New Algorithms for Planning Bulk Transfer via Internet and Shipping Networks

Brian Cho Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
{bcho2, indy}@illinois.edu

***Abstract*—Cloud computing is enabling groups of academic collaborators, groups of business partners, etc., to come together in an ad-hoc manner. This paper focuses on the group-based data transfer problem in such settings. Each participant source site in such a group has a large dataset, which may range in size from gigabytes to terabytes. This data needs to be transferred to a single sink site (e.g., AWS, Google datacenters, etc.) in a manner that reduces both total dollar costs incurred by the group as well as the total transfer latency of the collective dataset.**

This paper is the first to explore the problem of planning a group-based deadline-oriented data transfer in a scenario where data can be sent over both: (1) the internet, and (2) by shipping storage devices (e.g., external or hot-plug drives, or SSDs) via companies such as Fedex, UPS, USPS, etc. We first formalize the problem and prove its NP-Hardness. Then, we propose novel algorithms and use them to build a planning system called Pandora (People and Networks Moving Data Around). Pandora uses new concepts of time-expanded networks and delta-time-expanded networks, combining them with integer programming techniques and optimizations for both shipping and internet edges. Our experimental evaluation using real data from Fedex and from PlanetLab indicate the Pandora planner manages to satisfy deadlines and reduce costs significantly.

I. Introduction

The emerging world of cloud computing is expected to grow to a $160 Billion industry by 2011 [26]. It is enabling *groups* of collaborators to come together in an ad-hoc manner to collect their datasets to a single cloud location and quickly run computations there [14], [20]. In a typical cloud computation project, very large datasets (each measuring several GBs to TBs) are originally located at multiple geographically distributed sources. They need to be transferred to a single sink, for processing via popular tools such as Hadoop, DryadLINQ, etc. [15], [7], [24], [31]. The participants in such a group may be academic collaborators in a project, business partners in a short-term venture, or a virtual organization.

Significant obstacles to the growth of cloud computing are the long latency and high cost involved in the data transfer from multiple sources to a single sink. Using *internet transfer* can be cheap and fast for small datasets, but is very slow for large datasets. For instance, according to [20], a "small" 5 GB dataset at Boston can be transferred over the internet to the Amazon S3 storage service in about 40 minutes, but a 1 TB forensics dataset took about 3 weeks! At Amazon Web Services (AWS) [1], which has data transfer prices of 10 cents per GB transferred, the former dataset would cost less than a dollar, but the latter is more expensive at $100.

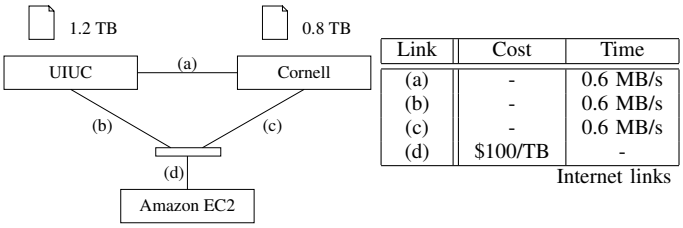
However, there is an alternative called *shipping transfer*, first proposed by Jim Gray [22], the PostManet project [29], and DOT [28]. At the source site, the data is burned to a storage device (e.g., external drive, hot-plug drive, SSD, etc.), and then shipped (e.g., via overnight mail, or 2-day shipping) to the sink site (e.g., via USPS [12], FedEx [3], UPS [13], etc.). AWS now offers an Import/Export service that allows data to be shipped and uploaded to the AWS storage infrastructure (S3) in this manner. Using shipping transfer is expensive for small datasets, but can be fast and cheap for large datasets. For instance, the same 5 GB dataset above costs about $50 via the fastest shipping option, which is overnight. The 1 TB dataset, on the other hand, would also cost $50 and reach overnight, which is both cheaper and faster than the internet transfer.

In this paper, we focus on the problem of satisfying a *latency* deadline (i.e., transfer finishes within a day) while minimizing *dollar cost*. The choice between using shipping vs. internet for a given scenario is far from clear, as we will illustrate via an extended example at the end of this section. Basically, choosing the correct option is a major challenge because there are: (1) multiple sites involved, (2) heterogeneity in sizes of datasets, (3) heterogeneity in internet bandwidth from each source to the sink, and (4) heterogeneity in the shipping costs from each source to the sink.

