Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 4 (December), pp. 681-700

# A Novel Parallel Transmission Strategy for Data Grid

Q. Ming-Cheng, W. Xiang-Hu, Y. Xiao-Zong

**QU Ming-Cheng, WU Xiang-Hu, Yang Xiao-Zong**
School of Computer Science and Technology, Harbin Institute of Technology
Harbin, Heilongjiang 150001, China
E-mail: qumingcheng@126.com, wuxianghu@hit.edu.com, yangxz@hit.edu.com

**Abstract:** Creation of multi-copies accelerates data transmission and reduces network traffic, but it causes overhead storage and additional network traffic. A variety of parallel transmission algorithms based on GridFTP and multi-copy can be used to accelerate data transmission further, but they can not adapt to a wide range of network, and they can not be used to solve the problems of storage space and network traffic waste. GridTorrent combined with BitTorrent and GridFTP has compatibility with grid and has flexible scalability, but the speed is very slow when there are few peers, to solve this problem multi-copy is needed also. To achieve multiple optimization objectives of storage space saving, suitable for two kinds of application modes(i.e. parallel transfer based on GridFTP and BitTorrent), adaptability for wide range of network and higher performance when there are fewer peers, based on the idea of GridTorrent, a distributed storage model, parallel transfer algorithm and virtual peer strategy are proposed. In experiments the performance is compared among the verification system VPG-Torrent and original parallel transfer algorithm (DCDA) only based on GridfTP & multi-copy and GridTorrent. When the same amount of data is deployed VPG-Torrent has better performance than DCDA, and when there are fewer peers VPG-Torrent also exceed GridTorrent, which prove the effectiveness of VPG-Torrent.
**Keywords:** Data grid, distributed storage model, parallel transmission.

## 1 Introduction

Data Grid is used during data-intensive computing applications to facilitate the efficient use of distributed data resources. It focuses on the management of data in a wide, heterogeneous, and distributed environment; acquisition of data from a variety of heterogeneous data resources, and extraction of useful information from the data source through collaboration and geographical distribution operation [1]. Data grid has been widely used in many fields currently. Such as in scientific computing, physics, biology, astronomy, oceans, atmosphere, manufacturing and so on [2–4], and its desktop operation interface has been implemented well [5–7]. Combining computing grid and data grid for data-intensive computing applications has become a trend now [8–11]. However, to meet time requirement of data-intensive computing applications, how to accelerate transmission is the key problem faced by data grid [12].

In order to improve the user's access speed and reduce the network load effectively, much work has been done in recent years on how to create a reasonable number of copies and locate them [13, 14] efficiently. Data transmission speed is improved and network traffic is reduced. However, these studies overlooked the mass properties of data in data grid, and storage and network overhead were caused by the creation of multiple copies. If there is a huge amount of data, then it is more harmful than good to create multi-copies [15, 16].

Bittorrent is a typical application example of P2P technology. In BT system every peers should upload data when they download. The download speed mainly depends on the number

of simultaneous peers. Now BT is mainly used in file sharing of public community where there are mass accesses. In scientific computing areas there are a few users in general, and computing tasks have strict time requirements in most cases, so Bittorrent reveals some shortcomings.

OverSim is a flexible overlay network simulation framework. All OverSim protocol implementations can be used without code modifications in real networks. Several applications like i3, Scribe, P2PNS, and SimMUD based on these protocols are available. All these implementations can be used for both simulation as well as real world networks. OverSim utilizes the graphical interface of OMNeT++ to display overlay and underlay topology and all network packets. The paper [17, 18] shows that with OverSim simulations of overlay networks with up to 100,000 nodes are feasible. Moreover it provides a fully configurable network topology with realistic bandwidths, packet delays and packet losses.

## 2 Related Works and Ideas

GridFTP provides a striped transmission mode to enable the GridFTP client to download the same data in different data blocks from multiple GridFTP servers in parallel mode, see Figure 1. Based on GridFTP protocol and multi-copy technologies, many parallel transmission algorithms have been developed [19–22], In these studies, a common feature is to divide original data into sub-blocks, then download these sub-blocks from different nodes through some decision-making in parallel mode, when download is finished the sub-blocks are merged into original data.

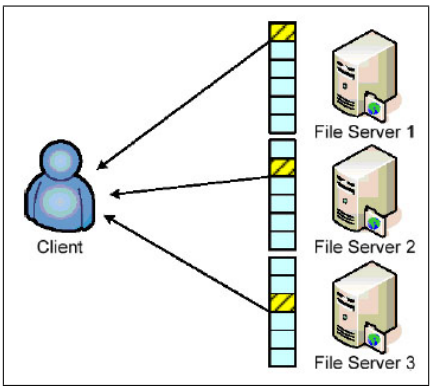


Figure 1: Striped transmission mode of GridFTP

For these parallel transfer algorithms based on client/server(C/S) mode, the performance will decrease dramatically as the amount of access increase, so new copy must be deployed to solve this problem. In spite of this defect, this C/S mode is still necessary for scientific computing community and computing tasks which have higher requirement in reliability, security and some other special data service. But the problems of storage resource waste and network traffic caused by the the deployment of multiple copies can not be neglected. So, how to use a less redundant storage and make full use of GridFTP protocol to improve data transmission speed is a challenge faced by data grid now.

Bittorrent has better adaptability in a wide range of network. But its performance also has much to be desired when there are a few peers or the peers do not allow to upload or upload speed is strictly limited. Integrated the advantages of GridFTP and Bittorrent, some scholars proposed GridTorrent as shown in [23–25].

GridTorrent (see Figure 2) is an implementation of the popular BitTorrent protocol designed to interface and integrate with well-defined and deployed Data Grid components and protocols (e.g. GridFTP, RLS). Just like BitTorrent, GridTorrent is based on peer-to-peer technique, that

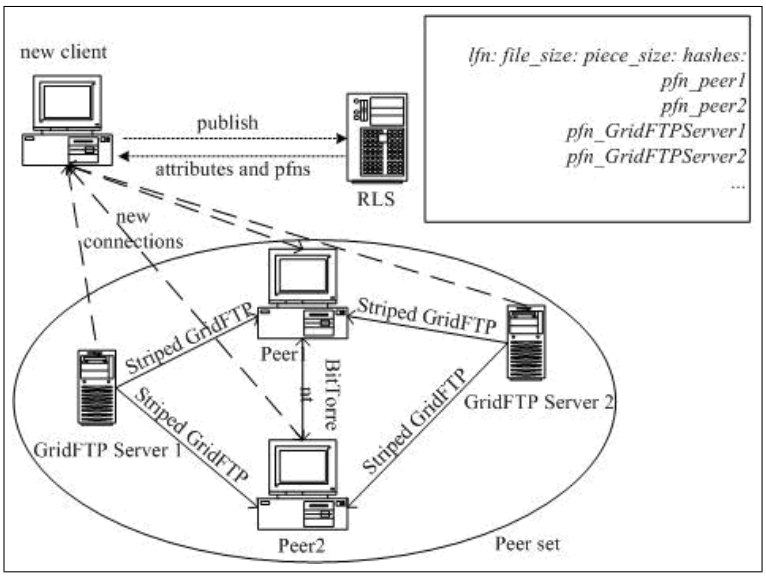


Figure 2: GridTorrent

allow clients to download files from multiple sources while uploading them to other users at the same time, rather than obtaining them from a central server. By dividing files into fragments, GridTorrent can combine the best out of the two protocols [25].

The data in Bittorrent community is generally obtained from other people's works, like music or movie file, whereas every scientific data is generated in scientific community. Therefore, scientific data are more sensitive than data used in Bittorrent community. So the number of concurrent peers will be very limited.

From Figure 2, when the sum of peers is small, the performance of GridTorrent will be degraded as GridFTP. So like other parallel transfer algorithms based on GridFTP and multi-copies, a number of copies should be deployed to enhance the transmission speed. Accordingly storage space and network traffic waste are serious.

Bittorrent algorithms do not rely on global data storage knowledge to schedule. Because the number of participating peers, network speed, peer-owned data blocks are dynamically changing. So if we directly divide overall data into sub-blocks of the same size and then deploy them to some servers, the parallel transmission capacity of the servers can not be maximized. And for the previous parallel transmission algorithms, it will inevitably lead to some of the nodes complete their download task first, i.e., all the data stored by the nodes have been downloaded. It can not guarantee each task continuously download from start to finish and all the tasks finish at or tend to the same time.

This paper is based on the idea of GridTorrent, and study how to achieve higher speed using less storage when the number of participating peers is small or P2P is not allowed, so that speed does not depend on the number of concurrent peers. To solve the problem from three levels: first, P2P is not allowed (only download from GridFTP servers); second, P2P is allowed, but the number of peers is small; third, large-scale peer transmission. Therefore, a data grid distributed storage model which uses less data redundancy and can ensure the reliability of data is proposed in this paper. A parallel scheduler and a parallel transmission algorithm based on the model and GridTorrent are thus presented.

