Improved Algorithms for Data Migration

Samir Khuller[1], Yoo-Ah Kim[2], and Azarakhsh Malekian[1]

[1] Department of Computer Science, University of Maryland, College Park, MD 20742.
Research supported by NSF Award CCF-0430650.
E-mail: samir,malekian@cs.umd.edu.

[2] Department of Computer Science and Engineering, University of Connecticut,
Storrs, CT 06269.
E-mail: ykim@engr.uconn.edu.

Abstract. Our work is motivated by the need to manage data on a collection of storage devices to handle dynamically changing demand. As demand for data changes, the system needs to automatically respond to changes in demand for different data items. The problem of computing a migration plan among the storage devices is called the *data migration problem*. This problem was shown to be NP-hard, and an approximation algorithm achieving an approximation factor of 9.5 was presented for the half-duplex communication model in [Khuller, Kim and Wan: Algorithms for Data Migration with Cloning, *SIAM J. on Computing*, Vol. 33(2):448–461 (2004)]. In this paper we develop an improved approximation algorithm that gives a bound of $6.5 + o(1)$ using various new ideas. In addition, we develop better algorithms using external disks and get an approximation factor of 4.5. We also consider the full duplex communication model and develop an improved bound of $4 + o(1)$ for this model, with no external disks.

1 Introduction

To handle high demand, especially for multimedia data, a common approach is to replicate data objects within the storage system. Typically, a large storage server consists of several disks connected using a dedicated network, called a *Storage Area Network*. Disks typically have constraints on storage as well as the number of clients that can access data from a single disk simultaneously. These systems are getting increasing attention since TV channels are moving to systems where TV programs will be available for users to watch with full video functionality (pause, fast forward, rewind etc.). Such programs will require large amounts of storage, in addition to bandwidth capacity to handle high demand.

Approximation algorithms have been developed [16, 17, 7, 11] to map known demand for data to a specific data layout pattern to maximize utilization, where the utilization is the total number of clients that can be assigned to a disk that contains the data they want. In the layout, we compute not only how many copies of each item we need, but also a layout pattern that specifies the precise subset of items on each disk. The problem is NP-hard, but there are polynomial-time approximation schemes [7, 16, 17, 11]. Given the relative demand for data, the algorithm computes an almost optimal layout. Note that this problem is slightly different from the *data placement problem* considered in [9, 15, 3] since all the disks are in the same location, it does not matter which disk a client is assigned to. Even in this special case, the problem is NP-hard [7].

Over time as the *demand for data changes*, the system needs to create *new* data layouts. The problem we are interested in is the problem of computing a data migration plan for the set of disks to convert an initial layout to a target layout. We assume that data objects have the same size (these could be data blocks, or files) and that it takes the same amount of time to migrate any data item from one disk to another disk. In this work we consider two models. In the first model (half-duplex) the crucial constraint is that each disk can participate in the transfer of only one item – either as a sender or as a receiver. In other words, the communication pattern in each round forms a matching. Our goal is to find a migration schedule to minimize the time taken to complete the migration (makespan). To handle high demand for popular objects, new copies will have to be dynamically created and stored on different disks. All previous work on this problem deals with the half-duplex model. We also consider the full-duplex model, where each disk can act as a sender and a receiver in each round for a single item. Previously we did not consider this natural extension of the half-duplex model since we did not completely understand how to utilize its power to prove interesting approximation guarantees.

The formal description of the *data migration problem* is as follows: data item i resides in a specified (source) subset S_i of disks, and needs to be moved to a (destination) subset D_i. In other words, each data item that initially belongs to a subset of disks, needs to be moved to another subset of disks. (We might need to create new copies of this data item and store it on an additional set of disks.) See Figure 1 for an example. If each disk had exactly one data item, and needs to copy this data item to every other disk, then it is exactly the problem of gossiping. The data migration problem in this form was first studied by Khuller, Kim and Wan [4], and it was shown to be NP-hard. In addition, a polynomial-time 9.5-approximation algorithm was developed for the half-duplex communication model.

