Towards Modeling Large-Scale Data Flows in a Multidatacenter Computing System With Petri Net

Weijing Song, Lizhe Wang, Rajiv Ranjan, Joanna Kolodziej, and Dan Chen

***Abstract*—There are several use cases that involve the need to transfer data between datacenters when processing large-scale data sets. Data transmission in a multidatacenter computing system has new characteristics that are hard to express and handle using traditional methods. Thus, it is necessary for a new transmission strategy to be developed for users of multidatacenter computing systems. This paper gives a general data model, a network model, and a multidatacenter model to describe the research problem clearly. We propose four new methods to meet the new features of data transmission among datacenters. Based on these models and methods, we propose a new data flow transmission model and analyze its validity.**

***Index Terms*—Large-scale data flow, multidatacenter computing, Petri net.**

I. Introduction

THE multidatacenter infrastructure is promising in that it provides feasible implementation for data-intensive computing. Data-intensive tasks running in multidatacenters may require large amounts of data to be transferred [19]. For example, the user downloads data to the TB level every month from the NASA datacenter [1] for his or her applications. If the computation and data are not available from the same datacenter, it might also be necessary to transfer data in order to complete the task [20]. Another example is the Chinese 863 project of 2013 called the Comprehensive Quantitative Remote Sensing System with Satellite, Aircraft and Receiving Station and Application Demonstration. This would achieve the capability of automatically and rapidly producing more than 80 kinds of multiscale, multiterm quantitative remote sensing at the global level. It can be used for comprehensive applications, such as detecting global food security, forest carbon sinks, cross-border rivers, and the ecological environment. The 80 kinds of quantitative remote sensing algorithm can be deployed for eight different datacenters. Quantitative remote sensing products can be divided into special and common remote-sensing products. Each special remote-sensing product requires several common remote-sensing products; for example, in the production of special products in terms of the country of crop growth, nine different common products are required. Therefore, in order to generate global remote-sensing products, more than one datacenter must be combined to process the data. As a result, large-scale data flow transmission based on multidatacenter architecture is required.

Data movement and storage management toolkits, such as GridFTP, GLobus Online, Data Mover-Lite (DML), and Bulk Data Mover (BDM), have been developed for large-scale data transmission in datacenters [2]. GridFTP allows files to be downloaded in pieces simultaneously from multiple sources, or even in separate parallel streams from the same source, which allows the bandwidth to be used more efficiently. DML supports downloading of a single file by splitting into multiple HTTPS connections for faster downloads. Specifically, partial files are downloaded from each https stream to compose a whole file, or partial files are downloaded from multiple replicas to compose a whole file. BDM may be good at processing different kinds of files, handling extreme variance in file sizes efficiently, and supporting many kinds of transfer protocols. Therefore, data can be transferred in various forms. Data flow models can be supported by those data movement and storage management toolkits.

Based on the works discussed above, some research has focused on data flow analysis. Some of this information has been applied toward standard program optimization [3]. Data flow models are often used to specify the behavior of signal processing and data flow applications as a set of tasks, actors, or processes with data and control dependence relations [4], [5]. The differences between various data flow models can be characterized by their expressive power and the availability of techniques to analyze the correctness of the model and performance properties such as throughput and absence of deadlock. Such analysis is becoming very important for hardware/software codesign of modern streaming systems. However, previous studies have only considered the case of single-path transmission and are not suitable for new features of data transmission using multiple datacenters. Furthermore, the methods of data partition are not taken into account in these models.

Traditional approaches to data transmission, such as single-path transmission, are likely to cause communication blockages on a single path due to the amount of data being transmitted, thereby reducing the transmission efficiency. When multiple

Manuscript received March 12, 2013; revised September 3, 2013; accepted September 22, 2013. The work of L. Wang was supported by the National Natural Science Foundation of China under Grant 61361120098. *(Corresponding authors: L. Wang and D. Chen.)*

W. Song is with the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China.

L. Wang is with the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100094, China, and also with the School of Computer Science, Chinese University of Geosciences, Wuhan 430074, China (e-mail: lizhe.wang@gmail.com).

R. Ranjan is with the ICT, CSIRO, Clayton, Vic. 3169, Australia.

J. Kolodziej is with the Institute of Computer Science, Cracow University of Technology, 31-155 Cracow, Poland.

D. Chen is with the School of Computer Science, Chinese University of Geosciences, Wuhan 430074, China (e-mail: danjj43@gmail.com).

Digital Object Identifier 10.1109/JSYST.2013.2283954

datacenters cooperatively process a data-intensive task, data transmission speed relative to the speed of data processing is the main bottleneck. Therefore, it is necessary to improve the efficiency of data transmission. With the popularity of high-speed Internet and the enhancement of personal computer computing and storage capacity, peer-to-peer (P2P) transmission technology has been rapidly developing. Compared with the conventional transmission mode, this has many significant advantages, specifically its decentralization, scalability, robustness, high cost, privacy protection, load balancing, and so on. The core idea of P2P technology is that each node acts both as a server and as a user enjoying the services provided by other nodes. However, it mainly considers the selection of a single datum from multiple data and does not take advantage of multiple data together.

In order to transform large-scale data rapidly and accurately, a model is needed to describe them in a simple and intuitive way. The graphic description for a data flow such as a directed acyclic graph (DAG) or Petri net is an intuitive approach compared with script-based methods. DAG is easier to use and more intuitive. However, it offers only a limited expression. Moreover, as DAG only has a single node type, data flows through the net cannot be easily modeled [6]. Petri net is a modeling tool used for modeling discrete, dynamic, parallel, and asynchronous systems. Because of the simple graphical description and interpretation ability, Petri net has been widely used for system modeling and performance analysis in recent years. Many studies have already used this method to establish workflow models [6]–[8], [15], [23]; for example, [18] applies colored time Petri nets to express various authorization constraints to be modeled, including role, temporal, cardinality, binding of duty, separation of duty, and role hierarchy constraints.

In this paper, models for large-scale data flows based on multiple datacenters are proposed. From the new features of large-scale data transmission, several methods are considered to improve data transfer rates. According to these models and methods, a large-scale data flow model is presented in this paper. This is validated using reachability tree techniques, and its completeness is demonstrated.

This paper is organized as follows: Section II discusses new data transmission characteristics of datacenters, whereas Section III presents the data model running in multiple datacenters, the network model among datacenters, and the multi-datacenter model. Section IV presents four methods to improve the data transmission rate. The large-scale data flow model is proposed in Section V. Section VI gives a validated analysis of the model. Finally, Section VII concludes our work.

II. Data Transmission Characteristics of Datacenters

Large-scale data processing in datacenters generally follows the near-data principle. That is to say, if the computing and data for an application are not located in the same datacenter, computing is usually moved to the datacenter where the data are stored. However, there are still some cases where data need to be transferred.

A. Cases Requiring Data Transfer

Original Data Are Unevenly Distributed: For remote sensing data, for example, each data-receiving station's received data are generally based on geographical segments, and a range of regional data is stored. Each data-receiving station has overlapping but not identical information. This means that the data distribution is uneven. Original and large-scale data with complete product information and unique characteristics belong to the simple point-to-point transmission mode.

Algorithms in Datacenters Are Distributed Based on Specific Applications: At NASA, for example, the data management centers are divided according to different functions and the generation of different types of products. Therefore, more than two kinds of algorithm processing for a datum are needed, it is necessary to transfer data between datacenters. Such data generally use the multiple-to-single transmission mode.

Multiple Datacenters Require Coprocessing: For example, senior products for remote sensing applications concerning the detection of vegetation and climate change in a certain region in the last ten years have require synergy among datacenters with different data and computing resources. Products that users demand are gradually developed from junior to senior. There is also a need for coprocessing between datacenters.

B. New Features of Datacenters

The data transmission speed is an important factor affecting data processing. In order to improve this speed, it is necessary to understand the data transmission features. The new features of datacenters are described below.

Simple Network Environment and Fixed Bandwidth: Compared with a variety of data transmission systems on the Internet, data transmission between multiple datacenters typically uses a dedicated high-speed bandwidth to transmit the data to ensure the speed and efficiency of data transmission. Therefore, the network transmission environment involves a single and fixed bandwidth.

Finite Data Sources: The data transfer between datacenters mainly from datacenters. Thus, the data source is known and limited.

Transmission Path Is Finite: The network interconnection between multiple datacenters is known, whereas the number of datacenters is limited, so that the transmission path between datacenters is both known and limited. It is possible to recognize the optimal paths from one to another.

