Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 5018213, 8 pages
http://dx.doi.org/10.1155/2016/5018213

Research Article

Research on Optimal Path of Data Migration among Multisupercomputer Centers

Gang Li,[1,2] Qingpu Zhang,[1] and Zhengqian Feng[2]

[1] *Department of Management Science and Engineering, Harbin Institute of Technology, No. 92, Xidazhi Street, Nangang, Harbin, Heilongjiang 150001, China*
[2] *Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Jinan, China*

Correspondence should be addressed to Gang Li; 15905315685@126.com

Received 28 January 2016; Revised 1 April 2016; Accepted 18 May 2016

Academic Editor: Dantong Yu

Data collaboration between supercomputer centers requires a lot of data migration. In order to increase the efficiency of data migration, it is necessary to design optimal path of data transmission among multisupercomputer centers. Based on the situation that the target center which finished receiving data can be regarded as the new source center to migrate data to others, we present a parallel scheme for the data migration among multisupercomputer centers with different interconnection topologies using graph theory analysis and calculations. Finally, we verify that this method is effective via numeric simulation.

1. Introduction

The development level of supercomputing is an important manifestation of the comprehensive power, which has become the strategic highland which every country competes for. A series of challenging problems can be solved in economic construction, social development, technological innovation, industrial upgrading, national security, and other aspects employing supercomputers.

Supercomputing power and storage capacity of supercomputer centers can help enterprises and scientific research institutions more easily carry out the large-scale data-intensive computing tasks and have been increasingly applied in numerous fields, including global climate simulations, global climate modeling, gene mapping, gamma ray bursts, financial investment decisions, economic policy simulations, aerospace monitoring and control, and electronic-very long baseline interferometry [1–7]. In particular, petascale computing can detail the numerical simulation of multiphysics, cosmological evolution, molecular dynamics, and biomolecules [8]. It is important to note that some data-intensive computing tasks such as large hadron collider experiment require supercomputer centers to work cooperatively. However, supercomputer centers located in different regions will inevitably result in data migration between various centers. As for TB or even PB-class data size, the limited bandwidth of supercomputer centers will inevitably give rise to longer data migration time which will prolong the overall task completion time [9, 10]. In the case of multifiles transfer, time delay can also cause the increase of data migration time which also can prolong completion time of the overall task. Therefore, when there is a transmission task between different centers, an optimizing transmission path to make data transmission path the shortest will reduce the time of occupying bandwidth resources and can avoid effects on the migration of other data which will ultimately improve the efficiency of supercomputer centers [11].

Data migration among supercomputer centers is a kind of shortest path problem in graph theory. There are several classic algorithms for this problem, such as Dijkstra, Floyd, Bellman-Ford, and SPFA. However, it must be noted that these algorithms cannot be directly used for searching the shortest path of data migration for the scenario that the source node can transfer data with a plurality of nodes connected to it and that a node which has obtained data from the previous one can also be used as a source node. In computer science, a lot of related researches have been done. Zhu et al. proposed a new method called Ap-proximate

Path Searching (APS) for constructing the broadcast index at mobile clients with soft arrival times to destinations for this problem [12]. Ward and Wiegand researched on complexity results on labeled shortest path problems from wireless routing metrics [13]. Xie et al. found alternative shortest paths in spatial networks [14]. Buchholz and Felko presented a new approach to model weighted graphs with correlated weights at the edges. Such models are meaningful in describing many real world problems like routing in computer networks or finding shortest paths in traffic models under realistic assumptions [15]. Sommer has solved the shortest path queries in static networks [16]. Many other researchers have also done a lot of work on this issue. However, for shortest path of data migration among multisupercomputer centers with different interconnection topologies, the relevant researches and algorithms are very limited. Therefore, we do some research work on this issue.

2. Factors of Data Migration

Assume that there are five supercomputer centers located in different regions (or countries) and they are defined as A, B, C, D, and E, respectively, of which the distributions are shown in Figure 1. Data migration needs to be carried out between each other due to business collaborations. In engineering practice, factors affecting data migration time are mainly physical factors, network link factors, transmission protocols used, and so forth. However, in this paper, we assume that physical factors and transmission protocols are the same, so the main factors affecting migration time are network link factors as follows.

(a) Bandwidth. Network link resource used in supercomputer centers is provided by network operators instead of private networks due to the high cost so that the transmission medium and bandwidth are deterministic.