Thus, it would be unwise for each participant site in the group to independently make the decision of whether to "ship physically or transfer using the internet?" Instead, transferring data *via* other sites might be preferable. In other words, we can leverage an *overlay consisting of many sites and both shipping links and internet links*. This motivates us to model the data transfer as a graph and formulate the problem as finding optimal *flows over time*.

After formulating the problem, we prove its NP-Hardness. Then, we present a solution framework using *time-expanded networks* that can find optimal solutions for small problem instances. We further optimize the solution framework to make it practical for larger problem instances, by applying knowledge about the transfer networks, and by using Δ-condensed time expanded networks. Finally, we build our solution into a system called *Pandora* (People and Networks Moving Data Around). Using the implementation, we present experimental

This paper was supported in part by NSF grant IIS 0841765



Link	Cost	Time
(a)	-	0.6 MB/s
(b)	-	0.6 MB/s
(c)	-	0.6 MB/s
(d)	$100/TB	-

Internet links

Item	Cost	Time
Overnight	$42	24 hrs
Two-Day	$17	48 hrs
Ground	$6	96 hrs

(a)

Cost	Time
$51	24 hrs
$27	48 hrs
$7	96 hrs

(b)

Cost	Time
$59	24 hrs
$28	48 hrs
$8	120 hrs

(c)

Item	Cost	Time
Device Handling	$80/device	-
Data Loading	$18.13/TB	40 MB/s

(d) Amazon EC2 Import/Export Service

Fig. 1: An example network, with internet and shipment link properties. The cost of shipment is for a single 2 TB disk (weighing 6 lbs).

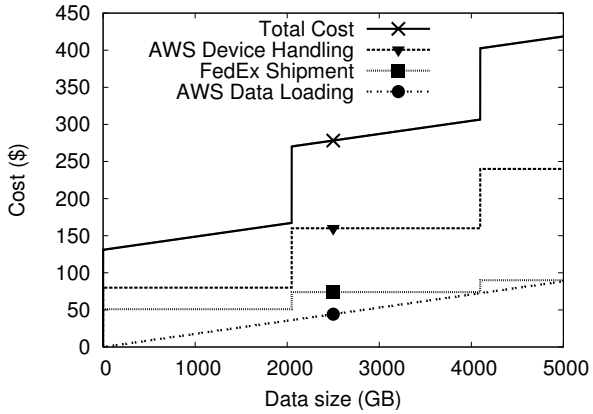


Fig. 2: Cost of sending 2 TB disks from UIUC to Amazon, including FedEx overnight shipping and Amazon handling charges.

results, based on real internet traces from PlanetLab and real shipping costs from FedEx. Our experiments show that optimal transfer plans can be significantly better than naive plans, and realistic-sized input can be processed in a reasonable time using our optimizations.

The Pandora planning system takes in as input the following: the dataset sizes at source sites, the interconnectivity amongst different sites including sources and sink (bandwidth, cost and latency of both internet and shipping links) and a latency deadline that bounds the total time taken for transfer. Its output is a transfer plan for the datasets that meets the latency deadline and minimizes dollar cost.

This paper represents a significant step forward from the state of the art in several fields. Our results extend not only the extensive class of cooperative network transfer systems [25], [2], but also shipping-only approaches such as Jim Gray's SneakerNet [22], the PostManet project [29], and DOT [28]. In addition, our algorithmic results solve a stronger (and more practical) version of the problems addressed in the theoretical literature on network flow over time [18], [23].

Extended Example: Before delving into the technical details of Pandora, we present an extended example to illustrate how different constraints can result in different solutions, even for a fixed topology and dataset sizes. Consider the topology of Figure 1, where there are two source sites (UIUC and Cornell) and one sink site (Amazon EC2). The costs and latencies for both internet transfers (using AWS prices) as well as shipping transfers (using AWS Import/Export prices, with 2TB disks) are shown with the figure.

In the dollar cost minimization version of the problem, the sole objective is to minimize the total dollar cost. The optimal solution to this problem is: send data from Cornell to UIUC via the internet (no cost), load data at UIUC onto a disk and ship to EC2. The total cost of $120.60 is significantly lower than other options such as transferring all data directly to EC2 via the internet ($200) or via ground shipment of a disk from each source ($209.60).