A slightly different formulation was considered by Hall et al. [10] in which a particular transfer graph was specified. While they can solve the problem very well, this approach is limited in the sense that it does not allow (a) cloning (creation of several new copies) and (b) does not allow optimization over the space of transfer graphs. In [4] it was shown that a more general problem formulation is the one with source and destination subsets specified for each data item. However, the main focus in [10] is to do the transfers without violating space constraints. Another formulation has been considered recently where one can optimize over the space of possible target layouts [12]. The resulting problems are also NP-hard. However, no significant progress on developing approximation algorithms was made on this problem. A simple flow based heuristic was presented for the problem, and was demonstrated to be effective in finding good target layouts.

Job migration has also been considered in the scheduling context recently as well [2], where a fixed number of jobs can be migrated to reduce the makespan by as much as possible. There is a lot of work on data migration for minimizing completion time for a fixed transfer graph as well (see [6, 14] for references).

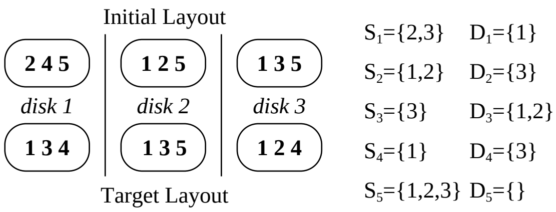

Fig. 1. An initial and target layout, and their corresponding S_i's and D_i's. For example, disk 1 initially has items $\{2, 4, 5\}$ and in the target layout has items $\{1, 3, 4\}$.

1.1 Communication Model

Different communication models can be considered based on how the disks are connected. In this paper we consider two models. The first model is the same model as in the work by Hall et al. [10, 1, 4, 13] where the disks may communicate on any matching; in other words, the underlying communication graph allows for communication between any pair of devices via a matching (a switched storage network with unbounded backplane bandwidth). Moreover, to model the limited switching capacity of the network connecting the disks, one could allow for choosing any matching of bounded size as the set of transfers that can be done in each round. We call this the *bounded-size matching model*. It was shown in [4] that an algorithm for the bounded matching model can be obtained by a simple simulation of the algorithm for the unbounded matching model with excellent performance guarantees.

In addition we consider the full duplex model where each disk may act as a sender and a receiver for an item in each round. Note that we do not require the communication pattern to be a matching any more. For example, we may have cycles, with disk 1 sending an item to disk 2, disk 2 to disk 3 and disk 3 to disk 1. In earlier work we did not discuss this model as we were unable to utilize the power of this model to prove non-trivial approximation guarantees. Note that this does not correspond directly to edge coloring anymore.

1.2 Our Results

Our approach is based on the approach initially developed in [4]. Using various new ideas lets us reduce the approximation factor to $6.5 + o(1)$. The main technical difficulty is simply that of "putting it all together" and making the analysis work.

In addition we show two more results. If we are allowed to use "external disks" (called bypass disks in [10]), we can improve the approximation guarantee further to $3 + \frac{1}{2}\max(3, \gamma)$. This can be achieved by using at most $\lceil \frac{\Delta}{\gamma} \rceil$ external disks, where Δ is the number of items that need to be migrated. We assume that each external disk can hold γ items. This gives an approximation factor of 4.5 by setting $\gamma = 3$.

Finally, we also consider the full-duplex model where each disk can be the source or destination of a transfer in each round. In this model we show that an approximation guarantee of $4 + o(1)$ can be achieved. Earlier, we did not focus on this model specifically as we were unable to utilize the power of this model in any non-trivial manner.

The algorithm developed in [4] has been implemented, and we performed an extensive set of experiments comparing its performance with the performance of other heuristics [8]. Even though the worst case approximation factor is 9.5, the algorithm performed very well in practice, giving approximation ratios within twice the optimal solution in most cases.

2 The Data Migration Algorithm

Our algorithms make use of known results on edge coloring of multigraphs. Given a graph G with max degree Δ_G and multiplicity μ the following results are known (see Bondy-Murty [5] for example). Let χ' be the edge chromatic number of G. Note that when G is bipartite, $\chi' = \Delta_G$ and such an edge coloring can be obtained in polynomial time [5].

