Cloud Computing for Online Visualization of GIS Applications in Ubiquitous City

Jong Won Park, Yong Woo LEE, Chang Ho Yun, Hyun Kyu Park, Seo Il Chang, Im Pyoung LEE, Hae Sun Jung*
The Ubiquitous (Smart) City Consortium,
The University of Seoul, *Korea University
{comics77, ywlee, touch011, ajnick31, schng, iplee}@uos.ac.kr, holylife7@hotmail.com

***Abstract*— Cloud computing can be used to generate the 3D noise maps in ubiquitous cities. Here in this paper, we present our cloud computing approach, its performance and a performance comparison for it. The 3D image processing with GIS data requires great amount of computational resource because of complex and large amount of spatial information. The cloud computing can solve the problem with an easy and transparent way. We use Hadoop which is a framework that includes the HDFS (Hadoop Distributed File System) and MapReduce as cloud computing methodology to do massively parallel processing of 3D GIS data. We found the computing time is vastly reduced with a cluster of computing nodes. We also present the performance comparison when we use MPI instead of MapReduce and Hadoop.**

Keywords- cloud computing; the noise map; GIS; Hadoop; MapReduce; MPI.

I. INTRODUCTION

In the 1990s, the noise map was presented to develop the environmental policy to reduce the noise in cities. Afterward, in 2002, Directive 2002/49 relating to the assessment and management of environmental noise was adopted by the European Parliament and Council for the developments of the long-term noise policy. The European Environmental Noise Directive (2002/49/EC) is one of the European Community's policies which have the goal to avoid, prevent, and decrease their displeasure and harmful effect caused by environmental noise exposure [1].

We find that immediately after the standard about the noise map was adopted by the EC, European Initiatives on the research of the noise map have been activated. The noise map combines noise information with GIS map. It requires a large amount of computing power and cannot be timely done with personal computers. In the reason, the noise map is usually made offline mode for long time and not in three-dimension but in two-dimension. However, current cities have high-rising buildings and we need to show the noise difference on each floor. In consequence, it is important to generate the 3D noise map [2][3]. The 3D image processing with GIS data should deal with complex and large amount of spatial information and requires great amount of computational resource.

In this paper, we present our approach to solve the problem in two ways and compare the performance. One way is to use MapReduce [4] with Hadoop system [5] and the other way is to use MPI.

The structure of this paper is as follows. In Section 2, we introduce, compare and analyze the state-of-the art works related to our research. In Section 3, we explain the steps of noise map. In Section 4, we describe our cloud computing approach to do it. In Section 5, we give performance evaluation. Finally, we conclude and explain the future work in Section 6.

II. RELATED WORK

EU has been actively researched noise map. Table 1 shows EU countries and their participating cities in the research [6]. Figure 1 and Figure 2 show some of their research results, that is, two noise maps in two-dimension. They do not produce online noise maps but makes the noise maps in offline mode and do not use cloud computing. The research on the 3D-noise map is an arising topic and not found except our work.

TABLE I. EUROPEAN UNION (EU) COUNTRIES AND THEIR CITIES MAKING THE NOISE MAP

Country	City
United Kingdom	London, Birmingham
Germany	Berlin
France	Paris
Netherlands	Amsterdam
Czech	Prague
Italy	Bologna
Switzerland	Geneva
Austria	Vienna
Sweden	Stockholm
Finland	Helsinki
Belgium	Brussels

