Distributed Processing of Elevation Data by Means of Apache Hadoop in a Small Cluster

Jitka Komarkova, Jakub Spidlen, Devanjan Bhattacharya and Oldrich Horak
Faculty of Economics and Administration, University of Pardubice, Studentska 95, 532 10 Pardubice, Czech Republic

Keywords: Distributed Processing, Apache Hadoop, Elevation Data, Small Cluster.

Abstract: Geoinformation technologies require fast processing of high and quickly increasing volumes of all types of spatial data. Parallel computational approach and distributed systems represent technologies which are able to provide required services, with reasonable costs. MapReduce is one example of such approach. It has been successfully implemented in large clusters in several instances. The applications include spatial and imagery data processing. The contribution deals with its implementation and operational performance using only a very small cluster (consisting of a few commodity personal computers) to process large-volume spatial data. Open-source implementation of MapReduce, named, Apache Hadoop, is used. The contribution is focused on a low-price solution and it deals with speed of processing and distribution of processed files. Authors run several experiments to evaluate the benefit of distributed data processing in a small-sized cluster and to find possible limitations. Size of processed files and number of processed values is used as the most important criteria for performance evaluation. Point elevation data were used during the experiments.

1 INTRODUCTION

Increasing volumes of stored and processed data is today one of the important issues which must be addressed by many types of information systems. This issue is particularly very important for geographic information systems and geoinformation technologies in general, as far as the volumes of spatial data are high and they are quickly increasing. Elevation data is an example of large volume spatial data. Parallel computational approach represents a suitable way how to process large volumes of spatial data at a reasonable time and shows how to scale the processing.

Importance of parallelization was recognized by Google several years ago. Google proposed a suitable architecture able to handle high workload and to improve web search applications performance (Barroso et al., 2003). The proposed architecture was oriented on throughput, software reliability and reduction in costs. The idea was to use many commodity-class personal computers (PC) together with fault-tolerant software. At that time, Google involved 15 000 commodity PCs. According to the authors, this solution was more cost-effective than utilization of a smaller number of high-end servers. The solution was named MapReduce and it was focused on general web search tasks connected to web search like word frequency counting; on processing of large volume data (e.g. terabytes) add on utilization of large-scale distributed systems based on commodity PCs (Barroso, et al., 2003; Dean and Ghemawat, 2004).

After that, just a few ways of utilization of MapReduce application for processing of spatial data were proposed. Cary et al. (2009) proposed a way of utilization of MapReduce to create R-Trees and compute quality of aerial imagery. They used Hadoop framework but it was run on Google & IBM cluster which contained around 480 computers. Chu, S.-T., Yeh, C.-C., Huang (2009) proposed creation of trajectory index scheme based on Hadoop and HBase as a part of StreetImage 2.0 service available on a web site to all users. The main goal was to keep response time of searching for trail logs (trajectories) reasonable. An application model suitable for GIS and based on cloud computing model was proposed by Zhou, Wang, and Cui (2012). This model includes Hadoop but no performance evaluation is provided by the authors. Cardosa et al. (2012) focused on optimization of MapReduce running in cloud environment to achieve high utilization of machines and decrease energy consumption by

DOI: 10.5220/0004591303400344
In *Proceedings of the 8th International Joint Conference on Software Technologies* (ICSOFT-EA-2013), pages 340-344
ISBN: 978-989-8565-68-6

means of dynamically changing size of MapReduce cluster. Zhu et al. (2009) focused on performance of MapReduce in the case supercomputing applications are run in its environment. They used 17 commodity PCs. They identified a significant problem – frequent data communication led to an overhead on network.

2 IDENTIFIED ISSUES AND GOALS

Authors use point elevation data, both regular grids and LIDAR clouds, to create Digital Surface Models (DSM). During the calculations they have experienced significant problems with computational performance of available hardware. To improve the computational performance, they decided to use distributed data processing powered by several commodity PCs and run by an open-source solution – Apache Hadoop which implements MapReduce approach.