Theorem 1. *(Vizing [20]) If G has no self-loops then $\chi' \leq \Delta_G + \mu$.*

Theorem 2. *(Shannon [18]) If G has no self-loops then $\chi' \leq \lfloor \frac{3}{2}\Delta_G \rfloor$.*

As in [4] let β_j be $|\{i | j \in D_i\}|$, i.e., the number of different sets D_i, to which a disk j belongs. We then define β as $\max_{j=1...N} \beta_j$. In other words, β is an upper bound on the number of items a disk may need. Note that β is a lower bound on the optimal number of rounds, since the disk j that attains the maximum, needs at least β rounds to receive all the items i such that $j \in D_i$, since it can receive at most one item in each round. Moreover, we may assume that $D_i \neq \emptyset$ and $D_i \cap S_i = \emptyset$. This is because we can define the destination set D_i as the set of disks that need item i and do not currently have it. Before performing data migrations, we first choose several representative sets from S_i and D_i.

2.1 Selecting Representative sets

1. For an item i decide a primary source $s_i \in S_i$ so that $\alpha = \max_{j=1,\ldots,N}(|\{i | j = s_i\}| + \beta_j)$ is minimized. In other words, α is the maximum number of items for which a disk may be a primary source (s_i) or destination. *Note that α is also a lower bound on the optimal number of rounds.* This step is the same as in [4].

2. Find $R_i (\subseteq D_i)$ for each item i.
   (a) We divide set D_i into $\lceil \frac{|D_i|}{q} \rceil$ subgroups of size at most q (q is a parameter that will be specified later.) That is, we create $\lfloor \frac{|D_i|}{q} \rfloor$ subgroups of size q and (if $|D_i|$ is not a multiple of q) one subgroup of size $|D_i| - q \cdot \lfloor \frac{|D_i|}{q} \rfloor$.
   (b) We find $R_i \subseteq D_i$ and assign subgroups to disks in R_i so that for each disk in R_i the total size of subgroups assigned to the disk is at most $\beta + q$. (We describe how to find R_i and the assignment, later in detail.) Let r_i be the disk in R_i to which the small subgroup (a subgroup with size strictly less than q) is assigned. Note that if $|D_i|$ is a multiple of q, there is no disk r_i. We define $\overline{R_i}$ to be $R_i \setminus r_i$.
3. For each item, we select $G'_i \subseteq D_i$ as follows.
   (a) Compute $G_i \subseteq D_i$ such that $|G_i| = \lfloor \frac{|D_i|}{\beta} \rfloor$ and they are mutually disjoint. This step is the same as in [4].
   (b) For each item i for which $G_i = \emptyset$ but $\overline{R_i} \neq \emptyset$, we select a disk g_i. Let $G'_i = G_i$ if G_i is not empty and $G'_i = \{g_i\}$ otherwise.

We now describe the details of Step 2 and Step 3.

Step 2: Select R_i for each item i. Let D_{ik} ($k = 1, \ldots, \lceil \frac{|D_i|}{q} \rceil$) be k-th subgroup of D_i. The size of D_{ik} is q for $k = 1, \ldots, \lfloor \frac{|D_i|}{q} \rfloor$ and D_{ik}, $k = \lfloor \frac{|D_i|}{q} \rfloor + 1$ consists of the remaining $|D_i| - q \cdot \lfloor |D_i|/q \rfloor$ disks (the last set is possibly empty). We make use of the following theorem by Shmoys and Tardos to choose R_i.

Theorem 3. *(Shmoys-Tardos [19]) We are given a collection of jobs $\mathcal{J}$, each of which is to be assigned to exactly one machine among the set $\mathcal{M}$; if job $j \in \mathcal{J}$ is assigned to machine $i \in \mathcal{M}$, then it requires p_{ij} units of processing time, and incurs a cost c_{ij}. Suppose that there exists a fractional solution (that is, a job can be assigned fractionally to machines) with makespan P and total cost C. Then in polynomial time we can find a schedule with makespan $P + \max p_{ij}$ and total cost C.*

We can think of each subgroup D_{ik} as a job and each disk as a machine. If disk j belongs to D_i, then we can assign job D_{ik} to disk j with zero cost. The processing time is the size of D_{ik}, which is at most q. If disk j does not belongs to D_i, then the cost to assign D_{ik} to j is ∞ (disk j cannot be in R_i).

