Improved Algorithms for Data Migration

Azarakhsh Malekian [*]

December 8, 2005

Abstract

In this paper, we focus on the data migration problem. This problem is motivated by the need to manage data on a set of parallel disks and was proposed in [1]. In this paper, it was shown that one can develop a polynomial time algorithm with a 9.5 approximation factor for this NP-hard problem. We show how to improve the approximation guarantee to 8.5.

1 Introduction

To handle high demand, especially for multimedia data, a common approach is to replicate data objects within the storage system, which means that each item exists on a subset of disks.The layout specifies for each disk the subset of items it needs to store. Over time, as the demand for the items changes, the layout is changed. We need to find a way to transfer items to change the initial configuration to the final configuration. Lots of optimization problems on this model are defined with different objectives. In our problem, we want to minimize the total number of rounds needed to reach the target configuration. The restriction here is that in most of the storage systems, disks have limits on the number of the items that they can send/receive simultaneously. In our problem we assume that each disk can send/receive one item at each point of time. The goal is to minimize the total number of rounds to reach the final configuration. More formally, as in [1], in *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 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. In [1], it was shown that computing the optimal solution is NP-hard; A 9.5 approximation algorithm

[*]Department of Computer Science. University of Maryland, College Park, MD 20742.
Email: {malekian}@cs.umd.edu

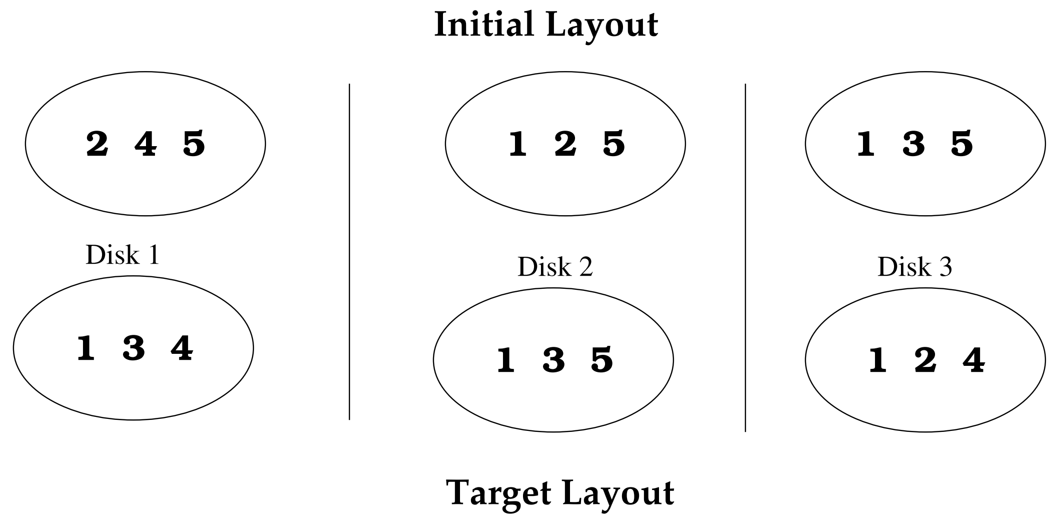

Figure 1: Demands may change over time. An example of initial and target layouts is shown in this figure. In this case, $S_1 = \{2, 3\}$, $D_1 = \{1\}$, $S_2 = \{1, 2\}, \ldots$.

was developed for the problem. In their method, they didn't use any internal bypass disks. So no unnecessary transfer exists in the final method. The method described in this paper, gives an improvement to their method. Here, we use internal bypass disks to improve the approximation ratio to 8.5. By internal bypass disks we mean to allow to send items to any of the disks even if it does not need that item. In the rest of the paper, the improved method for the *data migration problem with cloning* is described.

2 The Data Migration Algorithm and Definitions

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 [2] for example). Let χ' be the edge chromatic number of G.

Theorem 1. *(Vizing [6]) If G has no self-loops then $\chi' \leq \Delta_G + \mu$.*

Theorem 2. *(Shannon [4]) If G has no self-loops then $\chi' \leq \lfloor \frac{3}{2}\Delta_G \rfloor$.*

Define β_j as $|\{i|j \in D_i\}|$, i.e., the number of different sets D_i, to which a disk j belongs. We then define β as $\max_{j=1\ldots N} \beta_j$. 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. Also, we define $\bar{\beta} = \sum_{i=1}^{\Delta} \frac{|D_i|}{N}$. easily we can see that $2\bar{\beta}$ is a lowerbound for the optimal number of rounds since in each round at most $\lfloor \frac{N}{2} \rfloor$ receiving can happen. 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.