 ISBN: 978-1-61208-106-9

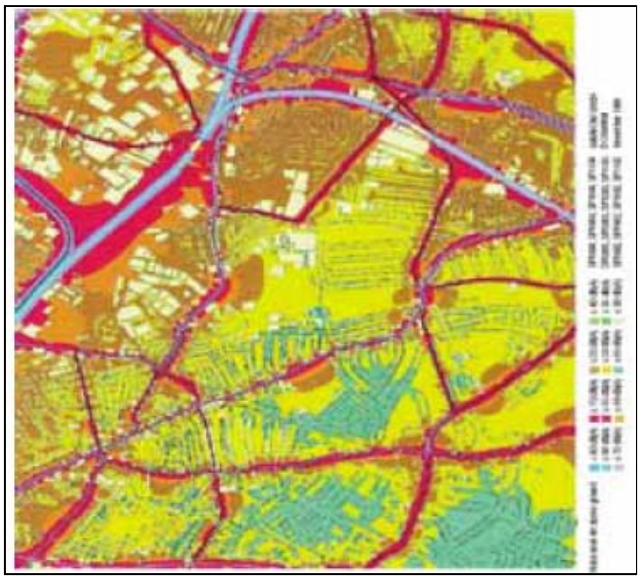

Figure 1. A noise map of Birmingham, U.K.

Figure 2. A noise map of Amsterdam, Netherlands

III. How to Generate the Noise Map?

The noise map is currently made in two dimensions. However, in this research, we are interested in the three dimension noise map. In this paper, we explain our way to generate the three dimension noise map, not the two dimension noise map. In order to make a 3D noise map, the following three-step-process is needed: 1) Making a noise database. 2) Generating the 3D city model. 3) Integrating the noise values with the 3D city model.

A. *Making a Noise Database*

In our ubiquitous cities, the noise data are collected through ubiquitous sensor network from remote sensors and sent to the database. We can also use an interpolation approach to make the database. That is, we measure the noise at important areas and use a noise prediction model to predict noise values at unmeasured areas using the measured data at the area nearby the unmeasured area [7].

B. *Generatinga 3D City Model*

The generation of 3D city model includes the terrain modeling as shown in Figure 3 and the building modeling [8] as shown in Figure 4. It needs big computing power because generation of the 3D building model is very complicated. To solve the problem, we use the cloud computing [9].

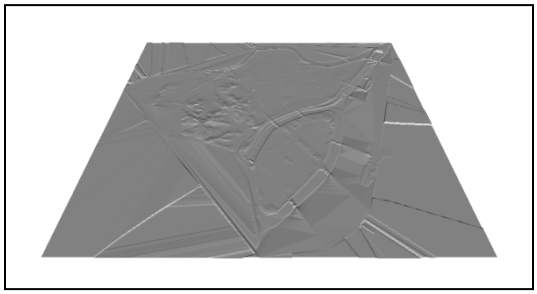

Figure 3. A digital elevation model

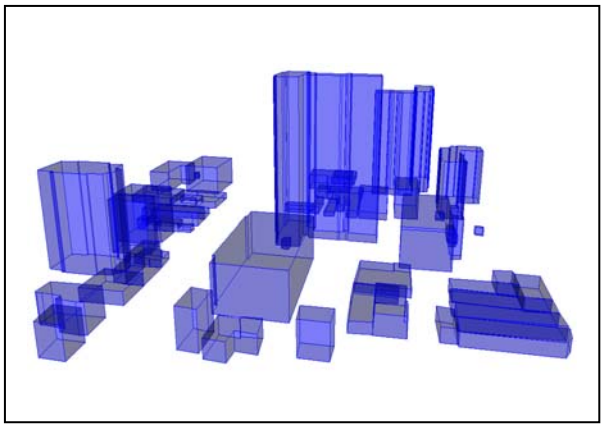

Figure 4. Building models

C. *Integrating the Noise Values with the 3D City Model*

Now, we integrate the noise values with the 3D city model. Because the data of 3D city model is very large, converting each noise value into RGB value and mapping the RGB value onto the texture file of the 3D city model requires a large amount of computing power. To solve the problem, again, we use the cloud computing. Thus we can reduce the running time to the level of online processing. Figure 5 shows a sample digital map of an experimental area.

