ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 1, pp: (17-26), Month: January-March 2014, Available at: www.researchpublish.com

Minimization of Distributed Data Items in the Clouds

V.Srikanth[1], S. Dhamodaran[2]

[1] Student of Computer Science and Engineering Department , Sathyabama University,
[2] Assistant professor of Computer Science and Engineering Department , Sathyabama University,
Chennai 639 119,Tamilnadu ,India.

Abstract: **In this paper, we study this data staging problem by leverging the dynamic programming (DP) techniques to optimally migrate, replicate, and cache the shared data items in cloud systems with or without some practical resource constrains in an efficient way whiling minimizing the monetary cost for transmitting and caching the data items. Monetary cost is our primary interest as on one hand, the provision of the resources in cloud systems are usually based on pay-as-you-go fashion, and thus effective use of the platforms within budget constraint is always the user's concern. In general, data staging algorithm in our applications is not only needed to figure out the user's access patterns (i.e., a sequence of requests, each being made by certain node at a predicted time instant) but also required to migrate, replicate, or cache the involved data items across the network at particular time instants to optimize some performance metric for future accesses.**

Keywords: **Cloud computing, data staging and caching, resource constrains, data placement and migration.**

I. INTRODUCTION

With increasing data accessibility demands on clouds, data availability maximization seems to be important problem to consider to maintain high-fidelity and time-bounded service expectations in clouds. For example, one of the pressing needs by the cloud service providers (CSPs) is to efficiently serve the needs of the user requests that demand single or multiple data items in the shortest possible time. Thus, with growing population of cloud users, the problem of making the requested data available to the users becomes an imperative issue for CSPs to guarantee high-quality services. An particularly appeal-ing approach to maximizing such data availability is to stage the requested data to some vantage sites and cache the data for a period of time so that the quality of service (e.g., latency minimization, network traffic reduction) for user's future accesses can be greatly improved. We refer to this integrated functionality as data staging. An exemplary scenario is a mobile user who may want a service to suggest probable alternative routes to her destination depending on current traffic patterns. This service requires not only making the queried data highly available at vantage nodes along her current path but also staging the data at key sites on probable paths based on her navigation history. Clearly from this example, data staging could effectively facilitate the service accesses. However, achieving the benefits is not free due to the costs for access pattern detection and data staging.

In this paper, we study this data staging problem by leveraging the dynamic programming (DP) techniques to optimally migrate, replicate, and cache the shared data items in cloud systems with or without some practical resource constraints in an efficient way while minimizing the monetary cost for transmitting and caching the data items (glory details are discussed in the later of this section). Monetary cost is our primary interest as on one hand, it is a very flexible concept to reflect the qualities of various network features such as network bandwidth, link latency, and storage utilization, and on the other hand, the provision of the resources in cloud systems are usually based on pay-as-you-go fashion, and thus effective use of the platforms within budget constraint is always the user's concern.
Due to the optimality, our solutions are unique and advantageous over other (suboptimal) methods to provide the cloud-based services with the flexibility that they can not only decide the duration of each data item to be cached at

some vantage sites but also make a tradeoff between transmission cost and caching cost to meet the constraints imposed by the underlying Infrastructure as a Service Providers (IaaSPs), information item owners or CSPs' budget. As cloud computing is gaining its prominence, we believe these benefits are more important than ever before to the success of the traditional network-based services migrating to clouds such as the distributed collaborative document editing process and multimedia personalized services where a document or video clip may be requested by users in a sequence of predefined time instants.

A. Background: Two-Phase Data Staging

In general, data staging algorithm in our applications is not only needed to figure out the user's access patterns (i.e., a sequence of requests, each being made by certain node at a predicted time instant) but also required to migrate, replicate, or cache the involved data items across the network at particular time instants to optimize some performance metric for future accesses.

I explicitly distinguish the two functional phases of the data staging algorithm with a burning focus on the second one. The first phase takes as input a variety of kinds of data such as a network graph, a historical data trace, or other modeled access distributions. The output of this phase is an inferred access pattern which is taken as an input to the second phase.
The output of the second phase is the total cost to achieve the data staging which in our particular case is the total monetary cost, a major concern in cloud computing. Distinguishing between these two phases is reasonable as the functionality of each phase is different, so is the measure metric.Additionally, another apparent advantage is that the prediction/selection element of the algorithm is separated from the migration/replication element. Consequently, the two elements could be designed and implemented independently and their combinations could be composeable to achieve the best performance.