MapReduce programming model is proposed to allow parallelized computing in a distributed system. The solution allows automatic parallelization and distribution of a computational task among available nodes. It is implementable on large clusters of commodity PCs, it enables scalability of the system but it is able to provide high level of fault tolerance at the same time. The idea of the programming model is to split calculations into two stages: map and reduce (Dean and Ghemawat, 2004). Apache Hadoop is an open-source software which includes MapReduce functionality together with Hadoop Distributed File System (HDFS) and framework for resource management (YARN). There are several other modules and functionalities available too (Apache, 2013).

Previous research on MapReduce and Hadoop was more focused on large scale clusters and cloud computational models where hundreds of computers included into a cluster were used, e.g. Dean, and Ghemawat (2004), Cary et al. (2009), Zhou, Wang, and Cui (2012), Zhu et al. (2009). Several authors pointed out the problem of high network load and Dean and Ghemawat (2004) describe a possible way of reducing amount of transmitted data through network by partial summing of results of map tasks. Another study was focused on optimization of input/output performance of small files (size from 1 KB to 10 KB) which were published by means of Web-based GIS application because HDFS was proposed to store large files (Xuhui et al., 2009).

Contrary to these, authors focus on a purely distributed solution which can be easily implemented in small laboratory conditions, with low costs and without utilization of a cloud service. The main goal of this paper is to describe and evaluate the proposed solution of utilization of MapReduce approach for processing of elevation data within a small-sized cluster consisting only of a few commodity PCs. In this way authors extend the previously done work.

3 DISTRIBUTED PROCESSING IN A SMALL CLUSTER – CASE STUDY

The proposed solution is based on Apache Hadoop 1.0.4. and Java 1.7. Ubuntu 12.10 is used as an operating system running on all PCs. Design model is simplified in comparison to the cloud GIS model proposed by Zhou, Wang, and Cui (2012). Principle of the used model is shown in the Figure 1.

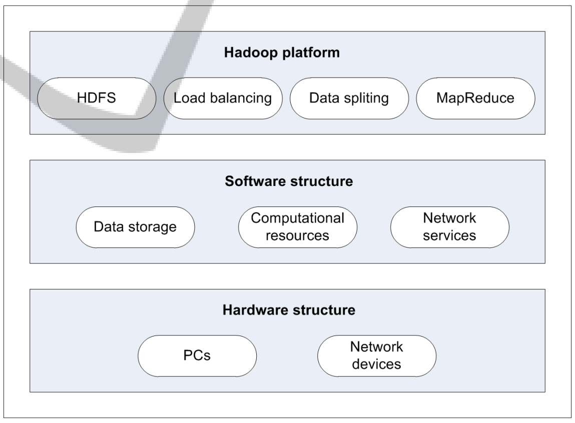

Figure 1: Design model of used solution.

3.1 Architecture

A very simple architecture is proposed. Only 5 commodity PCs are used. One of them works as a master, the others as slaves. Configuration of PCs: 4 GB RAM, quad-core four-thread Intel® Core™ i3-3220 3.3 GHz CPU. All the computers are connected by 100 Mbps Ethernet to a dedicated VLAN (Virtual Local Area Network) within university local network.

Used architecture and basic principles of communication are shown in Figure 2. The principle of communication is based on Hadoop properties (Apache, 2013). Input data are stored in local data storage. At first, data are split into blocks and