(b) Delay. Delay includes Processing Delay (T_p), Transmission Delay (T_t), Propagation Delay (T_{ew}), and Queuing Delay (T_q). T_p and T_q are determined by the computing capability and the hardware performance of each node (physical device). T_t is determined by bandwidth. T_{ew} is determined by length of the link [17, 18]. When the migrated data is M bits, the delay is

$$D(M) = T_p + T_{ew} + T_q + T_t = T_p + T_{ew} + T_q + \frac{M}{\mathrm{Bw}}, \qquad (1)$$

where $D(M)$ is the total delay. Bw is bandwidth.

The optimal path of data migration among supercomputer centers can be obtained when

$$T = \min(D(M)), \qquad (2)$$

where T is data migration time.

Supercomputer centers are very advanced in terms of physical devices (such as network cards), so that T_p and T_q can be ignored. Electrical signal transmission speed is approximately equal to the speed of light and the route between supercomputer centers is less than 300000 kilometers, so T_{ew} depending on the two factors is very short and it can also be ignored. In this case, the optimal path is mainly depending on T_t, what is determined by the size of data migrated and the bandwidth.

It is assumed that data migrated is M bits and bandwidth between each other is Bw_i; the data migration time determined by these two parameters is as in Table 1.

Table 1: Data migration time between any two centers (units: hours).

	A	B	C	D	E
A	0	20	240	120	4.8
B	20	0	120	2.4	30
C	240	120	0	12	24
D	120	2.4	12	0	120
E	4.8	30	24	120	0

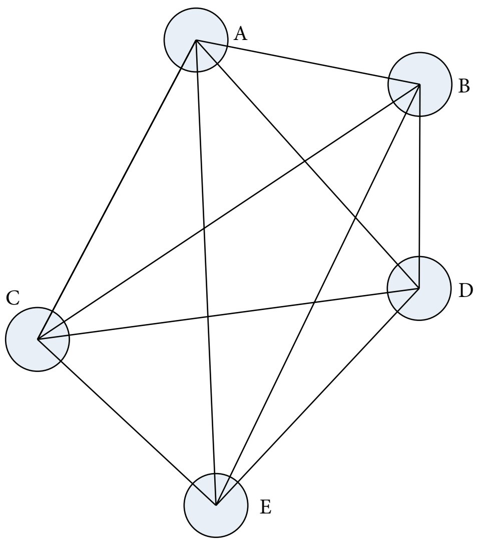


Figure 1: Distribution of supercomputer centers.

3. Optimal Path Planning for Data Migration

3.1. Main Theory. Shortest path problem is a classical algorithm in graph theory, which is intended to find the shortest path between two nodes in the graph. Therefore, this paper carries out relevant research using graph theory.

(1) Graph is a data structure consisting of vertices and edges which is usually expressed as $G(v, e)$.

(2) A value $w(e)$ can be assigned to each edge e in $G(v, e)$. $w(e)$ is called the weight which represents the delay of each link.

(3) Given two vertices in weighted graph, the path with the minimum weight is the shortest path between them [19].

(4) For weighted graph, the weighted adjacency matrix can be expressed as

$$a_{ij}=\begin{cases} w_{v_iv_j}, & \text{if } \left(v_i,v_j\right)\in E \\ 0, & i=j \\ \infty, & \text{if } \left(v_i,v_j\right)\notin E, \end{cases} \tag{3}$$

where v_i and v_j represent the vertices of weighted graph. w_{ij} denotes the weight of edge determined by v_i and v_j. a_{ij} is the value of this weighted adjacency matrix. E denotes the set of all edges.

3.2. Searching Method: Floyd. Floyd algorithm is the easiest shortest path algorithm which can obtain the shortest path between any two nodes in graph. To make it more convenient to discuss the shortest path between supercomputer centers under a variety of scenarios, this paper takes the Floyd algorithm as an example. Constructor method of Floyd is as follows.

Assuming that vertices of weighted graph are $V=\{v_1,v_2,\ldots,v_\nu\}$, for $k=1,2,\ldots,\nu$,

$$\begin{aligned} D^{(k)} &= \left(d_{ij}^{(k)}\right)_{\nu\times\nu} \\ d_{ij}^{(k)} &= \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)}+d_{kj}^{(k-1)}\right\} \\ R^{(k)} &= \left(r_{ij}^{(k)}\right)_{\nu\times\nu}, \end{aligned} \tag{4}$$

where $d_{ij}^{(k)}$ denotes the length of the shortest path in all paths from v_i to v_j. $D^{(k)}$ is distance matrix. $R^{(k)}$ is path matrix and $r_{ij}^{(k)}$ is the node numbered shortest path to pass from v_i to v_j.

$R^{(\nu)}$ can be obtained at the same time with $D^{(\nu)}$ and the shortest path between any nodes can be found from $R^{(\nu)}$ [20, 21].