B. Research Goal, Cost and Constrains

1. Research Goal

In this paper, we do not intend to improve the first phase of the algorithm for a high-quality access pattern. Instead, with a such kind of information presumably known in advance, our goal is to efficiently migrate, replicate, or cache the multiple requested data items defined by the access pattern in a fully connected network with or without resource constraints[2] so that the sequence of service demands are satisfied with minimum data transmission and caching costs.

Achieving this goal will have two immediate advan-tages, which are critical to the success of applying such techniques to the maximization of data availability. First, as the cost in the data staging and caching accounts for a part of the total cost, minimizing it can accordingly minimize the total cost. Second, with the cost of the second phase decreasing, the first phase of the algorithm could have more opportunities to adjust the frequency of running the prediction algorithms to improve the prediction accuracy so that the overall cost could be further reduced.

2. Cost Models

The cost model adopted in our research could be hetero-geneous or homogeneous in the senses that whether or not the transmission costs are identical and caching costs at all sites are also identical. As our algorithms are mainly designed for CSPs who usually demand the infrastructure services from an IaaSP (e.g., Amazon AWS, GoGrid, Flexiscale and Mosso), in this paper we are particularly interested in the situation when the homogeneous cost model is employed due to two reasons: First, the rented infrastructure for a particular service is always organized as a subset of homogeneous resources to entail the hosted applications to meet its Service Level Agreement (SLA) targets. This in general results in homogeneous computation and communication in the clouds.

Homogeneous computation: IaaSPs commonly provide users with a set of different virtual machine types, each of which has different resource capacities in terms of CPU capacity, RAM size, disk I/O bandwidth, and the like. The

performance of different types of virtual machines are obviously heterogeneous. However, the performance of multiple virtual machines of the same type which usually host a particular service is practically similar.

Homogeneous communication: the current topologies of data center networks are in general structured as either two- or three-level trees of switches or routers with an low bandwidth edge tier at the leaves and high-bandwidth fat-tree at the top of the tree, roughly leading to homogeneous communication in nature.

Second, as both the transmission cost rate and caching cost rate are determined by the IaaSPs, it is unlikely for them to offer a heterogeneous cost model as it could pose difficulties in public clouds since at present only a few of IaaSPs are willing to expose some low-level information about the containers and sub networks to their users.

3. Resource Constraints

Given multiple copies allowed for each data item, we also consider the situation when some restrictions are imposed on the volume of communications and the number of copies during a scheduling interval. These restrictions are rational in practice as they would be required either by some SLA, technical requirements or by some legal or economic reasons.
Overall, our network and cost models together with the resource constraints are reasonable and representative to conduct this research to reflect the reality.

C. Contributions

Under the homogeneous cost model, our results include two parts. First, when there are no resource constraints, we give a DP-based optimal algorithm in polynomial time to stage and cache k distinct items to satisfy a predetermined sequence of requests, each item having single or multiple copies to minimize the cost. Based on this algorithm, we further show that the cost of a single-copy algorithm is within a factor of 1+C/S from that of the multicopy (optimal) algorithm. This result indicates that a single copy of each data item can efficiently serve all the user requests to it provided that the ratio of transmission cost rate (C) and caching cost (S) is low. Moreover, when S > C, the optimal algorithm is a single-copy algorithm, and the cost of any suboptimal single-copy algorithm is not greater than twice that of the optimal single-copy algorithm.
Second, the constraints on the volume of communications and the number of copies are studied in the form of upper bounds that can be used during a scheduling interval. The upper bounds are given either on per-item basis or on all-item basis, which make it flexible enough to describe the restrictions. We optimize the total cost under these constraints by proposing a suite of optimal solutions in polynomial time given that the upper bound is polynomially bounded by the number of requests and the number of distinct data items. We validate the algorithms by implementing a data staging solver whereby conducting extensive simulation studies.

As for the case with a heterogeneous cost model, it is generally believed that this problem is NP-Complete. We thus consider its optimality in a very restricted yet practical form which is tractable when no simultaneous communications are allowed and the maximum of the instant copies is upper bounded by a small constant.

