

Distributed and Parallel Processing of Noise Information for 3D GIS in a U-city

Hyun Kyu Park, Jong Won Park, Chang Ho Yun, Yong Woo LEE (Corresponding Author)

School of Electrical & Computer Engineering, University of Seoul, Korea

{ajnick31, comics77, touch011, ywlee}@uos.ac.kr

Abstract— To generate the 3D GIS urban noise map, we need a lot of computing power. To meet the requirement, this research used MPI with our own distributed and parallel processing model. MPI is known to have the advantage of portability and reduce the processing time of the map. The distributed and parallel model is used when the text file that has the noise values of a designated area is processed to make jpg files. We experimented the distributed and parallel model and found that the processing time was increased linearly when the size of the file was increased. Thus, we prove that the model works well in generating the 3D GIS urban noise map.

Keywords— MPI, Noise Map, GIS, Distributed and Parallel Processing

I. INTRODUCTION

U-city[1] is a city that provides ubiquitous city service through the constructed infrastructure using technology of ubiquitous city in everywhere and anytime for quality of life and competitiveness of city. It aims to make future city using ubiquitous computing technology. The study which is related to noise is researched to enhance the quality of citizen's life in a U-city. The noise map is studied in 1990s to reduce the noise level in a city and the European Parliament and Council adopted the Directive 2002/49 which is related to policy of the assessment and the management of noise environment in 2002.[2] This means that the researches of the noise map are growing more than before and it pushes the development of the technology of the noise map. They regard the noise map as combination of the noise information and GIS technology. Therefore, to generate the noise map, GIS technology is required and it means that it require the technology of the image processing. However, to make the 3D GIS urban noise map, GIS data has a lot of spatial information in each grid. To process this 3D image is required many computing resources. So we can induce from it that to generate the 3D GIS urban noise map is required a lot of computing power.[3] Therefore, the problem is how we can provide the computing power in a real-time when we make the 3D GIS urban noise map.

To solve this problem, we use MPI[4] with our own distributed and parallel processing model and we will show how we use MPI and how we provide a lot of computing power in a real-time. After we propose the key way, we implemented it and we experiment it to show the performance of it.

This paper is organized as follows. Section 2 discusses the generation of the 3D GIS urban noise map. We divide the whole steps into three steps. Section 3 explains how we propose the model of the distributed and parallel processing to solve the problem which is to provide a lot of computing power. Section 4 shows the performance of the model which is proposed in section 3 and we will show how the processing time is reduced. Section 5 discusses the results of this paper.

II. 3D GIS URBAN NOISE MAP

The 3D GIS urban noise map is showed in figure 1.

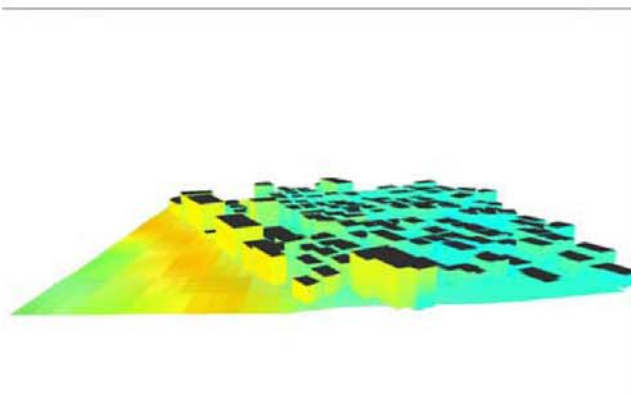


Figure 1. The 3D GIS urban noise map

To generate this 3D GIS urban noise map, ground needs to be modelled at first. After that, the buildings need to be modelled. The modelling is completed after ground and buildings are combined together. The output of this step is showed in figure2.

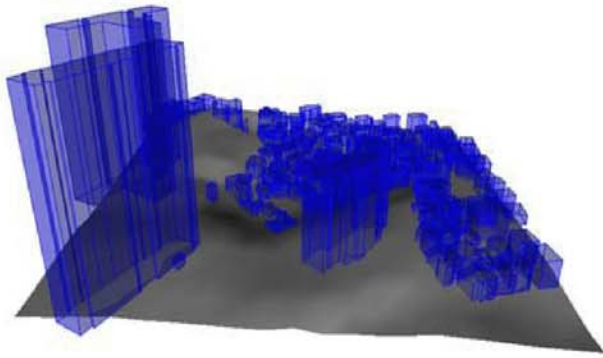


Figure 2. Modelling of the ground and building

After modelling ground and buildings, we need to process the text file to jpg file. The text file has the noise level information of a designated area. After processing it, the model of building is wrapped in the jpg file which is matched in each building so we generate the output file of the 3D GIS urban noise map as wrl file which is the extent name of it.[5]

III. DISTRIBUTED AND PARALLEL PROCESSING MODEL

A. 3D GIS Urban Noise Map in a Node

We choose the test bed where is nearby Cheonggyecheon as the width is 200m and the length is 200m. In the test bed, the total number of buildings is 52. The size of test bed is 1/150000 times compared with the size of Seoul city which is 605 square kilometres in area.

