Architectural Design Pemaralelan (Parallel) for Finding the Shortest Path Algorithm Dijkstra

Ponco Rahmad Hadi, Sugiawan Sulastro & Andika Irawan

Dr. Ponco Rahmad Hadi, Sugiawan Sulastro, M.Sc., & Andika Irawan, M.Si.
Department of Mathematics, Adhi Tama Institute of Technology (ITATS), Surabaya
Department of Engineering & Education, Ma Chung University, Malang
Departement of Informatics, Wijaya Kusuma University, Surabaya
Indonesia

Abstract: Pemaralelan architectural design is one of the important stages in parallel computing. This phase is intended that the complexity of computing and communications can be efficient. This paper is a study of architectural design pemaralelan looking Dijkstra Shortest Path Algorithm. This design aspects are reviewed by both the computational complexity analysis of algorithms and communication

Keywords: Design pemaralelan architecture, parallel computing, computational complexity and Shortest Path.

1. INTRODUCTION

Solution to seek Shortest Path by Dijkstra's algorithm has been widely investigated in a sequential manner the experts [McHugh, 1990]. As is known aspect sequential programming run into obstacles include the limitations of the data transfer, and the calculation speed limitations [Lewis, 1992]. With the development of hardware and software technology today, an alternative that has been developed is a problem solving approach to process in parallel. Generally expected to improve performance and efficiency in dealing with a problem. Moreover, the part of the user also wanted a quick problem resolution system and can solve the problem much bigger and complicated [Lewis, 1992], [Kumar, 1994].

Similarly to the solution of problems with the shortest path Dijkstra's algorithm requires the completion of a problem with the parallel process approach, when sequentially solution approach has not been able to provide a quick solution and are faced with a number of much larger vertex and complicated.
In parallel programming involving multiple processors, the next load of problems distributed to various processors. By involving many processors it will have an impact on the communication aspect. Issues important to note is the communication process to keep a low-overhead.

Issues that will have an impact on a variety of issues, among others include setting and synchronization of computer architecture, communication and data transfer processes, and methods of parallels [Lewis, 1992], [Kumar, 1994].

This paper examines the parallels architectural design for the shortest path problem with Dijkstra's algorithm, in order to obtain efficiency and increased speedup compared with a sequential manner.

2. PARALLEL COMPUTING FOR DIJKSTRA SHORTEST PATH ALGORITHM

Glance directed graph with non-negative weights G = (V, E), the shortest path problem with single-source is to find the shortest path from a vertex v ∈ V to all other vertex in V. A shortest path from u to v is the path with minimum weight. Besides finding the shortest path from a single vertex v to each vertex to another, may also to find the shortest path between every pair of points. Formally, each pair shortest path problem is to find the shortest path between all pairs of vertex vI, vjV such that i≠j. For a graph with n vertex, its output is a matrix of nxn size of D = d (i, j) such that dI, j is the cost of the shortest path from vertex to vertex vj vI.

Weights can represent time lines, costs, penalties, damages, or some other quantity cumulatively.

Dijkstra's algorithm to find the single-source shortest paths from a single vertex s, done in increments seeking the shortest path from s to another vertex in G and always choose an edge to a vertex closest enclosed, with complexity $\Theta(n^2)$. The search algorithm is being all-pairs shortest path from one vertex to all other vertex, for all couples with a single-source algorithm executing on each processor, for every vertex v. This algorithm requires complexity $\Theta(n^3)$.

The following program segment shows Sequential Algorithm for Shortest Path s Dijkstra's Single Source [Brassard, 1996]. In this procedure for each vertex u ∈ (V-VT), put l [u], as a minimal cost to reach vertex u from vertex s where the vertex-vertex is in VT.

1. Procedure DIJKSTRA-SINGLE-SOURCE-SP(V,E,w,s)
2. Begin
3. $V_T$={s};
4. For all v ∈ (V-VT) do
5. If(s,v) exists set l[v]=w(s,v);
6. Else set l[v]=∞;
7. While $V_T$≠V do
8. Begin
9. Find a vertex u such that l[v] = min { l[v] | v ∈ (V-VT) };
10. $V_T$=VT∪{u};
11. For all v ∈ (V-VT) do
12. l[v] = min { l[v], l[u] + w(u,v) };
13. Endwhile
14. End DIJKSTRA-SINGLE-SOURCE-SP

3. ARCHITECTURE PARALLEL COMPUTING FOR DIJKSTRA SHORTEST PATH ALGORITHM

According to [Kumar, 1994], architectural models chosen for the implementation of the parallelism must be adjusted with the processor and hardware, in order to create process efficiencies. This should be taken into account, because it is not impossible that this communication problem, will be much more complex than the problem of architecture, and it is often overlooked in the performance calculation. Then, from the aspect of software (operating systems, compilers) can be done dynamically (detecting system itself) or static (programmer must specify the location keparalelannya) [Chaudhuri, 1992].
Furthermore, to obtain optimum results in addition to the design of parallel algorithms that right, must also consider the cost of communication, because sometimes the complexity of communication is higher than the computational complexity, or the time taken to set up data between the processor are higher than the time to process the data manipulation [Quinn, 1987]. It is also worth noting computer architecture, it is important for the process of synchronization between processors and processing.