II. PROBLEM FORMULATION

Suppose there are k distinct shared data items initially stored at one node, say p_1 and later migrated, replicated, and cached in a fully connected network with m nodes *(p1,p2,…,pm)* to serve a sequence of requests $\sigma \ \Delta = \sigma 1\ \sigma 2 \ldots\ \sigma n$ in which each
which each $\sigma_i \ \Delta = (t_i, p_i, R_i)$, $1 \leq i \leq$ n is specified by the predicted access pattern and represents a request made for a data subset R_i by node p_i at time t_i. We therefore have the complete knowledge of all such information as an input to our algorithms. Further, for simplicity we also assume that there exists only one request per stage.

To satisfy a request for a particular data item, we define the following primitive operations to preform on the cached data items, which may involve caching and transmission costs:

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 1, pp: (17-26), Month: January-March 2014, Available at: www.researchpublish.com

1. *retainment:* cache the data item at a node p_u from time t_u to t_v by paying $(t_v \_ t_u)S_u$, S_u is the rate of caching cost at node p_u, $1 \leq u \leq m$.
2. *migration:* move the data item from a node p_u to a node p_v at a cost equal to the distance C_{uv}.
3. *replication:* copy the item to the request node p_v from a node p_u at a cost of C_{uv}.
4. *excursion*: satisfy the request at a node p_v by using the copy at a node p_u without migration at a cost of E_{uv}.
5. *creation/deletion*: create/delete the selected copies at some nodes without incurring any cost.

We thus follow this principle in our model to allow $E_{uv} = C_{uv}$ for any pair of p_u and p_v in the network. Each new request can trigger one or more operations, leading to the creation and deletion of the data items at arbitrary nodes.

III. OPTIMAL DATA STAGING ALGORITHMS

In this section, with the homogeneous cost model, we first present the multicopy algorithm to handle multiple distinct data items, and then propose our extension to address the constraints on the maximum number of transmissions and copies. Note that we do not discuss the single-copy algorithm separately as it can be treated as a special case of multicopy algorithm.

1. Multiple Copies without Constraints

Our multicopy algorithm for handling k distinct data items with total cost is shown in DP Recrusion (2) where $Fj(i)$ is maximum total cost for accessing item j till $t_i \geq 1$:

$$F_j(i) = \begin{cases} F_j(i-1) & j \notin R_i \\ \min\limits_{l \in E_i^j} \{F_j(l) + SP(p_l^j, p_i^j) + \sum\limits_{h \in A_l^j} \min\limits_{t \in E_h^j} \{\beta_{lh} SP(p_t^j, p_h^j)\}\} & j \in R_i, \\ F_j(0) = 0 & \end{cases} \quad (2)$$

Recrusion (2) indicates that when there is no request for item j made t_i $(j \in R_i)$the minimum cost till t_i depends on the source point set whose elements are candidate copies to service the current request,and the shortest path from the selected source point to the current request point. In addition, we also need to update the costs for the request points in the affected set because of the side effects of the shortest path computation. The final scheme for the data staging obtained from Recursion (2) contains all the information pertain to how primitive operations are combined to achieve the minimum cost.

2. Resource Constraints on Per-Item Basis

Since the resource concepts of transmission and copy in our research are highly related, in this section we uniformly investigate the optimality when the constraints are given on per-item basis to serve the entire request sequence (i.e., on per-scheduling-cycle basis).

TABLE 1
Computation of SP(pl, pi, y)

	Transmission		Interval Copy	
	y>0	*y=0*	*y>0*	*y=0*
$p_l = p_t$	$S\Delta_{i-t}$	$S\Delta_{x-1}$	$S\Delta_{i-t}$	$S\Delta_{i-1}$
$p_l \neq p_t$	C+ $S\Delta_{i-1}$	$+\infty$	C+ $S\Delta_{i-1}$	$S\Delta_{i-1}$ C+ $S\Delta_{i-1}$

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 1, pp: (17-26), Month: January-March 2014, Available at: www.researchpublish.com

3. Dynamic Programming Recursion

For description, we define $F_j(i, c^j_i)$ as the minimum cost for item j till stage t_i given the maximum number of resource c^j_i ($c^j_0 = c^j$), here the resource is either the number of copies or the number of transmissions. The total cost for all k items at
t_i is $F(i)=\sum_{i=1}^{t_i} F_j(i,c^j_i)$ where $F_j(I,c^j_i)$ defined by DP .

The recursion is not difficult to understand. If no request for item j is made at t_i, the total cost remains unchanged, otherwise, the resource at stage ^{ti} will be deducted by x units from the current available c^j_i and the remaining is applied to some source point l.