However, this does not work for the latency-constrained version of the problem - the above solution takes 20 days! With a deadline of, say 9 days, the optimal solution is: ship a 2 TB disk from Cornell to UIUC, add in the UIUC data, and finally ship it to EC2. This takes far less than 9 days while also incurring a reasonably low cost of $127.60.

Given an even tighter latency constraint, the best options are: send a disk from Cornell through UIUC to EC2 using overnight shipping, or send two separate disks via 2-day shipping from Cornell and UIUC each to EC2. Given the prices in our input, the latter ($207.60) gives us a price advantage over the former ($249.60). However, it is important to note that even small changes in the rates could make the former a better option.

Finally, we present the case when all data cannot fit into a single 2 TB disk – consider the above example where UIUC has 1.25 TB (an extra 50 GB). Figure 2 shows the shipment and handling costs (for overnight shipping) when the number of disks increases. The cost jumps by over $100 when shipping two disks instead of one. Thus, for the 50 GB of extra data that doesn't fit into a disk, it is better to send it through the internet than to combine it on disk.

II. Problem Formulation

In this section, we present a formal definition of the data transfer problem. We model our problem as a network flow graph with both shipping links and internet links connecting all pairs of sites. We formalize the flow and constraints on the graph *over time*, because data movement is dynamic and time-sensitive.

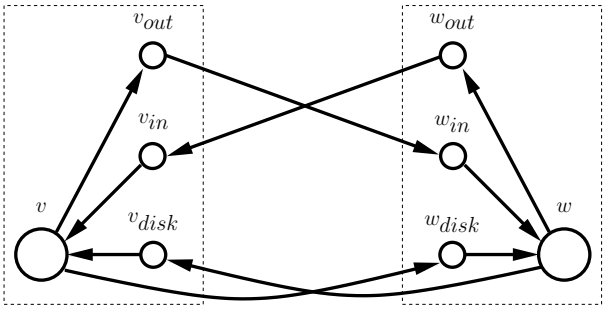


Fig. 3: Two connected sites v, w in our model. A site v is represented by vertices v, v_{out}, v_{in}, and v_{disk} and the edges between them.

A. Graph model

We model our problem as a directed graph, where edges have capacities, costs, and transit times. The edge properties are based on the characteristics of the links connecting sites, and the bottlenecks that appear within sites.

1) Internet and Shipping links: Graph edges represent the transportation of data through the network, composed of internet and disk shipment links. Each link has a *capacity*, a *cost*, and a *transit time*. An internet link has a constant capacity equal to the average available bandwidth, a transit time set to zero (since millisecond latencies are negligible), and a cost of zero (except when terminating in the sink).

Shipping disks in packages has entirely different properties. The first thing to note is that there are many levels of service (e.g. Overnight, Two-day, Ground, etc.) when shipping between sites. We treat each level of service as a distinct link, and the discussion below pertains to any one of these links.

The price paid (cost) for shipment with 2 TB disks is depicted by the "FedEx Shipment" line in Figure 2. This cost is a *step function* of the amount of data transferred. The cost grows with the number of disks, but not linearly with the amount of data inside each disk. For example, sending either 0.2 TB or 1.8 TB has the same cost because a single disk is used, but sending 2.2 TB has a higher cost because an additional disk is needed.

The time that a package takes to reach its destination (transit time) is on the order of hours or days from the time it was sent, however the time in transit is not fixed. Rather, it depends on the time of day – for instance, with overnight shipping, all packages sent from UIUC anytime before 4pm may arrive at Cornell the next day at 8am. The amount of data shipped at once (i.e., capacity) grows with the number of disks sent, on which shipping links impose practically no limits.

In summary, a shipment link has a cost that follows a step function of the amount of data transferred, capacity that is infinite, and a transit time that depends on the time sent.

2) Site bottlenecks: However, knowing the capacity, cost, and transit time of each link is not sufficient. Our model also combines end-site constraints. The combined model is depicted in Figure 3 and elaborated below.