 ISBN: 978-1-61208-106-9

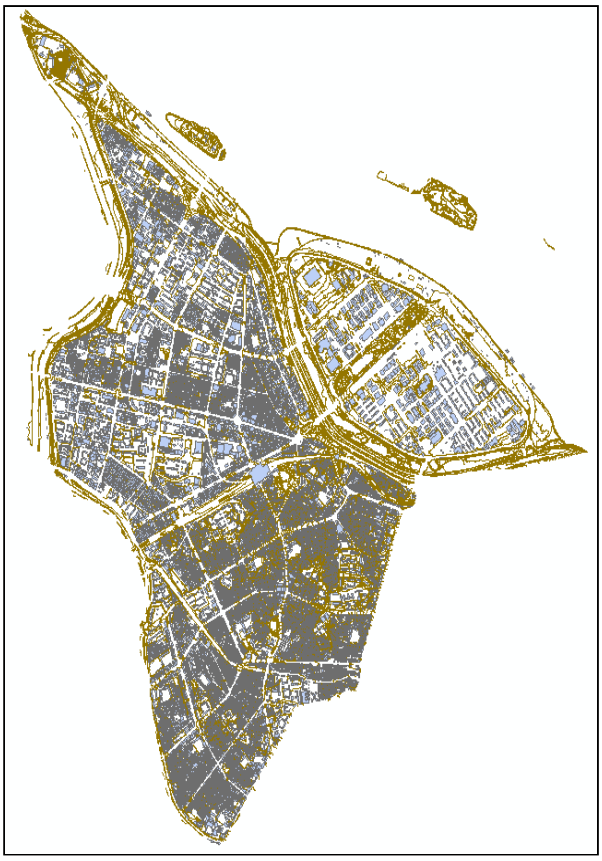

Figure 5. The digital map of Yeongdeungpogu Disrict, Seoul, Korea

IV. THE CLOUD COMPUTING

The cloud computing process to make the 3D noise map is shown in Figure 6. To process 3D data, we employ the ways that the data of the digital map are divided into grid cell units. The data of the digital map make a huge file so we use the MapReduce with Hadoop that is one of cloud computing technologies to do massively distributed and parallel processing. Distributed and parallel programming greets the new trends due to the cloud technologies such as Hadoop, an open source Java framework. It consists of Hadoop Distributed File System (HDFS) and MapReduce. HDFS uses a scheme of replication to ensure that the stored files are always kept intact in separate places of a Hadoop cluster. It enables us to solve a large scale of data intensive problems.

We divide the data of the 3D GIS images into the unit of grid cell for the MapReduce processing and later integrate the result since the MapReduce uses Single-Program Multiple-Data (SPMD) methodology [10]. As shown in Figure 7, we used MapReduce to make the 3D city model and the 3D noise map and the generated 3D city model is reused as an input to the 3D noise map.