The x units are then distributed among computing the shortest path cost (i.e., y units) and the costs for all elements in the affected point set (i.e., z units). The minimum sum of these three parts over the source point set E^j_i and the current available resources c^j_i is the total cost until stage t_i. As we defined, the source point set is a constant set, the minimum cost hence can be obtained independent of the scheduling path.

Note that SP (u, v, c) returns + ∞ if the resource c is not sufficient enough to compute any constrained shortest path. In this situation, $F_j(i,c^j)$ is also failed to compute and returns + ∞ as well. The optimal solution is $F_j(n, c^j)$. Obviously, when we have the transmission constraints, more copies will be used. Similarly, when we have a constraint on the copies, more transmissions will be used.

4. Resource Constraints on All-Item Basis

In this section, we consider an optimal solution when the maximum number is given to all the k distinct data item instead of each individual one, which is more effective to utilize the available resources. The proposed algorithm should have the capability of distributing the number among these items in an optimal way to minimize the total costs while respecting the constraints. More specifically, let the maximum number resources for all k items be c (i.e.,$c=\sum_{j\in|L,|} c^j, c^j \geq 0$), we try to find the optimal solution from the following equation:

$$L(k,c)=\sum_{j\in|1,l|}^{cj=c,cj\geq 0} \sum_{j\in|,k|} F_j(n,c^j).$$

This equation requires c to be partitioned into k parts,$c^j, j=1...k,$,in such a way that the sum of $F_j(n,c^j)$ is minimum. To this end, we put the k data item along a line and index them from 1 to k. Suppose the current resource is r, and index i is the right most item that is assigned a nonzero units, we assign q units to item i and the remaining r - q units to the first i - 1 items. Let Q[I,q] store the precomputed value of F_i(n, q), we can obtain the DP matrix as follows when 1≤ j≤ k and 0 ≤ r ≤ c

$$\begin{cases} L[j,r] = \min\limits_{\substack{1\le i\le j \\ 1\le q\le r}} \left\{ L[i-1, r-q] + Q[i,q] + \sum\limits_{l=i+1}^{j} Q[l,0] \right\} \\ L[0,q] = 0 \end{cases} \qquad q \in [0,c],$$

The optimal solution is $L[k,c]$.
To analyze the time complexity for above equation, we first compute $Q[i,r]$, for $1 \le i \le k$, $0 \le r \le c$. we only need to compute $F_i(n,c)$ to get all the values for Q[i,r], 0≤r≤c. Thus, the complexity is $O(kcnm^2 \log m)$. Then, the complexity of DP matrix in above equation is $O(k^2c(k + c))$. So the total time complexity is $O(kcnm^2 \log m + k^2c(k+c)) \le O(k^2n(m\log m+k^2)$. Obviously, given the fixed number of n,m,and k, our algorithm needs quadratic time on c (i.e., $O(c^2)$ time) to serve the sequence.

IV. ON HETEROGENEOUS COST MODEL

Up to now we have investigated the data staging problem under the homogeneous cost model, together with some

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 1, pp: (17-26), Month: January-March 2014, Available at: www.researchpublish.com

practical constraints. For completeness, in this section we briefly discuss the problem under the heterogeneous cost model where each node has its own caching cost rate, and the transmission cost rate between any pair of nodes is not always identical. As this general case is overly complex and somehow connected to the Rectilinear Steiner Arborescence (RSA) problem it is generally believed to be an NP-Complete problem .However, by extending the DP algorithm where $S_k \geq C_{ij}$, $i, j, k = 1, ..m$ is considered for optimal data staging solution, we can show that this heterogeneous case is still tractable under some restricted conditions. In the following discussion, we concentrate on a single data item thereby designing our algorithms to this problem.

