Algorithms for Data Migration with Cloning*

Samir Khuller † Yoo-Ah Kim ‡ Yung-Chun (Justin) Wan §

Abstract

Our work is motivated by the problem of managing data on storage devices, typically a set of disks. Such high demand storage servers are used as web servers, or multimedia servers for handling high demand for data. As the system is running, it needs to dynamically respond to changes in demand for different data items. In this work we study the *data migration problem*, which arises when we need to quickly change one storage configuration into another. We show that this problem is NP-hard. In addition, we develop polynomial time approximation algorithms for this problem and prove a worst case bound of 9.5 on the approximation factor achieved by our algorithm. We also compare the algorithm to several heuristics for this problem.

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. With the recent interest in autonomic computing[1], a relevant goal is to have the system automatically respond to changes in demand patterns and to recompute data layouts.

Approximation algorithms have been developed [21, 22, 9, 16] to map known demand for data to a specific data layout pattern to maximize utilization[2]. 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 [9, 22, 16]. Given the relative demand for data, the algorithm computes an almost optimal layout.

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. The crucial constraint is that each disk can participate in the transfer of only one item – either as a sender or as a receiver. Our goal is to find a migration schedule to minimize the time taken to complete the migration (makespan).

A special case of this was studied in [10]—they compute a movement schedule, but this *does not allow* the creation of new copies of any data object. It addresses only the data *movement* problem. (So for example, one cannot create extra copies of any data item, but can just change which disks they are stored on.) The problem studied in [10] is formally defined as follows: given a set of disks, with each storing a subset of items and a specified set of move operations (each move operation specifies which data object needs to be moved from one disk to another), how do we schedule these move operations? If there are no storage constraints, then this is exactly the

*Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This research was supported by NSF Awards CCR-9820965 and CCR-0113192.

†E-mail : `samir@cs.umd.edu`.

‡E-mail : `ykim@cs.umd.edu`.

§Contact Author Phone: (301) 405-2857. Fax: (301) 314-1353. E-mail : `ycwan@cs.umd.edu`.

[1] `http://www.research.ibm.com/autonomic/`

[2] Utilization refers to the total number of clients that can be assigned to a disk that contains the data they want.

problem of edge-coloring the following multi-graph. Create a graph that has a node corresponding to each disk, and a directed edge corresponding to each move operation that is specified. Algorithms for edge-coloring multigraphs can now be applied to produce a migration schedule since each color class represents a matching in the graph that can be scheduled simultaneously. Computing a solution with the minimum number of rounds is NP-hard, but several good approximation algorithms are available for edge coloring. With space constraints on the disk, the problem becomes challenging. In [10] it is shown that with the assumption that each disk has one spare unit of storage, very good constant factor approximations can be developed. The algorithms use at most $4\lceil\Delta/4\rceil$ colors with at most $n/3$ bypass nodes, or at most $6\lceil\Delta/4\rceil$ colors without bypass nodes[3].

On the other hand, to handle high demand for popular objects, new copies will have to be dynamically created and stored on different disks. This means that we crucially need the ability to have a "copy" operation in addition to "move" operations. In fact, one of the crucial lower bounds used in the work on data migration [10] is based on a degree property of the multi-graph. For example, if the degree of a node is Δ, then this is a lower bound on the number of rounds that are required, since in each round at most one transfer operation involving this node may be done. For copying operations, clearly this lower bound is not valid. For example, suppose we have a single copy of a data item on a disk. Suppose we wish to create Δ copies of this data item on Δ distinct disks. Using the transfer graph approach, we could specify a "copy" operation from the source disk to each of the Δ disks. Notice that this would take at least Δ rounds. However, by using newly created copies as additional sources we can create Δ copies in $\lceil\log(\Delta + 1)\rceil$ rounds, as in the classic problem of broadcasting by using newly created copies as sources for the data object. (Essentially each copy spawns a new copy in each round.)

The *most general problem* of interest is the **data migration problem with cloning** when 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. We show that this problem is NP-hard by reduction from edge coloring and develop a polynomial time 9.5-approximation algorithm for it. The algorithm has been implemented by Svetlana Shargorodskaya [24] and we are comparing its performance to several other heuristics for the problem. For all our algorithms, we move data only to disks that need the data. Thus we use no bypass nodes. The total number of data transfers performed is thus the minimum possible.

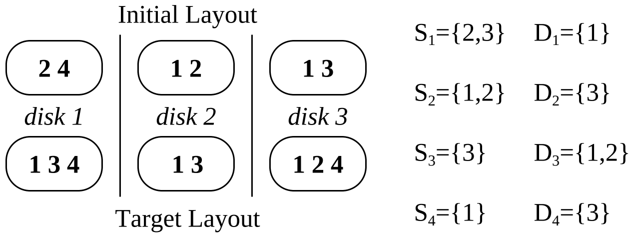

Figure 1: An initial and target layouts, and their corresponding S_i's and D_i's

Different communication models can be considered based on how the disks are connected. We use the same model as in the work by [10, 1] 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). Later we will also discuss a restricted model where the devices may communicate on a matching, but the size of the matching is constrained (we call this the bounded-matching model).