Main contributions: a distributed storage model is put forward, the model can ensure data reliability using less storage. A scheduler is presented based on model, and further a parallel transmission algorithm (PTABM) is put forward based on model and scheduler. Comparative

experiments are carried, our algorithm PTABM achieved the same performance compared with DCDA, meanwhile the strategy proposed in this paper has absolute advantage in storage space, and also network load can be reduced greatly during copy deployment and migration.

Virtual peer(Vpeer) is presented in section 4.4 based on PTABM and the idea of GridTorrent. During the scheduling, Vpeer is regarded as a general peer, only the transmission from Vpeer is based on PTABM. This system is called VPG-Torrent, and it can adapt to the situation of large-scale networks.

Compared with GridTorrent, when the number of peers is small (dozens of peers), VPG-Torrent make the average speed of every peers be higher, as the number of peers increases gradually, the performance of them gets closer gradually, it is because that BitTorrent accounts for a dominant role when large-scale peers exist. The outcome achieved by this paper is:Using less storage space achieved objective of rapid transmission; and the performance is not affected by the sum of concurrent peers; VPG-Torrent can met two kinds of application modes, i.e., one is parallel transmission only based on GridFTP, the other one is BitTorrent service in large-scale network. This is the lack of previous studies.

The experiments include two parts: parallel transmission only based on GridFTP, simulation of VPG-Torrent.

(1)The first part is to verify the performance of VPG-Torrent when BitTorrent is not allowed. We implemented a Data Dispatcher, RLS server and PTABM, and built environment platform composed of 7 computers. Good results were achieved compared with previous parallel transmission algorithm (DCDA).

(2)The second part is to verify the performance of VPG-Torrent when BitTorrent is allowed, and compare with GridTorrent.

In order to simulate large-scale concurrent access, oversim is used in simulation experiment, and also good results were achieved.

In following chapters, firstly a distributed storage model is put forward, then a scheduler based on model is proposed, further a parallel transmission algorithm is given, and then expounded the verification system modules and work process, at last experiments and analysis are given.

## 3 Distributed Storage Model

In this section we will put forward a block-devision and block-storage strategy (called distributed storage model), so some necessary definitions will be given first. Definitions 1-6 are given to explain how to divide data file and how to storage the blocks into grid nodes. Definitions 7-10 are used to show the properties of distributed storage model

**Definition 1.** (metasum partition and metadata) Let M represent the total amount of data. Divide the overall data into sub-blocks of equal size, so that metablks=k(k-1)*metasum, and k is the number of copies, and metablks is divided shares. firstly the data is divided into k(k-1) shares, then each share is divided into metasum shares, and metasum is a variable parameter. So let the amount of data for each share be metadata, and can be expressed as:

$$metadata = \frac{M}{k(k-1) * metasum} \tag{1}$$

**Definition 2.** (local data) The k(k-1)*metasum shares of data defined in definition 1 are evenly distributed among k nodes. So each node contains (k-1)*metasum shares of data. These data is called local data $LND_i$ of node $N_i$.

**Definition 3.** (local data virtual group) The local data of $N_i$ is classified into some virtual groups.The metasum shares of data is classified as one group, and the classified groups are numbered uniquely within nodes. From definition 2, we can see that the local data of each node can be classified into (k-1) virtual groups. Let the virtual group be $G_i{}^j$ ($0 \le i \le$ k-1, $0 \le j \le$ k-2). The sum of data blocks that one group contains is metasum. Let $G_i$ represent all the vitural groups that node $N_i$ contains, and G represents the current vitural group.

**Definition 4.** (Remaining Node Set) After excluding node $N_i$, all remainder grid nodes is called remaining node set: $\overline{N}_i^e (0 \le e \le k-2)$ *and* $\bigcup_{y=0}^{k-2} \overline{N}_i^y = \bigcup_{j=0, j\neq i}^{k-1} N_j$ , $\overline{N}_i^y = N_j$, $\begin{cases} y = j & \text{if } j < i \\ y = j-1 & \text{if } j > i \end{cases}$

**Definition 5.** (Cross Storage) Croup $G_i$ of $N_i$ is distributed into $\overline{N}_i^e$, that is $G_i \to \overline{N}_i^e$ , and $\left( \bigcup_{r=e}^{w} (G_i^r \to \overline{N}_i^e) \right) \Big|_{e=0}^{k-2} \{w = (e + (p-1)) \bmod k, p \le k-1\}$ , where p is the specified constant, and symbol $\to$ means 'storage into'. This is why such storage rule is called cross storage.

**Definition 6.** (Distributed storage Model) All the data $AND_i$ that node $N_i$ contains includes local data $LND_i$ and cross storage data $OND_i$. From the definitions 3 to 5 above: $AND_i$=($LND_i$)∪

($OND_i$)= $\left( \bigcup_{j=0}^{k-2} G_i^j \right) \cup \left( \bigcup_{(a=0, a\neq i)}^{k-1} \left( \bigcup_{r=e}^{w} G_a^r \right) \right) \begin{cases} e = i-1 (a < i) \\ e = i \quad (a > i) \end{cases}$ &

$\begin{cases} w = (e + (p-1)) \bmod k \\ p \le k-1 \end{cases}$ . p is a specified constant, so such storage mode is called Distributed storage Model. (Note: OND represents local data and $OND_i$ represents local data of node $N_i$ )

**Theorem 1.** *(p integrity) If data storage meets Distributed Storage Model, when there are arbitrary p nodes are not available, the merger of data at the remaining k-p nodes is still equivalent to M, that is, the data is complete, this property is called p integrity.*

**Proof:** For an arbitrary local data $G_i^x$, $i \in (0, k-1)$, $x \in (0, k-2)$, *where* $G_i^x$ represents arbitrary one local virtual group of one arbitrary node $N_i$.

From definition 5, we can know that $e \in (x-(p-1), k-2)$, $r \in (e, w)$, and w=(e+(p-1) mod k), p≤k-1. So from e to w there are p numbers. Retrieve a subset e and $e \in (x-(p-1), x) \subset (0, k-2)$.Because of each value of e, r starts from e will get p values.

When e varies from x-(p-1) to x, r is in the range of { (x - (p - 1),x),...,(x,x + (p - 1))} .The 'x' is repeated p times because e has undergone P changes, that is,$G_i^x$ has been stored into node $\overline{N}_i^e$ for p times. Here e∈(x-(p-1),x).

Adding the node that virtual group $G_i^x$ lies in, there are p +1 nodes where store $G_i^x$ Therefore, if there are arbitrary P nodes are unavailable, a node store $G_i^x$ is available. As $G_i^x$ is arbitrary, so all the virtual groups of all the nodes meet the above derivation. The proof is completed. □

**Lemma 1.** *If the data storage met the Distributed storage Model, then the overall amount of stored data can be expressed as: k(1+p)(k-1)*metasum.*

**Proof:** Derivated from definition 6:

$$\begin{aligned} \sum_{i=0}^{k-1} AND_i &= \sum_{i=0}^{k-1} ((k-1) * metasum|_{LND_i} + (1+p) * metasum * (k-1)|_{OND_i}) \\ &= k(k-1)(1+p) * metasum \end{aligned}$$

The proof is completed. □

**Definition 7.** (Storage Space Usage Rate:SSUR) The total amount of data that stored by Distributed storage Model divided by the total amount of data that stored by k-complete copies is defined as SSUR. From definition 1 and lemma 1:

$$SSUR = \frac{k(k-1)(1+p) * metasum}{k * k(k-1) * metasum} = \frac{1+p}{k} \tag{2}$$

**Definition 8.** (Limit Speed Ratio: Vratio) Let Vratio=$V_i/V_j$ represents the limit ratio of the amount of data that downloaded from nodes $N_i$ and $N_j$ seperately. Then:

$$\frac{1}{(k-1)(1+p) * metasum} \leq V_{ratio} \leq (k-1)(1+p) * metasum \tag{3}$$

**Definition 9.** (Maximum Speed Ceiling: $V_top$) During the period of downloading data, the amount of data downloaded from any node can not excess $AND_i$. Then: $\left(V_{max}/\sum_{i=0}^{k-1} V_i\right) k(k-1) * metasum \leq (k-1)(1+p) * metasum \Rightarrow V_{top} \leq \frac{(1+p)}{k}(\sum_{i=0}^{k-1} V_i)$. $V_top$ is called Maximum Speed Ceiling.

**Example 1** Given: k=4,P=2,metasum=4. Questions: (A), Give the calculus course that distributes the data to grid nodes according to model. (B). Calculate SSUR.

Solution A:

(1) From definition 1: The data divided shares=4*(4-1)*4=48. These data blocks are as follows: (1), (2),..., (48).