3.3. Searching Process. According to data migration time between any two centers as assumed in Table 1, the constraint network graph with weight $w(e)$ is shown in Figure 2.

In this way, we can get result after each node $(v_1,v_2,\ldots,v_\nu)$ insertion through iterative process according to the constructor method and constraint network graph with weight $w(e)$ ($D^{(1)}$ represents $k=1$, which means that v_1 is inserted. Similarly, $D^{(2)}$ represents $k=2$, which means that v_2 is inserted. The rest can be done in the same manner. Moreover, "=" in the matrix is the element that has changed after iteration and A, B, C, D, and E are the five nodes). The corresponding distance matrix and path matrix are

$$D^{(0)}=\begin{bmatrix} 0 & 20 & 240 & 120 & 4.8 \\ 20 & 0 & 120 & 2.4 & 30 \\ 240 & 120 & 0 & 12 & 24 \\ 120 & 2.4 & 12 & 0 & 120 \\ 4.8 & 30 & 24 & 120 & 0 \end{bmatrix},$$

$$R^{(0)}=\begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \end{bmatrix},$$

$$D^{(1)}=\begin{bmatrix} 0 & 20 & 240 & 120 & 4.8 \\ 20 & 0 & 120 & 2.4 & \underline{\underline{24.8}} \\ 240 & 120 & 0 & 12 & 24 \\ 120 & 2.4 & 12 & 0 & 120 \\ 4.8 & \underline{\underline{24.8}} & 24 & 120 & 0 \end{bmatrix},$$

$$R^{(1)}=\begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \underline{\underline{\mathrm{A}}} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{A} & \underline{\underline{\mathrm{A}}} & \mathrm{C} & \mathrm{D} & \mathrm{E} \end{bmatrix},$$

$$D^{(2)}=\begin{bmatrix} 0 & 20 & \underline{\underline{140}} & \underline{\underline{22.4}} & 4.8 \\ 20 & 0 & 120 & 2.4 & 24.8 \\ \underline{\underline{140}} & 120 & 0 & 12 & 24 \\ \underline{\underline{22.4}} & 2.4 & 12 & 0 & \underline{\underline{27.2}} \\ 4.8 & 24.8 & 24 & \underline{\underline{27.2}} & 0 \end{bmatrix},$$

$$R^{(2)}=\begin{bmatrix} \mathrm{A} & \mathrm{B} & \underline{\underline{\mathrm{B}}} & \underline{\underline{\mathrm{B}}} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{A} \\ \underline{\underline{\mathrm{B}}} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \underline{\underline{\mathrm{B}}} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \underline{\underline{\mathrm{A}}} \\ \mathrm{A} & \mathrm{A} & \mathrm{C} & \underline{\underline{\mathrm{A}}} & \mathrm{E} \end{bmatrix},$$

$$D^{(3)}=\begin{bmatrix} 0 & 20 & 140 & 22.4 & 4.8 \\ 20 & 0 & 120 & 2.4 & 24.8 \\ 140 & 120 & 0 & 12 & 24 \\ 22.4 & 2.4 & 12 & 0 & 27.2 \\ 4.8 & 24.8 & 24 & 27.2 & 0 \end{bmatrix},$$

$$R^{(3)}=\begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{E} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{A} \\ \mathrm{B} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{E} \\ \mathrm{B} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{A} \\ \mathrm{A} & \mathrm{A} & \mathrm{C} & \mathrm{A} & \mathrm{E} \end{bmatrix},$$

$$D^{(4)} = \begin{bmatrix} 0 & 20 & \underline{\underline{34.4}} & 22.4 & 4.8 \\ 20 & 0 & \underline{\underline{14.4}} & 2.4 & 24.8 \\ \underline{\underline{34.4}} & \underline{\underline{14.4}} & 0 & 12 & 24 \\ 22.4 & 2.4 & 12 & 0 & 27.2 \\ 4.8 & 24.8 & 24 & 27.2 & 0 \end{bmatrix},$$

$$R^{(4)} = \begin{bmatrix} A & B & \underline{\underline{D}} & B & E \\ A & B & \underline{\underline{D}} & D & A \\ \underline{\underline{D}} & \underline{\underline{D}} & C & D & E \\ B & B & C & D & A \\ A & A & C & A & E \end{bmatrix},$$

$$D^{(5)} = \begin{bmatrix} 0 & 20 & \underline{\underline{28.8}} & 22.4 & 4.8 \\ 20 & 0 & 14.4 & 2.4 & 24.8 \\ \underline{\underline{28.8}} & 14.4 & 0 & 12 & 24 \\ 22.4 & 2.4 & 12 & 0 & 27.2 \\ 4.8 & 24.8 & 24 & 27.2 & 0 \end{bmatrix},$$

$$R^{(5)} = \begin{bmatrix} A & B & \underline{\underline{E}} & B & E \\ A & B & D & D & A \\ \underline{\underline{E}} & D & C & D & E \\ B & B & C & D & A \\ A & A & C & A & E \end{bmatrix}. \qquad (5)$$