C. New Transmission Features

Based on the new transmission features of multidatacenters mentioned above, it is clear that conventional transmission cannot meet the demand for data transmission between datacenters. The traditional data transmission mode, which focuses on the optimization of a single transmission path, does not consider the optimization of multiple transmission paths. For example, when it is necessary to distribute to several datacenters, the conventional transmission strategy creates an equal number

of paths and respectively transmits to the specific datacenter, without full use of the data processing flow.

The traditional data transmission strategy generally involves divided data blocks for large-scale data transmission, but the partitioning strategies do not consider the need for a data processing algorithm. Therefore, the data need to be rechunked during processing, increasing I/O times [22].

The conventional data transmission scheme uses a single path to transmit data with a single destination. This easily leads to the blocking of a single path, as it is not suitable for large-scale data transmission.

Given these considerations, a new data transmission model incorporating multiple datacenters is necessary to cope with the new features of data transmission.

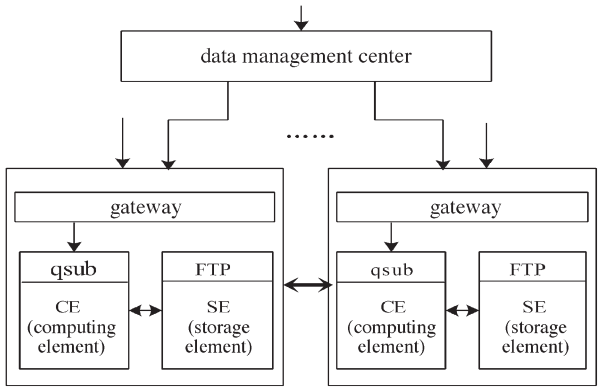

Fig. 1. Multidatacenter architecture.

III. Modeling Conditions

This section presents the multidatacenter model, the data model running in multiple datacenters, and the network model among datacenters. First, we analyze transmission involving multiple datacenters. The multidatacenter system is composed of multiple datacenters, which are peer, hierarchical, unified, or middle deployed. They provide services to the user through a gateway, which coordinates management of each datacenter's resources. Multidatacenter architecture allows users not only to process data through a single datacenter but also to handle large-scale data from multiple datacenters.

A. Multidatacenter Architecture

In order to clarify the multidatacenter paradigm, before discussing the multidatacenter architecture, we give an example of a multidatacenter called the earth system grid (ESG) center. Thus uses new technologies, having successfully established a new capability for serving data from distributed centers. The ESG Federation effectively combines the new P2P architecture with the more traditional client-server model. The infrastructure forms the network of a geographically distributed global federation that is based on standard protocols and application programming interfaces, such as OpenID, Solr, Security Assertion Markup Language (SAML), and Open Archive Initiative (OAI), thereby allowing seamless access to a large and diverse user community [2].

First, the ESG offers gateways that form the main entry points for users to access the data and services. Gateways allow users to browse and search for data, examine detailed metadata, download and subset files, request high-level data products such as analysis and visualization, and register and apply for specific group memberships. The ESG data node was developed to act as the data services back-end to the user interface provided by a gateway. Each ESGF node can offer different services, depending on how it is configured; nodes with different flavors can be scaled differently (for example, to provide increased computational resources or failover search capabilities), and all nodes interact as equals, so there is no single point-of-failure.

In order to meets different kinds and different data transmit requests, many data movement and storage management toolkits are required, for example, GridFTP, GLobus Online, DML, and BDM. GridFTP allow files to download in pieces simultaneously from multiple sources, or even in separate parallel streams from the same source, which allows the bandwidth to be used more efficiently; DML supports downloading of a single file by splitting into multiple HTTPS connections for faster downloads. BDM can efficiently handle extreme variance in file sizes and supports multiple transfer protocols. Therefore, data can be transferred in various forms.

The system enables users to access, analyze, and visualize data using a globally federated collection of networks, computers, and software. It currently provides more than 25 000 users access to more than half a petabyte of climate data (from models and observations) and has been the topic of over 1000 scientific publications.

Summarizing from the above example, we first give the general single datacenter and multidatacenter architectures used in this paper (see Fig. 1).

The general single datacenter is mainly composed of a gateway, compute element (CE), and storage element (SE) [9]. The multidatacenter architecture generally uses a hybrid architecture [21] for workflow enactment based on a centralized control flow and distributed data flow [17]. In detail, there is a management center to manage resource and distribute tasks. Tasks from users or management centers are submitted to the gateway, which distributes the tasks to the CE, and then data processing, which reads from the SE. Finally, the result is returned by the FTP. For large-scale data processing tasks, the data-management center distributes tasks to different datacenters. It controls data transmission and is responsible for returning the processing results. Datacenters interconnect with each other and the main data management center. Therefore, data can be directly transmitted among them.

Consequently, the multidatacenter architecture makes full use of the advantages of P2P and the centralized structure. It not only conveniently manages the overall information but also effectively decreases intermediate data and network traffic.

The general transmission process in the datacenter is as follows. First, data are transmitted to the appropriate datacenter, where they are processed. Then, the processed data are ready for transmission to another datacenter following the processing order (see Fig. 2).

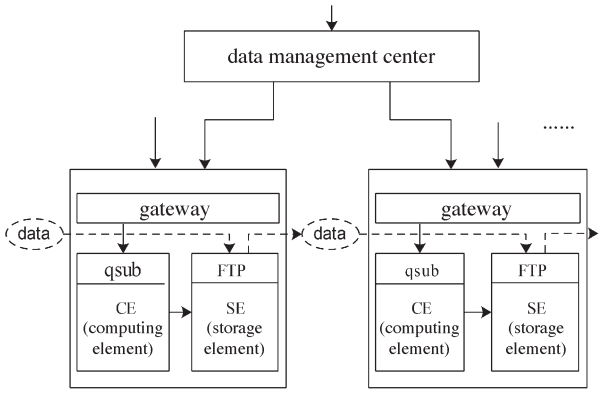

Fig. 2. Data flow in the multidatacenter architecture.

B. Multidatacenter Model

Based on the datacenter and multidatacenter architectures, the major elements of the multidatacenter model can be identified as the number of datacenters, the data stored in each datacenter, and the network among the datacenters. Therefore, the multidatacenter model can be described as follows:

$$M = (C, ID, N).$$

1) $C = \{C_1, C_2, \ldots, C_K\}$ is the collection of datacenters, K is the number of datacenters.
2) $ID = \{ID_{C_1}, ID_{C_2}, \ldots, ID_{C_K}\}$ is the collection of data in datacenters. ID_{C_k} represents the data set in datacenter C_k. ID_{C_k} consists of many data and is expressed as $ID_{C_k} = \{Id_{C_k,1}, Id_{C_k,2}, \ldots, Id_{C_k,N_k}\}$. N_k is the number of data in datacenter C_k. Data $Id_{C_k,n}$ in datacenter C_k can be divided into several data blocks, $Id_{C_k,n,1}, Id_{C_k,n,2}, \ldots, Id_{C_k,n,p}$; p is the number of data blocks.
3) N is the network model among datacenters.

In order to describe the multidatacenter architecture clearly, there are three elements that may need to be modeled. Data represent the major parameter for data transmission. The data information provides the relationship with others and determines whether the data in question need to be transmitted. The data block is the unit of data and the unit of transmission. Network influences how the data are transmitted, as well as the speed of data transmission. Consequently, the data and network models need to be clearly described.

C. Data Model

One of the data features is that data can be divided into many data blocks. A data block not only represents independent data for use but also can be divided again into smaller data blocks. Therefore, the data and data block have the same characteristics. The difference between them is that the data block is usually smaller than the full data set. Generally, before data processing begins, the complete information is called data. The results of data block processing are called data blocks. Thus, the same model could be used to describe data and data blocks. Data may contain or intersect with one another or be independent. In the data transmission process, whether the data need to be transferred depends on whether they are stored in the datacenter. Therefore, the data storage location is an important data attribute.

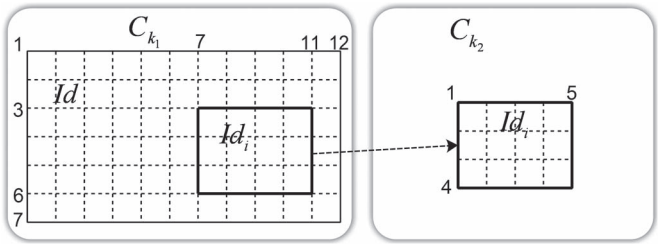

Fig. 3. Data blocking example.