Lemma 1. *There exists a fractional assignment such that the max load of each disk is at most β.*

Proof. We can assign $\frac{1}{|D_i|}$ fraction of subgroup D_{ik} to each disk $j \in D_i$. It is easy to check that every subgroup D_{ik} is completely assigned. The load on disk j is given by

$$\sum_{i:j \in D_i} \sum_k \frac{|D_{ik}|}{|D_i|} = \sum_{i:j \in D_i} \frac{1}{|D_i|} \sum_k |D_{ik}| = \sum_{i:j \in D_i} 1 \leq \beta$$

Lemma 2. *There is a way to choose R_i sets for each $i = 1 \ldots \Delta$ and assign subgroups D_{ik} such that for each disk in R_i the total size of subgroups D_{ik} assigned to the disk is at most $\beta + q$.*

Proof. By Theorem 3, we can convert the fractional solution obtained in Lemma 1 to a solution such that each subgroup is completely assigned to one disk, and the maximum load on a disk is at most $\beta + q$ as maximum size of D_{ik} is q.

Let r_i be the disk in R_i that is assigned the small subgroup (a subgroup with size strictly less than q). Note that if $|D_i|$ is a multiple of q, there is no disk r_i. We define $\overline{R_i}$ to be $R_i \setminus r_i$. We will need the following fact later in the algorithm.

Fact. *For each disk j, at most $\beta/q + 1$ different large subgroups D_{ik} (of size exactly q) can be assigned to the disk j.*

Step 3: Select $G'_i \subseteq D_i$. We can find disjoint sets $G_i \subseteq D_i$ using the same algorithm as in [4]. To deal with the remaining items i for which $G_i = \emptyset$ but $\overline{R_i} \neq \emptyset$, we find a disk g_i. Note that if $|G_i| = 0$ then $|D_i| < \beta$, and therefore, $|\overline{R_i}| < \beta/q$. We define G'_i to be G_i if $G_i \neq \emptyset$ and $G'_i = g_i$ otherwise.

Lemma 3. *For each item i for which $G_i = \emptyset$ but $\overline{R_i} \neq \emptyset$, we can find g_i so that for a disk j, $\sum_{i:j=g_i} |\overline{R_i}| \leq 2\beta/q + 1$.*

Proof. We reduce the problem to the following scheduling problem. In this problem, each disk acts like a machine. For each item such that $|G_i| = 0$ we create a job of size $|\overline{R_i}|$. The cost of assigning job i to disk j is 1 iff $j \in \overline{R_i}$, otherwise it is infinite. Note that there is a fractional assignment such that the load to each disk is at most $\beta/q + 1$. (Assign each job fractionally $(\frac{1}{|\overline{R_i}|})$ to each machine (disk) in its $\overline{R_i}$ set. The load due to this job on the machine (disk) is 1. Since a disk is in at most $\beta/q + 1$ different $\bar{R}_i$ sets, its fractional load is at most $\beta/q + 1$.) By applying the Shmoys-Tardos [19] scheduling algorithm (see Theorem 3), we can find an assignment of jobs (items) to machines (disks) such that the total cost is at most the number of items and the load on each machine (disk) is at most $2\beta/q + 1$. (Note that the size of each job is at most β/q.) Let g_i denote the disk that item i is assigned to.

2.2 Performing Data Migrations

1. Send data item i from S_i to G'_i. For this step, we first send items from S_i to a subset of G'_i. We have to carefully choose which disk in S_i sends to a disk in G'_i (see Lemma 4). For sets with $|G'_i| > 1$, note that those G'_i sets are disjoint. Therefore, we can double the number of copies in every round (cloning) once each set receives at least one copy.
2. Send item i from G'_i to $\overline{R_i} \setminus G'_i$. We can create a transfer graph with maximum degree and multiplicity $O(\beta/q)$.
3. Send item i from s_i to r_i if r_i has not received item i. This step can be done in $3\alpha/2$ rounds.

4. We now create a transfer graph from R_i to $D_i \setminus R_i$. We find an edge coloring of the transfer graph and the number of colors used is an upper bound on the number of rounds required to ensure that each disk in D_i gets item i. In Lemma 5 we derive an upper bound on the number of required colors.

We describe the details of each step in data migration.