One interesting generalization would be for the situation when clusters of disks are connected in a wide area network. The time required to transfer one unit of data between a pair of disks in different clusters may be an order of magnitude higher

[3] A bypass node is a node that is not the target of a move operation, but used as an intermediate holding point for a data item.

than the time required to transfer data between a pair of disks in the same cluster. We can model this by a communication graph model where the number of rounds required to transfer one unit of data between a pair of disks in different clusters is a certain number of rounds, and one round is required to transfer one unit of data between a pair of disks in the same cluster.

1.1 Relationship to Gossiping and Broadcasting

The problems of Gossiping and Broadcasting have been the subject of extensive study [17, 12, 14, 3, 4, 15, 7, 8]. These play an important role in the design of communication protocols in various kinds of networks. The *gossip problem* is defined as follows: there are n individuals. Each individual has an item of gossip that they wish to communicate to everyone else. Communication is typically done in rounds, where in each round an individual may communicate with at most one other individual. Some communication models that allow for the full exchange of all items of gossip known to each individual in a single round. Other models allow the sending of only one item of gossip from one to the other (half-duplex) or allow each individual to send an item to the individual they are communicating with in this round (full-duplex). In addition, there may be a communication graph whose edges indicate which pairs of individuals are allowed to communicate directly in each round. (In the classic gossip problem, also called the telephone model, communication may take place between any pair of individuals; in other words, the communication graph is the complete graph.) In the *broadcast problem*, one individual needs to convey an item of gossip to every other individual. The two parameters typically used to evaluate the algorithms for this problem are: the number of communication rounds, and the total number of telephone calls placed.

The problems we study are generalizations of the above mentioned gossiping and broadcasting problems. The basic generalizations we are interested in are of two kinds (a) each item of gossip needs to be communicated to only a subset of individuals, and (b) several items of gossip may be known to an individual.

The communication model we use is the half-duplex model, where only one item of gossip may be communicated between two communicating individuals during a single round. Each individual may communicate (either send or receive an item of data) with at most one other individual in a round. *This model best captures the connection of parallel storage devices that are connected on a network and is most appropriate for our application.* This model is one of the most widely used in all the work related to gossiping and broadcasting.

1.2 Other Communication Models

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 as *bounded-size matching model.* Using the constant factor approximation algorithm for the full matching model, we can also develop a constant factor approximation algorithm for the bounded-size matching model. Details omitted from this abstract.

1.3 Contributions and Outline of Paper

In Section 2 we define the basic model of communication and the notation used in the paper.

This is the first work that relates broadcasting and gossiping with the data migration problem. In Section 3 we develop a 9.5 approximation algorithm for the general data migration problem. This is the first approximation algorithm for this problem. While two of the lower bounds used are quite simple, we develop a new lower bound using network flows, and use this in the algorithm. (Without this lower bound, the best bound one can obtain is a $O(\log n)$ factor.) We show the data migration problem is NP-hard at the end of Section 3.

In Section 4 we give description of other heuristics and briefly compare their performance.

1.4 Related Work

The paper by Liben-Nowell [19] considers a problem very similar to multi-source multicast which is exactly the data migration problem with restrictions that each disk contains at most one source item and each item has at most one source. However, the model that he uses is different than the one that we use. In his model, in each telephone call, a pair of users can exchange all the items of gossip that they know. The objective is to simply minimize the total number of phone calls required to convey item i of gossip to set D_i of users. In our case, since each item of gossip is a data item that might take considerable time to transfer between two disks, we cannot assume that an arbitrary number of data items can be exchanged in a single round. Several other papers use the same telephone call model [2, 6, 11, 15, 26].

Other related problems that have been studied are the set-to-set gossiping problem [18, 20] where we are given two possibly intersecting sets A and B of gossipers and the goal is to minimize the number of calls required to inform all gossipers in A of all the gossip known to members in B. Liben-Nowell [19] generalizes this work by defining for each gossiper i the set of relevant gossip that they need to learn. This is just like our multi-source multicast problem with $\Delta = N$, except that the communication model is different, as well as the objective function.

2 Models and Definitions

In the *data migration problem*, we have N disks and Δ data items. For each item i, there is a subset of disks S_i and D_i. Initially only the disks in S_i have item i, and all disks in D_i want to receive i. Note that after a disk in D_i receives item i, it can be a source of item i for other disks in D_i that have not received the item as yet. Our goal is to find a migration schedule using the minimum number of rounds, that is, to minimize the total amount of time to finish the schedule. We assume that the underlying network is fully connected and the data items are all the same size, in other words, it takes the same amount of time to migrate an item from one disk to another. The crucial constraint is that each disk can participate in the transfer of only one item - either as a sender or receiver. Moreover, as we do not use any bypass nodes, all data is only sent to disks that desire it.

Our algorithms make use of known results on edge coloring of multi-graphs. Given a graph G with max degree Δ_G and multiplicity μ the following results are known (see [5] for example). Let χ' be the edge chromatic number of G.