The data model can be described as follows:

$$Id = \{dn, lcs, S, E\}.$$

1) dn is the data name.
2) $lcs = (ls, cs)$ represents the number of lines and columns of data Id.
3) $S = \{C_{k_1}, C_{k_2}, \ldots, C_{k_i}\}$ represents the data storage location. This is a collection of datacenters, which means that data dn are distributed in datacenter $C_{k_1}, C_{k_2}, \ldots, C_{k_i}$. $C = \{C_1, C_2, \ldots, C_n\}$ is a collection of all datacenters; $C_{k_1}, C_{k_2}, \ldots, C_{k_i}$ belong to C. i is the number of datacenters where data Id are stored.
4) $E = \langle E_1, E_2, \ldots, E_m\rangle$ records the division process of generated data dn, whereas $E_i = (Dn_i, F_i)$ reflects the relationship between dn and Dn_i. Data dn are obtained directly or indirectly by the data Dn_i diced. $F_i : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_{i1}(x) \\ f_{i2}(y) \end{pmatrix} (x \in \psi_{i1}, y \in \psi_{i2})$ represents the line and column function of data dn and data Dn_i. $f_{i1}(x)$ is the line function, and $f_{i2}(y)$ is the column function; these may be linear or piecewise functions. ψ_{i1} is the defined domain of $f_{i1}(x)$, whereas ψ_{i2} is the defined domain of $f_{i2}(y)$. Moreover, u and v represent the data line and column variables, respectively, of data Id. If $E_{i+1} = (Dn_{i+1}, F_{i+1})$, then E_i and E_{i+1} meet $E_i < E_{i+1}$. Data Dn_i can be completely represented by Dn_{i+1}. That is to say, data Dn_i are obtained by directly dividing data Dn_{i+1}. Consequently, data dn is the data block that data Dn_1 divides directly into. Data Dn_m are the raw and undivided data. Finally, m is the number of partitions from the original data to data dn.

For example, data $Id_{C_{k_2},n_2}$, which is one data block divided by raw data $Id_{C_{k_1},n_1}$ in datacenter C_{k_1}, needs to be transmitted to datacenter C_{k_2} (see Fig. 3).

Data Id can be represented as follows:

$$Id_{C_{k_1},n_1} = \left\{Id_{C_{k_1},n_1}, \{C_{k_1}\}, (7, 12), \left\langle (Id_{C_{k_1},n_1}, F)\right\rangle\right\}$$

$$F : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} (1 \le x \le 7, 1 \le y \le 12).$$

Data Id_i is denoted as follows:

$$Id_{C_{k_2},n_2} = \left\{Id_{C_{k_2},n_2}, \{C_{k_1}, C_{k_2}\}, (4,5), \left\langle (Id_{C_{k_1},n_1}, F)\right\rangle\right\}$$

$$F : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x-2 \\ y-6 \end{pmatrix} (3 \le x \le 6, 7 \le y \le 11).$$

The data model reflects not only the storage location of data but also the relationship between the data. In the following, we need to define several operations between data, such as contain, intersect, union, and subtract. These are similar to the concepts used in discrete mathematics; however, they have specific meanings in the field of data processing. We first assume there are data $Id_{C_{k_1},i}$ and data $Id_{C_{k_2},j}$.

$Id_{C_{k_1},i} = (dn_i, S_i, lcs_i, E_i)$, $E_i = \langle E_{i1}, E_{i2}, \ldots, E_{im_1}\rangle$, specially $E_{im_1} = (Dn_{im_1}, F_{im_1})$

$$F_{im_1} : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_{im_1 1}(x) \\ f_{im_1 2}(y) \end{pmatrix} (x \in \psi_{im_1 1}, y \in \psi_{im_1 2}).$$

$Id_{C_{k_2},j} = (dn_j, S_j, lcs_j, E_j)$, $E_j = \langle E_{j1}, E_{j2}, \ldots E_{jm_2}\rangle$, specially $E_{jm_2} = (Dn_{jm_2}, F_{jm_2})$

$$F_{jm_2} : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_{jm_2 1}(x) \\ f_{jm_2 2}(y) \end{pmatrix} (x \in \psi_{jm_2 1}, y \in \psi_{jm_2 2}).$$

1) *Contain* means that a piece of data is part of another one, represented by $\subseteq$ or $\supseteq$. For example, $Id_{C_{k_1},i} \supseteq Id_{C_{k_2},j}$ means that data $Id_{C_{k_2},j}$ and data $Id_{C_{k_1},i}$ satisfy $Dn_{im_1} = Dn_{jm_2}$ and $\psi_{jm_2 1} \subseteq \psi_{im_1 1}, \psi_{jm_2 2} \subseteq \psi_{im_1 2}$. If data $Id_{C_{k_1},i}$ and $Id_{C_{k_2},j}$ meet $Id_{C_{k_1},i} \supseteq Id_{C_{k_2},j}$ and $Id_{C_{k_1},i} \subseteq Id_{C_{k_2},j}$, called data $Id_{C_{k_1},i}$ equals to data $Id_{C_{k_2},j}$ represented by $Id_{C_{k_1},i} = Id_{C_{k_2},j}$.
2) *Intersect* refers to the overlapping area between two data, represented by $\cap$. For example, $Id_{C_{k_1},i} \cap Id_{C_{k_2},j} \neq \phi$ means that data $Id_{C_{k_2},j}$ and data $Id_{C_{k_1},i}$ satisfy $Dn_{im_1} = Dn_{jm_2}$, $\psi_{jm_2 1} \cap \psi_{im_1 1} \neq \phi$, and $\psi_{jm_2 2} \cap \psi_{im_1 2} \neq \phi$. If $Id_{C_{k_1},i} \cap Id_{C_{k_2},j} = \phi$, that is to say $Dn_{im_1} \neq Dn_{jm_2}$, $\psi_{jm_2 1} \cap \psi_{im_1 1} = \phi$, or $\psi_{jm_2 2} \cap \psi_{im_1 2} = \phi$, called data $Id_{C_{k_2},j}$ and data $Id_{C_{k_1},i}$ Independent. For remote sensing image processing, the block would have a fixed overlapping border area. Therefore, two independent data refers to no other overlapping part except a fixed overlapping boundary area. The intersection of two data is expressed as $Id_{C_{k_1},i} \cap Id_{C_{k_2},j} = (dn_q, S_q, lcs_q, E_q)$, $S_q = S_i \cup S_j$, $E_{qm} = (Dn_{qm}, F_{qm})$, $F_{qm} : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_{qm1}(x) \\ f_{qm2}(y) \end{pmatrix} (x \in \psi_{qm1}, y \in \psi_{qm2})$, $\psi_{qm1} = \psi_{im_1 1} \cap \psi_{jm_2 1}$, $\psi_{qm2} = \psi_{im_1 2} \cap \psi_{jm_2 2}$.
3) *Union* means that the two data covering all ranges and is represented by $\cup$. The union of data $Id_{C_{k_2},j}$ and data $Id_{C_{k_1},i}$ can be expressed as $Id_{C_{k_1},i} \cup Id_{C_{k_2},j}$. Two data must satisfy $Dn_{im_1} = Dn_{jm_2}$ constraints in order to form a union. If $Id_{C_{k_1},i} \cup Id_{C_{k_2},j}$ can be expressed as $Id_{C_{k_1},i} \cup Id_{C_{k_2},j} = (dn_g, S_g, lcs_g, E_g)$, then $S_g = S_i \cap S_j$, $E_{gm} = (Dn_{gm}, F_{gm})$, $F_{gm} : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_{gm1}(x) \\ f_{gm2}(y) \end{pmatrix} (x \in \psi_{gm1}, y \in \psi_{gm2})$, $\psi_{gm1} = \psi_{im_1 1} \cup \psi_{jm_2 1}$, $\psi_{gm2} = \psi_{im_1 2} \cup \psi_{jm_2 2}$.
4) *Subtract* means that $Id_{C_{k_1},i} \cap Id_{C_{k_2},j}$ is removed from the data $Id_{C_{k_1},i}$, as represented by $Id_{C_{k_1},i} - Id_{C_{k_2},j}$. If $Id_{C_{k_1},i} - Id_{C_{k_2},j} = (dn_h, S_h, lcs_h, E_h)$, then $S_h = S_i$, $E_{hm} = (Dn_{hm}, F_{hm})$, $F_{hm} : \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_{hm1}(x) \\ f_{hm2}(y) \end{pmatrix} (x \in \psi_{hm1}, y \in \psi_{hm2})$, $\psi_{hm1} = \psi_{im_1 1} - \psi_{im_1 1} \cap \psi_{jm_2 1}$, $\psi_{hm2} = \psi_{im_1 2} - \psi_{im_1 2} \cap \psi_{jm_2 2}$.

