Large-Scale Graph Processing Analysis using Supercomputer Cluster

Alfrido Vildario¹, Fitriyani², Galih Nugraha Nurkahfi³

¹,²,³Department of Computational Science, Telkom University, Bandung, West Java, Indonesia

¹bukan.rido@gmail.com, ²fitriyani@telkomuniversity.ac.id, ³galihnugrahanurkahfi@gmail.com

Abstract. Graph implementation is widely used in various sectors such as automotive, traffic, image processing and many more. They produce graph in large-scale dimensions, causing the processing need long computational time and high specification resources. This research addressed the analysis of implementation large-scale graph using supercomputer cluster. We implemented graph processing by using Breadth-First Search (BFS) algorithm with single destination shortest path problem. Parallel BFS implementation with Message Passing Interface (MPI) used supercomputer cluster at High Performance Computing Laboratory Computational Science Telkom University and Stanford Large Network Dataset Collection. The result showed that the implementation gives the speed up averages more than 30 times and efficiency almost 90%.

Keywords: Large-scale Graph, BFS, Shortest Path, Parallel

1. Introduction

Large web, huge database, social network, and many scientific problems have produced large-scale graphs [1,12,13]. The graph is used in various sectors. Such as GPS, social media, network system, traffic, search engine and many more[5]. This is growing and more widely used. Large-scale graph processing becomes very interesting to analyze and important to solve.

As these problems grow in scale, parallel computing resources are required to meet their computational needs. In the earlier times, supercomputers and mainframe had dominated the world of computing. These are very exotic machines with an integrated technologies inside them, and the scale at which they are built are very high, up to thousands of processors make up a single machine. This is where high performance computing emerges as a solution to provide such needs. Supercomputers are expensive and poor scalability. High performance computing (HPC) offer a scalable, cost-efficient, and easily-maintained resources to conduct a very large computational process. HPC system generally takes one of two paths: cluster computing, and grid computing. Cluster computing is a large number of dedicated computers which are connected in a very close proximity, managed in a centralized way, and have a very high speed interconnect. Among many important graph algorithms, graph traversal and shortest path routing are frequently used to solve other graph problems. For example, shortest path algorithm is an important tool to find between-nodes centrality, a measure of influence in social network.

Graph500 benchmark is the new standard for supercomputer benchmark beside Top500. This benchmark use Breadth-first search (BFS) for searching graph’s vertices. The BFS do searching that start from root vertex and continue searching for their neighbor vertices until find the path [2].
Previous studies had implemented BFS on Graphics Processing Unit (GPU) [10] and processing large graph on cloud [3]. In this research we implemented BFS algorithm on cluster supercomputer, with given problem single destination shortest path. Hope the result in this research is provide knowledge about processing large scale graph with breadth-first search algorithm given problem single destination shortest path and show the performance of the supercomputer cluster.

2. Literature Review

2.1. Graph Adjacency List

Adjacency list is a list from set of couple vertex in graph which between the couple vertex is the edge [4]. The example of the representation is given below:

A Adjacent to B = (A, B)

Given directed graph G as Figure 1 above, the adjacency list form on that graph G is: (1,2), (1,4), (2,4), (3,5), (4,5).

2.2. Breadth-first Search Algorithm

The BFS is searching algorithm for tree or graph. Start from the root (starting vertex) and search their neighbour from that vertices before continue to the another neighbour on another level [2]. This algorithm used in many applications that using in graph modelling for searching purpose, example the shortest path problem. For example, the BFS algorithm for graph G at Figure 1. The searching start form vertices that neighbor with vertex 1, they are vertices 4, 3 and 2. Then the seraching continue to neighbor of vertices 4, 3 and 2. Base on the grap, vertex 4 neighbor with vertex 5 and vertex 3 neighbor with vertex 5. So vertex 5 become the next and last vertex. Final result of traversal graph sequentially are vertices 1, 4, 3, 2, 5.

2.3. Parallel processing

Parallel computation is a technique for doing computation simultaneously by using many computer, processor or thread which are available [7]. Usually it is needed when the problem is large, either the data or the computing process [5,7]. In computer parallel machine, consist of the node for solve the computation. Head node is for give the instruction computation to compute and compute node for execution received from head node.

Message Passing Interface (MPI) is the system message-passing for parallel computation architecture. It support many programming language such as C or C++ [7].