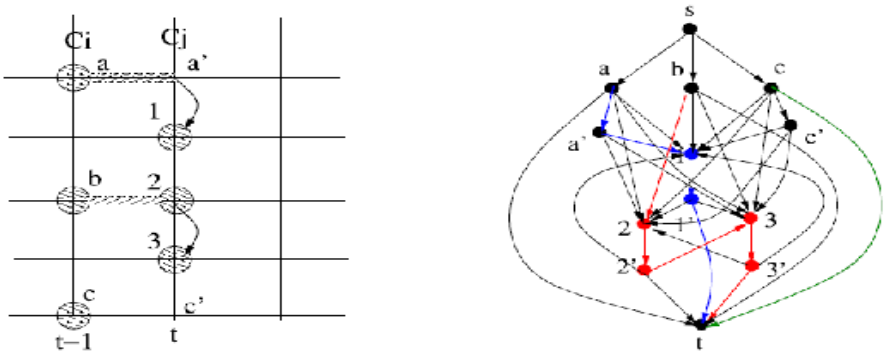
1. Configuration Distance

We define a metric space *(M,d)*, which is the space-time diagram with a distance *d(u,v)* defined as the shortest path between any pair of points (p_u,t_u) and (p_v,t_v) in the space-time diagram. A k-configuration for *(M,d)* is any multiset C of size k over M. In this section, we focus on the computation of the minimum cost of transforming one _-configuration to another, which forms the basis for our

2. *Definitions*

Suppose C_i and C_j are two k-configurations then $d(C_i,C_j)$ is the distance between C_i and C_j, which is the minimum distance/cost traveled/spent by all the data items and their copies that change configuration from C_i to C_j. Formally,

given $C_i=(p^1_i,p^2_i,\ldots p^k)$ and $C_j=(p^1_j,p^2_j,\ldots.p^k_j)$, formally we have



(a) Example of path interference in configuration mappings

(b) Network-flow Solution

Fig 2. Mapping path interference and its workflow

$$d(C_i, C_j) = \min_{\pi} \left\{ \sum_{l=1}^{\kappa} d(p_i^l, p_j^{\pi(l)}) \right\},$$

where π is the permutation of {1,2,…k}. In general, the computation of $d(p^l_i, p\pi^{(l)})$ in (M,d) is very complicated as it needs to map a set of nodes at *t-1 (i.e., C_i)* to another set of nodes at *t (i.e., C_j)* given that the mapping paths may interference with each other for a minimum cost. Fig. 2a is an example to illustrates the difficulties in computing

$d(C_i,C_j)$ where C_i ={ a,b,c } will map to C_j={1,2,3} with a minimum cost. In this example, b first mapping to 2 and then making a replication at 3 are cheaper than mapping b; c to 2 and 3, respectively. Apparently from this example, figuring out the optimal mapping paths between a pair of configurations is the key point to attain the distance.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 1, pp: (17-26), Month: January-March 2014, Available at: www.researchpublish.com

3. Bijective Mapping Function

There are two major issues in the above network-flow algorithm. One is the restriction on the cost model and the other is to allow multiple continuous replications to shift C_i to C_j at time t (see the reason why copy c in the example Fig. 2a is not involved in the mapping). Although the restricted cost model is not always satisfactory, it is still acceptable under certain conditions. In contrast, making multiple replications at a time instant is in general infeasible in practice as the resource cost of replication is usually high. Therefore, in the following discussion we always assume that the function that maps C_i to C_j from t -1 to t is a bijective function, which technically prohibits the multiple replications from happening simultaneously.

4. Greedy Algorithm

Not surprisingly, due to the hardness of this problem, the time complexity of the DP-based algorithm is prohibitively high even though we have imposed some constraints on the problem. In this section, we propose a simple greedy algorithm to the problem in its general form, whose staging cost is at most twice the optimum.
The basic idea is very simple. Given σ=σ1,σ2..σn, we first construct a service graph *G(V,E)* in the following way:
1. *V = {1,2. . . . n}* represents the n requests making at *(pi, ti), i € [1, . . . , n]*

2. for each request i, compute its source point set Ei

3. for each point *l € Ei*, add an arc from *l to i* in G with the weight being equal to the cost of the shortest path from *l to i*

We compute a minimum spanning branching (MSB) of the service graph to satisfy all request demands. Since $|E^i| \leq m$, the total number of arcs is not greater than O(mn). and thus the time complexity of constructing the service graph is O(mn).

V. EMPIRICAL STUDIES

To verify our algorithms and study their performance behaviors in reality, we developed a network data staging solver in C++ which efficiently implements the proposed optimal algorithms for both the constrained and nonconstrained versions of the problem as well as the models upon which the algorithms were built. Again, as the monetary cost is our primary interest, in the solver we did not take into account some properties and features of the network platform such as network traffic, bandwidth capacity, link latency, and CPU power, Rather, we focus squarely on the factors closely related to our research goal.