Step 1: Sending item i from S_i to G'_i. In the first step, we send data from S_i to G'_i. We claim that this can be done in $2OPT + O(\beta/q)$ rounds. We develop a lowerbound on the optimal solution by solving the following linear program $\mathbf{L(m)}$ for a given m.

$$\mathbf{L(m)}: \quad \sum_j \sum_{k=1}^{m} n_{ijk} x_{ijk} \geq |G'_i| \quad \text{for all } i \tag{1}$$

$$0 \leq x_{ijk} \leq 1 \tag{2}$$

where $n_{ijk} = \min(2^{m-k}, |G'_i|)$ if disk j belongs to S_i and $n_{ijk} = 0$ otherwise. Intuitively, x_{ijk} indicates that at time k, disk j send item i to some disk in G'_i. Let M be the minimum m such that $\mathbf{L(m)}$ has a feasible solution. Note that M is a lowerbound for the optimal solution.

Lemma 4. *We can perform migrations from S_i to G'_i in $2 \cdot M + O(\beta/q)$ rounds.*

Proof. Given a fractional solution x^* to $L(M)$, we can obtain an integral solution x^{**} such that for all i, $\sum_j \sum_k x^{**}_{ijk} \geq \lfloor \sum_j \sum_k x^*_{ijk} \rfloor$ (see [4] for details). For each item i, we arbitrarily select $\min(\sum_j \sum_k x^{**}_{ijk}, |G'_i|)$ disks from G'_i. Let H_i denote this subset. We create the following transfer graph from S_i to H_i: create an edge from a disk $j \in S_i$ to a disk H_i if $x^{**}_{ijk} = 1$. (Make sure every disk in H_i has an incoming edge from a disk in S_i.) Note the indegree of a disk in this transfer graph is $2 + \beta/q$ since a disk can belong to H_i for at most $2 + \beta/q$ different items i. (A disk can be g_i for at most $\beta/q + 1$ different items and also may belong to one G_i.) The outdegree is M and the multiplicity is $2\beta/q + 4$. Therefore, we can perform the migration from S_i to H_i in $M + O(\beta/q)$ rounds. For items with $|G'_i| = 1$, we are done for this step. For other items, since sets $G'_i (= G_i)$ are disjoint, we can double the number of copies in each round until the number of copies becomes $|G_i|$. After M rounds, the number of copies we can make for item i is at least

$$
\begin{aligned}
2^M |H_i| = 2^M \min(\sum_j \sum_k x^{**}_{ijk}, |G_i|) \\
&\geq \min(2^{M-1} \cdot 2 \sum_j \sum_k x^{**}_{ijk}, |G_i|) \\
&\geq \min(2^{M-1} \cdot (\sum_j \sum_k x^{**}_{ijk} + 1), |G_i|) \\
&\geq \min(2^{M-1} \sum_j \sum_k x^{*}_{ijk}, |G_i|) \\
&\geq \min(\sum_j \sum_k n_{ijk} x^{*}_{ijk}, |G_i|) \geq |G_i|.
\end{aligned}
$$

The second inequality comes from the fact that $\sum_j \sum_k x^{**}_{ijk} \geq 1$. Therefore we can finish the first step in $2 \cdot M + O(\beta/q)$ rounds.

Step 2: Sending item i from G'_i to $\overline{R_i}$. We now focus on sending item i from the disks in G'_i to disks in $\overline{R_i}$. We construct a transfer graph to send data from G'_i to $\overline{R_i}$ sets so that each disk in $\overline{R_i} \setminus G'_i$ receives item i from one disk in G'_i. We create edges as follows: Add directed edges from disks in G_i to disks in $\overline{R_i}$ first. Recall that $|G_i| = \lfloor \frac{|D_i|}{\beta} \rfloor$ and $|\overline{R_i}| = \lfloor \frac{|D_i|}{q} \rfloor$. Since G_i sets are disjoint, there is a transfer graph where each disk in G_i has at most $\Theta(\beta/q)$ outgoing edges. For items with $G_i = \emptyset$, we create edges from g_i to all $\overline{R_i}$. The outdegree of the disks can be increased by at most $2\beta/q + 1$. The indegree of a disk in $\overline{R_i}$ is at most $\beta/q + 1$ and the multiplicity is $2\beta/q + 2$. Therefore, this step can be done in $O(\beta/q)$ rounds.

Step 3: Sending item i from s_i to r_i. We create a transfer graph where there is an edge from s_i to r_i if r_i has not received item i in the previous steps. The indegree of a disk j is at most β_j since a disk j is selected as r_i only if $j \in D_i$ and the outdegree of disk j is at most $\alpha - \beta_j$. Using Theorem 2, this step can be done in $3\alpha/2$ rounds.