The file size for modelling of ground and buildings is 1MB and the file size for noise level information of a designated area is 50M. We implemented it as C language and we measured the processing time which is the generation time of the 3D GIS urban noise map in a node. The result of the experiment is showed in figure 3.

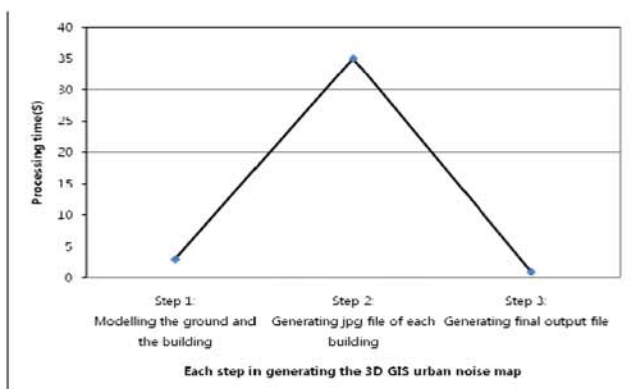


Figure 3. Processing time in generating the 3D GIS urban noise map

In the figure, the x axis is steps for generation of the 3D GIS urban noise map. First step is to model the ground and

buildings and to combine the models. Second step is to generate jpg files of a designated area's noise level using text file which has the noise level of a designated area. Third step is to generate the final output of the 3D GIS urban noise map which has the extent "wrl". The y axis is the processing time which means to complete each step. As showed in figure 3, the total time for generating the 3D GIS urban noise map is 39 seconds and in the step2, most of the time is taken. It is over 90% to generate the 3D GIS urban noise map and we can know that a lot of time is taken in step2. In this step, the text file which has the noise level information of a designated area is read and it is processed to make jpg files. We can analogize that to generate the faces of one specific zone in a designated area and to generate the faces of another specific zone in a designated area which is needed to read the text file is not related to each other so it is independent each other to generate the faces of specific zone. Therefore, as the conclusion is drawn in figure3, we can know that the step2 satisfies the two conditions below.

(1) The step2 take a lot of time in generating the 3D GIS urban noise map.

(2) The data (in this case, the text file) is independent when the data is used to generate the jpg files in a designated area.

As the step 2 satisfies the two conditions, we process it as distributed and parallel computing.

B. Framework Choice

To use the distributed and parallel processing, we need to choose what is fit for it on framework. There are many frameworks like mapreduce[6] and cgl-mapreduce[7] but we choose MPI to model the distributed and parallel processing.

1) The Strength of MPI

We use MPI (Message Passing Interface) to provide a lot of computing power in a real-time. The reasons that we choose are as follows.

(1) MPI is defacto standard in distributed and parallel processing so it guarantees and provides portability in source level. This means that if the operating system is changes, the source code is not needed any changes in it so it can transfer as it is. Therefore, because the portability is guaranteed, we can use the source code immediately without any changes.

(2) Because the programmer can use each node independently, the performance is higher than other frameworks through appropriate distributed and parallel processing model. This means that various distributed and parallel processing models can be made and we can choose one of them that is more fit in the 3D GIS urban noise map than others. Therefore, it can reduce the processing time more than others.

We choose MPI because of these two main reasons. The reasons are strengths of MPI.

C. Distributed and Parallel Processing Model in the 3D GIS Urban Noise Map

We make a model for the distributed and parallel processing for step 2 to reduce the processing time.

We divide the text file which has noise level values in a designated zone. It means that all of nodes are not needed the

whole information of the file to generate the 3D GIS urban noise map. The nodes just need to read the part of the text file that they want to know the specific zone. In this test bed, the total number of buildings is 52 so the total number of the files is divided in 52 files. So we can reduce the read time because of the above reason. We regard these 52 files as an area and we put them on a directory. After putting on them in a directory, we copy this directory repeatedly because we need to increase the input files.