THEOREM 2.1. *(Vizing [27]) If G has no self-loops then $\chi' \leq \Delta_G + \mu$.*

THEOREM 2.2. *(Shannon [23]) If G has no self-loops then $\chi' \leq \lfloor \frac{3}{2}\Delta_G \rfloor$.*

3 The Data Migration Algorithm

Define β_j as $|\{i | j \in D_i\}|$, i.e., the number of different sets D_i, that a disk j belongs to. We then define β as $\max_{j=1 \dots N} \beta_j$. 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 i that attains the maximum, needs at least β rounds to receive all the items j such that $i \in D_j$, since it can receive at most one item in each round.

Moreover, we may assume that $D_i \neq \emptyset$ and $D_i \cap S_i = \emptyset$. (We simply define the destination set D_i as the set of disks that need item i and do not currently have it.)

Since the algorithm is somewhat complex, we first give a high level description of the algorithm and then discuss the various steps in the following Lemmas. Dealing with multiple data items sharing common disks causes some difficulty.

Algorithm Data Migration.

1. For an item i decide a unique source $s_i \in S_i$ so that $\alpha = \max_{j=1,\dots,N}(|\{i | j = s_i\}| + \beta_j)$ is minimized. In other words, α is the maximum number of items that a disk may be a source (s_i) or destination for. *Note that α is also a lower bound on the optimal number of rounds.* In Lemma 3.1 will show how we can find a source for each item.

2. Find a transfer graph for items such that $|D_i| \geq \beta$ as follows.

   (a) We first compute a disjoint collection of subsets $G_i, i = 1 \dots \Delta$. Moreover, $G_i \subseteq D_i$ and $|G_i| = \lfloor \frac{|D_i|}{\beta} \rfloor$. (In Lemma 3.2, we will show how such G_i's can be obtained.)

   (b) We have each item i sent to the set G_i as shown in Lemma 3.5.

   (c) We now create a transfer graph as follows. Each disk is a node in the graph. We add directed edges from disks in G_i to

$(\beta-1)\lfloor\frac{|D_i|}{\beta}\rfloor$ disks in $D_i \setminus G_i$ such that the out-degree of each node in G_i is at most $\beta - 1$ and the in-degree of each node in $D_i \setminus G_i$ is 1. Figure 2 shows an example of the transfer graph constructed in this step. We redefine D_i as a set of $|D_i \setminus G_i| - (\beta - 1)\lfloor\frac{|D_i|}{\beta}\rfloor$ disks which do not receive item i so that they can be taken care of in Step 3. Note that the redefined set D_i has size $< \beta$.

3. Find a transfer graph for items such that $|D_i| < \beta$ as follows.

   (a) For each item i, find a new source s_i' in D_i. A disk j can be a source s_i' for several items as long as $\sum_{i \in I_j} |D_i| \le 2\beta - 1$ where I_j is a set of items of which j is a new source. See Lemma 3.7 for the details of this step.

   (b) Send each item i from s_i to s_i'.

   (c) Create a transfer graph. We add a directed edge from the new source of item i to all disks in $D_i \setminus \{s_i'\}$. Lemma 3.9 will show that the out-degree of a disk does not exceed $2\beta - 4$.

4. We now find an edge coloring of the transfer graph obtained by merging two transfer graphs in Step 2(c) and 3(c). The number of colors used is an upper bound on the number of rounds required to ensure that each disk in D_j gets item j. In Lemma 3.10 we derive an upper bound on the number of required colors.

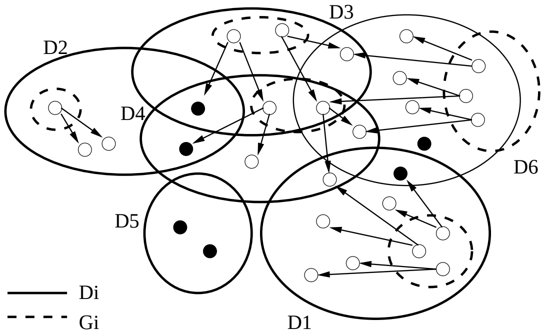

Figure 2: An example of the transfer graph constructed in Step 2(c) where $\beta = 3$. Disks marked as black do not receive some data items and will be taken care of in Step 3.

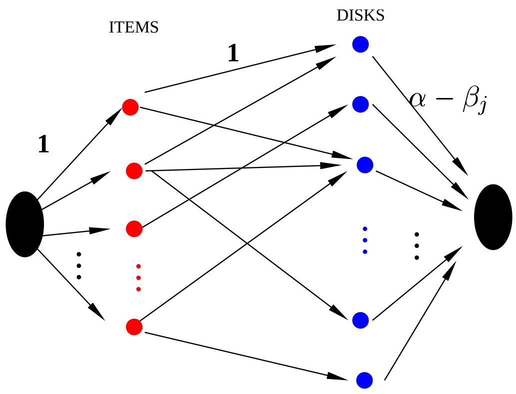

Figure 3: Flow network to find α

Lemma 3.1. *We can find a source $s_i \in S_i$ for each item i so that $\max_{j=1,\ldots,N}(|\{i | j = s_i\}| + \beta_j)$ is minimized, using a flow network.*