(2) From definition 2: $LND_0$=(1),(2)...(12); $LND_1$=(13),(14)...(24); $LND_2$=(25),(26),...,(36); $LND_3$= (37),(38),...,(48)

(3) From definition 3:

(a) $G_0^0$=(1),(2),(3),(4), $G_0^1$=(5),(6),(7),(8), $G_0^2$=(9),(10),(11),(12);

(b) $G_1^0$=(13),(14),(15),(16), $G_1^1$=(17),(18),(19),(20), $G_1^2$=(21),(22),(23),(24);

(c) $G_2^0$=(25),(26),(27),(28), $G_2^1$=(29),(30),(31),(32), $G_2^2$=(33),(34),(35),(36);

(d) $G_3^0$ =(37),(38),(39),(40), $G_3^1$=(41),(42),(43),(44), $G_3^2$=(45),(46),(47),(48).

(4) From definition 4: (a)$\overline{N}_0^0$=$N_1$, $\overline{N}_0^1$=$N_2$, $\overline{N}_0^2$=$N_3$; (b)$\overline{N}_1^0$=$N_0$, $\overline{N}_1^1$=$N_2$, $\overline{N}_1^2$=$N_3$; (c)$\overline{N}_2^0$=$N_0$, $\overline{N}_2^1$=$N_1$, $\overline{N}_2^2$=$N_3$; (d)$\overline{N}_3^0$=$N_0$, $\overline{N}_3^1$=$N_1$, $\overline{N}_3^2$=$N_2$;

(5) From definition 5:(a) $G_0 \to \overline{N}_0^0 = G_0 \to N_1 = \{G_0^0, G_0^1\} \to N_1$, $G_0 \to \overline{N}_0^1 = G_0 \to N_2 = \{G_0^1, G_0^2\} \to N_2$, $G_0 \to \overline{N}_0^2 = G_0 \to N_3 = \{G_0^2, G_0^0\} \to N_3$; (b) $G_1 \to \overline{N}_1^0 = G_1 \to N_0 = \{G_1^0, G_1^1\} \to N_0$, $G_1 \to \overline{N}_1^1 = G_1 \to N_2 = \{G_1^1, G_1^2\} \to N_2$, $G_1 \to \overline{N}_1^2 = G_0 \to N_3 = \{G_1^2, G_1^0\} \to N_3$; (c)(d) Omitted

Solution B:

From definition 7: $SSUR = \frac{(1+P)}{k} = \frac{1+2}{4} = 0.75$. From the calculus above we can know each node stores 36 shares of data, and 4 nodes in all. So all the shares of data are 36*4, 36*4/(48*4)=0.75. the theoretical value equals to the actual value, that is, SSUR=0.75.

# 4 Parallel Scheduler

## 4.1 The Basic Idea for Scheduler

**Definition 10.** (Local Data Remainder: $S_i$) Let $x_i$ represent the ideal data downloaded from node $N_i$, where $x_i = V_i/(\sum_{j=0}^{k-1} V_j)$. Let $S_i = x_i - LND_i$, it is called Local Data Remainder.

This indicator reflects the node load. If $S_i$ tends to or equals to 0, then it shows that each node can roughly finish downloading at the same time. If $S_i$ is positive, then it shows that the download speed of the current node is fast and it can share the load of other nodes whose $S_i$ is negative. From definition 9 we can see that under ideal circumstances, there is $\sum S_i = 0$.

**Examples (1)** Given k=4, p=1, metasum=3, $V_1$=12, $V_2$=10, $V_3$=4, $V_4$=28. Scheduling objective: As far as possible to ensure that each node finishes downloading at the same time, and the data is non-repeated. We first give final results of scheduling algorithm as shown in table 1. Download the gray data blocks from node $N_i$. The ideal download data blocks from each node is (8, 6.7, 2.7, 18.7), the scheduling result is (8, 7, 3, 18). The load($S_i$) of each node happens (-1, -2, -6, 10) both before (I-A) after scheduling , that is, (0, -0.3, -0.3, 0.7), the load is balanced more or less.

As the sum of data blocks is an integer, so the actual scheduling result is rounded off. Here, SSUR=50%, Compared with the full copy deployment, only 50% storage space is used.

Table 1 Example (1)

| *Node* | *V* | $S_i$ | *Ideal (I)* | *Actual (A)* | *I-A* | *LND data* | *OND data* |
|---|---|---|---|---|---|---|---|
| $N_1$ | 12 | -1 | 8 | 8 | 0 | (1)(2)(3)(4)(5)(6)(7)(8)(9) | (10)(11)(12)(19)(20)(21)(28)(29)(30) |
| $N_2$ | 10 | -2 | 6.7 | 7 | -0.3 | (10)(11)(12)(13)(14)(15)(16)(17)(18) | (1)(2)(3)(22)(23)(24)(31)(32)(33) |
| $N_3$ | 4 | -6 | 2.7 | 3 | -0.3 | (19)(20)(21)(22)(23)(24)(25)(26)(27) | (4)(5)(6)(13)(14)(15)(34)(35)(36) |
| $N_4$ | 28 | 10 | 18.7 | 18 | 0.7 | (28)(29)(30)(31)(32)(33)(34)(35)(36) | (7)(8)(9)(16)(17)(18)(25)(26)(27) |

From Table 1 we can see that for the fast nodes they not only complete LND download mission, but they also share simultaneously the download mission of the slow nodes from their OND. For example, the data blocks (25)(26)(27) lie in LND of node $N_3$, but they are downloaded from the OND of node $N_4$.

**4.2 Description of Scheduler**

Objective: Optimize the $S_i$ of each node, make them close to or tend to 0, to make load balance for each node.

Explanation: For any two nodes ZN and FN, if the $S_i$ of ZN is positive and there are remainder parts in LND of FN that are not shared by the OND of ZN, then ZN can share the load of FN. When a block is shared, do ZN.$S_i$–, FN.$S_i$++. If a data block is shared, then the flag which points out download position is marked, which contains the same data block stored at other nodes. So the same block at other nodes will not be operated.

Input: k, p, $V_i$ , and the data distributed according to storage model.

Output: Program for how to download data blocks.

(1) Build list ZSi_list consisted with the nodes whose Si are positive, and sort ZSi_list in descending order.

(2) Build list FSi_list consisted with the nodes whose Si are negative, and sort ZSi_list in ascending order.

/* Deal with each node in FSi_list. */

(3) For every node FN in FSi_list deal from first

/* Let the nodes in ZSi_list share the load of nodes in FSi_list*/

(4) For every node ZN in ZSi_list deal from first

(5) IF FN.$S_i \geq 0$ THEN FSi_list.Move_to_Next, go to(3)

(6) IF ZN.$S_i > 0$ THEN

(a)Let the data blocks in OND of ZN share the load in LND of FN. According to the sharing sum, update ZN.Si. Share one block each time, and judge the

below two conditions timely: (b)IF FN.$S_i \geq 0$ THEN FSi_list.Move_to_Next, go to(3); (c) IF ZN.$S_i \leq 0$ THEN ZSi_list.Move_to_Next, go to(4).

(7) In stages (1)-(6) above, let the fast node share the load of slow node. If either list has reached the tail, then exit.

(8) If there are still some nodes(FN) whose Si are still negative in FSi_list, then deal with ZSi_list from first. If the intersection for OND(non-processed) of ZN and LND(non-processed) of FN is not null, then let ZN share the load of FN until the intersection is null, in this process regardless of whether the Si of ZN is already negative.

(9) There are two circumstances for the nodes in FSi_list, their Si is positive or non-positive. Let the nodes whose Si are positive share the load of negative ones until they can not share.

Now the nodes in ZSi_list exist in two cases, their Si is positive or non-positive. For every node in ZSi_list, repeat (1)-(6). Then output the scheduling sequence.

# 5 Parallel Tranfer Algorithm: PTABM

If the transmission speed is stable, then the scheduler can be used directly. As network transmission speed changes dynamically, a parallel transmission algorithm, which is based on scheduler and can meet the dynamic speed, is proposed in this chapter.

**5.1 Model-based storage policy**

Assuming there are four nodes. As shown in Figure 3, let the data meet sub-block model storage, that is, each data block meets Distributed storage Model separately. In fig.1 data groups $< B_i^1, B_i^2, B_i^3, B_i^4 >$ $(1 \leq i \leq 4)$ and $B_i = B_i^1 \cup B_i^2 \cup B_i^3 \cup B_i^4$ meet storage model separately(every group meets the model separately).

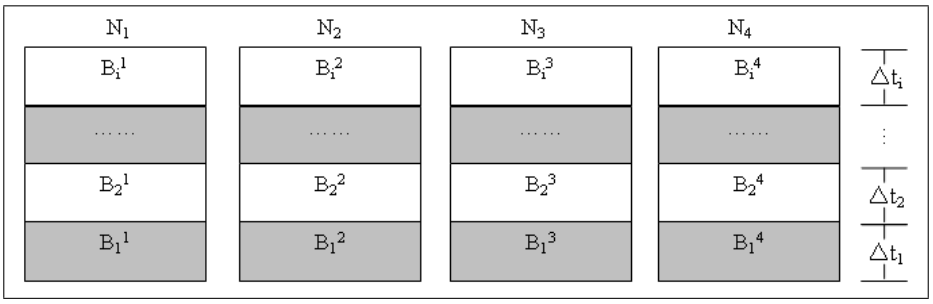


Figure 3: Storage policy sample

**5.2 Parallel Transmission Algorithm Based on Model (PTABM)**

In fig.1, as any $B_i^j$ is only part of $B_i$, so if download $B_i$ from four nodes in parallel, the following restriction must be met: (a) Uninterrupted download data from the four nodes at the same time; (b) Non-repeated download.