The shortest path of data migration between any two supercomputer centers can be obtained from the distance matrix. Route of data migration between any two supercomputer centers can be traced from the path matrix. As we can see from the matrix $R^{(5)}$, to find the shortest path from A to D, we first get to the node |B|. Then we search the shortest path from B to D and find it can be direct to D, so the migration path is A $\rightarrow$ B $\rightarrow$ D. We can see from the matrix $D^{(5)}$ that the shortest path is 22.4:

$$D^{(5)} = \begin{bmatrix} 0 & 20 & 28.8 & |22.4| & 4.8 \\ 20 & 0 & 14.4 & 2.4 & 24.8 \\ 28.8 & 14.4 & 0 & 12 & 24 \\ 22.4 & 2.4 & 12 & 0 & 27.2 \\ 4.8 & 24.8 & 24 & 27.2 & 0 \end{bmatrix},$$

$$R^{(5)} = \begin{bmatrix} A & B & E & |B|^1 & E \\ A & B & D & |D|^2 & A \\ E & D & C & D & E \\ B & B & C & D & A \\ A & A & C & A & E \end{bmatrix}. \qquad (6)$$

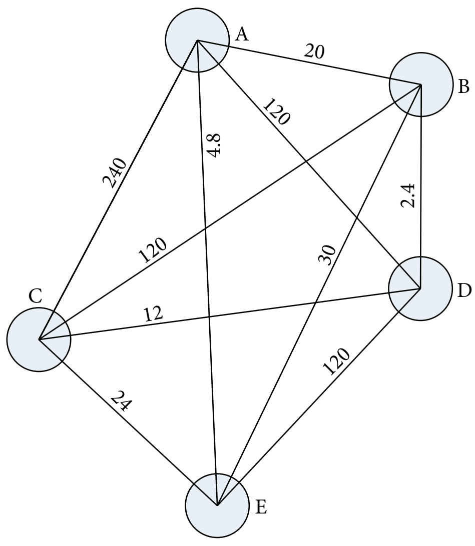


Figure 2: Constraint network graph with weight $w(e)$.

4. Application Model

We can easily obtain the shortest path between any two nodes from the searching process above. That is, if one supercomputer center is fixed, the other supercomputer center which can perform large-scale data-intensive computing tasks with it is also determined. However, the algorithm above cannot be applied to searching the shortest path when the data is distributed from one node to all nodes.

4.1. TSP Model That Does Not Return to the Source Node. Typically, data migration between supercomputer centers is interpreted as selecting a node as the source node and migrating data to other supercomputer centers from this source node and each node can be routed through only once. This scenario is TSP (Traveling Salesman Problem) model that does not return to the source node [22, 23]. TSP is a problem that given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? There has been a lot of research work in this field to solve the optimal data migration path problem based on the TSP model. This paper will no longer investigate for this scenario. Taking Figure 2 as example, if source node is 3 (C), the shortest path is 39.2 and migration path is C $\rightarrow$ D $\rightarrow$ B $\rightarrow$ A $\rightarrow$ E.

4.2. Data Migration in Parallel. The scenario introduced in Section 4.1 is typical, but it does not consider the case when a node is migrating data with another node; it can also migrate data with other nodes at the same time. Moreover, a node which has obtained data from the previous node can also be used as the source node and all the source nodes can migrate data with a plurality of nodes connected to it in parallel. In order to solve this problem well, this paper presents an optimal path algorithm, which is as follows.

Definition 1. $V = \{v_1, v_2, \ldots, v_n\}$; V is the set of nodes.

Table 2: Shortest path and migration path obtained by enumeration method in case 1.

Source point	A	B	C	D	E
Shortest path	28.8	24.8	28.8	27.2	27.2
Migration path	A → E → C A → B → D	B → D → C B → A → E	C → D → B C → E → A	D → B → A → E D → C	E → A → B → D E → C

```
    function[result, routes] = parallel(start, results)
        count = size (results, 1);
        arrived = [start, start, 0];
        while size(arrived) ~= count
            temp = Inf;
            from = 0;
            to = 0;
            for index = 1:count
                if ismember(index, arrived(:, 2)) == 0
                    for index2 = 1:size(arrived, 1)
                        if results(arrived(index2, 2), index) < temp
                                temp =
results(arrived(index2, 2), index);
                                from = arrived (index2, 2);
                                to = index;
                        end
                    end
                end
            end
            results(to,:) = results(to,:) + temp;
            arrived = [arrived;[from to temp]];
        end
        result = max(arrived(:,3));
        routes = [arrived((2:count), 1), arrived((2:count), 2)];
```

Algorithm 1

Definition 2. A^k is the set of nodes reached at step k, and $\overline{A^k}$ denotes the nodes not reached yet, $k \in \{1, 2, \ldots, n-1\}$.