Proof. We create a flow network with a source s and a sink t as shown in Figure 3. We have two set of nodes corresponding to disks and items. Add directed edges from s to nodes for items and also directed edges from item i to disk j if $j \in S_i$. The capacities of all those edges are one. Finally we add an edge from the node corresponding to disk j to t with capacity $\alpha - \beta_j$. We want to find the minimum α so that the maximum flow of the network is Δ. We can do this by checking if there is a flow of Δ with α starting from $\max \beta_j$ and increasing by one until it is satisfied. If there is outgoing flow from item i to disk j, then we set j as s_i. $\square$

Lemma 3.2. *There is a way to choose disjoint sets G_i for each $i = 1 \ldots \Delta$, such that $|G_i| = \lfloor\frac{|D_i|}{\beta}\rfloor$ and $G_i \subseteq D_i$.*

Proof. First note that the total size of the sets G_i is at most N.

$$\sum_i |G_i| \le \sum_i \frac{|D_i|}{\beta} = \frac{1}{\beta}\sum_i |D_i|.$$

Note that $\sum_i |D_i|$ is at most βN by definition of β. This proves the upper bound of N on the total size of all the sets G_i.

We now show how to find the sets G_i. We create a flow network with a source s and sink t. In addition we have two sets of vertices U and W. The first set U has Δ nodes, each corresponding to a disk that is the source of an item. The set W has N nodes, each corresponding to a disk in the system. We add directed edges from s to each node in U, such that the edge (s, i) has capacity $\lfloor \frac{|D_i|}{\beta} \rfloor$. We also add directed edges with infinite capacity from node $i \in U$ to $j \in W$ if $j \in D_i$. We add unit capacity edges from nodes in W to t. We find a max-flow from s to t in this network. The min-cut in this network is obtained by simply selecting the outgoing edges from s. We can find a fractional flow of this value as follows: saturate all the outgoing edges from s. From each node i there are $|D_i|$ edges to nodes in W. Suppose $\lambda_i = \lfloor \frac{|D_i|}{\beta} \rfloor$. Send $\frac{1}{\beta}$ units of flow along $\lambda_i \beta$ outgoing edges from i. Note that since $\lambda_i \beta \leq |D_i|$ this can be done. Observe that the total incoming flow to a vertex in W is at most 1 since there are at most β incoming edges, each carrying at most $\frac{1}{\beta}$ units of flow. An integral max flow in this network will correspond to $|G_i|$ units of flow going from s to i, and from i to a subset of vertices in D_i before reaching t. The vertices to which i has non-zero flow will form the set G_i. $\square$

For Step 2(b), the simple solution would be to broadcast the data to each group G_i from the chosen source, since the groups are disjoint. The only thing we have to be careful of is that the sources for many data items are shared. However, this broadcast takes at least $\max_i \log |G_i|$ rounds. Unfortunately, we cannot argue that this is a valid lower bound since even though D_i is large, if S_i is large, then there could be a solution using $O(1)$ rounds. This would give us an $O(\log N)$ approximation guarantee. The method described below, develops stronger lower bounds for this situation.

Let M be the number of steps required to send all items i to all disks in G_i in an optimal schedule of Step 2(b). To find a lower bound of M, we construct the following flow network F_m (parameterized by an integer m) as shown in Figure 3. We have a source s and two sets of nodes U and V. U has $N \cdot m$ nodes $x_{jk} (j = 1 \ldots N, k = 1 \ldots m)$.

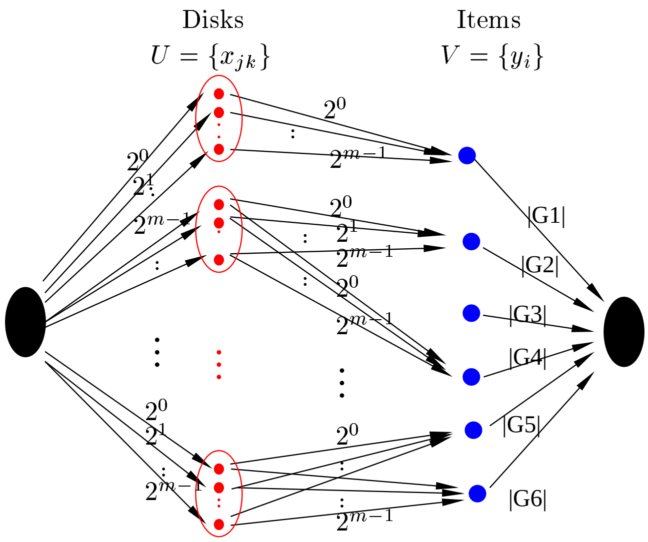

Figure 4: An example of constructing F_m where $\Delta = 6$

V has Δ nodes $y_i (i = 1 \ldots \Delta)$ and y_i has demand $|G_i|$. There is an edge e_{ijk} from x_{jk} to y_i and its capacity c_{ijk} is 2^{m-k} if a disk j has item i initially. There are edges from s to nodes x_{jk} in U with capacity 2^{m-k}.