As the speed changes dynamicly, let $\Delta t_i$ represent the time interval from the time of first finishing downloading $B_{i-1}$ to the time of first finishing downloading $B_i$. For example, the time of completion of $B_{i-1}$ downloaded by $N_1$ is $t_{i-1}$, the time of completion of $B_i$ downloaded by $N_3$ is $t_i$, so the time interval $\Delta t_i = t_i - t_{i-1}$. As the speed changes dynamically, the size of each time interval may also be different.

The amount of $B_i$ data ($x_i$) $(0 \leq i \leq k-1)$ that each node should commit is determined as follows:(a)Take the $V_i$ at $t_{i-1}$ $(0 \leq i \leq k-1)$ as the input speed of scheduler; (b)The remainder data that every node did not complete is $m_i$ ; (c)Formula (2) must be met in theory, that is, as far as possible to ensure that each node fulfils its download mission at the same time.

$$\frac{m_0 + x_0}{V_0} = \frac{m_1 + x_1}{V_1} = ... = \frac{m_{k-1} + x_{k-1}}{V_{k-1}} \tag{4}$$

Formula (2) is changed into formula (3) through Identity Theorem. From formula (3), $x_i$ is deduced as formula (4). Take $x_i$ as the input value of scheduler to calculate $S_i$.

$$\left(\frac{m_i + x_i}{V_i}\right)_{i=0}^{k-1} = \frac{\sum_{i=0}^{k-1}(m_i + x_i)}{\sum_{i=0}^{k-1} V_i} = \frac{\sum_{i=0}^{k-1} m_i + \sum_{i=0}^{k-1} x_i}{\sum_{i=0}^{k-1} V_i} = \frac{\sum_{i=0}^{k-1} m_i + M}{\sum_{i=0}^{k-1} V_i} = \lambda \tag{5}$$

$$x_i = \lambda V_i - m_i \ (0 \le i \le k-1) \tag{6}$$

According to formulas and ristriction in this section, the schematic diagram of parallel transmission architecture is shown in Figure 4 The client side consists of two parts. At the end of every $\Delta t$ Download Unit passes $V_i$ and $m_i$ to Scheduler Unit. Scheduler Unit calculates $x_i$ and the data block $B_i^j$, and then output them to Download Unit. Download Unit downloads data from the nodes according to $B_i^j$, and real-time observe whether there is a new generation of $\Delta t$, and then repeat the process above.

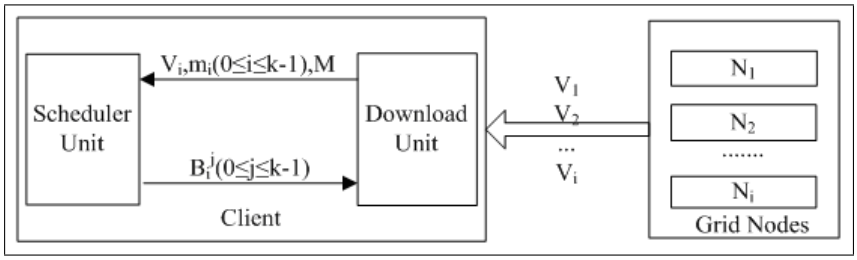


Figure 4: Schematic diagram of parallel transmission

**5.3 Reliability analysis**

The reliability of algorithm includes two aspects, which are scheduler reliability and algorithm reliability.

(1) From theorem 1, if there are random p nodes fail, then the data ,the other k-p nodes contain, is integrated. (2) If there are nodes fail during the download period, then (a) get $V_i$ of current normal nodes; (b) Let the speed of failure nodes be 0; (c) Mark the downloaded blocks of $B_i$, and the undownloaded blocks are $B_i^{'}$; (d) Take the amount of data(M') of $\sum B_i^{'}$ , $V_i$ and $m_i$ as the input values of scheduler; (e)Use Scheduler to recompute the scheduling sequences.

**5.4 Verification System (VPG-Torrent)**

Verification System contains four parts, as shown in Figure 5: Data Dispatcher, RLS server, virtual node (Vpeer), Peer node.

Data Dispatcher: Data Dispatcher take p, k, list of servers and original data as input, through model computing unit, original data was divided and deployed into servers. Meanwhile, the storage information of data blocks(M-table) was sent to RLS server.

RLS Server: the Replica Location Service (RLS) keeps track of the physical locations of files. The information stored in the Replica Location Service defines the number of replicas for each file as well as the physical location of the actual data. The physical location is identified by a unique physical file name (PFN) such as a GridFTP URL. To enable the use of the GridTorrent protocol we introduce the GridTorrent URL.

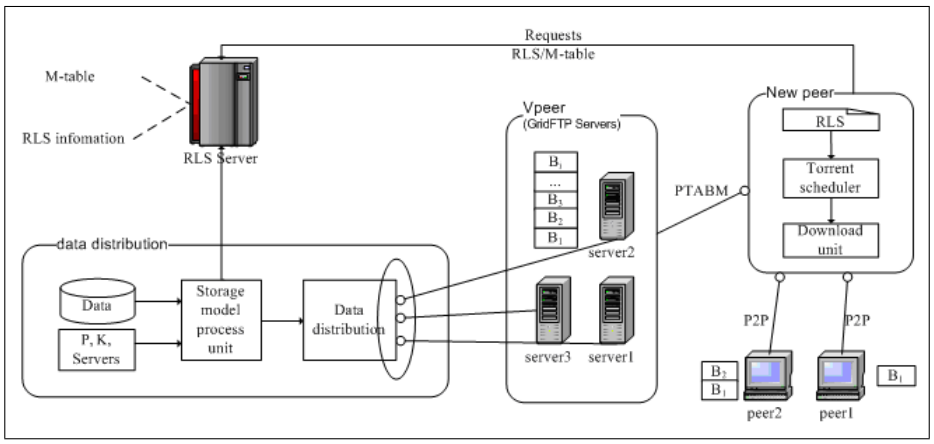


Figure 5: Data dispatcher and download

**Virtual peer (Vpeer)**: In the process of data download, GridFTP servers are regarded as a virtual peer node (Vpeer). So Vpeer has a fast transmission speed. The other peers regard Vpeer as a general peer to schedule. Of course, the transmission of Vpeer is based on PTABM.

**Peer node**: Each peer contains scheduling algorithm of BitTorrent and a download unit. Scheduling algorithm treat Vpeer as a normal peer. Download unit responsible for data transmission from other peers or Vpeer.

Each new peer must first request access to RLS server to get RLS and M-talbe information, and upload the information of data blocks they owned to RLS server periodically. As Vpeer has fast speed, so when the number of peers is small, its contribution to transmission will be greatest. When download from Vpeer PTABM must be used. Verification system consists of two major experiments: Real experiment 5-3 is to verify PTABM, and compare with previous algorithm. Simulation experiments 5-4, 5-5 are to verify the performance of large-scale concurrent access, so experiment is carried out in oversim.

# 6 Experiments

### 6.1 Purpose

From Figure 5, (a)if there are no peers willing to upload, then VPG-Torrent will degenerate into the most common parallel transmission system based on GridFTP and multi-copy, this transmission mode plays an irreplaceable role in some applications in data grid. Therefore, the performance between VPG-Torrent and traditional DADC must compared; (b) For large-scale network where BitTorrent plays an important role, the performance of VPG-Torrent must be verified as the number of concurrent peers increase from small to large, and should be compared with GridTorrent.

There is a natural experiment. If storage model is not used, but the data is directly divided into uniform shares, and then stored them into multiple nodes, so the bandwidth is also increased. If so, Vpeer is not exist, accordingly PTABM can not be used too, but only BitTorrent can be used. As bittorrent algorithms do not rely on global data storage knowledge to schedule, so it leads to the service capacity of servers can not be maximized. This experiment is carried out in 5-4(2).

Simulation experiments (1) (2) (3) are made to verify the performance of scheduler; real experiments (4) (5) are made to verify the performance of Parallel Transmission Algorithm based on Storage Model & full copies and compare with the performance, experiment (6) (7) are

made to verify the performance VPG-Torrent. Experiment 5.5 is made to verify the reliability of sheduler and algorithm. In experiment (4) (5) the client host lies in education network, and the other four grid nodes are all telecommunication network access. GridFTP is used as transmission protocol.