The files are put on NFS (Network File System). The reason of this is that all of the nodes do not need to copy the files. Therefore, as removing the copy of the files, we can reduce the initial setup time so we can save the setup time.

Because of the advantages of the hierarchical structure, we choose one node as a master node and the others are selected as slave nodes. The master node processes step 1 and step 3 and distributes the job to slave nodes, in this paper step2, and manages the slave nodes. The slave nodes processes step 2 if the request comes from master node. Figure 4 briefly shows above explanations.

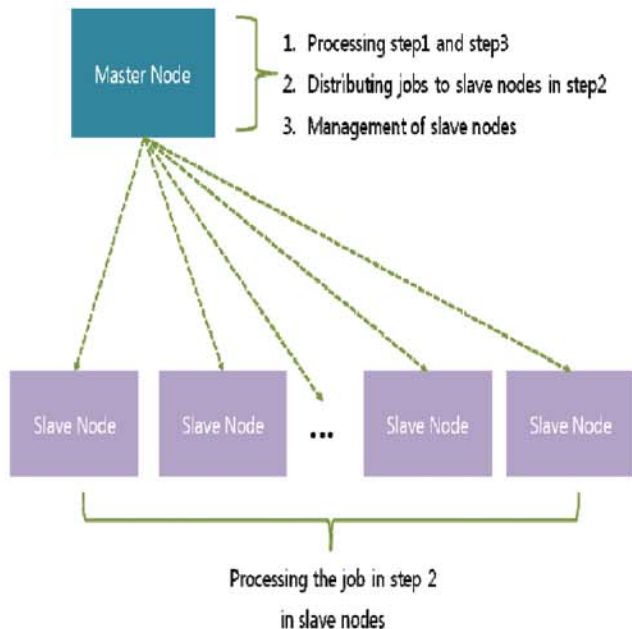


Figure 4. Hierarchical struture of master-slave

Before sending a job, master node check that slave nodes are tied. After checking this, master node sends a message to slave nodes. The message is that the text file which has the noise level values of specific zone is processed to make jpg files in the zone. After receiving this message, the slave node reads the text file that it needs from NFS. From this text file, the slave node executes some functions and completes the request. After finishing the request which come from master node, slave node sends a message to master node. The message is that the request which come from master node is finished. Master node notices that the job was over from the message. The master node checks that jobs are still remained

or not. After checking this, if there are some jobs remained, master node sends a message to slave node. The message is a request as explained above. Slave node repeat the job as it is also explained in the above. However, if there is no job remained, master node and slave nodes which were completed their job wait for other nodes which have not yet completed their jobs. After all of the slave nodes completed their jobs, master node sends a message to all slave nodes. The message is that there is no remained job so all of work is done. The figure 5 briefly shows these explanations.

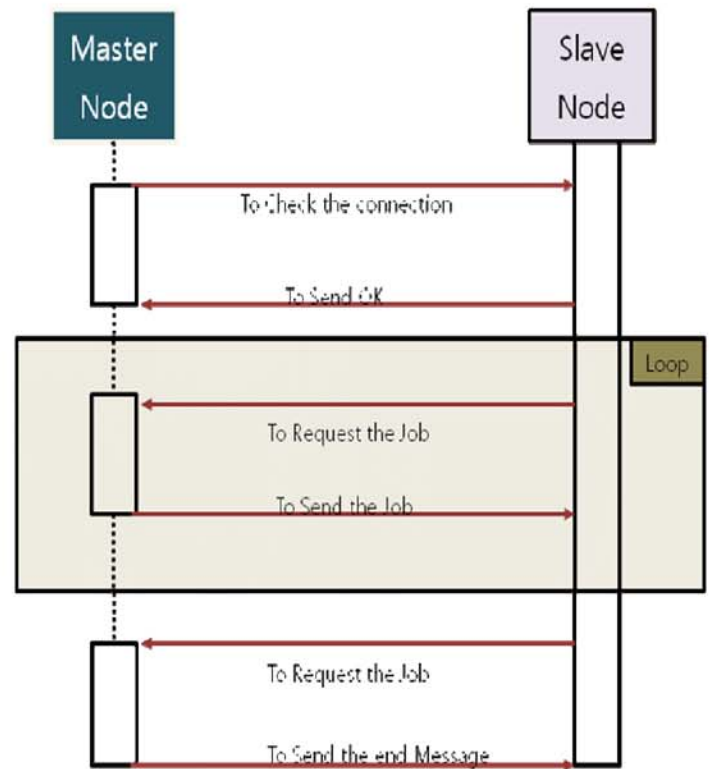


Figure 5. The model of the distributed parallel in master node and slavenodes

IV. EXPERIMENTS

In this chapter, we will show the result of the experiments that is conducted for the distributed and parallel processing model which was implemented using C language. All experiments were conducted on a 4-core Intel i5 2.80GHz machine, with 8GB of physical memory.