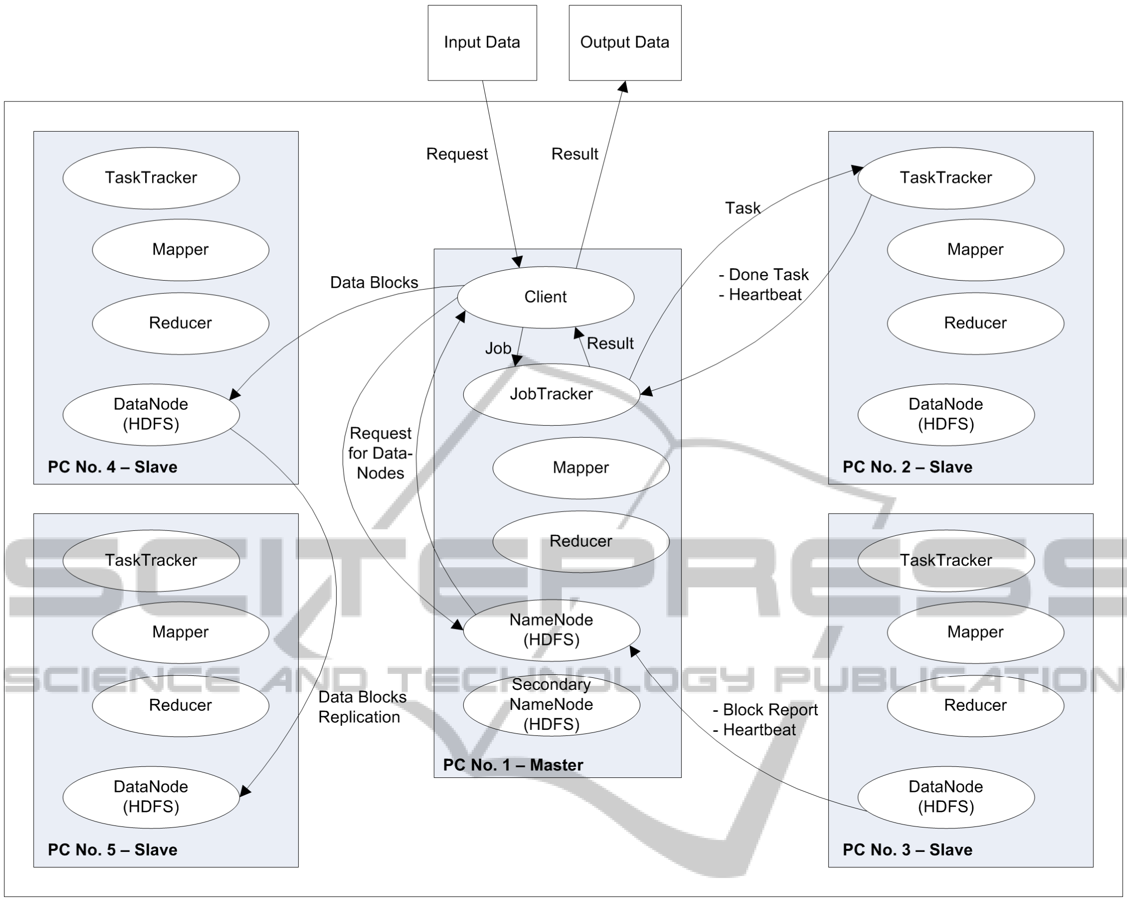

Figure 2: Architecture of used solution.

replicated to DataNodes. NameNode knows how blocks of data are distributed within the cluster. JobTracker is responsible for assigning TaskTrackers specific computational tasks to be solved. TaskTrackers regularly send heartbeat signal (a special kind of a message that they are alive and working properly) to let JobTracker know that they are available for the next task. DataNodes do the same, they send a heartbeat signal to the NameNode.

3.2 Data Processing

Point elevation data was used as input. Each record represented one elevation point – its X, Y and Z coordinates in text file containing 12 780 000 records, totalling 579 MB. Calculation was done for 38 340 000 values..

For the testing purposes splitting of input data was se into the default value – 128 MB blocks. Parameter "dfs.permission" was set to false to prevent problems with read/write permissions.

The first experiment was focused on the influence of number of PCs involved into the cluster. Obtained results are shown in the Figure 3.

Number of reduce tasks is an important parameter which can significantly influence computational performance of the system. It is set by the parameter "mapred.reduce.tasks" in the configuration file of Hadoop. By default, number of reduce tasks is set to 1 which means that reduce task is not distributed. An appropriate increasing of the reduce tasks can speed up the calculation. Inappropriate number of reduce tasks can slower the calculation because of increased network load (see Figure 4). The same testing file was used as in the previous step, so 38 340 000 values were processed.