The solver can be configured by several parameters including the size of network, the number of distinct data items, and the available resources. It also accepts as an input a sequence of request demands that is made for any subset of k distinct data items. Of course, there constantly exist studies on modeling the generation of various access sequences in different research contexts and scenarios .However, this kind of information is irrelevant to our empirical studies as the sequence of request has been presumably known in advance. Unlike the sequence of requests, there is no such a readily available model that is well accepted to specify the distribution of the accessed items in each request. To address this issue, we intentionally allow that the requests for each item are automatically generated by following some probability distribution with item-dependent parameters to mimic the real case where the access opportunities among the k items are not equal. More specifically, we assume that the request for item j at a particular time step follows the Bernoulli distribution with a successful probability.

1. Impact of Caching and Transmission Rates on Total Costs

We first investigate the impact of caching and transmission rates on the total costs. To this end, we fix a 50-node fully connected network, and conduct a balanced and a unbalanced request sequences of length 100 by following the discussed methods. Each request is made for at most 10 data items. Fig. 4 shows our experimental results when the maximum number of copies is restricted by 10 and transmission constraint of 308, respectively. These numbers allow

the requests to have the minimal resources to access the required data items. Both caching and transmission rates in all cases have linear impact on the final costs. These observations are easy to understand since the total cost undergoes a proportional increase with the number of copies and transmissions.

Comparing with balanced and unbalanced requests, balanced requests under both constraints result in more costs due to their wider dispersion among the network nodes than their unbalanced counterpart. In addition, another interesting observation is the comparison between the two constraints. Although the two kinds of resources are minimum to both balanced and unbalanced requests, the transmission constraints will lead to more total costs than the copy constraints, especially for the unbalanced requests. This phenomenon is not as easy as that of the compared request distribution to understand because the constraints on one type of resources will naturally increase the use of the other type of resources to serve the entire request sequence, and both types of resources have the same range of cost rates. However, we found that the rationale behind this phenomenon is the time-invariant transmission cost to serve a request, which is different from the time-dependent caching cost. As a consequence, when transmissions are restricted, the algorithm needs to keep more cached copies for the subsequent accesses, which, by and large, increases the total costs compared with limiting the number of copies where each used transmission only has a constant cost rate C. Such benefits of copy constraints are more pronounced for the unbalanced requests due to the relative bad performance of transmission constraints where some nodes may accept a quite few requests, and thus the first copy maybe needed to kept for a long time for accesses.

2. Impact of Copy Constraints on Total Costs and Solver Performance

By following the same configuration in the last experiment, in this section, we conduct experiments to study the impact of available resources on total costs and evaluate the performance of the solver by measuring its execution time. Given constraints on the number of copies, Table 2 shows our results for both balanced and unbalanced requests when fixing C = 60 and varying S from 5 to 120. The given numbers of copies are optimally distributed among the 20 requested data items to minimize the total cost. Comparing the balanced and unbalanced cases, we found that the reduction is relatively significant for unbalanced requests. We think this is because if more copies are allowed, for most of the unbalanced requests, the algorithm is easier than in the balanced case to find a nearby copy with a minimum cost to serve a request. This explanation can be verified by observing the situations when C=S is high in which the cached copies are usually biased toward serving requests. This also explains why as C=S decreases, the benefits of adding more resources are gradually diminished. although the total costs decrease linearly as the copy numbers increase, the staging time increases in a nonlinearly quadratic fashion, which verifies our analysis on the time complexity where given the fixed number of nodes, the length of request sequence as well as the total number of distinct data items, the time complexity of our algorithm is a quadratic polynomial in the number of copies.

3. Empirical Studies on Transmission Constraints

In addition to the copy constraints, we also conducted the same set of empirical studies on the transmission constraints. Like what we have observed the results under the transmission constraints more or less exhibit the same relation forms as those under the copy constraints, which are consistent with our intuition as the copy and transmission are highly inter-related for data staging.