Definition 3. $w^k_{v_i v_j}$ is weight between v_i and v_j at step k.

Definition 4. V^k_s is the source node of step k and V^k_e is the end node of step k.

At step k, we have

$$\begin{aligned} w^{k-1}_{v^k_s v^k_e} &= \min\left\{w^{k-1}_{v^k_i v^k_j}\right\}, \quad v^k_i \in A^{k-1},\ v^k_j \in \overline{A^{k-1}} \\ A^k &= A^{k-1} \cup \left\{v^k_e\right\}, \\ \overline{A^k} &= \overline{A^{k-1}} - \left\{v^k_e\right\} \\ w^k_{v^k_e v^k_j} &= w^{k-1}_{v^k_e v^k_j} + w^{k-1}_{v^k_s v^k_e}, \quad v^k_j \in \overline{A^k}. \end{aligned} \tag{7}$$

Shortest path can be obtained when all nodes have reached ($V = A^{n-1}$). Of course, shortest path and migration path can be obtained through enumeration method when the number of nodes is small. In order to verify the correctness of this algorithm, we compared the results with that of enumeration. Taking Figure 2 as example, the shortest path obtained by enumeration is shown in Table 2 with each node as the source point.

However, as the number of nodes increases, the difficulty of enumeration will increase rapidly. The proposed algorithm can easily obtain the shortest path results, but it is likely to have mistakes. So we developed a function using this algorithm to simplify the calculation which has been tested in MATLAB. Function is as Algorithm 1.

For example, selecting 3 (C) as resource point, the simulation result is shown in Figure S.1 in Supplementary Material available online at http://dx.doi.org/10.1155/2016/5018213.

Result shows that shortest path is 28.8 and data migration path is {C → D → B, C → E → A}. By comparison, the simulation result is the same as the result of enumeration, so the correctness of the algorithm is verified.

It can be concluded that this algorithm is accurate and friendly (machine-executable). Although the time complexity is $O(n^2)$, the algorithm is feasible in solving these complex problems.

4.3. Data Migration among Nonfully Connected Centers. The above discussion is a perfect case, in which all nodes are connected to each other. If the nodes given are not connected

Table 3: Shortest path and migration path obtained by enumeration method in case 2.

Source point	A	B	C	D	E
Shortest path	40.8	60.8	48.8	60.8	36
Migration path	A → E → C → D A → B	B → A → E → C → D	C → E → A → B C → D	D → C → A → E → B	E → A → B E → C → D

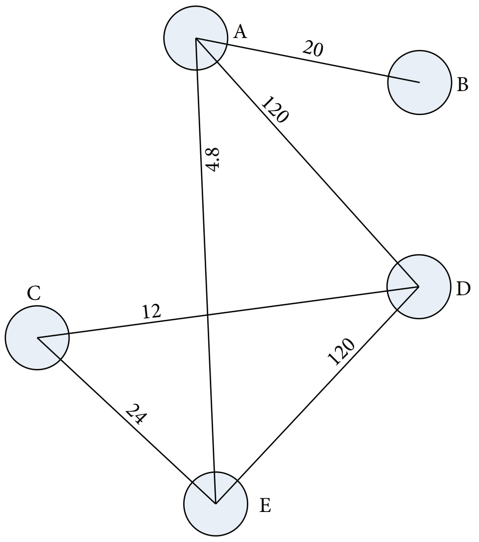


Figure 3: Entire constraint network graph.

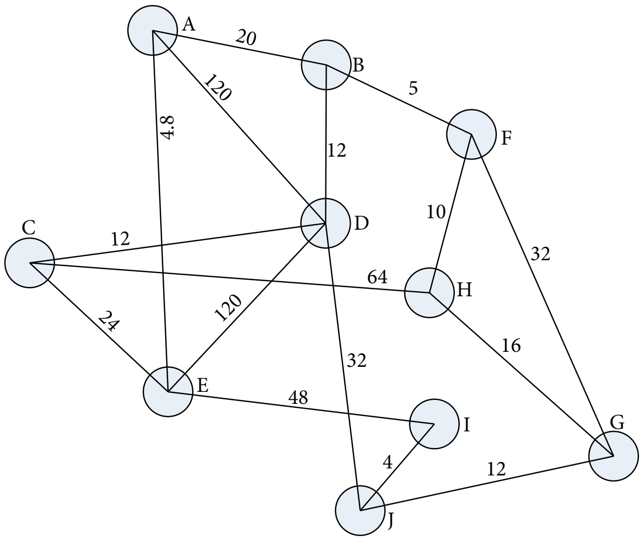


Figure 4: Entire constraint network graph of 10 nodes.