Step 4: Sending item i from R_i to $D_i \setminus (R_i \bigcup G'_i)$. We now create a transfer graph from R_i to $D_i \setminus (R_i \bigcup G'_i)$ such that there is an edge from disk $a \in R_i$ to disk b if the subgroup that b belongs to is assigned to a in Lemma 2. We find an edge coloring of the transfer graph. The following lemma gives an upper bound on the number of rounds required to ensure that each disk in D_i gets item i.

Lemma 5. *The number of colors we need to color the transfer graph is at most $3\beta + q$.*

Proof. First, we compute the maximum indegree and outdegree of each node. The outdegree of a node is at most $\beta + q$ due to the way we choose R_i (See

Lemma 2). The indegree of each node is at most β since in the transfer graph we send items only to the disks in their corresponding destination sets. Multiplicity of the graph is also at most β since we send data item i from disk j to disk k (or vice versa) only if both disk j and k belong to D_i. By Theorem 1, we see that the maximum number of colors needed is at most $3\beta + q$.

To wrap up, in the next theorem we show that the total number of rounds in this algorithm is bounded by 6.5+o(1) times the optimal solution.

Theorem 4. *The total number of rounds required for the data migration is at most* $6.5 + o(1)$ *times OPT.*

Proof. The total number of rounds we need is $2M+3\alpha/2+3\beta+O(\beta/q)+q$. Since M, α, and β are the lowerbounds on the optimal solution, chooosing $q = \Theta(\sqrt{\beta})$ gives the desired result.

3 External Disks

Until now we assumed that we had N disks, and the source and destination sets were chosen from this set of disks and only essential transfers are performed. In other words, if an item i is sent to disk j, then it must be that $j \in D_i$ (disk j was in the destination set for item i), hence the total number of transfers done is the least possible. In several situations, we may have access to idle disks with available storage that we can make use of as temporary devices to enable a faster completion of the transfers we are trying to schedule. In addition, we exploit the fact that by performing a small number of non-essential transfers (this was also used in [13, 10]), we can further reduce the total number of rounds required. We show that indeed such techniques can considerably reduce the total number of rounds required for performing the transfers from S_i sets to D_i sets.

We assume that each external disk has enough space to pack γ items. If we are allowed to use $\lceil \frac{\Delta}{\gamma} \rceil$ external disks, the approximation ratio can be improved to $3 + \max(1.5, \frac{\gamma}{2})$. For example, choosing $\gamma = 3$ gives a bound of 4.5.

Define $\bar{\beta} = \sum_{i=1}^{\Delta} \frac{|D_i|}{N}$. We can see that $2\bar{\beta}$ is a lowerbound on the optimal number of rounds since in each round at most $\lfloor \frac{N}{2} \rfloor$ data items can be transferred. The high level description of the algorithm is as follows:

1. Assign γ items to each external disk. Send items to their assigned external disks.
2. For each item i, choose disjoint $\overline{G_i}$ sets of size $\lfloor \frac{D_i}{\bar{\beta}} \rfloor$.
3. Send item i to all disks in the $\overline{G_i}$ set.
4. Send item i from the $\overline{G_i}$ set to all the disks in D_i. We will also make use of the copy of item i on the external disk.

We now discuss the steps in detail.

First step can be done in at most $\max(\alpha, \gamma)$ rounds by sending the items from their primary sources to the external disks (for this step we will compute α as before, with the change that we can ignore the β_j term). The maximum degree of each disk is at most $\max(\alpha, \gamma)$. Since the graph is bipartite, transferring items to their assigned external disks can be finished in $\max(\alpha, \gamma)$ rounds.

We can easily choose disjoint set $\overline{G_i}$ as we are allowed to perform non-essential transfers (i.e., a disk j can belong to $\overline{G_i}$ even if j is not in D_i.) Hence we can use a simple greedy method to choose $\overline{G_i}$. Broadcasting items inside $\overline{G_i}$ can be done in $2M$ rounds as described in Section 2.