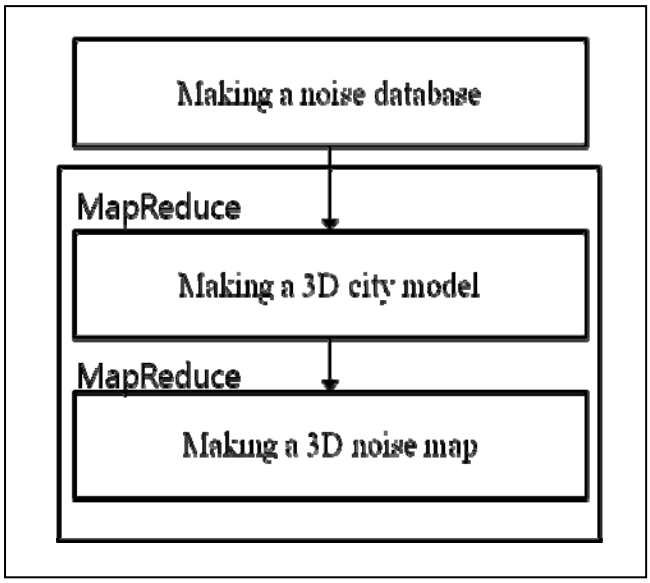

Figure 6. The cloud computing process to make the 3D noise map

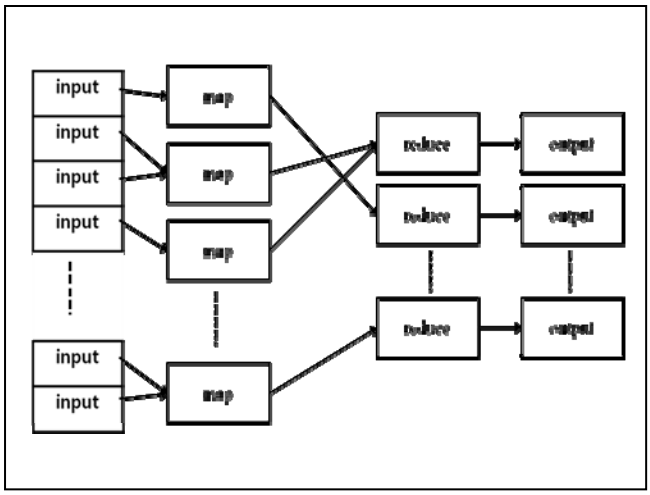

Figure 7. The MapReduce execution

A. *Cloud Computing to Make a 3D City Model*

Here, we explain how we do cloud computing with MapReduce to make the 3D city model. We use two kinds of map functions: map_1 and map_2. Map_1 plays a role of making a Digital Elevation Model (DEM), also called as a Digital Terrain Model (DTM), which has the topology information and height information of ground surface for the 3D city model. Map_2 plays a role of making a building model which has the object topology information and the height information as shown in Figure 4. Reduce integrates the DEM and the building model: this process is called as reduce_1. The output of the reduce_1 is a 3D city model. Table 2 explains the three functions: map_1, map_2 and reduce_1.

 ISBN: 978-1-61208-106-9

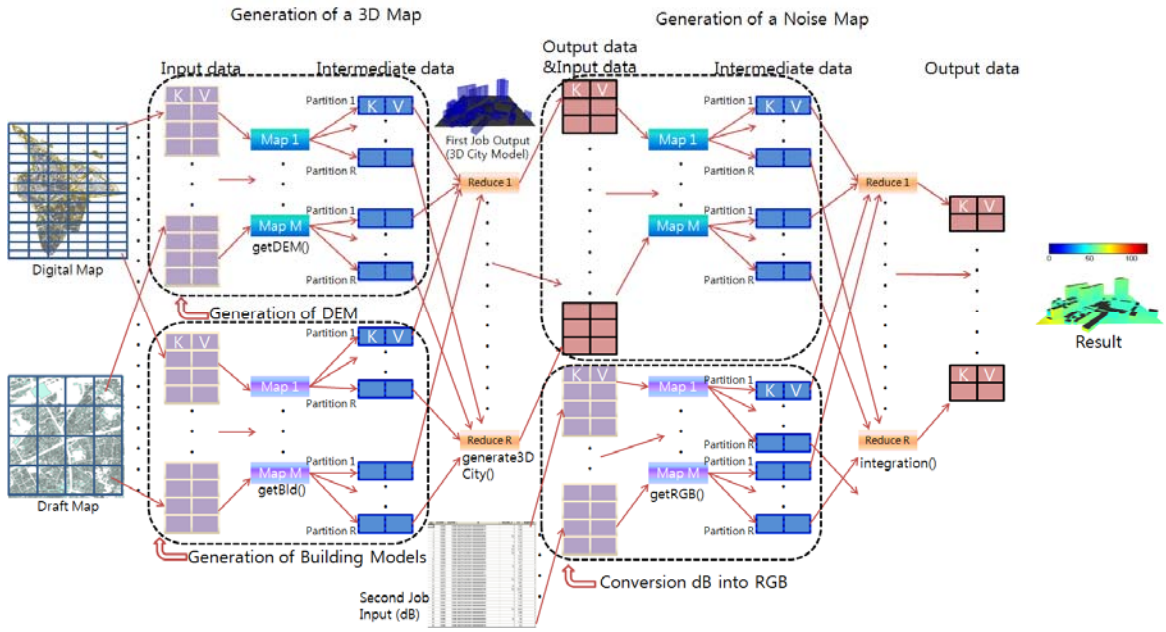

Figure 8. The process of MapReduce function in visualization of the 3D noise map

TABLE II. THE TASKS TO MAKE A 3D CITY MODEL

Function	Task	Key-Value Pair
Map_1	To make a DEM.	<<Sub-area ID, x, y coordinates>,<z coordinates, topography ID >>
Map_2	To make a building model.	<<Sub-area ID, x, y coordinates>,<z coordinates, building ID >>
Reduce_1	To integrate the DEM and the building model.	<<Sub-area ID, x, y coordinates>, <z coordinates, value of 3d city model>>

