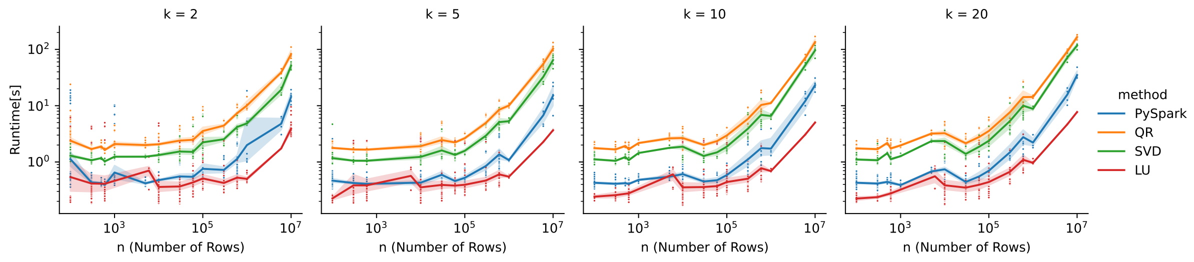

Figure 2: *Runtime comparision for the different implementations with logarithmic scale*

1.2 Node scalability

To evaluate the horizontal compute scaling capabilities, we analized the runtime for the same amount of data with different cluster sizes. Expectedly, the results show, that increasing the cluster size reduced the runtime, with increasing effectiveness as the amount of data increases. Figure 3 shows, that performing the calculations on a cluster for `n=100` increases the runtime, which is obvious since a lot of overhead work has to be done in order to distribute the tiny workload, which itself takes a lot longer than the actual computation. The map-reduce LU implementation obviously fails on more than eight nodes, since the rows are evenly distributed in the cluster and each node does not have enough datapoints to compute the coefficients. For $n = 10^4$ eight node seems to be an efficient sizing, for $n = 3 \cdot 10^5$ 32 nodes bring a considerable performance improvement, while the improvement from 32 to 64 nodes for $n = 6 \cdot 10^5$ does not improve computation time significantly.

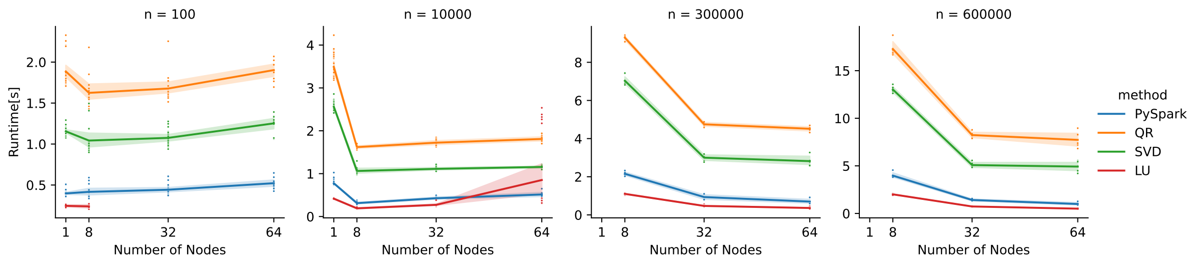

Figure 3: *Comparision of runtimes for the same amounts of data on different cluster sizes, with k=10*

1.3 Conclusion

The results show, that the implemented methods are scaleable, with ~$O(N)$ for SVD and QR and ~$O(log(N))$ for the map-reduce LU implementation (Fig. 1). Each implementation scales well with increasing cluster sizes.