Data-intensive Spatial Indexing on the Clouds

Abdelmounaam Rezgui\textsuperscript{a}, Zaki Malik\textsuperscript{b}, Jizhe Xia\textsuperscript{c}, Kai Liu\textsuperscript{c}, Chaowei Yang\textsuperscript{c}

\textsuperscript{a}Department of Computer Science & Engineering, New Mexico Tech, Socorro, NM, USA, rezgui@cs.nmt.edu
\textsuperscript{b}Department of Computer Science, Wayne State University, Detroit, MI, USA, zaki@wayne.edu
\textsuperscript{c}Center for Intelligent Spatial Computing, George Mason University, Fairfax, VA, USA, \{jxia3, kliu4, yang3\}@gmu.edu

Abstract

In this paper, we investigate the potential of cloud computing for data-intensive spatial indexing. We assess the benefits in performance and cost. Specifically, we consider a data- and compute-intensive spatial application, namely, the construction of very large R-tree spatial indexes. We selected this application because of its high computing and memory requirements. We implemented this application and deployed it on various types of cloud configurations. We report our findings and provide insights useful when considering cloud computing for data- and compute-intensive spatial applications.

© 2013 The Authors. Published by Elsevier B.V.
Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science

Keywords: Cloud computing, Data-intensive spatial indexing, R-Trees.

1. Introduction

Data- and compute-intensive applications require significant computing and/or storage resources. Cloud computing is a computing paradigm that has the potential to quickly provide adequate resources for these applications. However, in many cases, it is not easy to predict the benefits of running data- and compute-intensive applications on clouds [11]. In this paper, we consider a particular application, namely, the construction of very large R-tree spatial indexes. We assess the benefits in terms of performance of running this application on various cloud platforms. We selected this application because of its high computing and memory requirements. We implemented this application and deployed it on various types of cloud configurations. We report our findings and provide insights useful when considering cloud computing for data- and compute-intensive spatial applications.

2. Construction of Large R-Tree Spatial Indexes

Earth observation generates massive spatial data. For example, the amount of data downloaded from Terra’s instruments is about 195 GB of Level 0 data each day, which represents about 850 terabytes when processed to higher level science products ([9], [13]). It is essential to provide efficient search capabilities for these massive
data. Spatial search operations such as: intersects, crosses, within, contains, etc. are known to be compute-intensive. This is particularly true for large spatial datasets. In order to handle these complex spatial operations, it is important to have an index mechanism that enables the retrieval of data quickly according to their spatial location. The R-tree [6] is one of the most popular spatial indexing methods. Many index technologies based on R-trees have been extensively researched such as R*-trees [2] and R+-trees [12]. The R-tree family is widely used in mainstream database systems such as Oracle Spatial [7] and PostGIS [10].

Building and updating large R-trees is very computationally complex and time consuming. The time to build an R-tree or update a node is almost linear in the number of spatial features. These operations usually require high speed CPUs, large memory, and large secondary storage. Substantial research has been conducted to handle this complexity. For example, in [8], Mondal et al. proposed an R-tree structure based on peer-to-peer environments. Recent research, e.g., Cary et al. [3], explored the use of MapReduce [4] to construct R-tree indexes. Although MapReduce is generally suitable for the parallel processing of large data sets, recent results do not show a significant improvement when using MapReduce to build hierarchical indexes such as R-trees [1].

In this paper, we explore the suitability of cloud computing for the construction of very large R-tree indexes. We experimented with an existing R*-tree software package developed at Hong Kong University of Science Technology (www.rtreeportal.org). We considered different R-tree sizes and different amounts of cloud computing resources. Specifically, we measured the construction time of an R*-tree with 1000000, 4000000, 10000000 spatial features on different types of cloud instances. For the sake of brevity, we only report results for the two latter cases. We used two cloud platforms: Amazon EC2 and Microsoft Windows Azure.

Figure 1 and Figure 2 show the experimental results for building an R-tree with 4M and 10M features respectively. Figure 2 shows that we could not run the application on a small instance for the case of 10M spatial features. This is normal since the computation requires the entire R*-tree to be loaded in memory at runtime. Also, the previous results show that the performance obtained does not always conform to the intuitive prediction. In almost all cases, cloud instances with large system configurations perform worse than instances with smaller system configurations. An exception is that local computer X-large instances with 8 CPU cores and 16 GB memory perform better than local computer large instances with 4 CPU cores and 4GB memory. Among all cloud deployments, the ones on “High Memory X Large” instances give the best performance. The “High Memory X Large” instance has two CPU cores less and 2.1 GB more memory than the “X Large” instance. However, it performs much better than the “X Large” instance which indicates that the nature of the application (i.e., having high memory requirements) plays a more important role than computing power in determining the overall performance. This can be further seen when comparing the performance results of the “High Memory X Large” and the “High CPU X Large” instances. For example, in the case of 10M spatial features, the former gives an execution time of 923.92 seconds while the latter gives an execution time of 1425.89 seconds. Execution time increases by %54 when the computing power increases from 6.5 ECUs to 20 ECUs and memory decreases from 17.1 GB to 7 GB.