Parallel to this problem formulation principle is iteration. Each iteration seeking a vertex with minimal achievement of a vertex origin, between the vertex-vertex unvisited connected directly to a vertex already dukanjungi. This achievement allowed to select more than one vertex, if there are more than two choices already visited berhubungan directly with unvisited vertex then been the closest distance. Weighted adjacency matrix partitioned by using block-striped mapping.

Architecture so that each processor is assigned sequentially p n / p columns of a matrix of weighted adjacent matrix, and calculate the value of n / p on the array l.

3.2. Parallel Architecture All-pairs Dijkstra Shortest Path Algorithm

Architectural design for all-pairs problem Dijkstra shortest path can diparalelisasikan in two different ways.

a. Source-partitioned Formulation

Formulation partitioned parallel source of Dijkstra's algorithm using n processors. Each ploking shortest vI shortest path from vertex to all other vertex by executing a sequential algorithm Dijkstra from single-source shortest path. Thus there is no inter-processor communication process.

Thus, the parallel execution of this formulation is \( T_p = \Theta(n^2) \). Communication processors like no, this is a parallel formulation excellence. However, this is not true, because the algorithms used for n processors. Furthermore, the function isoefisiensi for konkurensinya process is \( \Theta(p^3) \), where this is the average functionality isoefisiensi this algorithm. If the number of processors available to resolve this problem is small (that \( n = \Theta(p) \)), then this algorithm has a good performance. However, if the number of processors is greater than n, another algorithm usually adopts this algorithm because its scalability is very small.

b. Source-Parallel Formulation

The main problem with source-partitioned formulation formulation is that if you only use n processors will happen busy at work. Source parallel formulation is the same as the source partitioned formulation, except that the single-source algorithm running on a separate processor subset.

In particular, p (= 16) processors are divided into n (= 4) partitions, each with p / n processors (this formulation is emphasized only if p > n). Each of these n single-source problem solved by the shortest path from n partitions. In other words, the first problem pemaralelan all-pairs shortest path by assigning each vertex to a portion of a collection of processors, and single-source pemaralelan algorithm using p / n processors for menyelaskanannya. The total number of processors used efficiently with this formulation is \( \Theta(n^3) \).

3.3. Evaluation of Computing and Communication Overhead

Assume that the architecture built have p processors with mesh structure, such that \( \sqrt{p} \) is multiplication in \( \sqrt{n} \) mesh structure \( \sqrt{px\sqrt{p}} \) partitioned into n submesh that each measuring \( \sqrt{(P / n) \times \sqrt{P / n}} \).

Furthermore, single-source algorithm executed on every submesh, the parallel execution time is \( T_p = \Theta(n^3/p) + \Theta(\sqrt{(np)}) \), where the computing time \( \Theta(n^3/p) \) and time of communication \( \Theta(\sqrt{np}) \). Medium sequential execution time is \( W = \Theta(n^3) \), then Speedup and Efficiencies are \( \Theta(n^3) / \{ \Theta(n^3/p) + \Theta(\sqrt{np}) \} \), and \( 1 / \{ 1 + \Theta(p^{1.5}/n^{2.5}) \} \).
From these results it appears that the formulation is a minimal fee $p^{1.5}/n^{2.5} = \theta(1)$. Furthermore, these formulations can be used to be $\theta(n^{1.66})$ efficient processor. This shows the case isoefisiensi for communication is $\theta(p^{1.8})$.

As for the parallel architecture Dijkstra formulations for all couples, it seems that in the formulation of the source partitioned no communication, using a processor numbering no more than $n$ processors, and solve problems in time $\theta(n^2)$. As a contrast, the formulations used to $n^{1.66}$ source parallel processors, have time (overhead) communication, and resolve the problem in time $\theta(n^{1.33})$ when used as $n^{1.66}$ processor. Thus, the formulation further exploit the parallel source than the source-partitioned parallels.

4. CONCLUSION

Dijkstra Algorithm Architecture for Single-Source Shortest Paths. This requires each processor is assigned sequentially $p n/p$ columns of a matrix of weighted adjacent matrix, and calculate the value of $n/p$ on the array $l$. In the single algorithm Source-Parallel Formulation executed on an architecture where each submesh, the parallel execution time is the sum of computing time $\theta(n^3/p)$ with a time of communication $\theta(\sqrt{np})$, is $T_p = \theta(n^3/p) + \theta(\sqrt{np})$. It also shows isoefisiensi function for communication is $\theta(p^{1.8})$, with isoefisiensi for concurrent process is $\theta(p^{1.5})$.

Speedup for architectural model of Source-partitioned Formulation is $\theta(n^3)/\theta(n^2)$ and its efficiency is $\theta(1)$, where there is no communication overhead. It is not a parallel formulation excellence, because when using $n$ processors, obtained isoefisiensi function for concurrency process of $\theta(p^3)$.

In two parallel architecture model for all couples, for the formulation of the source partitioned no communication, when using a processor numbering no more than $n$ processors, and solve problems in time $\theta(n^2)$. While the source parallel formulations using up $n^{1.66}$ processors, have time (overhead) communication, and resolve the problem in time $\theta(n^{1.33})$ when used as $n^{1.66}$ processor.

5. BIBLIOGRAPHY