To make the building model, the getBld() of the map_2 extracts the coordinates of 2D building boundary from the digital map and extracts the z value of the building from the draft map by establishing the correspondence between the building in the digital map and it in the draft map.

We divided the test area into a number of sub-areas and assigned an ID to each of them. When the test area is processed in the map function, the coordinates of the specific area is assigned with a sub-area ID. Therefore, the key value becomes <sub-area-ID, x, y-coordinate>.

The outputs of "map_1" and "map_2" are sorted and grouped according to the ID by the partitioner. The outputs of the partitioner become the input of "reduce_1". It means that the key-value pairs of the DEM and the building model that are sorted and grouped become the inputs of "reduce_1". "Reduce_1" calls and process the generate3DCity() function to generate the 3D city model. Each "reduce_1" task is matched to each sub-area and therefore the number of the "reduce_1" task is same as the number of the sub-areas. The output of "reduce_1" will be used as the input of "map_4" in the next step.

B. *Cloud Computing to Making a Noise Map*

Here, we explain how we combine the 3D city model with the noise information to generate the 3D noise map. We use two kinds of map functions: map_3 and map_4. Map_3 takes the noise information of buildings as the inputs of reduce_2. As the output, we take the key-value pair of <<building ID, x, y coordinates> and <z coordinates, value of noise level>>. Map_4 transfers the result of reduce_1 to make the 3D noise map. Table 3 explains the three functions: map_3, map_4 and reduce_2.

TABLE III. THE TASKS TO MAKE A NOISE MAP

Function	Task	Key-Value Pair
Map_3	To take the noise information of buildings as the inputs of reduce_2.	<<building ID, x, y coordinates>, <z coordinates, value of noise level>>
Map_4	To transfer the result of reduce_1 to reduce_2.	<<Sub-area ID, x, y coordinates>, <z coordinates, value of 3d city model >>
Reduce_2	To integrate the 3D city model and noise information.	A noise map.

The input of map_3 has coordinates, building ID and noise value to make a noise map. Because it has a coordinates, the noise value can be matched to 3D City

model. As a key of map_3, we use a building_ID to distinguish each building.

The outputs of "map_1" and "map_2" are sorted and grouped according to the ID by the partitioner. The outputs of the partitioner become the input of "reduce_2". Reduce_2 plays a role of visualizing the RGB by combining the inputs with 3D city model. We divide noise level distribution of the test area into sub areas and find RGB color index using getRGB() function. When we convert it into color index, we use the equation as shown in Eq. (1) [3]:

$$NC = \frac{(C-1)*(N-N_{max})}{N_{max}-N_{min}} \quad (1)$$

In Eq. (1), NC is color index, Nmax is the maximum value of the noise pollution level, Nmin is its minimum value, C is the total number of colors and N is the noise level on each grid.

After noise level is converted into RGB values, the following steps are performed to generate the noise map. First, the group of the points corresponding to each wall facet is classified according to the proximity of each point to the facet. Second, the RGB of the classified points are then interpolated into a grid using the same encoding scheme presented as Eq. (1). Now, we can get facet image files and merge the files into one file so as to generate a 3D noise map. The final result of the merged file is the 3D noise map as shown in Figure 9.