The algorithm is as follows:

1. For an item i decide a 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 source (s_i) or destination. *Note that α is also a lower bound on the optimal number of rounds.*

2. Find a transfer graph for items that have $|D_i| \geq \bar{\beta}$ as follows.

   (a) We first compute a disjoint collection of subsets $G_i, i = 1 \ldots \Delta$ so that $|G_i| = \lfloor \frac{|D_i|}{\bar{\beta}} \rfloor$. Here, G_i is not necessarily a subset of D_i. (In Lemma 1, we will show how such G_i's can be obtained.)

   (b) We have each item i sent to the set G_i.

   (c) We create a transfer graph as follows. Each disk is a node in the graph. We add directed edges from disks in G_i to the disks in $D_i \setminus G_i$ such that the out-degree of each node in G_i is at most $\bar{\beta}$ and the in-degree of each node in $D_i \setminus G_i$ from G_i is at most 1. We redefine D_i as the set of disks in $D_i \setminus G_i$ which do not receive item i in the transfer graph. As we can see, for each item i, the number of remaining elements can not exceed $\bar{\beta}$. New D_i sets will be taken care of in Step 3. Note that the redefined set D_i has size $< \bar{\beta}$.

3. Find a transfer graph for redefined D_i sets 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 $\sum_{i \in I_j} |D_i| \leq \beta + \bar{\beta}$ where I_j is a set of items for which j is a new source. See Lemma 2 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 s'_i to all disks in $D_i \setminus \{s'_i\}$. We will prove later that the out-degree of a disk does not exceed $\bar{\beta} + \beta$.

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_i gets item i. In Lemma 3 we derive an upper bound on the number of required colors.

We can find primary sources for each item using the same method as in [1] so that we guarantee that for each disk, the number of items which this disk is chosen as its primary source plus the number of items that it should receive itself does not exceed α. Next, we should show how to choose G_i sets. We present two methods here.

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

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

$$\sum_{i=1}^{\Delta} |G_i| \leq \sum_{i=1}^{\Delta} \frac{|D_i|}{\bar{\beta}} = \frac{1}{\bar{\beta}} \sum_{i=1}^{\Delta} |D_i|.$$

Note that $\bar{\beta} = \frac{\sum_{i=1}^{\Delta} |D_i|}{N}$ by definition. This proves the upper bound of N on the total size of all sets G_i. Since, the only requirement for the G_i sets is disjointness, we can easily partition disks into sets of size $\frac{|D_i|}{\bar{\beta}}$ and assign each of them to one of the G_i's. However, if we want to maximize $\sum_{i=1}^{\Delta} |G_i \cap D_i|$ at the same time, we can use min-cost-flow. Make a flow network with a source s and target t. We have two set of nodes, one corresponds to the disks and the other to the items. Put edges between s and item nodes with capacity $|G_i|$ and cost 0. Also, put edges between all item nodes and disk nodes with capacity 1 and cost 0 if the disk belongs to the corresponding demand set of the item, otherwise assign cost 1. put edges between disk nodes and t with capacity 1 and cost 0. Now, since we showed above that we can choose all disjoint G_i sets, so this network should have a flow of size $\sum_{i=1}^{\Delta} |G_i|$. Also since we find the minimum cost flow of this kind, we guarantee that we minimize the assignments of items to disks out of the demand set which result in maximizing $\sum_{i=1}^{\Delta} |G_i \cap D_i|$. □

Next we will show how step 3(a) works. Again, the same as [1] we will use the Shmoys-Tardos result in [5].