An optimum number of reduce tasks can be calculated according to the number of available CPUs and cores. Increased number of reduce tasks allows the free cores to begin with reduce tasks before all map tasks are finished. It is recommended to dedicate one core to daemon and the rest to map and reduce tasks when there are more cores available (Stein, 2010; Apache, 2013). According to

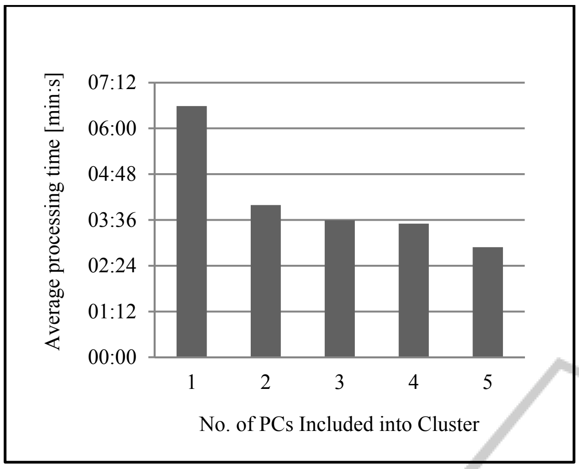

Figure 3: Average processing time according to the number of PCs included into cluster.

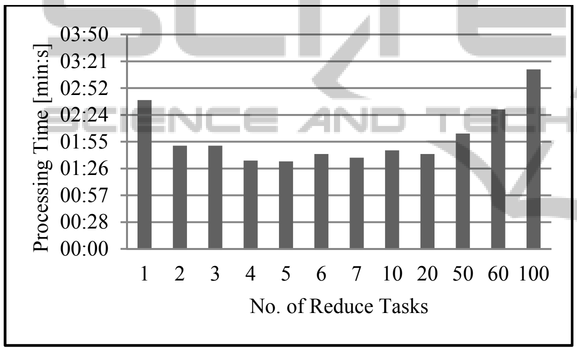

Figure 4: Average processing time according to the number of reduce tasks.

Stein (2010) it is recommended to set the number of reduce tasks: “somewhere between 0.95 and 1.75 times the number of maximum tasks per node times the number of data nodes”. In our case, 5 nodes and 4 threads per node (Hyper-Threading Technology) were available. It leads into an optimum interval <15; 26> reduce tasks. For the next experiment we set the parameter “mapred.reduce.tasks” to 15 because all of the used PCs were the same. Figure 5 shows processing times of bigger files containing higher numbers of values. Figure 6 confirms that map and reduce processes were partially run parallely. Adding times of map and reduce processes provides higher resulting time (Figure 6) than the ones which were really measured (Figure 5).

To illustrate computational demandingness, the same task was calculated using ArcGIS 10 for Desktop. Used hardware was dual-core AMD Opteron 8220 CPU, 48 GB RAM. In this case, processing of the frequency calculation of Z coordinate (elevation) took 36.3 min, without

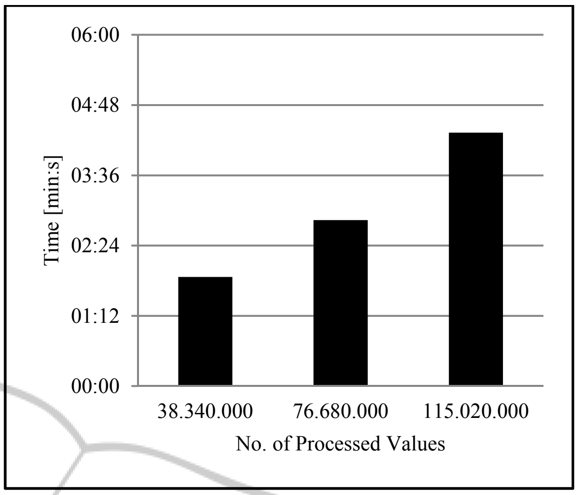

Figure 5: Measured Processing Times of Bigger Files Containing Higher Numbers of Values.

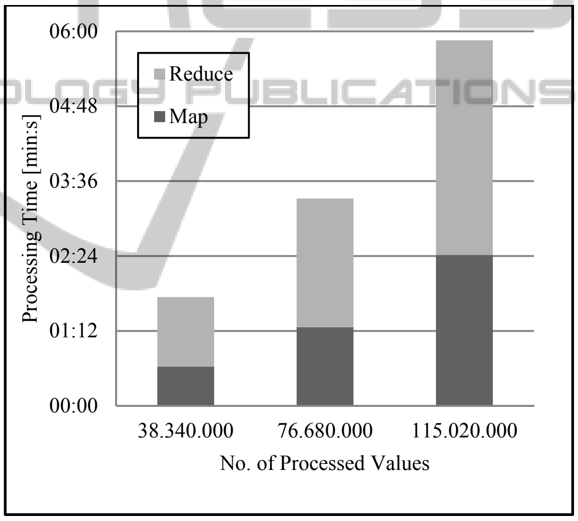

Figure 6: Processing Times of Bigger Files – Adding Map and Reduces Times Together.

keeping data visible. Only 12 780 000 values were processed by the tool “Frequency”. But the result cannot be used for an exact comparison because of different hardware. Yet it does point out and demonstrate the difference in processing times.