Data sent by disk does not enter a site instantaneously. Rather, an individual at the receiving end must remove the disk from its packaging, then plug in the disk and transfer the actual bytes. The transfer is done through a disk interface such as eSATA, which has a typical transfer rate of 40 MB/s. Only when this transfer is finished is the data transfer considered complete. In addition, this process is not always free. Cloud services charge per-disk and per-data fees, e.g., see the "AWS Device Handling" and "AWS Data Loading" lines of Figure 2.

We formulate our model to include this stage of the data transfer at node v in the following way. We first add a new vertex v_{disk} and new edge (v_{disk}, v). This edge represents both the capacity constraint (e.g. 40 MB for eSATA transfer) and the per-data linear cost (the data loading fee, if this is a sink node); these properties are similar to an internet edge. A shipment link from node w to node v is represented by the edge (w, v_{disk}). This edge has the shipment link properties (step function cost, infinite capacity, and send-time dependent transit time).

There are also end-site constraints for data transferred through the internet. Most often, not all connections can be used to capacity at once, because there is a common bottleneck at the incoming or outgoing ISP. We model this by including two extra vertices v_{in} and v_{out} and corresponding edges (v_{in}, v) and (v, v_{out}). The internet connection between sites v and w is depicted with (w_{out}, v_{in}) and (v_{out}, w_{in}). These edges take the properties of internet links, i.e., capacity equal to available bandwidth, zero transit time, and zero cost.

B. Data Transfer Over Time

Our graph model is formalized as a flow network $\mathcal{N}$ consisting of the set of directed edges A and vertices V discussed above, along with the following attributes on these elements. Each edge in the graph $e \in A$ has a capacity u_e, cost function c_e, and transit time function τ_e. Each vertex $v \in V$ has a demand attribute D_v. Demand represents the data that originates from that vertex, and needs to be transferred to the sink. Vertices having non-zero values of D_v constitute a terminal set $S \subseteq V$. This set is further partitioned into source terminals $v \in S^+$ which have $D_v > 0$ and sink terminals $v \in S^-$ with $D_v < 0$ such that $\sum_{v \in S} D_v = 0$. In this paper, we focus only on the single sink problem, where $|S^-| = 1$.

Now, expanding to consider time as well, flow is assigned to each edge at time unit θ, denoted as $f_e(\theta)$. A flow that is feasible and satisfies demands within deadline T adheres to the following four constraints:

i) Capacity constraints: $f_e(\theta) \leq u_e$ for all $\theta \in [0, T), e \in A$

This ensures that at any time, flow through an edge does not exceed its capacity.

ii) Conservation of flow I:

$$\int_0^{\xi} \left(\sum_{e \in \delta^+(v)} f_e(\theta) - \sum_{e \in \delta^-(v)} f_e(\theta - \tau_e(\theta)) \right) d\theta \leq 0$$

for all $\xi \in [0, T), v \in V \setminus S^+$

This ensures that any non-terminal vertex sends out only as much flow as it has received until then. This does not preclude the storage of flow at vertices.

iii) Conservation of flow II:

$$\int_0^T \left(\sum_{e \in \delta^+(v)} f_e(\theta) - \sum_{e \in \delta^-(v)} f_e(\theta - \tau_e(\theta)) \right) d\theta = 0 \quad \text{for all } v \in V \setminus S$$

This ensures that at time T, there is no leftover flow at any vertex other than the sink.

iv) Demands:

$$\int_0^T \left(\sum_{e \in \delta^+(v)} f_e(\theta) - \sum_{e \in \delta^-(v)} f_e(\theta - \tau_e(\theta)) \right) d\theta = D_v \quad \text{for all } v \in S$$

This ensures that the total amount of flow that has left and entered a terminal node by time T is equal to the amount specified.

Under the above constraints, we would like to find flows that are feasible and satisfy demand, and in addition meet monetary and performance goals. In particular, the problem we focus on in this paper is to minimize dollar cost while satisfying a deadline. That is:

$$\text{Minimize } c(f) := \sum_{e \in A} \int_0^T c_e(f_e(\theta)) d\theta$$

while completing transfer within deadline T. Here c_e is either a linear function or a step function.

Unfortunately, this problem turns out to be NP-Hard.

Lemma 2.1: Solving the data transfer problem is NP-Hard.