D. Network Model

Datacenters interconnect using a high-speed bandwidth. Data generally need to be divided when transmitting data great deal of information. For traditional data transmission, the bandwidth is the major parameter influencing the speed of data transmission. Other parameters have little influence that can be ignored, such as network delay and the time interval between continued data blocking. When the data scale grows larger and larger, such parameters cannot be ignored because they have increasing influence on data transmission. Therefore, data block transmission must consider not only the impact of the network bandwidth but also the impact of the network delay, overhead, and the time interval between continued data blocking. Therefore, the network model can be represented as follows:

$$N = \{L, o, g, B\}$$

1) L: an upper bound on the latency or delay when transmitting a small data block from datacenter to another datacenter [10].
2) o: the overhead defined as the length of time wherein a datacenter's processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations [10].
3) g: the gap, defined as the minimum time interval between consecutive message transmissions or consecutive message reception at a processor. The reciprocal of g corresponds to the available per-processor communication bandwidth [10].
4) B: bandwidth, the matrix of bandwidth among datacenters.

IV. Methods of Improving the Data Transmission Rate

This section analyzes approaches to improving the large-scale data transmission rate among datacenters. Considering the new transmission features among datacenters and the limitations of the previous data transfer strategy, we propose several methods to increase the transmission rate among datacenters as follows.

1) *Data reuse*: In the context of maintaining data consistency, data reuse means using a data copy for transmission or processing. This reduces not only the amount of data to be transmitted but also the data transmission time, because the optimal path is selected.

2) *Data subblock*: In the process of data transmission or processing, particularly for large-scale data, data subblocking means dividing data into blocks according to a special data processing algorithm in order to facilitate data transmission and processing. This can achieve the goal of the block transmission and avoid rechunking for data processing.
3) *Data cache*: A data cache is temporarily saved data for read or reread purposes. For example, when data are transmitted through a datacenter, they are temporarily stored in the datacenter and can be used within the validity period. When the number of data transmissions is minimized, the cache increases the number of data copies.
4) *Multipath transmission*: Multipath transmission refers to the selection of multiple paths to transfer data. This generally relates to data blocking. If we divide data into several data blocks and transmit them through multiple paths, this will increase the data transmission rate and make the distribution of data as uniform as possible.

Under the premise that no additional transmission tasks are used, *data reuse* and *data cache* increase the number of data copies as much as possible. The *data subblock* and *multipath transmission* increase the transmission rate from one-path transmission to multipath transmission. This means that parallel data transmission is achieved. At the same time, the *data subblock* strategy matches the algorithm, reducing the number of I/O operations and the processing time. Consequently, data *reuse*, *data cache*, *data subblock*, and *multipath transmission* improve the data transmission and processing efficiency in different ways.

The data transfer scenarios in a multidatacenter strategy may be divided into the following four cases: a single data source $\rightarrow$ single datacenter, a single data source $\rightarrow$ multiple datacenters, multiple data sources $\rightarrow$ single datacenter, and multiple data sources $\rightarrow$ multiple data sources. This paper mainly considers the case of a single data source $\rightarrow$ single datacenter and multiple data sources $\rightarrow$ single datacenter.

Single data source $\rightarrow$ single datacenter means that there is only one datum stored in one datacenter. For this kind of transmission scenario, the methods of improving data transmission efficiency are combined using *data subblocking* and *multipath transmission*.

Multiple data sources $\rightarrow$ single datacenter means that there are several data or data copies stored in several datacenters. For this kind of transmission scenario, the methods of improving data transmission efficiency are combined using *data reuse*, *data subblocking*, and *multipath transmission*.

In the above two cases, the key issues are the data blocking strategy and the multipath selection. First, the data block not only meets the requirements of transmission for the data block size—if the data block is too small, this will increase the number of data being transmitted and the data transfer network latency—but also makes full use of the blocking algorithm for processing data. The data block not only ensures that data can be freely split but also meets the traceability requirements. Another key issue is the basis for the selection of multiple paths. Multipath selection requires consideration of the data transfer rate; at the same time, data processing flow information is fully used, and datacenters require as much of these data as possible. Finally, the most pressing problem is how to match data blocks with multiple paths. There are two issues when it comes to transmitting data blocks: One is how to select more appropriate paths, and the other is how many data would be most appropriate for each path. The optimal transmission rate can be achieved according to the proportion of the bandwidth allocated for the data block size (deduced from the optimal solution by the least squares method). The data block cannot be divided down in an unlimited capacity, because the more data blocks, the more o and g time will be required in data transmission. In order to achieve a better transmission speed, it is necessary to meet the limiting conditions of the minimum data block size and make the ratio of data block size as close to the ratio of the bandwidth as possible.

Taking advantage of the proposed methods to improve the data transfer rate, the optimal data transfer processing strategy, which running on a multidatacenter system, is proposed below.

1) *Checking whether datacenter C_k stores data $Id_{C_k,n}$ or not*. There are three data processing cases. In the first, there are integral data $Id_{C_k,n}$, so data transfer is not necessary. In the second, there are no data $Id_{C_k,n}$, so integral data need to be transferred before step 3 is performed. In the last case, part of the data $Id_{C_k,n}$ is stored, so that the remaining part of the data is transferred before step 2.
2) *Computing* the size of *data that* needs to *be* transmit*ted*, equal to $Id_{C_k,n} - Id_{C_k,n,q}$, then go to step 3.
3) *Retrieving target data* $Id_{C_k,n}$ or $Id_{C_k,n} - Id_{C_k,n,q}$. The retrieval conditions are a subset of the target data and a collection containing the target data.
4) *Retrieving the maximum bandwidth path.* First, it is necessary to statics the datacenter that stores the data sets retrieved in step 3. Then, the optimal path for each datacenter to the destination datacenter C_k is calculated.
5) *Sorting the data*. First, the data are sorted in ascending order according to the amount of data for each datacenter. Then, they are sorted in descending order according to the bandwidth of the datacenter.
6) *Taking data sets' union in the order of step 5*, until the union set contains the target data. Then, the data sets are stored, and the order relations are retained.
7) *Removing the same range of the data block in the order of the path from short to long.*
8) *Processing data in reverse order.* We assume that the number of data is $j = NUM$. The purpose of this step is to decide which data blocks require transfer. Now, we may descript the operations using the following pseudocode:

BEGIN
FOR $j : NUM \rightarrow 1$ DO
  IF $j == NUM$ THEN
    IF $Id_{C_i,j} \cap Id_{C_k,n} \neq \phi$ OR
  $Id_{C_i,j} \cap (Id_{C_k,n} - Id_{C_k,n,q}) \neq \phi$ THEN
      $Id_{C_i,j}$=$Id_{C_k,n}$ OR $Id_{C_i,j}$=$Id_{C_k,n}$−$Id_{C_k,n,q}$
    ENDIF

ENDIF
FOR $v : j \to 1$ DO
IF $Id_{C_i,j} \cap Id_{C_r,v} \neq \phi$ THEN
$Id_{C_i,j} = Id_{C_i,j} \cap Id_{C_k,n} - Id_{C_r,v}$
ENDIF
ENDFOR
ENDFOR
END

9) *Checking the size of the data.* If the size of data does not meet the blocking requirements, then the data will wait for transmission, and step 12 is initiated. If the size of the data meets the requirements, then step 10 is carried out.
10) *Choosing other paths except the optimal path for the data that can be divided.*
11) *Dividing the data.* This corresponds to the select paths based on the transmission path selecting algorithm.
12) *Ready to transfer data.*

V. Large-Scale Data Flow Model

Based on the above data transfer processing, this section proposes the large-scale data flow model. This model is divided into a decision-making model and a transmission model. We first define the Petri net system.