In experiment the concurrent connections to Vpeer is limited to 20. The maximum connection speed of every server in Vpeer is 200k, and rated bandwidth is 5M. Each peer can start up 15 connections at the same time, the upload rate is up to 20kb.

**6.2 Basic Definitions and Explanations**

If the speed is determined, the ideal amount of data downloaded from each node can be: $M * V_i/(\sum_{j=0}^{k-1} V_j)$ $(0 \leq i \leq k-1)$. The result that scheduler can give and the ideal amount of download data are compared during experiments (1) (2) (3).

Literature [11] combines basic parallel transmission mode of GridFTP and P2P technologies, but the prerequisite is that there are multiple clients requesting; the research of parallel transmission algorithm in literature [9] [10] is based on GridFTP only, its dynamic collaborative algorithm (DCDA) achieved good performance. The performances of PTABM and DCDA are compared in experiments (4) (5).

**Definition 11.** (The percentage of time that scheduling results exceeds ideal value:TRER) Given the ideal download time is $T_{id}$, the download time outputted by scheduler is $T_{re}$. Let $TRER = (100(T_{re} - T_{id})/T_i)\%$. This parameter is used to detect the performance of scheduler.

**Definition 12.** (Unit Speed Variance:D(V)) Variance divided by the sum of the speed is called Unit Speed Variance, that is:

$$E(V) = \left(\sum_{i=0}^{k-1} V_i\right)/k \; and \; D(V) = \left(\sum_{i=0}^{k-1} (V_i - E(V))^2\right) / \left(k\left(\sum_{i=0}^{k-1} V_i\right)\right) \tag{7}$$

From definition 9 we can see that when k is fixed and for the same group $V_i$, $V_{top}$ increases by $\left(\sum_{i=0}^{k-1} V_i\right)/k$ when p increases by 1. From definition 7 we can see that the Storage Space Usage Rate increases by 1/k.

**6.3 Scheduler Performance Test**

(1) This experiment is to verify the impact of D(V), $V_{top}$, $V_{ratio}$ on TRER. Let p=1, k=4, metasum=15, the range for the speed distribution is (1,100), then the theoratical $V_{ratio}$=90. Consecutively record the results of algorithm for 20 times, the impact of D (V), $V_{ratio}$, and $V_{top}$ on TRER is listed in Table 2.

Calculate the average of D (V), that is, ED(V). Classified the observational results that meet D(V)>ED(V) into a group, and D(V)<ED(V) as another group. In table 2 the experiments numbered by 1-9 are of D (V)>ED (V). TRER is 4.40.

The experiments 1-9 and 10-20 calculate the average TRER respectively, the value are 7.40, 1.87 and they are greater or less than the overall average of 4.40.

Experiment (1) shows that if the distribution of node speed is very uneven and the difference between maximum and minimum speed is large, TRER may be increased. If Vtop exceeds the theoratical maximum value, then TRER will have a substantial increase.

**(2) This experiment is made to verify the impact of p and speed range on TRER.**

Let k=10, and change the range of p & speed, as shown in Table 3. There are nine combinations, observe ten times for every combination. Calculate the TRER average of every combination, the histogram is shown in figure 6.

Let k=2,p=2, observe the impact of speed range on TRER. Observing ten times of algorithm results in every speed range, and calculates TRER average. The curve is shown in Figure 7.

Table 2 Impact of D(V), $V_{ratio}$, $V_{top}$ on *TRER*

| *No.* | *>D(V) Average?* | *D(V)* | *V.* | *Actual* $V_{...}$ | *Theoretical* $V_{xy}$ | *Actual* $V_{xy}$ | *TRER* | **TRER Average** |
|---|---|---|---|---|---|---|---|---|
| 1 | Y | 5.34 | 37,10,88,93 | 9.30 | 114.00 | 93.00 | 2.70 | |
| **2** | **Y** | **4.47** | **12,23,55, 1** | **55.00** | **45.50** | **55.00** | **18.03** | |
| 3 | Y | 3.07 | 85,18,89,76 | 4.94 | 134.00 | 89.00 | 0.00 | |
| 4 | Y | 4.48 | 1,77,67,75 | 77.00 | 110.00 | 77.00 | 0.00 | |
| 5 | Y | 4.48 | 95, 9,72,49 | 10.56 | 112.50 | 95.00 | 3.22 | **7.40** |
| 6 | Y | 4.59 | 7,85,57,91 | 13.00 | 120.00 | 91.00 | 5.33 | **(>4.40)** |
| 7 | Y | 3.92 | 55,95,70,12 | 7.92 | 116.00 | 95.00 | 6.33 | |
| **8** | **Y** | **5.89** | **12,24,76,10** | **7.60** | **61.00** | **76.00** | **29.44** | |
| 9 | Y | 4.00 | 67,34, 1,34 | 67.00 | 68.00 | 67.00 | 2.37 | |
| 10 | N | 2.16 | 58,66,22,83 | 3.77 | 114.50 | 83.00 | 2.16 | |
| 11 | N | 0.46 | 36,28,41,21 | 1.95 | 63.00 | 41.00 | 2.50 | |
| 12 | N | 2.03 | 22,26,50,67 | 3.05 | 82.50 | 67.00 | 2.24 | |
| 13 | N | 1.99 | 33,94,89,73 | 2.85 | 144.50 | 94.00 | 2.48 | |
| 14 | N | 1.09 | 35,31,45,67 | 2.16 | 89.00 | 67.00 | 1.71 | |
| 15 | N | 0.75 | 19,10,32,24 | 3.20 | 42.50 | 32.00 | 0.00 | **1.87** |
| 16 | N | 2.35 | 19,13,46,55 | 4.23 | 66.50 | 55.00 | 5.00 | **(<4.40)** |
| 17 | N | 2.47 | 64,22,66,90 | 4.09 | 121.00 | 90.00 | 0.00 | |
| 18 | N | 2.62 | 70,68,44,14 | 5.00 | 98.00 | 70.00 | 1.46 | |
| 19 | N | 1.94 | 85,29,46,64 | 2.93 | 112.00 | 85.00 | 2.99 | |
| 20 | N | 1.82 | 51,19,16,45 | 3.19 | 65.50 | 51.00 | 0.00 | |
| **Average** | | **3.00** | | | | | **4.40** | |

Table 3 Experimental Methods

| P \ $V_i$ | 5-25 | 5-50 | 5-75 |
|---|---|---|---|
| 1 | 10 Times | 10 Times | 10 Times |
| 2 | 10 Times | 10 Times | 10 Times |
| 3 | 10 Times | 10 Times | 10 Times |

From Fig.6 we can see that when p=1, the change of speed has greater impact on TRER. But when p=2,3 as the speed range increases, TRER are almost unaffected. Scheduling results are approximately equivalent to the ideal download speed.

It can be seen from Figure 7, TRER gradually increases as the speed range increases. However, when the speed range is in 5-150 and TRER are in the vicinity of one, the effect is good.

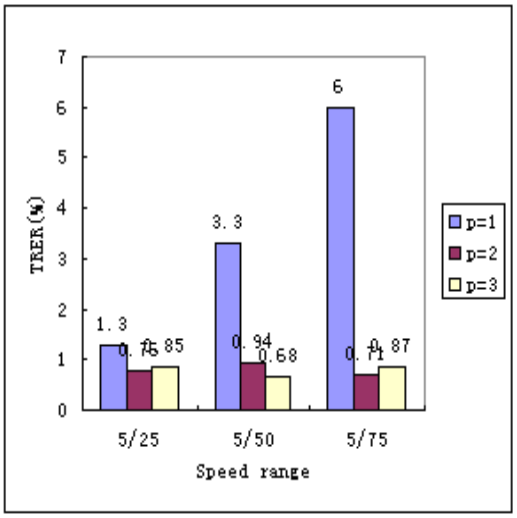


Figure 6: mpact of p and speed range on *TRER* when k=10

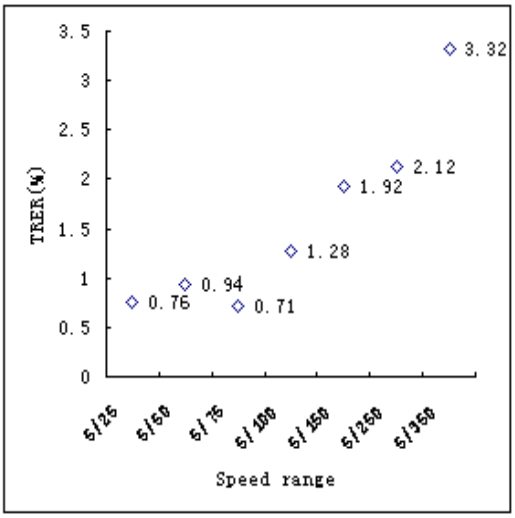


Figure 7: Impact of speed range on *TRER* when k=10, p=2

Experiment (2) shows that in a random speed range, if p has already guaranteed that TRER is maintained at an ideal value. Even when p continues to increase, TRER will not decrease.