Next step is to send the item to all the remaining disks in the D_i sets. We make a transfer graph as follows: assign to each disk in $\overline{G_i}$ at most $\bar{\beta}$ disks in D_i so that each disk in D_i is assigned to at most one disk in the $\overline{G_i}$ set. The number of unassigned disks from each D_i set is at most $\bar{\beta}$. Assign all of the remaining disks from D_i to the external disk containing that item. The outdegree of the internal disks is at most $\bar{\beta}$ since each disk belongs to at most one $\overline{G_i}$ set. The indegree of each internal disk is at most β since a disk will receive an item only if it is in its demand set. The multiplicity between two internal disks is at most 2. (Since each disk can belong to at most one $\overline{G_i}$ set.) So the total degree of each internal disk is at most $\beta + \bar{\beta}$. Each external disk has at most γ items and the number of remaining disks for each item is at most $\bar{\beta}$. So the outdegree of each external disk is at most $\gamma\bar{\beta} \leq \frac{\gamma}{2}OPT$.

So the maximum degree of each node in the whole graph is at most $\max(\beta + \bar{\beta}, \gamma\bar{\beta})$. and the maximum number of colors needed to color this graph is $\frac{1}{2}\max(3,\gamma)OPT + \max(2,\gamma)$. Adding up all these values the complete transfer can be done in $\alpha + 2m' + 3 + \frac{1}{2}\max(3,\gamma)OPT + \max(2,\gamma) \leq (3 + \frac{1}{2}\max(3,\gamma))OPT + 2\gamma + O(1)$.

4 Full Duplex Model

In this section we consider the full duplex communication model. In this model, we assume that each disk can send and receive at most one item in each round. In the half-duplex model, we assumed that at each round, a disk can either send or receive one item (but not both at the same time). In the full duplex model the communication pattern does not have to induce a matching since directed cycles are allowed (the direction indicates the data transfer direction).

We develop a $4 + o(1)$ approximation algorithm for this model. In this model, given a transfer graph G, we find an optimal migration schedule for G as follows: Construct a bipartite graph by putting one copy of each disk in each partition. We call the copy of vertex u in the first partition u_A, and in the other partition u_B. We add an edge from u_A to v_B in the bipartite graph if and only if there is a directed edge in the transfer graph from u to v. The bipartite graph can be colored optimally in polynomial time and the number of colors is equal to the maximum degree of the bipartite graph.

Note that β and M are still lower bounds on the optimal solution in the full-duplex model. The algorithm is the same as in Section 2 except the procedure to select primary sources s_i.

– For each item i, decide a primary source s_i so that $\alpha' = \max_{j=1\ldots N}(\max(|\{j|j = s_i\}|, \beta_j))$ is minimized. Note that α' is also a lower bound for the optimal solution. We can find these primary sources as shown in Lemma 6 by adapting the method used in [4].

We show how to find the primary sources s_i.

Lemma 6. *By using network flow we can choose primary sources to minimize* $\max_{j=1\ldots N}(\max(|\{j|j = s_i\}|, \beta_j))$

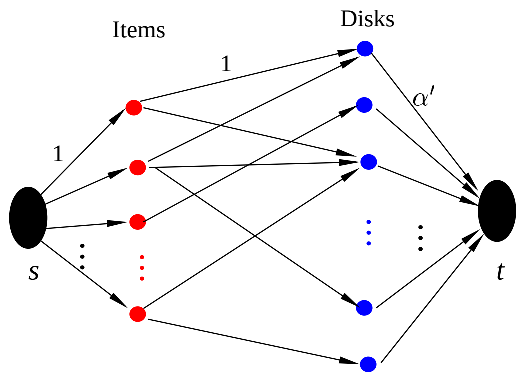

Fig. 2. Computing α'.

Proof. Create two vertices s and t. (See Figure 4 for example.) Make two sets, one for the items and one for the disks. Add edges from s to each node corresponding to an item of unit capacity. Add a directed edge of infinite capacity between item j and disk i if $i \in S_j$. Add edges of capacity α' from each node in the set of disks to t. Find the minimum α' (initially $\alpha' = \beta$), so that we can find a feasible flow of value Δ. For each item j, choose the disk as its primary source s_j to which it sends one unit of flow.

Theorem 5. *There is a* $4 + o(1)$ *approximation algorithm for data migration in the full duplex model.*

Proof. Step 1 (from S_i to G'_i) and Step 2 (from G'_i to $\overline{R_i}$) still take $2M + O(\beta/q)$ rounds and $O(\beta/q)$ rounds, respectively. For Step 3, if we construct a bipartite graph, then the max degree is at most $\max(\alpha', \beta)$, which is the number of rounds required for this step. For Step 4, the maximum degree of the bipartite graph is $\beta + q$. Therefore, the total number of rounds we need is $2M + \max(\alpha', \beta) + \beta + O(\beta/q) + q$. By choosing $q = \Theta(\sqrt{\beta})$, we can obtain a $4 + o(1)$-approximation algorithm.