to each other, can this algorithm still be used in searching shortest path? In order to verify the correctness of this algorithm in this case, this paper similarly takes Figure 2 as an example. We assume that links between node A and node C, node B and node C, node B and node D, and node B and node E are removed (of course, you can remove any links between the nodes); then the entire constraint network graph is shown in Figure 3.

First, get the shortest path and the migration path through enumeration method just as in Section 4.2. The results are shown in Table 3.

Second, get the shortest path and the migration path through simulation.

Finally, compare the two sets of results and determine whether the results are identical.

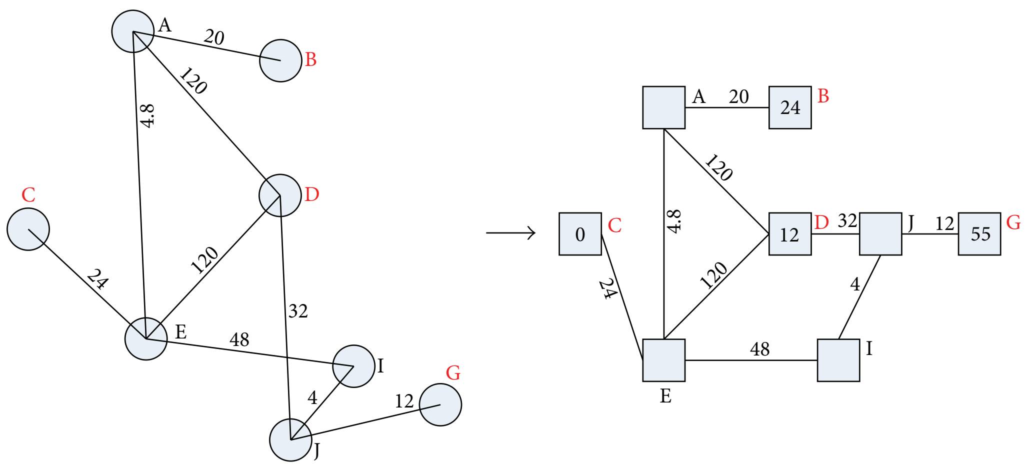
In addition, setting 3 (C) as the resource point, the simulation result is shown in Figure S.2 in Supplementary Material.

Result shows that the shortest path is 48.8 and the data migration path is {C → E → A → B, C → D}. By comparison, the simulation result is the same as the result of enumeration, so this algorithm can be used in searching shortest path in this case.

4.4. Data Migration When Nodes Are Increasing. It is easy to find out the shortest path when the number of nodes is small. However, as the number of nodes increases, what will be the result? In order to verify that this algorithm can be applied to an arbitrary weighted graph, we carried out further experiments. You can randomly determine the number of nodes and the weight between them. For example, we randomly determine ten nodes and the weight between them and the entire constraint network graph just as in Figure 4.

For example, setting 3 (C) as the resource point, the simulation result is shown in Figure S.3 in the Supplementary Material. Result shows that shortest path is 55 and data migration path is {C → D → B → F → H → G, C → E → A, D → J → I}. But what we have to note is that it will be very troublesome if we still use enumeration method directly just as in Sections 4.2 and 4.3 to verify the result, so we borrowed calculating process of Floyd. Searching shortest path in accordance with the process described in Section 3.2 and the result is as follows:

$$D^{(10)} = \begin{bmatrix} 0 & 20.0 & 28.8 & 32.0 & 4.8 & 25.0 & 51.0 & 35.0 & 52.8 & 56.8 \\ 20.0 & 0 & 24.0 & 12.0 & 24.8 & 5.0 & 31.0 & 15.0 & 47.0 & 43.0 \\ 28.8 & 24.0 & 0 & 12.0 & 24.0 & 29.0 & |55.0| & 39.0 & 48.0 & 44.0 \\ 32.0 & 12.0 & 12.0 & 0 & 36.0 & 17.0 & 43.0 & 27.0 & 36.0 & 32.0 \\ 4.8 & 24.8 & 24.0 & 36.0 & 0 & 29.8 & 55.8 & 39.8 & 48.0 & 52.0 \\ 25.0 & 5.0 & 29.0 & 17.0 & 29.8 & 0 & 26.0 & 10.0 & 42.0 & 38.0 \\ 51.0 & 31.0 & 55.0 & 43.0 & 55.8 & 26.0 & 0 & 16.0 & 16.0 & 12.0 \\ 35.0 & 15.0 & 39.0 & 27.0 & 39.8 & 10.0 & 16.0 & 0 & 32.0 & 28.0 \\ 52.8 & 47.0 & 48.0 & 36.0 & 48.0 & 42.0 & 16.0 & 32.0 & 0 & 4.0 \\ 56.8 & 43.0 & 44.0 & 32.0 & 52.0 & 38.0 & 12.0 & 28.0 & 4.0 & 0 \end{bmatrix},$$