Lemma 3.3. *If m' be the smallest number such that we can construct a solution of $F_{m'}$ that satisfies all demands $|G_i|$, then $M \geq m'$.*

Proof. Suppose that $M < m'$. Given a optimal schedule of Step 2(b), we can construct a solution of the flow network F_M as follows. If a disk j sends item i to a disk in G_i at round $t \leq M$ and it makes f copies in G_i subsequently, we send a flow f from x_{jt} to y_i. Note that f cannot be more than 2^{M-t} and therefore, it does not violate the capacity constraint. Since all disks in G_i receive item i after M rounds with this schedule, the corresponding flow satisfies all demands $|G_i|$. This is a contradiction to the assumption that m' is the smallest number to satisfy all demands $|G_i|$. $\square$

In the solution of the flow network $F_{m'}$, a node x_{jk} may send flow to several nodes. But since in our schedule, a disk can copy only one item to a disk at a round, the solution of the flow in $F_{m'}$ may not correspond to a valid schedule.

Lemma 3.4. *Given a solution of $F_{m'}$, we can con-*

vert it to a solution satisfying the following properties.

- *node x_{jk} sends flow to at most one node in V.*
- *the solution satisfies at least $|G_i| - 2^{m'-1}$ demands for each item i.*

Proof. First, we define a variable z_{ijk} for an edge from x_{jk} to y_i and set $z_{ijk} = f_{ijk}/c_{ijk}$ where f_{ijk} is the flow through e_{ijk} in solution $F_{m'}$. We substitute nodes $y_{il}(l = 1 \ldots \lfloor\sum_{j,k} z_{ijk}\rfloor)$ for each node y_i in V. We distribute edges having nonzero flow to y_i as follows. Sort edges in non-increasing order of their capacities. Assign edges to y_{i1} until the sum of z values of assigned edges is greater than or equal to one. If the sum is greater than one, we split the last edge (denote as $e_{ij'k'}$) into $e_{ij'k'_1}$ and $e_{ij'k'_2}$. Assign $e_{ij'k'_1}$ to y_{i1} and define $z_{ij'k'_1}$ so that the sum of z values of edges assigned to y_{i1} is exactly one. Set $z_{ij'k'_2} = z_{ij'k'} - z_{ij'k'_1}$. We repeat this so that for all nodes y_{il}, the sums of z values of the assigned edges are one. Let E_{il} be the set of edges assigned to y_{il} and $c^{max}_{il}(c^{min}_{il})$ be the maximum (minimum) capacity of the edges in E_{il}. In addition, we denote the edges not assigned to $y_{il}(l = 1 \ldots \lfloor\sum_{j,k} z_{ijk}\rfloor)$ as $E_{il'}$ and the maximum capacity of edges in $E_{il'}$ as $c^{max}_{il'}$ where l' is $\lfloor\sum_{j,k} z_{ijk}\rfloor + 1$.

In the resulting bipartite graph with U and $V' = \{y_{il}\}$, z makes a *fractional* matching which matches all vertices in V'. Therefore, we can find an integral matching that matches all vertices in V' and the matching satisfies the first property in the lemma.

Now we merge nodes y_{il} into y_i. Then each y_i matches exactly $\lfloor\sum_{j,k} z_{ijk}\rfloor$ edges. We prove that the sum of capacities of edges matched to y_i is at least $|G_i| - c^{max}$ where c^{max} is the maximum capacity of edges, using an analysis similar to that in [25]. The sum of capacities of edges matched to y_i is at least

$$
\begin{aligned}
&\sum_{l=1}^{\lfloor\sum_{j,k} z_{ijk}\rfloor} c^{min}_{il} \geq \sum_{l=2}^{\lfloor\sum_{j,k} z_{ijk}\rfloor+1} c^{max}_{il} \\
\geq\ & \sum_{l=1}^{\lfloor\sum_{j,k} z_{ijk}\rfloor+1} c^{max}_{il} - c^{max}_{i1} \\
\geq\ & \sum_{l=1}^{\lfloor\sum_{j,k} z_{ijk}\rfloor+1} \sum_{e_{ijk}\in E_{il}} c_{ijk} z_{ijk} - c^{max}_{i1} \\
\geq\ & \sum_{l=1}^{\lfloor\sum_{j,k} z_{ijk}\rfloor+1} \sum_{e_{ijk}\in E_{il}} f_{ijk} - c^{max}_{i1} \\
\geq\ & |G_i| - c^{max}.
\end{aligned}
$$

Since $c^{max} \leq 2^{m'-1}$, the second property can be satisfied by setting flow through e_{ijk} as c_{ijk} if e_{ijk} is matched. □

LEMMA 3.5. *Step 2(b) can be done in $\alpha + 2m' + 1$ rounds.*

Proof. We can do this with the following schedule. First we choose $\min(\lfloor\sum_{j,k} z_{ijk}\rfloor + 1, |G_i|)$ disks in G_i and denote those disks as H_i. Disk j sends item i to a disk in H_i if edge e_{ijk} is matched for some k. If $|H_i| > \lfloor\sum_{j,k} z_{ijk}\rfloor$, there is one disk in H_i which cannot receive item i. The disk receives item i from s_i. Then the maximum degree of a disk is at most $m' + \alpha$ and the multiplicity is 2 since the out-degree of disk j is at most $m' + \alpha - \beta_j$ and the in-degree is at most $\min(\beta_j, 1)$. Therefore, it can be done in $m' + \alpha + 2$ rounds.