The constraint token colored Petri net system:

$$TCPN = \{P, T; F, D, C, I, O, N, TCon, K, M_0\}.$$

1) P is a finite set of places used to store tokens. Places represent the state of the data in the data flow model. A token stands for data or a data block.
2) T is a finite set of transitions processed for tokens. A transition refers to processing of data or a data block.
3) F is a finite set of arcs connecting places and transitions. $F \subseteq (P \times S) \cup (S \times P)$. This is used to indicate the data processing order.
4) D is a finite set of colors. This indicates data attribute information.
5) C is a finite set of color functions representing the correspondence relationship between the state of the data and data attribute information.
6) I *and* O are the input and output arc functions, respectively. I denotes the attribute information that data should have before the strike transition. O refers to the attribute information that data should have after the strike transition.
7) N is divided into iN and oN. iN represents the constraints on the number of input tokens for transition T to be triggered. oN represents the constraints on the number of output tokens for the transition to be triggered.
8) K *is a set of capacity functions.* $K : P \to N \cup \omega$, $N = \{1, 2, 3, \ldots\}$ and ω denotes infinite. This represents the capacity of the places.
9) $M0$ is the initial token marking. This is the initial condition for the larger-scale data flow model.

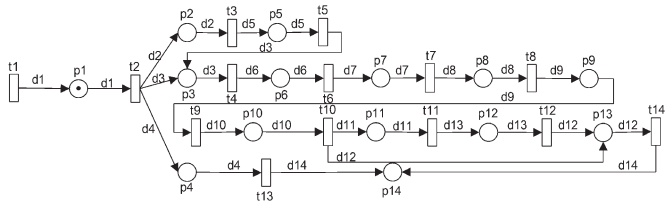

Fig. 4. Petri net of the decision-making model.

A. Decision-Making Model

Based on the constraint token colored Petri net system, we proposed the decision-making model (see Fig. 4).

$$TCPN_1 = \{P, T; F, D, C, I, O, N, TCon, K, M_0\}.$$

1) $P = \{p_i | 1 \leq i \leq 14\}$;
2) $T = \{t_i | 1 \leq i \leq 14\}$, t_1: submit the requirements of data transmission (assuming the data are $Id_{C_k,n}$); t_2 check whether datacenter C_k stores data $Id_{C_k,n}$ or not; t_3: check all the subsets of data $Id_{C_k,n}$ in datacenter C_k; t_4: retrieve the subset of the target data $Id_{C_k,n}$ and collection containing the target data in all datacenters; t_5: remove the part of the data $Id_{C_k,n,j}$ already in the datacenter C_k, $Id_{C_k,n} = Id_{C_k,n} - \{Id_{C_k,n,j} | j \geq 1\}$; t_6: select data based on the optimal path if the data are the same; t_7: sort the data. First, data are sorted according to the ascending order of the amount of data for each datacenter. Then, they are sorted according to the bandwidth descending order of the datacenter; t_8: the union set is obtained for all data from front to back, until $\cup Id_{C_i,j}$ first meets $\cup Id_{C_i,j} \supseteq Id$; t_9: data

$$Id_{C_i,j} = Id_{C_i,j} - \bigcup_{v=1}^{j-1} Id_{C_r,v} - (Id_{C_i,j} - Id) \; (j = N \ldots 2)$$

are obtained; t_{10}: whether the size of data meets the requirement of data blocking is determined; t_{11}: several paths are retrieved based on the data storage location for separable data; t_{12}: the data $Id_{C_i,j}$ are divided and the data blocks made to meet the constraints. $\forall x \; w$, st $Id_{C_i,j,x} \cap Id_{C_i,j,w} = \phi$; t_{13}: the task request is eliminated and the task completed; t_{14}: this is a complex transition representing the data transfer process.
3) $F \subseteq (P \times S) \cup (S \times P)$, $F = \{\langle t_1, p_1\rangle, \langle p_1, t_2\rangle, \langle t_2, p_2\rangle, \langle t_2, p_3\rangle, \langle t_2, p_4\rangle, \langle p_2, t_3\rangle, \langle t_3, p_5\rangle, \langle p_5, t_5\rangle, \langle t_5, p_3\rangle, \langle p_3, t_4\rangle, \langle t_4, p_6\rangle, \langle p_6, t_6\rangle, \langle t_6, p_7\rangle, \langle p_7, t_7\rangle, \langle t_7, p_8\rangle, \langle p_8, t_8\rangle, \langle t_8, p_9\rangle, \langle p_9, t_9\rangle, \langle t_9, p_{10}\rangle, \langle p_{10}, t_{10}\rangle, \langle t_{10}, p_{11}\rangle, \langle t_{10}, p_{13}\rangle, \langle p_{11}, t_{11}\rangle, \langle t_{11}, p_{12}\rangle, \langle p_{12}, t_{12}\rangle, \langle t_{12}, p_{13}\rangle, \langle p_4, t_{14}\rangle, \langle t_{13}, p_{14}\rangle, \langle t_{14}, p_{14}\rangle, \langle p_{13}, t_{14}\rangle\}$;
4) $D = \{d_i | 1 \leq i \leq 14\}$, d_1: tasks submitted by users; d_2: the intersection of the data $Id_{C_k,n}$ and the data stored in datacenter C_k is a proper subset of data $Id_{C_k,n}$; d_3: the intersection of data $Id_{C_k,n}$ and the data stored in datacenter C_k is a null set; d_4: the intersection of data $Id_{C_k,n}$ and the data storing in datacenter C_k is data $Id_{C_k,n}$; d_5: the data removed the intersecting part of data; d_6: data set $\cup Id_{C_i,j}$ including the data that contains

$Id_{C_k,n}$ and the data that $Id_{C_k,n}$ contains; d_7: the data set $\cup Id_{C_i,j}$ with the optimal transmission path; d_8: the data set $\cup Id_{C_i,j}$ after being sorted; d_9: the data set that first contains the data $Id_{C_k,n}$; d_{10}: the data set whose intersection is empty and union; d_{11}: the data set that meets the requirements of data blocking; d_{12}: the data set that is ready to transfer; d_{13}: the data set with several transmission paths and is ready to block; d_{14}: the data set that does not require transfer;

5) $C(p_1)=\{d_1\}$, $C(p_2)=\{d_2\}$, $C(p_3)=\{d_3\}$, $C(p_4)=\{d_4\}$, $C(p_5)=\{d_5\}$, $C(p_6)=\{d_6\}$, $C(p_7)=\{d_7\}$, $C(p_8)=\{d_8\}$, $C(p_9)=\{d_9\}$, $C(p_{10})=\{d_{10}\}$, $C(p_{11})=\{d_{11}\}$, $C(p_{12})=\{d_{13}\}$, $C(p_{13})=\{d_{12}\}$, $C(p_{14})=\{d_{14}\}$;
6) $\forall t\in T\Rightarrow I(t)\rightarrow D, O(t)\rightarrow D$: $I(t_1)=\phi$, $I(t_2)=\{d_1\}$, $I(t_3)=\{d_2\}$, $I(t_4)=\{d_3\}$, $I(t_5)=\{d_5\},\ldots,I(t_{11})=\{d_{11}\}$, $I(t_{12})=\{d_{13}\}$, $I(t_{13})=\{d_4\}$, $I(t_{14})=\{d_{12}\}$; $O(t_1)=\{d_1\}$, $O(t_2)=\{d_2,d_3,d_4\}$, $O(t_3)=\{d_5\}$, $O(t_4)=\{d_6\}$, $O(t_5)=\{d_3\}$, $O(t_6)=\{d_7\},\ldots,$ $O(t_9)=\{d_{10}\}$, $O(t_{10})=\{d_{11},d_{12}\}$, $O(t_{11})=\{d_{13}\}$, $O(t_{12})=\{d_{12}\}, O(t_{13})=\{d_{14}\}, O(t_{14})=\{d_{14}\}$;
7) $\forall t\in T\Rightarrow N(t)\rightarrow\langle n_1,n_2\rangle(n_1,n_2\in Z)$, n_1 is the input constraints for the number of *tokens* before touch off transition, n_2 is output constraints for the number of *tokens* after touch off transition, $N(t_1)=\langle\phi,n\rangle$, $N(t_2)=\langle 1,1\rangle$, $N(t_3)=\langle 1,n\rangle$, $N(t_4)=\langle 1,k\rangle$, $N(t_5)=\langle n,1\rangle$, $N(t_6)=\langle k,k\rangle$, $N(t_7)=\langle k,k\rangle$, $N(t_8)=\langle k,m\rangle$, $N(t_9)=\langle m,p\rangle$, $N(t_{10})=\langle p,p_1+p_2\rangle$, $N(t_{11})=\langle p_1,p_1\rangle$, $N(t_{12})=\langle p_1,q\rangle$, $N(t_{13})=\langle 1,1\rangle$, $N(t_{14})=\langle p_2+q,1\rangle$, $n,k,m,p\geq 1$, $p_1,p_2,q\geq 0$, and $p\leq m\leq k$, $p=p_1+p_2$, $p_1\leq q$;
8) $K(p_1)=K(p_3)=n$, $K(p_2)=K(p_4)=K(p_{14})=1$, $K(p_i)=n\ (i=5\ldots 13)$;
9) $M_0=\{k,0,0,0,0,0,0,0,0,0,0,0,0,0\}$, $1\leq k\leq n$.