Speedup is how faster execution of that parallel program. Speedup is widely used for show the performance after the resource is increase or changed. Efficiency is a measure of how much of your available power is being used. The speedup and efficiency can be count using $S_p = T_s / T_p$ and $E_p = S_p / p$. Where $S_p$ = Speedup in p processor or thread, $E_p$ = Efficiency in p processor or thread, $T_s$ = Sequential execution time, $T_p$ = Parallel execution time with p processor or thread, $p$ = amount of using processor or thread.
3. Parallel large-scale graph processing
In this research, we implemented BFS algorithm that given problem single destination shortest path on Supercomputer Cluster at HPC Lab Telkom University using MPI.

3.1. Computer Cluster
A computer cluster consists of a set loosely or tightly connected computers that work together with goal that computation can be more faster and efficient rather than single compute. The component of the cluster usually connected to each other through fast local area network (LAN) and each node running its own instance of an operating system [5].

The Supercomputer Cluster of HPC Lab is consists of three nodes, one head node and two compute nodes. Head node specification is inter Xeon E5-606 @2.13 GHz, 24 GB RAM 1333 Mhz DDR3, SSD 250 GB. Each compute node have dual intel Xeon E5-2670v3 12 core (@2.3GHz), 128 GB RAM DD4, SSD 250 GB with accelerato intel Xeon Phi 3120 and Xeon Phi 31 SP. All of node is connected with InfiniBand cable and use FDR-IB SwitchX Mellanox SX6005 Switch. Below is the topology of Supercomputer Cluster.

Figure. 2 : Supercomputer Cluster Topology HPC Lab

3.1.1. Graph Dataset.

Graph dataset adjacency list to simulate BFS algorithm on supercomputer cluster were obtained from Standford Large Network Dataset Collection[11]. The data type is directed and unweighted. Data is contain of set couple between vertex, which the vertex is showed by number. Number on the first column is starting vertex and on the other column is goal vertex. It means the vertex on first column have a directed edge to vertex on second column. For example, vertex 0 in first column and vertex 1 on second column, it means that vertex 0 have and directed edge to vertex 1.

We chose some of Gnutella dataset. They are p2p -Gnutella04, p2p -Gnutella05, p2p -Gnutella08, and p2p -Gnutella09. Gnutella dataset is a sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 2002. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts. Table 1 below describes the Gnutella dataset.

| Data name | Data type | Number of vertices | Number of Edges |
|-----------|-----------|--------------------|-----------------|
|...|...|...|...|

Table 1. Adjacency List Graph Dataset
4. Result and Discussion

BFS algorithm implemented in serial and parallel form. The parallel algorithm was designed as parallel data model. The data distributed to all processors by head node and each compute node doing computation by using this part of data. Then, compute node send the part of result to head node, so the head node compile all part of result become the final result. The serial algorithm implemented on single super computer that consist of 48 threads as explained before. And the parallel algorithm were implemented on super computer cluster with various thread. Table 2 show the speed up.

| Sum of threads | p2p-gnutella08 | p2p-gnutella09 | p2p-gnutella05 | p2p-gnutella04 |
|----------------|----------------|----------------|----------------|----------------|
| 4              | 3.98           | 3.64           | 3.97           | 3.61           |
| 16             | 15.47          | 14.84          | 15.54          | 15.04          |
| 32             | 28.75          | 25.42          | 28.80          | 23.32          |
| 48             | 42.36          | 44.22          | 46.39          | 31.75          |
| 64             | 62.12          | 60.44          | 60.29          | 44.77          |
| 80             | 75.06          | 78.73          | 78.14          | 59.86          |
| 96             | 94.56          | 90.62          | 91.19          | 80.63          |

Base on Table 1, the highest speed up of p2p-gnutella08, p2p-gnutella09, p2p-gnutella05, and p2p-gnutella04 dataset sequentially are 94.56, 90.62, 91.19, and 80.63. The greater number of threads, the higher the speed up. And the greater number of graph edges and vertices, the lower the speed up. But, when we compare the speed up of p2p-gnutella09 and p2p-gnutella05, p-gnutella09 get less speed up than p2p-gnutella05 even the number of vertices is almost the same. It’s because the number of edges from p2p-gnutella09 is less rather than p2p-gnutella05. So the number of edges more influential the computational time than vertices. Data p2p-gnutella04 as the largest graph gives the least speed up See figure 3 to show the comparison of speed up clearly.
The result of the efficiency in all data is quite fluctuating. On data p2p-gnutella04 the best efficiency is on 8 thread with 95.43% and the worst is on 40 thread with 66.03%. On data p2p-gnutella05 and p2p-gnutella08 the best is on 4 thread with 99.33% and 99.57% efficiency and the worst on 8 thread with 88.76% in p2p-gnutella05 and 48 thread with 88.24% in p2p-gnutella08. On data p2p-gnutella09 the best on 96 thread with 94.39% efficiency and the worst is on 8 thread with 74.68%. See Figure 4 below.