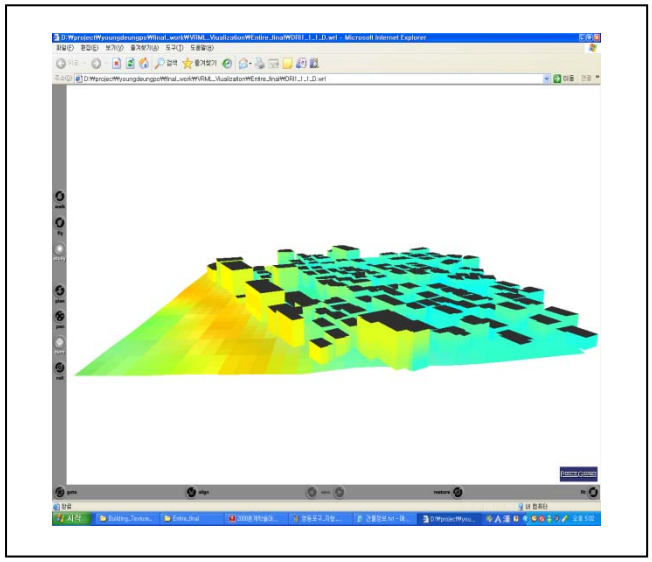

Figure 9. A snapshot of a 3D noise map

V. PERFORMANCE EVALUATION

Here, we show the performance of our approach and compare it to the performance of the approach with MPI to generate 3D noise map. The reason why we do this comparison is that we want to be sure of the advantage and disadvantage of our approach. We have been seeking the currently best cloud computing solution to process the large amount of 3D GIS data for ubiquitous city applications. To find out the answer, we did this performance comparison.

For the performance comparison, we used a ten nodes cluster, where 8 nodes had Dual Core Intel processor and 2 nodes had Quad Core Intel Processor and each node had 4 GB memory. Each node of the cluster was connected through a giga-bit Ethernet switch, runs a Ubuntu Linux 9.04 Server edition and used our own private Cloud based on OpenNebula. The JVM version 1.6.0_20 was used for Hadoop and the gcc version 4.4.1 compiler and MPICH2 were used for the MPI. For the noise map area, we selected Yeongdeungpogu District, Seoul, Korea, as shown in Figure 5, where the area size is about 24.5km^2 and the volume of the processed data was 250 GB. We processed both MapReduce and MPI experiments and measured the performance. We ran them 10 times and averaged the results. Figure 10 shows the performance and we know that the MPI case is faster than the MapReduce case.

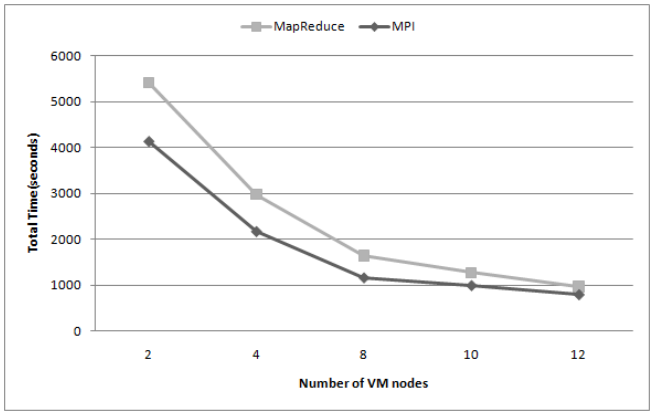

Figure 10. Performance comparison between MapReduce and MPI

Distributed and parallel processing based on message passing infrastructures such as PVM [11] and MPI [12] supports fine-grained parallelism, while workflow frameworks such as Kepler [13] and Taverna [14] supports coarse-grained parallelism. MapReduce also supports fine grained parallelism but it is different from MPI or PVM since it does not support any shared files but supports local files only. That is, by restricting the programming model, the MapReduce framework enables us to partition the given tasks into a large number of fine-grained sub-tasks, but it does not communicate each node since it only supports local files.

While MPI supports a wide variety of communication topologies for various kinds of distributed and parallel models. MapReduce only allows a communication topology from map to reduce. However, MapReduce allows us to use

 ISBN: 978-1-61208-106-9

a simple but convenient cloud computing environment, which eventually allows us to implement parallelism to run our applications. Also, MapReduce gives better support to quality of services such as fault tolerance and monitoring in data intensive parallel applications.