B. Transmission Model

In the decision-making model, transition t_{14} is a complex transition and represents the transmission process. When data are transferred to a datacenter, the datacenter not only transfers data but also judges whether to save these data. Therefore, we need give the transmission model.

The following gives the basic process of data processing when the data are transmitted to a datacenter.

1) Storing a copy of data $Id_{C_k,n}$ when it is transmitted to datacenter C_{k_1}.
2) Transferring the data to other datacenters.
3) Judging the relationship between data $Id_{C_k,n}$ and data that are stored in the datacenter. If there are data containing data $Id_{C_k,n}$, then the copy of data $Id_{C_k,n}$ is deleted. If there are data that data $Id_{C_k,n}$ contains, then they are deleted. If all data intersect data $Id_{C_k,n}$ equal to the null set, then data $Id_{C_k,n}$ are stored. If the intersection is not equal to the null set, then the intersection is removed and the last is stored.

According to the above data processing, the transmission model is as shown in Fig. 5.

$$TCPN_2=\{P,T;F,D,C,I,O,N,TCon,K,M_0\}.$$

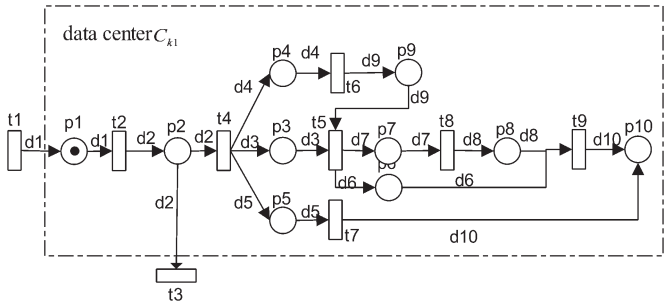

Fig. 5. Petri net transmission model.

1) $P=\{p_i|1\leq i\leq 10\}$;
2) $T=\{t_i|1\leq i\leq 9\}$, t_1: transfer data Id to datacenter C_{k_1}; t_2: produce a copy data of data $Id_{C_k,n}$; t_3: transfer data $Id_{C_k,n}$ from datacenter C_{k_1} to other datacenters; t_4: retrieve the relationship between data $Id_{C_k,n}$ and the data that is stored in datacenter C_{k_1}. There are three relations: $A\cap B=\phi$, $B\supseteq A$, $A-B\neq\phi$; t_5: determine whether the data can be merged; t_6: delete the data set where data $Id_{C_k,n}$ minus them not equal to null set; t_7: delete the copy of data $Id_{C_k,n}$; t_8: merge data that can be merged with data $Id_{C_k,n}$; t_9: archive data;
3) $F\subseteq(P\times S)\cup(S\times P)$, $F=\{\langle t_1,p_1\rangle, \langle p_1,t_2\rangle, \langle t_2,p_2\rangle, \langle p_2,t_3\rangle, \langle p_2,t_4\rangle, \langle p_2,t_4\rangle, \langle t_4,p_3\rangle, \langle t_4,p_4\rangle, \langle t_4,p_5\rangle, \langle p_3,t_5\rangle, \langle p_4,t_6\rangle, \langle p_5,t_7\rangle, \langle t_5,p_6\rangle, \langle t_5,p_7\rangle, \langle t_6,p_9\rangle, \langle t_7,p_{10}\rangle, \langle p_6,t_9\rangle, \langle p_7,t_8\rangle, \langle t_8,p_8\rangle, \langle p_8,t_9\rangle, \langle p_9,t_9\rangle, \langle t_9,p_{10}\rangle\}$;
4) $D=\{d_i|1\leq i\leq 10\}$, d_1: the data transferred to datacenter C_{k_1}; d_2: the data transferred to datacenter C_{k_1} and its copy; d_3: the data set that intersects with data $Id_{C_k,n}$ equal to the null set; d_4: the subset with data $Id_{C_k,n}$ minus the data stored in datacenter C_{k_1} that is not equal to the null set; d_5: the data set that contains data $Id_{C_k,n}$ stored in datacenter C_{k_1}; d_6: the data set that cannot be merged with data $Id_{C_k,n}$; d_7: the data set than can be merged with data $Id_{C_k,n}$; d_8: the data set after merging with data $Id_{C_k,n}$; d_9: the data set with the portion already stored in datacenter C_{k_1} removed; d_{10}: the data set archived in datacenter C_{k_1};
5) $C(p_1)=\{d_1\}$, $C(p_2)=\{d_2\}$, $C(p_3)=\{d_3\}$, $C(p_4)=\{d_4\}$, $C(p_5)=\{d_5\}$ $C(p_6)=\{d_6\}$, $C(p_7)=\{d_7\}$, $C(p_8)=\{d_8\}$, $C(p_9)=\{d_9\}$, $C(p_{10})=\{d_{10}\}$;
6) $\forall t\in T\Rightarrow I(t)\rightarrow D$, $O(t)\rightarrow D$: $I(t_1)=\phi$, $I(t_2)=\{d_1\}$, $I(t_3)=\{d_2\}$, $I(t_4)=\{d_2\}$, $I(t_5)=\{d_3\}$, $I(t_6)=\{d_4\}$, $I(t_7)=\{d_5\}$, $I(t_8)=\{d_7\}$, $I(t_9)=\{d_6,d_8,d_9\}$; $O(t_1)=\{d_1\}$, $O(t_2)=\{d_2\}$, $O(t_3)=\phi$, $O(t_4)=\{d_3,d_4,d_5\}$, $O(t_5)=\{d_7\}$, $O(t_6)=\{d_9\}$, $O(t_7)=\{d_{10}\}$, $O(t_8)=\{d_8\}$, $O(t_9)=\{d_{10}\}$;
7) $N:\forall t\in T\Rightarrow N(t)\rightarrow\langle n_1,n_2\rangle(n_1,n_2\in Z)$, n_1 represents the input constraints for the number of *tokens* before the touch off transition, n_2 denotes the output constraints for the number of *tokens* after the touch off transition, $N(t_1)=\langle\phi,1\rangle$, $N(t_2)=\langle 1,2\rangle$, $N(t_3)=\langle 1,1\rangle$, $N(t_4)=\langle 1,k(k_1+k_2+k_3)\rangle$, $N(t_5)=\langle k_1,k_{11}+k_{12}\rangle$, $N(t_6)=\langle k_2,m\rangle$, $N(t_7)=\langle k_3,k_3\rangle$, $N(t_8)=\langle k_{12},n\rangle$, $N(t_9)=\langle k_{11}+n+m,k_{11}+n+m\rangle$, $k\geq 1$, k_1, k_2,

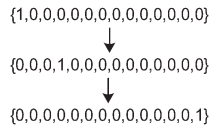

Fig. 6. Reachability tree for case 1.

k_3, k_{11}, k_{12}, n, m all greater than or equal to 0, $k = k_1 + k_2 + k_3$, $k_1 = k_{11} + k_{12}$, $n \leq k_{12}$, $m \leq k_2$;

8) $K(p_i) = n\ (i = 1 \ldots 10), K(p_2) = 2K(p_1)$;
9) $M_0 = \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$.

VI. Validity Analysis

A. Types of Graphics

Validity analysis is necessary for a model based on Petri net to ensure the success of the model in practice. For the decision-making model and the transmission model, we analyze the structure to verify correctness. The main work in structure analysis is reachable analysis. We can build reachability trees for the decision-making model and the transmission model to validate their reachability. The judgment conditions related to whether Petri net is reachable or not are as follows.