**(3) This experiment is done to verify the impact of k and speed range on TRER.**

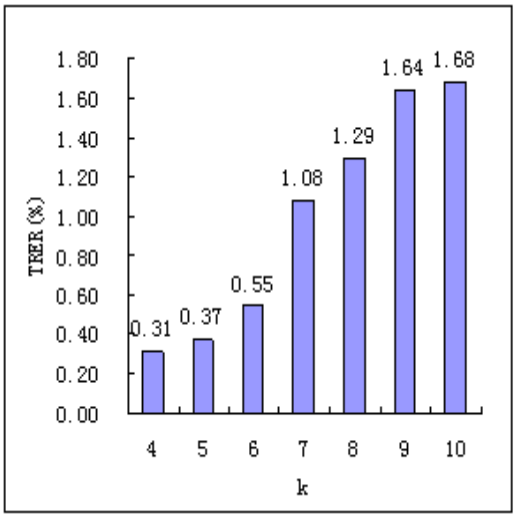


Figure 8: Impact of k on *TRER* when P=2, $V_i \in(1,50)$

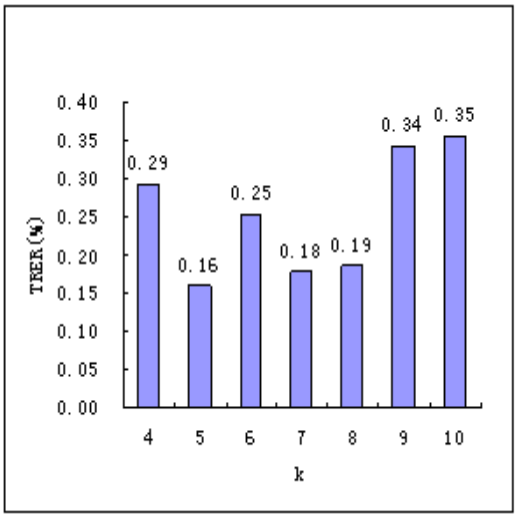


Figure 9: Impact of k on *TRER* when P=2,$V_i \in(5,50)$

Let the amount of data at every node is 300. In order to ensure the metadata will not be affected when k and the overall amount of data change, do as follows: When k=4 and metasum=100, from formula (1) metasum of other nodes can be calculated, as shown in table 4.

Because the sum of nodes increases, if the amount of data doesn't increase, then the download time will decrease, so the TRER will become inaccurate. In order to guarantee experimental parameters consistency, a special treatment is needed.

Table 4 Value of k, M and metasum

| K | M | metasum |
|---|---|---|
| 4 | 1200 | 100 |
| 5 | 1500 | 75 |
| 6 | 1800 | 60 |
| 7 | 2100 | 50 |
| 8 | 2400 | 43 |
| 9 | 2700 | 38 |
| 10 | 3000 | 33 |

Let p=2, $V_i \in(1,50)$, observe the effect of changes of k on TRER. For every k, observe 10 times, then calculate the average of TRER. From Figure 8 we can see that TRER increases gradually as k increases.

Let p=2, $V_i \in(1,50)$, observe the effect of the change of k on TRER. For every k, observe 10 times, then calculate the average of TRER. From Figure 9 we can see that as k increases TRER changes little.

When $V_i \in(1,50)$, the difference among the speeds of nodes is larger, and the maximum theoretical $V_{top}$=50; while $V_i \in(5,50)$, there is little difference among the speeds of nodes, and the maximum theoretical $V_{top}$=10 only.

Experiment (3) shows that the larger speed range affects TRER greatly. If the speed range is not large, then k has little effect on TRER.

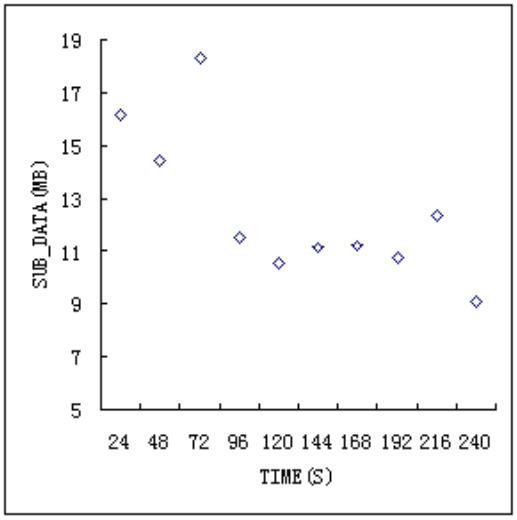


Figure 10: Distribution of SUB_DATA points when matadata=5M)

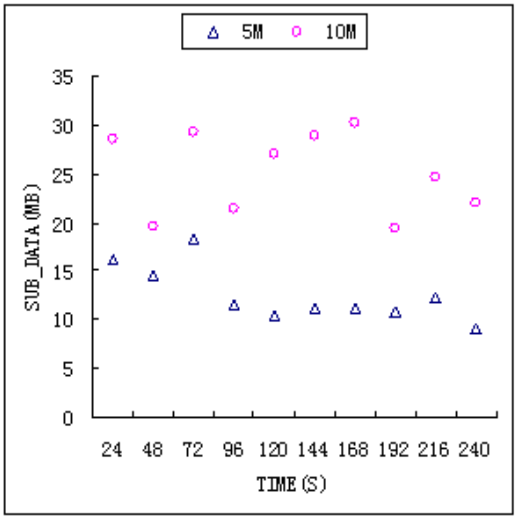


Figure 11: Distribution of SUB_DATA points when matadata=5M,10M

It is found through experiments that the value of TRER near one is caused by the sub-block. The ideal download data of each node is based on an arbitrary partition for the overall data, and the error generated by the scheduler can be reduced by increasing the metasum appropriately. Therefore for the value of TRER near one we can consider that the scheduler has achieved the ideal scheduling result.

### 6.4 PTABM Performance Test

(4) If there are no peers willing to upload, then VPG-Torrent will degenerate into the most common parallel transmission system based on GridFTP and multi-copy. This experiment is made to compare the transmission speed between PTABM and DCDA, as well as compare the amount of download data at the same time of the two algorithms. Let metadata be 5M and 10M, analyze the performance of PTABM.

Let matadata=5M, observe the amount of data downloaded by DCDA and PTABM separately ten times at different time. Let SUB_DATA equal the amount of download data by SUB_DATA minus PTABM's. The point distribution is shown in Figure 10.

Let matadata=5M,10M, observe the amount of data that downloaded by DCDA, PTABM separately ten times at different time. Let SUB_DATA equal the amount of data downloaded by DCDA minus PTABM's. Two type point distribution is shown in Figure 11.

Figure 10 in experiment (4) shows that SUB_DATA does not increase as the time increases, and shows the oscillation effects. It shows that SUB_DATA is non-cumulative. So as the amount of download data increases, the ratio of SUB_DATA and overall data tends to zero, so the performance of PTABM tends to DCDA. Fig.8. shows that when matadata is larger, SUB_DATA is also larger. But as time increases, it also shows the oscillation effects.

**(5) This experiment is made to compare the performance between PTABM and DCDA in download time and amount.**

Let one time unit contain 10 seconds and matadata=15M. Observe the amount of download data and calculate the ratio of data downloaded by DCDA and PTABM, take the percentage (DR). The curve for the relationship between DR and time units is shown in Figure 12.

Let metadata=5M,10M,20M, p=1. Separately observe the time that taken by DCDA and PTABM to download 960M and 1920M data. Then calculate the ratio of PTABM finishing time and DCDA's, take the percentage (TR). The TR bar graph is shown in Figure 113.

Figure 12 in experiment 5 shows that as the download time increases the speed of PTABM tends to DCDA. At 40 50s their speeds are similar and are approximately equivalent at 100s. From Figure 13 we can see that smaller matadata can achieve better effects than larger one. When matadata=5 transfer 1920M data the ratio of the time used by PTABM and DCDA is 1.05. So for vast data the two algorithms can achieve similar performances.

**6.5 VPG-Torrent Verification**

**(6) This experiment is made to verify performance of VGP-Torrent when p=1, P2P is allowed or not allowed (download from GridFTP servers), and compare performance with GridTorrent.**

Let original data size be 1G, and the amount of data permitted to be deployed is 2G. Within 20 minutes specific amount of peers are generated randomly. Each peer will continue to permit to be downloaded by other peers for 40 minutes from its completion.

Firstly, deploy two copies for GridTorrent system; secondly, given p=1, k=5,6,7 as the input of data dispatcher of VPG-Torrent, so the information of data storage is stored into RLS server later.

When P2P is allowed, from Figure 14, in the performance curves of different peers, the curve GT of GridTorrent is located in the bottom. When k=5,6,7, the curves of VPG-Torrent increased gradually. And the curves are all concave. It is because that when the number of peers is small, parallel transmission only based on GridFTP servers play an important role, as the number increases gradually, then BitTorrent plays an important role.