Theorem 3. *(Shmoys-Tardos [5]) 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.*

Lemma 2. *For each item i, we want to choose a secondary source $s'_i \in D_i$ to broadcast items to redefined D_i sets. We prove that it is possible to choose these secondary sources in a way so that for each disk, the number of disks that it is responsible for is not more than $\beta + \bar{\beta}$. More formally, if I_d be the set of items for which d is chosen as the secondary source, $\sum_{i \in I_d} |D_i| \leq \beta + \bar{\beta}$.*

Proof. We use the above theorem proved by Shmoys-Tardos to show our bounds. 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. (To guarantee that we choose sources only from the corresponding demand sets). Processing time of job i (corresponding to item i) is $|D_i|$ (uniform processing time on all machines). The scheduling algorithm 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. Here, since $|D_i| \leq \bar{\beta}$, the largest job is no more than $\bar{\beta}$. Next, we will argue that there is a fractional solution with makespan β. The fractional solution for the case where $s'_i \in D_i$ 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. So the makespan of the integral solution is at most $\bar{\beta} + \beta$ which means that each s'_i is responsible for at most $\bar{\beta} + \beta$ disks. □

This lemma guarantees that the outdegree corresponds to step 3(c) is no more that $\beta + \bar{\beta}$. We can use the same proof as in [1] to show that sending items to s'_i takes at most $\frac{3}{2}\alpha$ rounds. In their proof they argued that since s'_i is in D_i set, then it can be chosen as the secondary source for at most β_i different items and also if it is sender it will send to at most $\alpha - \beta_i$ disks. So as we see the fact that $s'_i \in D_i$ is useful here to make the β_i guarantee on the indegree. To complete their proof, they used the Shanon's theorem. Now, what we should do is to put the corresponding transfer graphs from step 2(c) and 3(c) together. The number of colors for a proper-coloring of this graph gives us the number of rounds we need to transfer the data.

Lemma 3. *The number of colors we need to color the combined graph is at most* $(3\beta + 2\bar{\beta} + O(1))$.

Proof. First, we compute the maximum indegree and outdegree of each node. The outdegree of a node is coming from either being in a G_i set (which results an outdegree of at most $\bar{\beta}$) or being a secondary source (which results in an outdegree of $\beta + \bar{\beta}$). The indegree of each node is at most β since in the transfer graph we send items only to the disks in their corresponding demand sets. The only remaining part is computing the multiplicity in the multigraph. Consider a pair of nodes j_1 and j_2. We have two kinds of edges either it is an edge between a node in a G_i set to one of the nodes in D_i, or it is a transfer from a secondary source to a demand node. We can see that we have at most two edge of the first type between j_1 and j_2. Since each node can belong to at most one G_i set. For counting the second type edges, note that if we transfer item i between j_1 and j_2 they should be both in D_i set. Now since j_1 belongs to at most β different D_i sets the number of this kind of edges can not exceed β so multiplicity is at most $\beta + O(1)$. Just by adding up all the terms, we see that the maximum number of colors needed is at most $2\bar{\beta} + 3\beta + O(1)$. □

To wrap up, in the next theorem we show that the total number of rounds in this algorithm is bounded by 8.5 times the optimal solution.

Theorem 4. *The total number of rounds required for the data migration is at most* 8.5 *times* OPT.

Proof. Since we use the same method as in [1] to send items to G_i sets , total number of this stage is bounded by $2m' + \alpha \leq 3OPT$.(α, β, m' and $2\bar{\beta}$ are lowerbounds for OPT.) We need $\frac{3}{2}\alpha \leq \frac{3}{2}OPT$ to send items from primary sources to secondary sources. The last stage is to send data items in the transfer graph which takes $3\beta + 2\bar{\beta} \leq 4OPT$. Putting all terms together, we conclude that data migration can be done in $8.5OPT$ rounds. □

3 Conclusion

In this paper, we gave an 8.5 approximation algorithm for the *Data Migration Problem* which improves the previous bound given in [1] by 1. The main idea was to use internal bypass nodes. Furthermore, if G_i sets are not completely disjoint and each disk in these sets is responsible for more than one item, the ratio can be improved to $8 + o(1)$. (The proof for this result will appear in the full version of the paper.)

References

[1] 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.

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

[3] S. Khuller, Y. A. Kim and Y. C. Wan. On Generalized Gossiping and Broadcasting. 11th *European Symposium on Algorithms*, LNCS 2832, pp. 373–384, Sep 2003 (Budapest).

[4] C.E. Shannon. A Theorem on Colouring Lines of a Network. *J. Math. Phys.*, 28:148–151, 1949.

[5] D.B. Shmoys and E. Tardos. An Aproximation Algorithm for the Generalized Assignment Problem. *Mathematical Programming*, A 62, pp. 461–474, 1993.

[6] V. G. Vizing. On an Estimate of the Chromatic Class of a p-graph (Russian). *Diskret. Analiz.* 3:25–30, 1964.