Now $|H_i|$ nodes in G_i have item i. Since $|G_i|/|H_i| \leq 2^{m'-1}$, we can make all disks in G_i have item i in additional $m' - 1$ rounds. □

Lemma 3.7 will show how Step 3(a) works. The lemma uses the following result from [25].

THEOREM 3.6. *[25] 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 we can find a schedule with makespan $P + \max p_{ij}$ and total cost C.*

LEMMA 3.7. *For each item i we wish to choose a source disk s'_i from D_i. Let I_j be the set of items for which disk j is a source. There is a way to choose the sources such that the following properties hold*

- *If $i \in I_j$ then $j \in D_i$.*

- $\sum_{i \in I_j} |D_i| \leq 2\beta - 1$.

Proof. We use Theorem 3.6 for this step. For example, we can create an instance of the problem of scheduling machines with costs. Items correspond to jobs and disks correspond to machines. For each item i we define a cost function as follows. $C(i, j) = 1$ if and only if $j \in D_i$, otherwise it is a large constant C. Processing time of job i (corresponding to item i) is $|D_i|$ (uniform processing time on all machines). As Theorem 3.6, the scheduling algorithm in [25] finds a schedule that assigns each job (item) to a machine (disk) to minimize the makespan. They show that the makespan is at most the makespan in a fractional solution plus the processing time of the largest job. Moreover, the cost of their solution is at most the cost of the optimal solution, namely the number of items. We cannot assign an item (job) to a disk (machine) if the disk is not in the destination set for the item.

In our case, it is easy to see that the maximum processing time of any job is $\beta - 1$. We will argue that there is a fractional solution with makespan β. It thus follows that by defining I_j to be the set of items (jobs) assigned to disk (machine) j, the result follows. The fractional solution is obtained by assigning each job fractionally to each machine by setting the assignment variable for job i on machine j to $\frac{1}{|D_i|}$ if $j \in D_i$ then the job is fully assigned fractionally and the fractional load on each machine is at most β. This also gives us a way of finding a fractional solution efficiently. □

LEMMA 3.8. *Step 3(b) can be done in* $\lfloor \frac{3}{2}\alpha \rfloor$ *rounds.*

Proof. Since j can be s'_i in Step 3(a) only if $j \in D_i$, $|I_j| \leq \beta_j$. Therefore, a disk j may need to send $\alpha - \beta_j$ items, and receive β_j items. That means the maximum degree is α and this transfer can make a multi-graph. Given a multi-graph with maximum degree Δ_G, we can find an edge coloring using $\lfloor \frac{3}{2}\Delta_G \rfloor$ colors (see Theorem 2.2 [23]) and the lemma follows. □

LEMMA 3.9. *The maximum out-degree of a disk in the transfer graph in Step 3(c) is at most* $2\beta - 4$.

Proof is omitted due to lack of space. Figure 5 shows an example of migrations in Step 3.

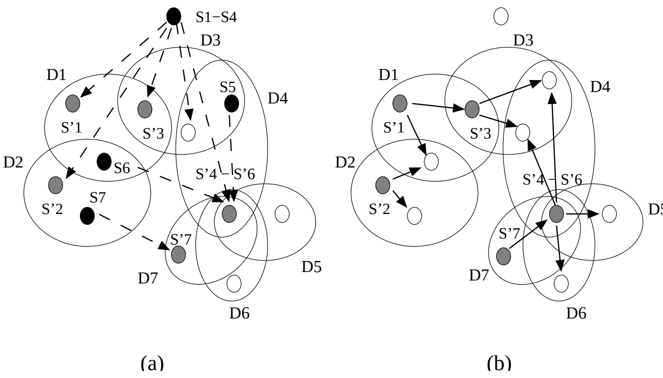

Figure 5: An example of Step 3 where $\alpha = 4$ and $\beta = 4$. (a) migration from s_i to s'_i (b) migration from s'_i to $D_i \setminus \{s'_i\}$.

LEMMA 3.10. *The number of colors we need for the final transfer graph in Step 4 is* $(4\beta - 5) + (\beta + 2)$.

Proof. The out-degree of a disk j can be at most $3\beta - 5$ ($\beta - 1$ in Step 2 and $2\beta - 4$ in Step 3). The in-degree is at most β by definition. We claim that the multiplicity of the final transfer graph is $\beta + 2$. Consider all the edges added in Step 2, we will show the multiplicity induced by these edges is 2. Since all G_i's are disjoint and each disk in G_i sends only item i to disks in $D_i \setminus G_i$, for any pair of disks j_1 and j_2, there can be at most one edge in each direction. Now consider all the edges added in Step 3, if there is an edge between disk j_1 and disk j_2, no matter which disk is the sender, both disks belong to D_i for some i. Thus, there are at most β edges between j_1 and j_2 since a disk can belongs to at most β different D_i's. Therefore, the result follows by Theorem 2.1. □

THEOREM 3.11. *The total number of rounds required for the data migration is at most* $\alpha + 2m' + \lfloor \frac{3}{2}\alpha \rfloor + 5\beta - 2$.