When P2P is not allowed, i.e., the transmission is only based on GridfTP. From Figure 15, VPG-Torrent has a great advantage over BitTorrent in average speed of peers. Most importantly, the storage space used by them is the same.

**(7) This experiment is made to test how impact of storage method on GridTorrent and compare performance with VPG-Torrent.**

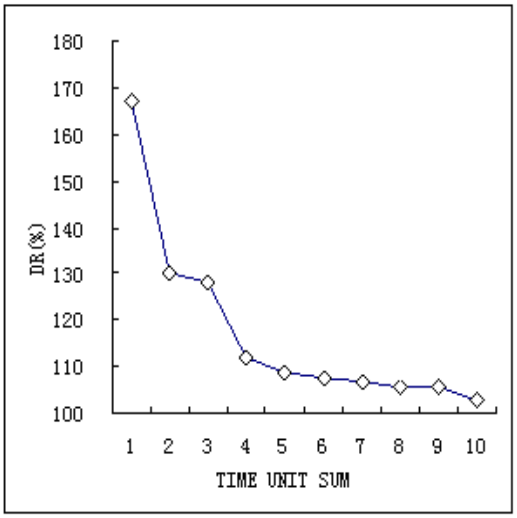


Figure 12: Download data ratio at the same time when matadata=15M

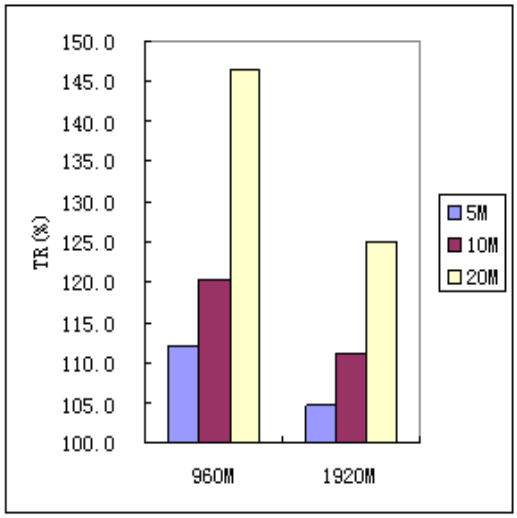


Figure 13: Impact of matadata on download speed

For GridTorrent we process the original data as follows: let original data size be 1G, and the amount of data permitted to be deployed is 2G. The original data were evenly divided into 3 shares, and are distributed to 6 nodes, so that there are two nodes have the same data. These six nodes will treated as normal peers for any new peers. Experiments were carried out in two conditions: P2P is allowed, and not allowed. When P2P is not allowed, there will be six threads downloading data concurrently, and the completion time depends on the node which complete transmission task at last.

In Figure 16, k=6 is the performance curve of VPG-Torrent, AS is the curve of GridTorrent when storage method is adopted, and GT is the curve of GridTorrent obtained in experiment 6-5-(6). Obviously AS is better than GT. So for BitTorrent to increase performance, the better way is to increase the bandwidth by deploying original data to multiple nodes under the condition of using the same size of storage space. However, the performance of AS is a little worse than VPG-Torrent's. It is because that BitTorrent will not utilize the storage knowledge to schedule, accordingly it can not make full use of server bandwidth.

In Figure 17, when P2P is not allowed, 'M' is the column of VPG-Torrent, and 'A' is the column of GridTorrent. Obviously the transmission time of VPG-Torrent is less than GridTorrent's. In this experiment VPG-Torrent can ensure that all the download tasks can be finished almost at the same time, therefore, it can make the load balance. Instead, BitTorrent can not. Although the storage space used is the same, VPG-Torrent exceeds BitTorrent in transmission time.

**6.6 Reliability Test**

This experiment is made to verify the performance of scheduler at normal conditions and when node failed, verify the performance of PTABM when one node failed, and compare the performance of PTABM and DCDA at this condition.

Let p=(1,2), $V_i$ ∈(5,25), k=(4,5,6,7), observe the performance of scheduler in normal and failure circumstances. In Figure 18 'F=1' represents that one node is failed. Draw the curve of k and TRER . From Figure 18 we can see that when p=1, there is one failure node and the performance of scheduler is a little worse than in normal circumstances. While p=2, there is also one failure node, but the performance of scheduler is good, with TRER kept near 1.

Let p=1, $V_i$ ∈(5,25), metasum=5, observe the performance of PTABM and DCDA in normal

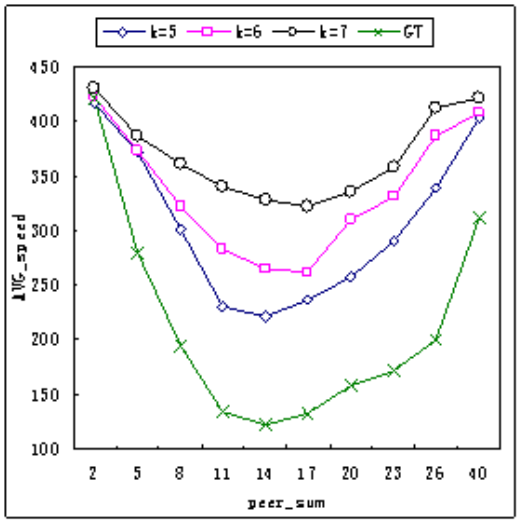


Figure 14: Impact of peers on average speed when BitTorrent is allowed

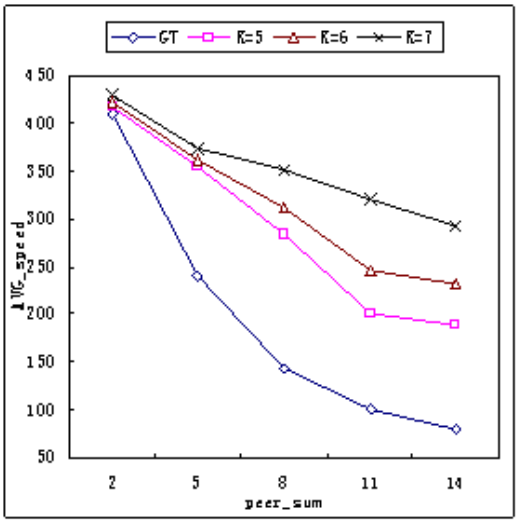


Figure 15: Impact of peers on average speed when BitTorrent is not allowed

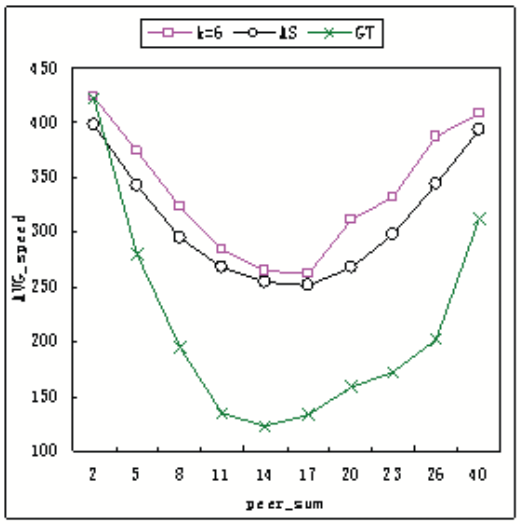


Figure 16: Performance compares for different storage methods when P2P is allowed

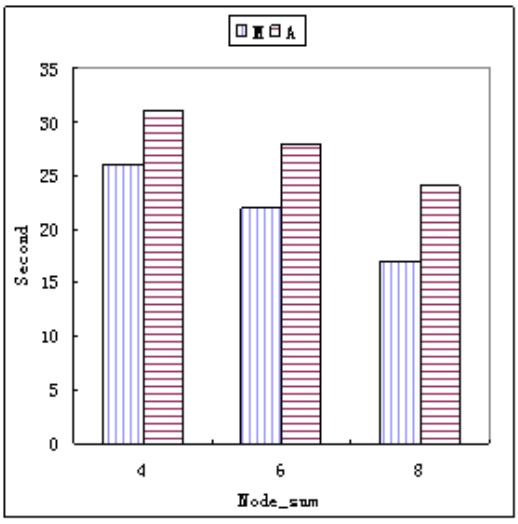


Figure 17: Performance compares for different storage methods when P2P is not allowed

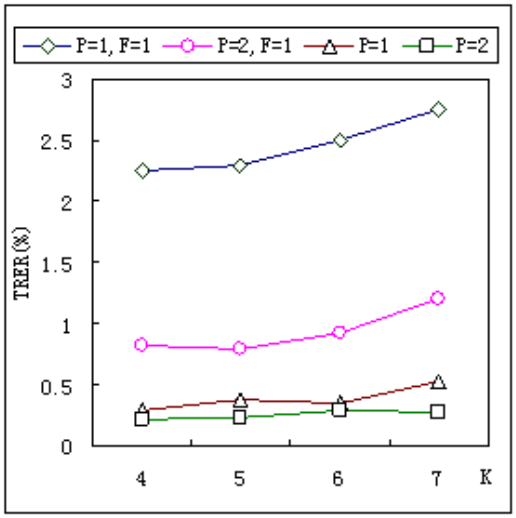


Figure 18: Comparison of scheduler performance between normal and failure curcumstances