1) The initial value only has the root node, i.e., the initial state M0.
2) We assume that x is a leaf node. If any transition cannot occur before ID x, then we call x a proper leaf node; if another node y is on the path from the root node to x, but My = Mx, then x is also called a proper leaf. If all of the leaf nodes in the model are proper leaf nodes, the algorithm is terminated.
3) If there exists a leaf node that is not a proper leaf node in the model, at least one transition has occurred.

For the decision-making model, the initial state is $M_0 = \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$, while the termination status is $\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1\}$. There is a conditional-decision transition in the decision-making model. In order to clearly verify the reachability, we divide three cases to model the reachability tree: 1) The data are stored in the target datacenter, as shown in Fig. 6; 2) the data are not stored in target datacenter at all, as shown in Fig. 7; and 3) part of the data is stored in the target datacenter. For this case, the initial state is described as $M_0 = \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, A1\}$ because of the assumed conditions, as shown in Fig. 8.

The number of tokens in the reachability tree in Fig. 7 satisfies the following conditions:

$$\begin{aligned} &\cup Id_{a1} \supset Id, a1 \geq 1 \\ &\cup Id_{a4} = \cup Id_{a3} = \cup Id_{a2} = \cup Id_{a1} \\ &a2 \geq a3 \geq a4 \geq 1 \\ &a4 = a41 + a42 \\ &a41 \geq 0, a42 \geq 0 \\ &\bigcup_{j}^{a41,a42} Id_j = Id. \end{aligned}$$

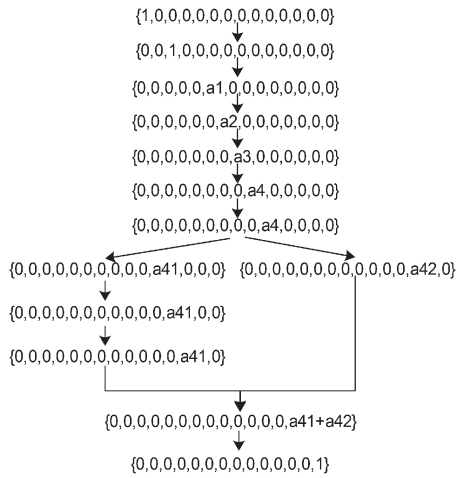

Fig. 7. Reachability tree for case 2.

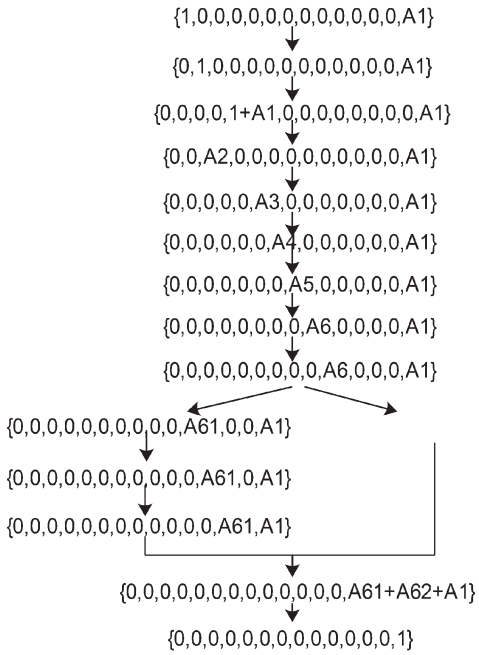

Fig. 8. Reachability tree for case 3.

The number of tokens in the following tree satisfies these conditions:

$$\begin{aligned} &Id, \cup Id_{A1} \subset Id, A1 \geq 1 \\ &\cup Id_{A2} = Id - \cup Id_{A1}, A2 \geq 1 \\ &\cup Id_{A6} = \cup Id_{A5} = \cup Id_{A4} = \cup Id_{A3} = \cup Id_{A2} \\ &A3 \geq A4 \geq A5 \geq A6 \geq 1 \\ &A6 = A61 + A62 \\ &A61 \geq 0, A62 \geq 0 \\ &\bigcup_{j}^{A61,A62,A1} Id_j = Id. \end{aligned}$$

For the data transmission model, the initial state is $M_0 = \{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$, while the termination status

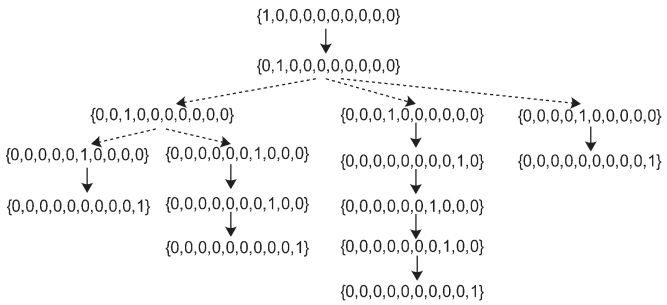

Fig. 9. Reachability tree of the data transmission model.

is $\{0,0,0,0,0,0,0,0,0,0,0,0,0,1\}$. The reachability tree is shown in Fig. 9. The imaginary line shows that only one path can be activated at one time.

VII. Related Work

Data flow management has been extensively studied, and as a result, it is well documented in related literature [16]–[18]. Much of this research has aimed at automating the execution and enhancing the performance of workflows in parallel and distributed systems [24], as well as scheduling and sending jobs to compute nodes that are "close" to the requested data [25], [26]. Some of this research has also utilized Petri nets or DAG to model workflow execution [6]. However, we had noted that in some cases, some data copies and data caches are not utilized.

Kosar and Balman in [16] discussed the limitations of the traditional schedulers in handling the challenging data-scheduling problem of large-scale distributed applications and provided a vision of a new paradigm in data-intensive scheduling. Moreover, they carried out a case study of the Stork data placement scheduler. Importantly, the level parallelism and concurrence increased, and for local area transfers, the transfer rate reached a threshold, after which time they had a negative impact on the transfer rate. However, the transfer rate incurred in the wide area transfers increased as expected. The interesting observation is that the use of a combination of concurrence and parallelism can result in higher performance than using parallelism only or concurrence only. For a Stork with multiple connections or single connections, the transfer speed slightly improved. However, Stork involves end-to-end workflow, improving the performance by increasing parallelism and concurrence levels. Unfortunately, it does not consider the optimization of multiple transmission paths. Transmitting data with a single destination easily leads to the blocking of a single path.

Compared with pure orchestration and pure choreography, Barker *et al.* in [17] introduced a hybrid architecture to workflow enactment based on centralized control flow and distributed data flow; they discussed web-services-based implementation that would decrease intermediate data and network traffic. However, they focused on a single path and did not consider reusing data duplicates and data caches.

Reference [6] modeled scheduling nets and job nets based on Petri net techniques and proposed a hierarchical colored Petri net for a scheduling net designed into four levels according to the granularity of parallel applications. The hierarchical scheduling model made each level scheduling pay attention only to its responsibility and reduced structural complexity. However, it did not consider multipath transmission and did not attempt to use data copies and caches. A general scheduling framework [7] modeled by Petri net was proposed located on the layer of the grid scheduler, but this is only used for independent tasks in a computational grid.

Compared with these models, the traditional data transmission strategy generally uses a single path to transmit data with a single destination. The new model can avoid this effectively and improve data transmission as much as possible.

VIII. Conclusion and Future Work

This paper has discussed a large-scale data flow model in multiple-datacenter architecture. First, we gave a detailed discussion of the new features and challenges in the research problem of transferring data among multiple datacenters. Second, in order to adapt these new features and challenges, we proposed four methods to increase the data transmission rate, specifically data reuse, data subblocks, data cache, and multipath transmission. Third, based on these methods, this paper defined a general multidatacenter. Then, models for data stored in the datacenter and a network between datacenters were proposed. A multiple-datacenter model was also proposed in this paper. Such models consider new features introduced by datacenter, for example, finite data sources and transmission paths. Based on these models, this paper gave a large-scale data flow model consisting of a decision-making model to transfer data and a transmission model. Then, we validate the decision-making and transmission models using reachability tree technologies. Validity analysis showed that the models are valid.

In the future, we will optimize these models further and emphasize the blocking algorithm in relation with the processing algorithm.

References

[1] B. R. Barkstrom, T. H. Hinke, S. Gavali, W. Smith, W. J. Seufzer, C. Hu, and D. E. Cordner, "Distributed generation of NASA earth science data products," *J. Grid Comput.*, vol. 1, no. 2, pp. 101–116, 2003.