VI. CONCLUSION

In this paper, we have presented our cloud computing approach to process a large amount of 3D GIS data to make the 3D noise map. We find that MapReduce with Hadoop is useful to reduce the turnaround time vastly. We also present the performance comparison when we use MPI instead of MapReduce and Hadoop. We find that the MPI case is faster than the case of MapReduce with Hadoop. However, we also find that MapReduce case has better fault-tolerance and more stable than MPI case in our experiment. We find that the MapReduce with Hadoop is not suitable for real-time interactive processing and thus have been studying real-time interactive processing of our work with MapReduce and any other useful cloud computing technology.

ACKNOWLEDGMENT

This study was supported by the Seoul Research and Business Development Program (10561), Smart (Ubiquitous) City Consortium and Seoul Grid Center. We would like to give thanks to Mr. Cheol Sang Yoon, Mr, Seung Woo Rho, Mr. Chang Won LEE, Mr. Kyoung Kyu LEE, Mr. Eui Dong Hwang, Mr. Sung Min Kim and the staffs of Seoul Grid Center and the members of Smart (Ubiquitous) City Consortium for their contribution to this research.

REFERENCES

[1] DIRECTIVE 2002/49/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 25 June 2002 relating to the assessment and management of environmental noise, “Official Journal of the European Communities”, 2002.

[2] Kurakula, V., “A GIS-Based Approach for 3D Noise Modelling Using 3D City Models”, MSc proposal, University of Southampton, UK, 2007.

[3] S. Oh, I. LEE, S. Tanathong, J. Ko, S. Chang, and T. Kim, “Generation of 3D Noise Map using a City Model”, 3D Geoinfo, pp. 155-160, 2008.

[4] J. Dean and S. Ghemaway, “MapReduce: Simplified Data processing on Large Clusters”, Communications of the ACM, vol. 51, January, 2008, pp. 107-113, doi:10.1145/1327452.1327492.

[5] Apache Hadoop Homepage [online], October 2010, Available from: http://hadoop.apache.org/common/.

[6] The noise map in Europe [online], October 2010, Available from: http://www.xs4all.nl/~rigolett/ENGELS/maps/.

[7] Cho, D. S., J. H. Kim, and D. Manvell. “Noise mapping using measured noise and GPS data”. Applied acoustics, vol.68 no.9, pp. 1054-1061, 2007, doi:10.1016/j.apacoust.2006.04.015.

[8] Oh, S., I. LEE, S. Kim, and K. Choi. “Generation of a Spatial city model using a Digital Map and Draft Maps for a 3D Noise Map”. Korean journal of remote sensing, vol.24 no.2, 2008, pp. 3-14.

[9] N. Golpayegani and M. Halem, “Cloud Computing for Satellite Data Processing on High End Compute Clusters”, Proceedings of the 2009 IEEE International Conference on Cloud Computing, 2009, pp. 88-92, doi: 10.1109/CLOUD.2009.71.

[10] F. Darema, “The SPMD model: past, present and future” Recent Advances in Parallel Virtual Machine and Message Passing Interface, 8th European PVM/MPI Users' Group Meeting, Santorini/Thera, Greece, 2001, Proceedings. Lecture Notes in Computer Science, 2001, Volume 2131/2001, pp. 1, doi: 10.1007/3-540-45417-9_1.

[11] Jack J. Dongarra, G. A. Geist, Robert Manchek, and V. S. Sunderam, “Integrated PVM Framework Supports Heterogeneous Network Computing” Computers in Physics, 1993, pp. 166-175.

[12] MPI (Message Passing Interface) [online], October 2010, Avaulable from: http://www-unix.mcs.anl.gov/mpi/.

[13] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao, Scientific workflow management and the Kepler system: Research Articles. Concurr. Comput. : Pract. Exper. 18(10), pp. 1039-1065, 2006.

[14] Hull, D., K. Wolstencroft, et al. “Taverna: a tool for building and running workflows of services” Nucleic Acids Res 34(Web Server issue): W729-32, 2006.

 ISBN: 978-1-61208-106-9