(a) Entire constraint network graph-simplified (b) Entire constraint network graph with time

FIGURE 5: Entire constraint network graph removed paths between the nodes having reached.

$$R^{(10)} = \begin{bmatrix} A & B & E & B & E & B & B & B & E & E \\ A & B & D & D & A & F & |F|^{3} & F & F & F \\ E & D & C & D & E & D & |D|^{1} & D & D & D \\ B & B & C & D & C & B & |B|^{2} & B & J & J \\ A & A & C & C & E & A & A & A & I & I \\ B & B & B & B & B & F & |H|^{4} & H & H & H \\ H & H & H & H & H & H & |G|^{5} & H & J & J \\ F & F & F & F & F & F & G & H & G & G \\ E & J & J & J & E & J & J & J & I & J \\ I & G & D & D & I & G & G & G & I & J \end{bmatrix}. \quad (8)$$

Result shows that the shortest path C $\rightarrow$ G has the largest weight and the corresponding data migration path is {C $\rightarrow$ D $\rightarrow$ B $\rightarrow$ F $\rightarrow$ H $\rightarrow$ G}. As the goal is to achieve data migration from node C to all the other nodes, this path {C $\rightarrow$ D $\rightarrow$ B $\rightarrow$ F $\rightarrow$ H $\rightarrow$ G} with the largest weight is certainly one of the paths we need. After determining this path, the entire constraint network graph shown in Figure 4 can be simplified to Figure 5(a) (the paths between the nodes that have already been reached are removed). For simplification of calculation, we changed the form of Figure 5(a) into Figure 5(b) and marked the arrival time of data at nodes (as shown in the box).

We have determined one path above ({C $\rightarrow$ D $\rightarrow$ B $\rightarrow$ F $\rightarrow$ H $\rightarrow$ G}), so there are 4 nodes A, E, I, and J not having been reached. As shown in Figure 5(b), it is easy to calculate the shortest path to the rest of the nodes and the shortest path is {C $\rightarrow$ E $\rightarrow$ A, D $\rightarrow$ J $\rightarrow$ I}. After the two steps mentioned before, each node has been reached, and the shortest data migration path from node C to all the other nodes is {C $\rightarrow$ D $\rightarrow$ B $\rightarrow$ F $\rightarrow$ H $\rightarrow$ G, C $\rightarrow$ E $\rightarrow$ A, D $\rightarrow$ J $\rightarrow$ I}. In addition, the weights of the migration paths are, respectively, {55, 28.8, 48}, so the shortest path is the largest one which is 55. By comparing with the result obtained by the simulation, the two results are the same. So this algorithm can be used in searching the shortest path when data migration is in parallel, and it can be applied to arbitrary weighted graphs.

5. Conclusions and Further Work

Based on graph theory calculations we present a parallel method to migrate data among multisupercomputer centers with different interconnection topologies when supercomputer centers are required to work cooperatively. Specifically, this paper gives a method of node selection, a method of searching the shortest path and migration path. At last, the correctness of this method has also been verified. It is worth mentioning that this method can provide a good reference for data migration between different supercenters. The calculation process is given in this paper. However, how load balancing of data transmission link, large data migration, and multiple and fewer files impact on data migration time and migration path selection is not considered. What is more, some other factors for data transfers across data centers in reality such as the availability of the data and the security issues are also not taken into account. Therefore, according to the actual application of the actual circumstances, we will take into account all influence factors above to explore more accurate optimal path selection next.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The authors are grateful for valuable comments from our instructors and classmates who improve the quality of the paper.

References

[1] J. Gondzio and A. Grothey, "Solving nonlinear financial planning problems with 109 decision variables on massively parallel architectures," in *Computational Finance and Its Applications II*, M. Costantino and C. A. Brebbia, Eds., vol. 43 of *WIT Transactions on Modelling and Simulation*, pp. 95–105, WIT Press, Southampton, UK, 2006.