Proof. We need $\alpha + 2m' + 1$ rounds for Step 2(b) by Lemma 3.5 and $\lfloor \frac{3}{2}\alpha \rfloor$ rounds for Step 3(b) by Lemma 3.8. Migration according to the coloring of the final transfer graph needs $(4\beta - 5) + (\beta + 2)$ by Lemma 3.10. Therefore, we have the theorem. □

COROLLARY 3.12. *Our algorithm is 9.5-approximation for the data migration problem.*

Proof. Since α, β, m' are lower bounds for the problem, the corollary follows. □

THEOREM 3.13. *The Data Migration Problem (with copy operation) is NP-hard. [Proof omitted]*

4 Heuristics

4.1 Edge-Coloring on Transfer Graph

We can find a transfer schedule using edge coloring on a transfer graph. Since a disk can get the same item from different source disks, we want to find a good way of selecting source disks. We build a flow network with a source s and a sink t. In addition we have two sets of nodes corresponding to items and source disks. We add edges from s to node i with capacity $|D_i|$ for all item i, and edges from source disks j to t with capacity $c - \beta_j$. Finally, we add edges from item i to source disk j if $j \in S_i$.

Suppose c' is the minimum c such that, for all i, the amount of flow from s to i is the same as its capacity. Such c' can be obtained by binary search and solving a network flow problem in each iteration. Let $f^*(i,j)$ be the flow value from item i to source disk j when c' is obtained. We build a transfer graph as follows. Each disk is a node in the graph. For each source disk j having item i initially, we put $f^*(i,j)$ directed edges from disk j to $f^*(i,j)$ different disks in D_i, meaning that source disk j would send item i to $f^*(i,j)$ disks in D_i. From the flow network, we know that $\sum_{j \in S_i} f^*(i,j) = |D_i|$ for all i. Thus, all destination disks are served. Moreover, each source disk j serves at most $c' - \beta_j$ disks, and receives β_j items from other disks. The total degree of each source disk in the transfer graph is at most c'.

Now we find an edge coloring of the transfer graph (which may be a multigraph) to obtain a valid schedule [5], and the number of colors used is an upper bound on the total number of rounds.

Consider the problem single source broadcast, where all Δ items are initially stored in a single disk s, and D_i is all the disks except the source disk s for all i. We claim that the approximation ratio of this method is $\Omega(\frac{\Delta(N-1)}{\lfloor \log N \rfloor + 2\Delta - 1})$. As there is only one source, the out-degree of the source disk equals to the total demand, which is $\Delta(N-1)$, in the transfer graph. Thus, the heuristic takes at least $\Delta(N-1)$ rounds. The optimal solution [7, 8] use pipelining and double the number of items in every two steps. This would take $\lfloor \log N \rfloor + 2\Delta - 1$ rounds[4].

4.2 Broadcast Items One by One

In this algorithm, we deal with one item at a time. We process each item i sequentially, and satisfy the demands by doubling the item in each round, which takes $\lceil \log(\frac{D_i}{S_i} + 1) \rceil$. The total number of rounds is $\sum_i \lceil \log(\frac{D_i}{S_i} + 1) \rceil$.

Consider the case where $|S_i|$ is 1 and $|D_i|$ is 3 for all i, and all S_i and D_i are disjoint. We claim that the approximation ratio of this method is $\Omega(\Delta)$. The algorithm takes 2 rounds for each item, and the total is 2Δ. The optimal solution is to double all items in parallel, which takes 2 rounds.

4.3 Heuristics Using Matching

The algorithm is as follows.

1. Build a weighted undirected graph as follows. Each disk is a node in the graph. Let the cost $c(i, v, w)$ of sending item i from disk $v \in S_i$ to disk $w \in D_i$ be $\log \frac{|D_i|}{|S_i|}$. Intuitively, if disk v and w are matched, we would like to send the item with greatest cost, and either v or w can be the sender. The weight between v and w in the graph is $\max(\max_i c(i,v,w), \max_i c(i,w,v))$.

2. Find a maximum-weight matching on this graph. For each matched v and w, suppose $c(i,v,w)$ corresponds to the weight between v and w, send item i from v to w. This takes one round to send items in the matched pairs. We then update the S_i's and D_i's.

3. If we have not satisfied all demands (there are some non-empty D_i's), go to Step 1.

4.4 Preliminary Performance Results

It is intuitive to see that the edge-coloring heuristic, which does not exploit the fact that all newly spawned copies can be sources, and broadcasting one-by-one heuristic, which does not exploit parallel transfer of different items, perform much worse than the other heuristics. Under the case where

[4]The bound is for odd N. For even N, the bound is $\lceil \frac{\Delta(N-1) - 2^{\lfloor \log_2 N \rfloor} + 1}{\lfloor \log N \rfloor + \lfloor N/2 \rfloor} \rceil$ rounds.

