Numerical Linear Algebra

Fall 2016

Team #24

Project proposal

### Project name

Implementation of NLA algorithm in distributed system

### Team

1. Anton Marin
2. Alena Shilova
3. Andrei Kvasov
4. Nadezhda Sviridenko

Tasks will be divided according to student’s prior experience, but also, it will be useful to learn something new to every of us. Also, we will have mutual responsibility. It means, that if someone fails to complete the task prior deadline, other members either help him or the whole team suffers from penalties.

### 

### Background

Developing new NLA algorithms and improving existing ones plays key role in improving the quality of mankind’s life. Optimization methods, data mining, signal processing, virtual reality - a small subset of NLA applications.

Due to the fact, that the amount of processing data is incredibly large, usually it can’t be stored in the memory of single computer. As data, stored on very large hard drives is very slow to process using a single CPU, and hard drives of huge capacity (>2 TB) are expensive, there is a need to distribute data among several computers and process it in parallel. So, a lot of distributed frameworks appear, offering a researcher to choose between different parallel data processing paradigms, which is a quite complicated task itself, because it requires not only knowledge of a model, but also an ability to implement particular algorithms in terms of this model.

Among different parallel data processing models, we can highlight a quite popular one, MapReduce [1]. Based on MapReduce, several distributes frameworks were created. While speaking about distributed frameworks, we mean distributed file system, which allows to store a consistent big amount of data, and a way to conduct distributed computations on this data. Project called Hadoop, developed by Apache Software Foundation, one of the most popular and convenient distributed frameworks. It introduces a fault-tolerant distributed file system (HDFS), supports MapReduce data processing paradigm and offers a quick way to run your own computational cluster (YARN). But it has a very meaningful drawback. A lot of iterative algorithms (especially NLA algorithms) require storage of results of intermediate steps during the work of algorithm. In Hadoop all computed data is stored on hard drives, which is slow itself, but also need to be replicated (fault tolerance requirement) and serialized (data consistency). If this is done after each step of algorithm, which has millions of iterations, computations degradate.

Apache Spark project was created to fight this problem. Based on HDFS, it introduces a new abstraction of data storage, called RDD [2] and a new way to conduct computations on RDDs (transformations). In addition to other advantages, RDDs are stored in memory, which is much faster, than storing on disks.

Among different NLA algorithms we decided to take LU decomposition and try to implement it on Spark. LU decomposition is used for solving systems of linear equations, inverting a matrix and computing the determinant. Also, LU decomposition can be parallelized well. Some researches [3] have implemented LU algorithm in Hadoop, so it will be challenging to implement it on Spark and compare performance.

### Problem formulation

A lot of algorithms such as finding an inverse matrix and calculation of a matrix determinant, solving linear system are based on computing LU factorization. It is a representation of a strictly regular matrix as a product of a lower triangular matrix and upper triangular matrix: *A =LU.*

This factorization can be found by means of Gaussian elimination which in general case can take *O(n3).* As we are going to work with a distributed file system, there is no usage of it. Our task is to implement Block LU decomposition adapted for the structure of Spark system.

### Data

### As we are going to compare our results with the results obtained by the previous researchers, we will use the same datasets, which were used in their works.

### Related work

Scalable Scientific Computing Algorithms Using MapReduce [3]

In this article a matrix inversion algorithm based on MapReduce paradigm and implemented on Hadoop is considered. It is shown that recursive block LU decomposition allows to effectively compute in parallel both the lower triangular (L) and upper triangular (U) matrices using MapReduce. The inverses of L, U and initial matrices are computed using MapReduce. Experiments provided in that paper show that this technique has good scalability, and it is simpler and more fault tolerant than MPI implementations such as ScaLAPACK

A parallel block LU decomposition method for distributed finite element matrices [4]

In this abstract authors implemented concept of the parallel LU decomposition for matrices resulting from finite element problems, which is based on a nested dissection approach. This approach is applied for different problems in PDE on HC3 Cluster machine.

### Scope

During the upcoming work we are going to:

1. Get acquainted with MapReduce paradigm, Spark framework
2. Implement parallel version of LU decomposition algorithm in Spark
3. If possible, deploy cluster or emulate it using virtual machines on a single multi-core PC
4. Evaluate performance, compare it with performance in related works
5. It everything goes well, we will develop parallel version of LU modification, e. g. sparse LU

### Evaluation

We are going to compare performance, like scalability, total computational time on the same data in different frameworks (using results of other researches). Also, scalability will be measured in Spark by increasing the number of computational nodes in the cluster. If it were possible to deploy our algorithm on a real cluster, we would try to evaluate performance using different settings, e. g. adding constraints on network latency and bandwidth, load balancing, different data partitioning strategies.

### References

1. MapReduce: Simplified Data Processing on Large Clusters / J. Dean and S. Ghemawat // OSDI 2004
2. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing / M. Zaharia, M. Chowdhury, T. Das et al. // OSDI. — 2011.
3. Scalable Scientific Computing Algorithms Using MapReduce / Jingen Xiang

# A parallel block LU decomposition method for distributed finite element matrices / D. Maurer, C. Wieners // Journal Parallel Computing, Volume 37 Issue 12, December, 2011, Pages 742-758