[2] C. Deissenberg, S. van der Hoog, and H. Dawid, "EURACE: a massively parallel agent-based model of the European economy," *Applied Mathematics and Computation*, vol. 204, no. 2, pp. 541–552, 2008.

[3] R. J. Small, J. Bacmeister, D. Bailey et al., "A new synoptic scale resolving global climate simulation using the Community Earth System Model," *Journal of Advances in Modeling Earth Systems*, vol. 6, no. 4, pp. 1065–1094, 2014.

[4] F. Checconi and F. Petrini, "Massive data analytics: the Graph 500 on IBM Blue Gene/Q," *IBM Journal of Research and Development*, vol. 57, no. 1-2, pp. 1–11, 2013.

[5] K. Kiuchi, Y. Sekiguchi, K. Kyutoku, M. Shibata, K. Taniguchi, and T. Wada, "High resolution magnetohydrodynamic simulation of black hole-neutron star merger: mass ejection and short gamma ray bursts," *Physical Review D—Particles, Fields, Gravitation and Cosmology*, vol. 92, no. 6, Article ID 064034, 2015.

[6] H. Sugawara, H. Yamada, and H. Yamada, "Introduction of supercomputer system for DNA data bank of Japan," *Fujitsu Scientific & Technical Journal*, vol. 44, no. 4, pp. 442–448, 2008.

[7] P. R. Spalart and V. Venkatakrishnan, "On the role and challenges of CFD in the aerospace industry," *Aeronautical Journal*, vol. 120, no. 1223, pp. 209–232, 2016.

[8] President's Information Technology Advisory Committee (PITAC), *Presidents Information Technology Advisory Committee (PITAC). Computational Science: Ensuring Americas Competitiveness*, President's Information Technology Advisory Committee (PITAC), Bethesda, Md, USA, 2005, http://www nitrd.gov/pitac/reports/20050609_computational/computational.pdf.

[9] D. Ziegler, Distributed Peta-Scale Data Transfer, http://www.cs .huji.ac.il/~dhay/IND2011.html.

[10] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[11] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Interdatacenter bulk transfers with netstitcher," in *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11)*, pp. 74–85, Toronto, Canada, August 2011.

[12] C. J. Zhu, K.-Y. Lam, and S. Han, "Approximate path searching for supporting shortest path queries on road networks," *Information Sciences*, vol. 325, pp. 409–428, 2015.

[13] C. B. Ward and N. M. Wiegand, "Complexity results on labeled shortest path problems from wireless routing metrics," *Computer Networks*, vol. 54, no. 2, pp. 208–217, 2010.

[14] K. Xie, K. Deng, S. Shang, X. Zhou, and K. Zheng, "Finding alternative shortest paths in spatial networks," *ACM Transactions on Database Systems*, vol. 37, no. 4, article 29, 2012.

[15] P. Buchholz and I. Felko, "PH-graphs for analyzing shortest path problems with correlated traveling times," *Computers & Operations Research*, vol. 59, pp. 51–65, 2015.

[16] C. Sommer, "Shortest-path queries in static networks," *ACM Computing Surveys*, vol. 46, no. 4, article 45, 2014.

[17] G. Almes, S. Kalidindi, and M. Zekauskas, RFC 2679—A One-Way Delay Metric for IPPM, http://www.ietf.org/rfc/rfc2679.txt.

[18] F. Jianru, *End to End Performance Measurement of Network Delay*, Beijing University of Posts and Telecommunications, Beijing, China, 2006.

[19] Y. Wang, Y. Liu, C.-Z. Zhang, and Z.-P. Li, "Finding the optimal bus-routes based on data mining method," in *Intelligent Computing Theories*, D.-S. Huang, V. Bevilacqua, J. C. Figueroa, and P. Premaratne, Eds., vol. 7995 of *Lecture Notes in Computer Science*, pp. 39–46, 2013.

[20] A. Barenghi, S. Crespi Reghizzi, D. Mandrioli, and M. Pradella, "Parallel parsing of operator precedence grammars," *Information Processing Letters*, vol. 113, no. 7, pp. 245–249, 2013.

[21] H. Djidjev, G. Chapuis, R. Andonov, S. Thulasidasan, and D. Lavenier, "All-Pairs Shortest Path algorithms for planar graph for GPU-accelerated clusters," *Journal of Parallel and Distributed Computing*, vol. 85, pp. 91–103, 2015.

[22] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, Cambridge, Mass, USA, 2004.

[23] I. D. I. D. Ariyasingha and T. G. I. Fernando, "Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem," *Swarm and Evolutionary Computation*, vol. 23, pp. 11–26, 2015.