The first experiment is as follow. As the input of the 3D GIS urban noise map is increased, the processing time for generating the 3D GIS urban noise map is measured. The environment is that master node is 1 and slave node is 8 and total cores are 33. The result of this experiment is show in figure 6.

In this figure, even though the input is increased, the processing time is increased linearly. This means that the increasing or decreasing in the file size is not changed sharply so the distributed and parallel processing model which was

proposed in chapter 3 is optimized in generating the 3D GIS urban noise map. We can also know that if the input is increased, the processing time can be reduced if there is enough computing power.

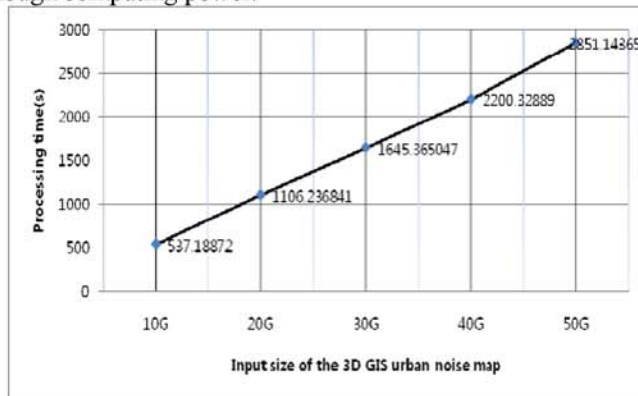


Figure 6. Processing time when generating 3D GIS urban noise map in various input size in 8 slave nodes

The second experiment is as follow. The processing time is measured, as the input file size is increased in generating the 3D GIS urban noise map in a node. The result of this experiment is showed in figure 7.

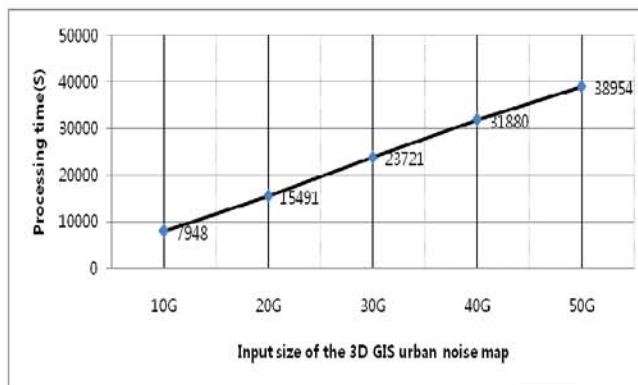


Figure 7. Processing time when generating 3D GIS urban noise map in various input size in a nodes

In this figure, we can know that the processing time is increased linearly even though the input file size is increased. From figure 6 and 7, we can know that the increased processing time is increased linearly, even though the input file size is increased. Therefore, the proposed distributed and parallel model is same performance even though it performs in a node or in nodes. This is means that the proposed distributed and parallel model is fit in generating the 3D GIS urban noise map.

V. CONCLUSION

In this paper, we explain how we provide a lot of computing power in generating the 3D GIS urban noise map. In U-city, the environment information is more important. To

visualize the environment information, many computing resources are needed. So the issue is that how we can provide the computing resources because the processing time is too long. We provide it using MPI and we solve the problem by proposing distributed parallel model.

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. Eui Dong Hwang, Mr. Kyoung Kyu LEE and the staffs of Seoul Grid Center and the members of Smart (Ubiquitous) City Consortium for their contribution to this research.

REFERENCES

- [1] U-City, http://en.wikipedia.org/wiki/Ubiquitous_city
- [2] 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
- [3] Kurakula, V. "A GIS-Based Approach for 3D Noise Modelling Using 3D City Models", MSc proposal, University of Southampton, UK, 2007
- [4] MPI (Message Passing Interface), <http://www.unix.mcs.anl.gov/mapi/>
- [5] S. Oh, I. Lee, S. Tanathong, J. Ko, S. Chang, T. Kim, "Generation of 3D Noise Map using a City Model", 3D Geoinfo, 2008
- [6] The Apache Software Foundation. Hadoop, <http://hadoop.apache.org/core/>
- [7] J. Dean, S. Ghemaway, MapReduce: Simplified Data processing on Large Clusters. Communications of the ACM. January, 2008.