[2] D. N. Williams, I. T. Foster, and D. E. Middleton, "DOE SciDAC's earth system grid center for enabling technologies (final report)," Lawrence Livermore Natl. Lab., Livermore, CA, USA, ESG-CET Final Progress Rep. LLNL-TR-501976, Oct. 1, 2006–Sep. 30, 2011.

[3] B. D. Theelen, M. C. W. Geilen, T. Basten, J. P. M. Voeten, S. V. Gheorghita, and S. Stuijk, "A scenario-aware data flow model for combined long-run average and worst-case performance analysis," in *Proc. 4th IEEE/ACM Int. Conf. Formal Methods Models Co-Design*, 2006, pp. 185–194.

[4] S. Sadiq, M. Orlowska, W. Sadiq, and C. Foulger, "Data flow & validation in workflow modeling," in *Proc. ACD*, Dunedin, New Zealand, vol. 27, pp. 207–214, Conferences in Research and Paractice in Information Technology.

[5] H. S. Hong, S. D. Cha, I. Lee, O. Sokolsky, and H. Ural, "Data flow testing as model checking," in *Proc. 25th Int. Conf. Softw. Eng.*, 2003, pp. 232–242.

[6] X. Zhao, C. Wei, M. Lin, X. Feng, and W. Lan, "Parallel application scheduling model based on petri net with changeable structure," in *Advances in Petri Net Theory and Applications.*. Rijeka, Croatia: Sciyo, Sep. 2010, ch. 9, pp. 153, 175.

[7] Z. Hu, R. Hu, W. Gui, J. Chen, and S. Chen, "General scheduling framework in computational grid based on petri net," *J. CSUT*, vol. 12, no. 1, pp. 232–237, Oct. 2005.

[8] Y. Han, C. Jiang, and X. Luo, "Resource scheduling model for grid computing based on sharing synthesis of petri net," in *Proc. 9th Int. Conf. Comput. Supported Cooperative Work Design*, May 2005, pp. 367–372.

[9] W. Zhang, L. Wang, W. Song, and D. Liu, "Towards building a multi-datacenter infrastructure for massive remote sensing image processing," *Concurrency Comput., Pract. Experience*, vol. 25, no. 12, pp. 1798–1812, Aug. 2013. [Online]. Available: http://onlinelibrary.wiley.com

[10] D. Culler, R. Karp, D. Patterson, and A. Sahay, "LogP: Towards a realistic model of parallel computation," in *Proc. 4th ACM SIGPLAN Symposium Principles Pract. Parallel Programm.*, San Diego, CA, USA, May 1993, pp. 1–12.

[11] J. Hidders, N. Kwasnikowska, J. Sroka, J. Tyszkiewicz, and J. Van den Bussche, "DFL: A dataflow language based on Petri nets and nested relational calculus," *Inf. Syst.*, vol. 33, no. 3, pp. 261–284, May 2008.

[12] A. Anjum, "Data intensive & network aware (DIANA) grid scheduling," Ph.D. dissertation, Univ. of the West of England, Bristol, U.K., 2007.

[13] S. Venugopal, "Scheduling distributed data-intensive applications on global grids," Ph.D. dissertation, Univ. of Melbourne, Melbourne, Australia, Jul. 2006.

[14] C. Yuan, *Petri Net Theory and Application*. Beijing, China: Publishing House of Electronics Industry, 2005.

[15] J. Yu and R. Buyya, "A taxonomy of scientific workflow systems for grid computing," *SIGMOD Rec.*, vol. 34, no. 3, pp. 44–49, Sep. 2005.

[16] T. Kosar and M. Balman, "A new paradigm: Data-aware scheduling in grid computing," *FGCS*, vol. 25, no. 4, pp. 406–413, Apr. 2009.

[17] A. Barker, J. B. Weissman, and J. van Hemert, "Orchestrating data-centric workflows," in *Proc. 8th IEEE Int. Symposium CCGRID*, 2008, pp. 210–217.

[18] L. He, C. Huang, K. Duan, K. Li, H. Chen, J. Sun, and S. Jarvis, "Modeling and analyzing the impact of authorization on workflow executions," *FGCS*, vol. 28, no. 8, pp. 1177–1193, Oct. 2012.

[19] Y. Luo and B. Plale, "Hierarchical mapReduce programming model and scheduling algorithms," in *Proc. CCGRID*, 2012, pp. 769–774.

[20] P. Chen, B. Plale, Y.-W. Cheah, D. Ghoshal, S. Jensen, and Y. Luo, "Visualization of network data provenance," in *Proc. HiPC*, 2012, pp. 1–9.

[21] B. Plale, E. C. Withana, C. Herath, K. Chandrasekar, and Y. Luo, "Effectiveness of hybrid workflow systems for computational science," *ICCS*, vol. 9, pp. 508–517, 2012.

[22] X. Shi, H. Jiang, L. He, H. Jin, C. Wang, B. Yu, and X. Chen, "Developing an optimized application hosting framework in Clouds," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1214–1229, Dec. 2013.

[23] L. Liu, L. He, and S. A. Jarvis, "Performance analysis for workflow management systems under role-based authorization control," in *Proc. GPC*, 2012, pp. 323–337.

[24] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Gen. Comput. Syst.*, vol. 25, no. 5, pp. 528–540, May 2009.

[25] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications," in *Proc. 11th IEEE Int. Symposium HPDC, IEEE Comput. Soc.*, Washington, DC, USA, 2002, pp. 352–358.

[26] K. Ranganathan and I. Foster, "Computation scheduling and data replication algorithms for data grids," in *Grid Resource Management: State of the Art and Future Trends*. Norwell, MA, USA: Kluwer, 2004, ch. 22, pp. 359–373.

Weijing Song received the Bachelor's degree from Henan University, Kaifeng, China, in 2010. She is currently working toward the Ph.D. degree with the Institute of Remote Sensing and Digital Earth (RADI), Chinese Academy of Sciences, Beijing, China.

Her research interests include remote sensing image processing infrastructures and earth observation applications.

Lizhe Wang received the B.E. and M.E. degrees from Tsinghua University, Beijing, China, and the Doctor of Eng. degree from University Karlsruhe, Karlsruhe, Germany.

He is a Professor with the Institute of Remote Sensing and Digital Earth (RADI), Chinese Academy of Sciences (CAS), Beijing, and a "ChuTian" Chair Professor with the School of Computer Science, Chinese University of Geosciences (CUG), Wuhan, China. He leads a group at CAS (on HPC and data-intensive computing) and the Scientific Computing Laboratory at CUG (on scientific cloud computing and GPGPU computing).

Dr. Wang is a Fellow of the IET and the British Computer Society.

Rajiv Ranjan received the Bachelor's degree in computer engineering from North Gujarat University, Patan, India, in 2002 and the Ph.D. degree in computer science and software engineering from the University of Melbourne, Melbourne, Australia, in 2009.

He is a Senior Research Scientist, a Julius Fellow, and a Project Leader with the CSIRO Computational Informatics, Canberra, Australia, where he is working on projects related to cloud and service computing. Previously, he was a Senior Research Associate (Lecturer level B) with the School of Computer Science and Engineering, University of New South Wales (UNSW), Sydney, Australia. He is broadly interested in the emerging areas of cloud, grid, and service computing. The main goal of his current research is to advance the fundamental understanding and state of the art of provisioning and delivery of application services in large, heterogeneous, uncertain, and evolving distributed systems (cloud, grids, data center, and web services).

Joanna Kolodziej received the MSD degree in theoretical mathematics and the Ph.D. degree in theoretical computer science from Jagiellonian University, Cracow, Poland.

Since September 1, 2012, she has been an Associate Professor with the Institute of Computer Science, Cracow University of Technology, Cracow. Previously, she was with the Department of Mathematics and Computer Science, University of Bielsko-Biala, Bielsko-Biała, Poland. The current topics of her research include grid and cloud computing, energy effectiveness and secure awareness in large-scale distributed systems, data-intensive computing, and text mining.

Dan Chen received the B.Sc. degree in physics from Wuhan University, Wuhan, China; the M.Eng. degree with the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan; and the M.Eng. and Ph.D. degrees with the School of Computer Engineering, Nanyang Technological University, Singapore.

He is a Professor, the Head of the Department of Network Engineering, and the Director of the Scientific Computing Laboratory with Chinese University of Geosciences, Beijing, China. His research interests include computer modeling and simulation, high-performance computing, and neuroinformatics.