Table 2

Impact of copy Constrains on Total Costs under Various C/S Ratio for balanced and Unbalanced Request Sequences (Nodes:50,Requests: 100,Data Items:20,Transmission Rate:60)

S/#Copies	Balanced sequence				Unbalanced Sequence			
	30	40	50	60	30	40	50	60
5	66745	66295	66295	66295	37185	36635	36085	35545
10	76540	76240	76150	76140	46950	46450	46950	46470
30	115590	115590	115590	115590	86010	85710	86010	86170
60	174120	174120	174120	174120	144600	144600	144600	144600
120	290580	290580	290580	290580	261060	261060	261060	261060

VI. RELATED WORK

Our work in data staging to facilitate a sequence of time-variant accesses to cloud services necessarily builds upon work in a number of related areas. In the replica placement problem, the requests made by users are to known in terms of their frequencies and positions in the network, while the minimum number and location of the servers to serve the requests with minimum total read and write costs are to be determined. For an arbitrary network, this is a NP-Complete problem even with no writers. As a result, the optimal solutions were usually studied based on some simple networks like tree. Our results can be viewed as an extension to this problem as we not only considered the frequency of a particular request but also dealt with its time sequence. On the other hand, we worked on a fully connected network and the replicas could be generated and destroyed automatically to optimize our goal.

Unlike the replica placement problem, the k-server problem allows k mobile servers at some vertices of a graph to serve a request sequence in time order .Each request specifies a vertex for service, the k-server problem is to decide how to move the servers in response to each request. The cost of handling a sequence of requests is equal to the total distance (i.e., transmission cost) moved by the servers, which is the optimization goal. It has been shown that the optimal offline algorithm can be achieved efficiently in polynomial time .Our problem is a typical variant of the offline k-server problem in that in addition to transmission, we also took caching cost into account, rendering this problem to be more difficult to address .

Clearly, our problem bears some similarity to the classic file allocation (FA) problem where multiple copies of a file are maintained via caching, migrating, and replicating at the nodes of a network to minimize communication costs for read/write requests. File copies could be created/ deleted at will with zero cost, and file caching cost is also free. As a consequence there are only a transmission cost defined for file replication, and write cost for file creation/update, which is typically a nonlinear function of the number of copies present. In contrast, our model considers the caching cost for read-only data which is the major difference that makes our findings to enrich the FA problem.

VII. CONCLUSIONS

In this paper, we studied the problem of staging a set of distinct data items in a fully connected network to facilitate cloud-based services with minimum cost. To this end, we investigated the optimal staging strategies based on the cost models in the paper to minimize the total staging cost. We also considered some practical constraints on this problem in terms of the maximum number of transmissions and copies.

We achieved an efficient optimal solution via dynamic programming to the situation when the numbers of Transmissions and copies are unbounded within the time complexity of $O(kmn^2)$. When $S > C$, this algorithm also ensures an optimal single-copy algorithm within the complexity of $O(kn^2)$. For arbitrary S and C, our results show that when C=S is low, a single copy of each data item can efficiently serve all the user request sequence.

In addition to the homogeneous cost model, we also briefly discuss this optimization problem under a heterogeneous cost model, which is generally believed to be NP-complete. We proposed a DP-based algorithm which is efficient under some restrictions and presented a 2-approximation algorithm to the general case.

In this paper, we assumed that the accesses to the k distinct items are independent with each other and the cost model is defined on per-item basis. Although these assumptions can simplify the algorithm design, it is not always sufficient to reflect the reality where some association rules may exist among the user accesses and caching or transmitting the data items in bulk could be cheaper than in each item individually. We will study these issues issues in the future work. Considering this problem under the storage constraints is another interesting problem. As each node is equipped with fixed storage size, the caching cost would be time dependent and not increase linearly with caching time. Therefore, solutions to the problem with this constraints is more practical in reality and thus worthwhile to study in the future.

REFERENCES

[1] B. Veeravalli, "Network Caching Strategies for a Shared Data Distribution for a Predefined Service Demand Sequence," IEEE Trans. Knowledge and Data Eng., vol. 15, no. 6, pp. 1487-1497, Nov,2003.
[2]J. Edmonds, "Optimum Branchings," J. Research of the Nat'l Bureau of Standards, vol. 71B, pp. 233-240, 1967.
[3] L. Breslau, P. Cue, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," Proc. IEEE INFOCOM, pp. 126-134, 1999.
[4]K. Kalpakis, K. Dasgupta, and O. Wolfson, "Steiner-Optimal Data Replication in Tree Networks with Storage Costs," Proc. Int'l Symp. Database Eng. & Applications, pp. 285-293, 2001.
[5] B. Awerbuch, Y. Bartal, and A. Fiat, "Competitive Distributed File Allocation," Information Computing, vol. 185, pp. 1-40, Aug. 2003.
[6] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive Algorithms for Distributed Data Management (Extended Abstract)," Proc. 24th Ann. ACM Symp. Theory of Computing (STOC '92), pp. 39-50, 1992.