Two reasons are, in part, behind the low performance achieved by this application: 1) the application was written in Java with all of its well-known memory management issues, and 2) the application’s code was not optimized to run on configurations with multiple CPU cores. In particular, because the application does not use multithreading, using more cores does not improve performance.

We also compared the performance of the R*-tree construction application on different cloud instances with regard to memory size, number of CPU cores, and CPU speed (Figures 3, 4, and 5). The figures show the following results: 1) instances with high CPU speed perform better than instances with low CPU speed, 2) utilizing more CPU cores/ECUs does not improve performance. In fact, performance sometimes degrades with more CPU cores/ECUs (for the same two reasons mentioned earlier), and 3) no obvious pattern is found with regard to the impact of increasing memory size. This is particularly true in the case of Amazon EC2 instances. Azure instances show moderate degradation in execution time when memory increases from 7-8 GB to 14-17
GB. This degradation is caused, in part, by the inability of the application to exploit the increase in the number of CPU cores from 4 to 8.

The results obtained in this second set of experiments highlight the fact that many of legacy, data- and compute-intensive software products that geospatial scientists have successfully used in traditional local settings are not adequate for the clouds. For example, almost all legacy Fortran-based data analysis code is based on sequential for loops. Running such a code on a cloud instance with a large number of CPU cores
would likely not improve performance. These legacy software products will have to be substantially redesigned to use cloud resources more efficiently.

Figure 5. Building an R*-tree with 4M Spatial Features (performance when varying CPU speed)

3. Conclusion

Cloud computing is now seen as a promising, cost-effective paradigm to support the execution of compute- and data-intensive spatial applications. Experts in academia and in industry view it as a key enabler of data-intensive scientific discovery [5]. From the experiments conducted in this work, it is clear that cloud platforms are not necessarily an easy solution to address the complex nature of compute- and data-intensive applications. Substantial research is still needed on how to best design spatial applications to be efficiently executed on the clouds. We argue that, to enable efficient data-/compute-intensive spatial computing on the clouds, efforts will have to go into four directions: (i) developing new design approaches for spatial applications specifically tailored for cloud environments, (ii) developing new mechanisms for accurate cost/benefit assessment of deploying applications on clouds, (iii) improving cloud platforms to better support science applications, and (iv) exploring new distributed computing alternatives on the clouds.

Acknowledgements

This research is supported by NSF grant CSR-1117300, the NASA SMD Cloud Computing Test Initiative (NNX07AD99G), and the ESIP Cloud Testbed Initiative.

References

[1] A. Akdogan, U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, Voronoi-Based Geospatial Query Processing with MapReduce, Proc. of the 2nd IEEE Intl. Cloud Computing Technology and Science (CloudCom), Indianapolis, IN, Nov. 30-Dec. 3, 2010.
[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. ACM SIGMOD, pages 322–331, May 1990.
[3] A. Cary and Z. Sun, V. Hristidis, and N. Rishe, Experiences on Processing Spatial Data with MapReduce. In Proc. of SSDBM, 2009.
[4] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters. OSDI, pp. 137-150, 2004.
[5] T. Hey, S. Tansley, and K. Tolle (Eds), The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, ISBN 0982544200, 2009.
[6] A. Guttman, R-Trees: A Dynamic Index Structure for Spatial Searching. Proc. of the ACM International Conference on Management of Data (SIGMOD), pp. 47-57. ISBN 0-89791-128-8, 1984.
[7] V. Kothuri, R. Kanth, R. Hanckel, and A. Yalamanchi, Using Oracle Extensibility Framework for Supporting Temporal and Spatio-Temporal Applications, Proc. of the 15th International Symposium on Temporal Representation and Reasoning (TIME), pp. 15-18, 2008.
[8] A. Mondal, Y. Lifu, and M. Kisturegawa, P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments. In Proc. of EDBT, 2004.
[9] NASA, Earth Science Reference Book, 2006.
[10] PostGIS, http://postgis.refractions.net, 2013.
[11] A. Rezgui, Z. Malik, and C. Yang, High-Resolution Spatial Interpolation on Cloud Platforms, 28th ACM Symposium on Applied Computing (SAC), March 18-22, 2013, Coimbra, Portugal, 2013.
[12] T. Sellis, N. Roussopoulos, and C. Faloutsos, The R+ tree: A Dynamic Index for Multidimensional Objects. In Proc. of the 13th International Conference on VLDB, pp. 507-518, England, 1987.
[13] Terra, http://terra.nasa.gov, 2013.