$\lfloor \frac{N}{\Delta+1} \rfloor$ disks have all the items and all the remaining disks are partitioned into Δ groups, and disks in each group i request item i, the data migration algorithm performs better than the matching heuristic. Under the case where $|S_i|$ and $|D_i|$ each follows Zipf distribution, matching heuristic usually takes β to 2β rounds while data migration algorithm takes almost 3β rounds. One reason is that in Step 4 of our algorithm does not treat newly spawned copies as sources, and there may be better ways to do Step 2(b) and 3(b) by using methods that do better in practice, but do not give good worst case bounds.

Acknowledgements We would like to thank Sudarshan Chawathe, Leana Golubchik, Liviu Iftode and John Kubiatowicz for useful discussions.

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*, 2001

[2] B. Baker and R. Shostak. Gossips and Telephones. *Discrete Mathematics*, 2:191–193, 1972.

[3] J. Bermond, L. Gargano and S. Perennes. Optimal Sequential Gossiping by Short Messages. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, Vol 86, 1998.

[4] J. Bermond, L. Gargano, A. A. Rescigno and U. Vaccaro. Fast gossiping by short messages. *International Colloquium on Automata, Languages and Programming*, 1995.

[5] J. A. Bondy and U. S. R. Murty. Graph Theory with applications. *American Elsevier*, New York, 1977.

[6] R. T. Bumby. A Problem with Telephones. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):13–18, March 1981.

[7] E. J. Cockayne, A. G. Thomason. Optimal Multi-message Broadcasting in Complete Graphs. *Utilitas Mathematica*, 18:181–199, 1980.

[8] A. M. Farley. Broadcast Time in Communication Networks. *SIAM Journal on Applied Mathematics*, 39(2):385–390, 1980.

[9] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella and A. Zhu. Approximation Algorithms for Data Placement on Parallel Disks. *Proc. of ACM-SIAM SODA*, 2000.

[10] J. Hall, J. Hartline, A. Karlin, J. Saia and J. Wilkes. On Algorithms for Efficient Data Migration. *Proc. of ACM-SIAM SODA*, 620–629, 2001.

[11] A. Hajnal, E. C. Milner and E. Szemeredi. A Cure for the Telephone Disease. *Canadian Mathematical Bulletin*, 15(3):447–450, 1972.

[12] S. M. Hedetniemi, S. T. Hedetniemi and A. Liestman. A Survey of Gossiping and Broadcasting in Communication Networks. *Networks*, 18:129–134, 1988.

[13] I. Holyer. The NP-Completeness of Edge-Coloring. *SIAM J. on Computing*, 10(4):718–720, 1981.

[14] J. Hromkovic and R. Klasing and B. Monien and R. Peine. Dissemination of Information in Interconnection Networks (Broadcasting and Gossiping). *Combinatorial Network Theory*, pp. 125–212, D.-Z. Du and D.F. Hsu (Eds.), Kluwer Academic Publishers, Netherlands, 1996.

[15] C. A. J. Hurkens. Spreading Gossip Efficiently. *Nieuw Archief voor Wiskunde*, 5(1):208–210, 2000.

[16] S. Kashyap and S. Khuller. Algorithms for Non-Uniform Size Data Placement on Parallel Disks. *Manuscript*, 2002.

[17] W. Knodel. New gossips and telephones. *Discrete Mathematics*, 13:95, 1975.

[18] H. M. Lee and G. J. Chang. Set to Set Broadcasting in Communication Networks. *Discrete Applied Mathematics*, 40:411–421, 1992.

[19] D. Liben-Nowell. Gossip is Synteny: Incomplete Gossip and an Exact Algorithm for Syntenic Distance. *Proc. of ACM-SIAM SODA*, 177–185, 2001.

[20] D. Richards and A. L. Liestman. Generalizations of Broadcasting and Gossiping. *Networks*, 18:125–138, 1988.

[21] H. Shachnai and T. Tamir. On two class-constrained versions of the multiple knapsack problem. *Algorithmica*, 29:442–467, 2001.

[22] H. Shachnai and T. Tamir. Polynomial time approximation schemes for class-constrained packing problems. *Proc. of Workshop on Approximation Algorithms*, 2000.

[23] C.E. Shannon. A theorem on colouring lines of a network. *J. Math. Phys.*, 28:148–151, 1949.

[24] S. Shargorodskaya. Implementation of Data Migration Algorithms. `http://www.cs.umd.edu/projects/smart/data-migration/`

[25] D.B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem *Mathematical Programming*, A 62, 461–474, 1993.

[26] R. Tijdeman. On a Telephone Problem. *Nieuw Archief voor Wiskunde*, 19(3):188–192, 1971.

[27] V. G. Vizing. On an estimate of the chromatic class of a p-graph (Russian). *Diskret. Analiz.* 3:25–30, 1964.