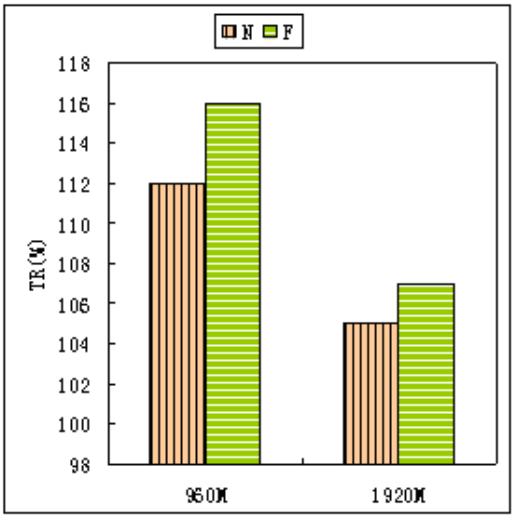


Figure 19: Comparison of algorithm performance between normal and failure curcumstances

and failure circumstances, make the ratio(TR) of PTABM and DCDA. In Figure 19 'N' represents normal circumstance while 'F' represents failure circumstance. From the fig we can see when the amount of data is 1920M the download time of the two algorithms is similar. It shows that PTABM has good reliability.

**6.7 Supplementary Analysis of Experimental Results**

The large speed range in experiment 6.3 is to verify the performance of scheduler in the extreme distribution of speed. So in practice as long as taking appropriate p value according to the speed range, the smaller TRER can be achieved. At the same time, given a reasonable matadata value, the performance of PTABM can be close to or approximately equal to the performance of DCDA. The most important is saving storage space and reducing the network traffic caused by the creation of multi-copies. Fortunately, the Distributed storage Model and PTABM meet those requirements.

In experiment 6.4 p=1, matadata=5M, the effects are very good. Derived from definition 7: when p=1, the SSUR and k are calculated as in table 5. When k=4, the SSUR is 50%; but when k=10, the SSUR is only 20%. As achieved, the objectives of rapid data transmission, the storage space and network traffic are also significantly optimized at the same time.

Table 5 Relationship between p=2, k and SSUR

| *k* | *4* | *5* | *6* | *7* | *8* | *9* | **10** |
|---|---|---|---|---|---|---|---|
| SSUR | 0.5 | 0.4 | 0.33 | 0.29 | 0.25 | 0.22 | 0.2 |

From experiment 6.5, whether P2P is allowed or not, the performance of VPG-Torrent is higher than GridTorrent's, even if increasing the bandwidth by adopting uniform storage method.

## 7 Conclusions

Parallel transmission algorithms based on multi-copy cause a waste of storage space seriously and are difficult to adapt to a wide range of network access. Instead, BitTorrent for wide range of file-sharing has a unique advantage. Although GridTorrent combined the two above,

the performance is poor when the peers are few. In this paper we achieved multiple optimization objectives: storage space saving, suitable for two kinds of application modes (i.e. parallel transfer based on GridFTP and BitTorrent), adaptability for wide range of network and higher performance when there are fewer peers. The proposed storage model, scheduler, parallel transmission algorithm, virtual peer and VPG-Torrent verification system achieved very good result, and the system proposed has certain advantages compared with the previous algorithms.

## Bibliography

[1] CHEN Lei, LI San-li. A Calking Dynamic Replication Distribution Algorithm in Data Grid. *ACTA ELECTRONICA SINICA*, 34(11):1-4, 2006

[2] XIE Xiao-lan, LIU Yu, ZHOU De-jian. Research on Manufacturing Grid Data Access and Integration Key Technology. *JOURNAL OF WUHAN UNIVERSITY OF TECHNOLOGY*, 31(6):1-4, 2009

[3] ZHANG Guangzhi, HE Jieyue. Application Research on Biological Data Grid. *Computer Engineering*,(2):1-4, 2004

[4] QIN Xin, LUO Ze, NAN Kai etal. Design and Implementation of Problem Solving Environment for Astronomy Application Based on Science Data Grid. *Application Research of Computers*,(4):1-4, 2009

[5] H.A. James, K.A. Hawick. Scientific Data Management in a Grid Environment. *Journal of Grid Computing*,3: 39-51, 2005

[6] Mingwei Wang, Shusheng Zhang, Jingtao Zhou etal. An Architecture of Semantic Desktop Data Grid. *Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design*,IEEE Computer Society ,1-6, 2006

[7] S. Fiore, M. Mirto, Cafaro. A GRelC based Data Grid Management Environment. *21st IEEE International Symposium on Computer-Based Medical Systems*, IEEE Computer Society, 355-360,2008

[8] Richard McClatchey, Ashiq Anjum etal. Data Intensive and Network Aware (DIANA) Grid Scheduling. *Journal of Grid Computing*,5:43-64, 2007

[9] H. Liu, et al., Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, *Future Generation Computer Systems*:1-8,2009

[10] Xiangang Zhao, Bai Wang, Nan Du. Qos-based Algorithm for Job Allocation and Scheduling in Data Grid. *Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops (GCCW'06)*, IEEE Computer Society:1-7,2006

[11] Nhan Nguyen Dang, Soonwook Hwang, Sang Boem Lim. Improvement of Data Grid's Performance by Combining Job Scheduling with Dynamic Replication Strategy. *The Sixth International Conference on Grid and Cooperative Computing(GCC 2007)*, IEEE Computer Society:1-8,2007

[12] Esther Pacitti. Patrick Valduriez. Marta Mattoso. Grid Data Management: Open Problems and New Issues. *Journal of Grid Computing*,5:273-281, 2007

[13] Jiang Jianjin, Yang Guangwen. Replication Strategies in Data Grid Systems with Clustered Demands. *JOURNAL OF COMPUTER RESEARCH AND DEVELOPMENT*,46(2):1-8,2009

[14] W u Chang-ze, Chen Shu-yu, Ti an Dong. The strategy of creating replica based on cost shared in data grid. *Huazhong Univ. of Sci. & Tech. (Nature Science Edition)*,35(2):1-4, 2007

[15] Pangfeng Liu. Jan-Jan Wu, Optimal Replica Placement Strategy for Hierarchical Data Grid Systems. *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*:IEEE Computer Society: 1-4, 2006

[16] Tim Ho, David Abramson. A Unified Data Grid Replication Framework. *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*: IEEE Computer Society: 1-8, 2006

[17] Ingmar Baumgart, Bernhard Heep, Stephan Krause, OverSim: A scalable and flexible overlay framework for simulation and real network applications, *Proceedings of the 9th International Conference on Peer-to-Peer Computing (IEEE P2P'09)*, pp. 87-88, Seattle, WA, USA, Sep 2009

[18] Ingmar Baumgart, Bernhard Heep, Stephan Krause, OverSim: A Flexible Overlay Network Simulation Framework, *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007*, p. 79-84, Anchorage, AK, USA, May 2007

[19] R.S.Bhuvaneswaran, Yoshiaki Katayama, Naohisa Takahashi. Dynamic Co-allocation Scheme for Parallel Data Transmission in Grid Environment. *Proceedings of the First International Conference on Semantics, Knowledge, and Grid, IEEE Computer Society*: 1-6, 2006

[20] Sudharshan, Vazhkudai. Distributed Downloads of Bulk, Replicated Grid Data. *Journal of Grid Computing*,2:31-42, 2005

[21] Gaurav Khanna, Umit Catalyurek, Tahsin Kurc, et al. A Dynamic Scheduling Approach for Coordinated Wide-Area Data Transfers using GridFTP. *The 22nd International Parallel and Distributed Processing Symposium (IPDPS '08)*. IEEE Computer Society, 2008,1-12

[22] Liu Dongmei, Liu Dongmei. Multi-path parallel transmission scheme for optical grid systems. *Chinese High Technology Letters*,5:1-4,2008

[23] A. Zissimos, K. Doka, A. Chazapis and N. Koziris. GridTorrent: Optimizing data transfers in the Grid with collaborative sharing. in *Proceedings of the 11th Panhellenic Conference on Informatics (PCI2007)*, Patras, Greece, May 2007:1-12

[24] Athanasia Asiki, Katerina Doka, Ioannis Konstantinou, et al. A Distributed Architecture for Multi-Dimensional Indexing and Data Retrieval in Grid Environments. *In Proceedings of the Cracow 2007 Grid Workshop (CGW'07)*, Krakow, Polland, October 16-17, 2007:1-8

[25] A. Kaplan, G.C. Fox and G. von Laszewski, GridTorrent Framework: A High-performance Data Transfer and Data Sharing Framework for Scientific Computing. *Proc Grid Computing Environments, Supercomputing Workshops*, Reno, NV, USA, November 2007:1-10