4 CONCLUSIONS

Distributed data processing can significantly improve computational performance and decrease time needed to process data.

Authors deal with processing LIDAR data and interpolation of digital surface models based on the

LIDAR data in real time (Hovad et al., 2012). The main aim of the authors is to propose a fast, easy and less resource hungry solution to interpolate LIDAR data and create 3D realistic surface models which can be used e.g. by public administration authorities or units of the Integrated Rescue System during appropriate steps of crisis management.

The main goal of the paper is to describe utilization of Apache Hadoop for processing of elevation data in a small-sized cluster of commodity PCs. Authors used only 5 PCs and partial steps are completed successfully. Solution of these steps, however, resulted in other issues which will be dealt with as further research.

ACKNOWLEDGEMENTS

The Ministry of Interior partly supported this work, by project VF20112015018. The Ministry of Education, Youth and Sports of the Czech Republic, projects CZ.1.07/2.3.00/30.0021 "Strengthening of Research and Development Teams at the University of Pardubice", and CZ.1.05/4.1.00/04.0134 "University IT for education and research" partly financially supported this work as well.

REFERENCES

Apache Software Foundation, Welcome to Apache™ Hadoop®! (online), 2013. [cit. 2013-06-04]. URL: < http://hadoop.apache.org/index.html>.

Barroso, L. A., Dean, J., Holzle, U., 2003. Web search for a planet: The Google cluster architecture, *IEEE MICRO*, 23 (2), 22-28.

Cardosa, M. et al., 2012. Exploiting Spatio-Temporal Tradeoffs for Energy-Aware MapReduce in the Cloud, *IEEE Transactions on Computers*, 31 (12), 1737-1751.

Cary, A. et al., 2009. Experiences on Processing Spatial Data with MapReduce, In *Scientific and Statistical Database Management, Proceedings, Lecture Notes in Computer Science*, vol. 5566, 302-319. Springer-Verlag.

Chu, S.-T., Yeh, C.-C., Huang, C.-L., 2009. A Cloud-Based Trajectory Index Scheme, In *ICEBE 2009: IEEE International Conference on E-Business Engineering, Proceedings*, 602-607. IEEE.

Dean, J., Ghemawat, S., 2004. Map Reduce: Simplified data processing on large clusters, In *OSDI'04 Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, 137-149.

Hovad, J. et al., 2012. Data Processing and Visualisation of LIDAR Point Clouds. In *Proceedings of the 3rd International conference on Applied Informatics and Computing Theory (AICT '12)*, 178-183. WSEAS Press.

Stein, J., 2010. Tips, Tricks And Pointers When Setting Up Your First Hadoop Cluster To Run Map Reduce Jobs (online). URL: <http://allthingshadoop.com/2010/04/28/map-reduce-tips-tricks-your-first-real-cluster/>.

Xuhui L. et al., 2009. Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS. In *Proceedings of the 2009 IEEE International Conference on Cluster Computing, August 31 - September 4, 2009, New Orleans, Louisiana, USA*, 1-8. IEEE.

Zhang C. et al., 2010. Case Study of Scientific Data Processing on a Cloud Using Hadoop. In *High Performance Computing Systems and Applications, Lecture Notes in Computer Science*, 5976, 400-415. Springer-Verlag.

Zhou, L. L., Wang, R. J., Cui, C.Y., 2012. GIS Application Model Based on Cloud Computing, *Communications in Computer and Information Science: Network Computing and Information Security*, 345, 130-136.

Zhu, S. et al., 2009. Evaluating SPLASH-2 Applications Using MapReduce, *Advanced Parallel Processing Technologies, Proceedings, Lecture Notes in Computer Science*, 5737, 452-464. Springer-Verlag.