References

1. E. Anderson, J. Hall, J. Hartline, M. Hobbes, A. Karlin, J. Saia, R. Swaminathan and J. Wilkes. An Experimental Study of Data Migration Algorithms. *Workshop on Algorithm Engineering*, pages 145–158, London, UK, 2001. Springer-Verlag
2. G. Aggarwal, R. Motwani and A. Zhu. The load rebalancing problem. *Symp. on Parallel Algorithms and Architectures*, pages 258–265, (2003).
3. I. D. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. *Proc. of ACM-SIAM SODA*, pp. 661–670, 2001.
4. S. Khuller, Y.A. Kim and Y.C. Wan. Algorithms for Data Migration with Cloning, *Siam J. on Comput.*, Vol. 33, No. 2, pp. 448–461,Feb. 2004.
5. J. A. Bondy and U. S. R. Murty. Graph Theory with Applications. *American Elsevier*, New York, 1977.
6. R. Gandhi and J. Mestre. Combinatorial algorithms for Data Migration to minimize the average completion time. *APPROX* (2006) (to appear).
7. L. Golubchik, S. Khanna, S. Khuller, R. Thurimella and A. Zhu. Approximation Algorithms for Data Placement on Parallel Disks. *Proc. of ACM-SIAM SODA*, pages 661–670, Washington, D.C., USA, 2000. Society of Industrial and Applied Mathematics.
8. L. Golubchik, S. Khuller, Y. Kim, S. Shargorodskaya and Y. C. Wan. Data migration on parallel disks. *Proc. of European Symp. on Algorithms* (2004). LNCS 3221, pages 689–701. Springer. To appear in Special Issue of *Algorithmica* from ESA 2004.
9. S. Guha and K. Munagala. Improved algorithms for the data placement problem, 2002. *Proc. of ACM-SIAM SODA*, pages 106–107, San Fransisco, CA, USA, 2002. Society of Industrial and Applied Mathematics.
10. J. Hall, J. Hartline, A. Karlin, J. Saia and J. Wilkes. On Algorithms for Efficient Data Migration. *Proc. of ACM-SIAM SODA*, pp. 620–629, 2001.
11. S. Kashyap and S. Khuller. Algorithms for Non-Uniform Size Data Placement on Parallel Disks. *Conference on FST&TCS Conference*, LNCS 2914, pp. 265–276, 2003. Full version to appear in *Journal of Algorithms* (2006).
12. S. Kashyap, S. Khuller, Y. C. Wan and L. Golubchik. Fast reconfiguration of data placement in parallel disks. *2006 ALENEX Conference*, Jan 2006.
13. S. Khuller, Y. Kim and Y. C. Wan. On Generalized Gossiping and Broadcasting. *ESA Conference*. pages 373–384, Budapest, Hungary, 2003. Springer.
14. Y. Kim. Data Migration to minimize the average completion time. *Proc. of ACM-SIAM SODA*, pp. 97–98, 2003.
15. A. Meyerson, K. Munagala, and S. A. Plotkin. Web caching using access statistics. In *Symposium on Discrete Algorithms*, pages 354–363, 2001.
16. H. Shachnai and T. Tamir. On Two Class-constrained Versions of the Multiple Knapsack Problem. *Algorithmica*, 29:442–467, 2001.
17. H. Shachnai and T. Tamir. Polynomial Time Approximation Schemes for Class-constrained Packing Problems. *Workshop on Approximation Algorithms*, LNCS 1913, pp. 238–249, 2000.
18. C.E. Shannon. A Theorem on Colouring Lines of a Network. *J. Math. Phys.*, 28:148–151, 1949.
19. D.B. Shmoys and E. Tardos. An Aproximation Algorithm for the Generalized Assignment Problem. *Mathematical Programming*, A 62, pp. 461–474, 1993.
20. V. G. Vizing. On an Estimate of the Chromatic Class of a p-graph (Russian). *Diskret. Analiz.* 3:25–30, 1964.