Fugure 4 indicates that speed up is not linear with efficiency. Efficiency depend on speed up and number of threads, number of threads refer to cost. So the efficiency means the optimal speed up with make sense cost. For dataset p2p-gnutella08, p2p-gnutella09, p2p-gnutella05, the efficiency approximately 80-99 %, and p2p-gnutella08 is 66-94%. It’s show that, the supercomputer cluster is quitely enough to solve the parallel BFS algorithm. To optimize the use of supercomputer cluster we can use the larger dataset and improve the parallel algorithm.
References

[1] Ajwani D, et al. 2015 Processing Large Graphs: Representations, Storage, Systems and Algorithms, Proceedings of the 24th International Conference on World Wide Web

[2] Beamer S., 2013 Distributed Memory Breadt-First Search revisited: Enabling Bottom-Up Search Berkeley, California. 1-2.

[3] Chen R, et al. 2012 Improving large graph processing on partitioned graphs in the cloud Proceedings of the Third ACM Symposium on Cloud Computing Article No. 3

[4] Bondy, J.A. and Murty, U.S.R. 1976. Graph Theory With Applications Ontario, Canada. Elsevier Science Publishing Co., Inc.

[5] Fitriyani, Izzatul U, 2013 A Research On Parallel Computing Using Ant Colony Optimization Algorithm To Distribute Jobs Among Processors In The System Environment, Proceeding, The 7th International Conference on Information & Communication Technology and Systems (ICTS)

[6] Fujisawa, Katsuki, et al. 2016 Advanced Computing and Optimization Infrastructure for Extremely Large-Scale Graphs on Post Peta-Scale Supercomputers Springer Japan. 1-13.

[7] Grama A, Gupta A, Karypis G, and Kumar V, Introduction to Parallel Computing 2nd Edition, Addison-Wesley, 2003.

[8] Kuck D, (1978). Computers and Computations, Vol 1. John Wiley & Sons, Inc. p. 12.

[9] Leskovec J., Stanford Large Network Dataset Collection. “Internet networks dataset”.

[10] Luo, Lijuan. Wong, Martin. Hwu, Wen-mei. 2010 An effective GPU implementation of breadth-first search. Proceedings of the 47th Design Automation Conference pp 52-55

[11] Malicevic, Jasmina,. Roy, Amithaba, et al. 2014 Scale-up graph processing in the cloud: challenges and solutions, Proceedings of the Fourth International Workshop on Cloud Data and Platforms

[12] Ummah, Izzatul., Fitriyani, 2013 Parallel Computing for Newton Raphson Method (Case Study: Stewart Platform), KNSI 2013

[13] U. Kang, C. Tsonarakis, A. P. Appel, C. Faloutsos, and J. Leskovec. 2008 HADI: Fast diameter estimation and mining in massive graphs with hadoop. Technical Report CMU-ML-08-117, CMU.

[14] U. Kang, C. E. Tsourakakis, and C. Faloutsos 2009 PEGASUS: A peta-scale graph mining system - implementation and observations. ICDM

[15] Murphy, Richard. 2010 Introducing the Graph 500 Albuquerque. 1-2

[16] Sadashiv, Naidila, and SM Dilip Kumar 2011 Cluster, grid and cloud computing: A detailed comparison, The 6th International Conference on Computer Science & Education (ICCSE). IEEE.

[17] Sibaroni, Yuliant. Fitriyani., Nhita, Fhira. 2016.. The Optimal High Performance Computing Infrastructure for Solving High Complexity Problem, Journal of Telkomnika. Vol.14, No.4, December 2016 p 281

[18] Willmore, Frank. 2012 Introduction to Parallel Computing Texas : Texas Advanced Computing Center.