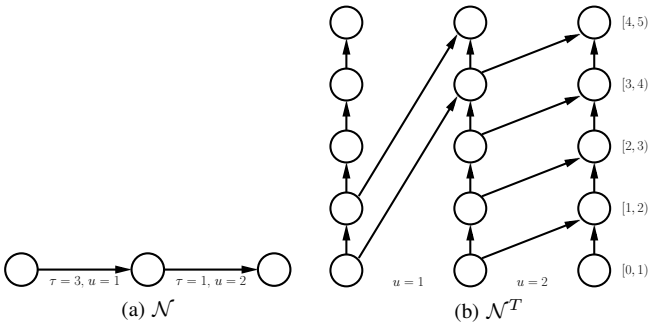
Proof: Our problem is NP-Hard because it is a generalization of the known NP-Hard problem of minimum cost flows over time with finite horizon [23]. That problem has only linear cost functions, where the cost on an edge is defined as a single coefficient k_e on an edge, such that $c_e(f_e(\theta)) = k_e * f_e(\theta)$, while our problem additionally considers costs that are defined as step functions. Also the transit times in [23] are fixed constants τ_e, while in our problem the transit time is a function τ_e that depends on the sending time. ■

III. Pandora Solution

In this section, we obtain transfer plans by deriving an *optimal solution* to the data transfer problem. For small problem sizes this computation can be done quickly. We later explore optimizations for larger problem sizes.

Our approach uses the following 4 steps to solve the problem:

- **(Step 1: Formulate)** Formulate inputs into a data transfer problem with flow network $\mathcal{N}$.
- **(Step 2: Transform)** Transform $\mathcal{N}$ into a static T-time-expanded network $\mathcal{N}^T$.



(a) $\mathcal{N}$ (b) $\mathcal{N}^T$

Fig. 4: An example network, and its corresponding time-expanded network with $T = 5$. Intuitively, a vertex in the time-expanded network represents the original vertex at a point in time. The time range to the right reflects that time passes as we advance up the network.

- **(Step 3: Solve)** Solve static min-cost flow problem; resulting flow is x.
- **(Step 4: Re-interpret)** Re-interpret x as flow over time f.

We showed in Section II how the problem is formulated (Step 1). We expand on the other individual steps in the rest of this section.

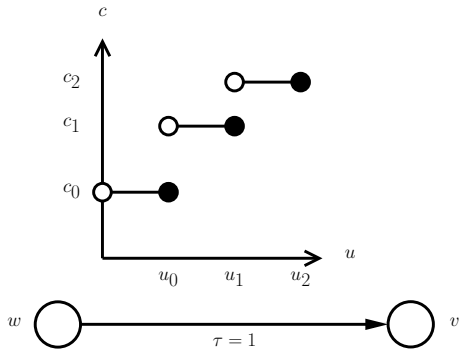
A. Building a Time-Expanded Network

Recall that a flow over time network $\mathcal{N}$ specifies for each edge $e \in A$, a transit time τ_e. A time expanded network is an invaluable concept for solving flow over time problems [18]. This approach works by emulating and absorbing time into an expanded graph representation, and then solving the problem therein. We apply the canonical time-expanded network conversion to edges with linear cost. However for edges with a step function cost we apply a novel conversion.

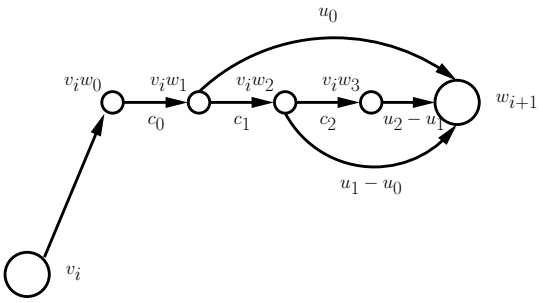
Intuitively, a T-time-expanded network represents the possible flows through all edges and vertices, from 0 up to T time units. Figure 4 depicts an example with 3 nodes and time moving upwards. A canonical T-time-expanded network $\mathcal{N}^T$ creates T copies of each vertex (labeled v_i) in the network. It then replaces edge $e = (v, w)$ in the original graph $\mathcal{N}$ with edges $(v_i, w_{i+\tau_e})$ for $i = 0, 1, ..., (T - \tau_e)$. Intuitively, flow on this edge represents flow originating at v in the original graph at exactly time i. In addition, in order to account for the storage of flow at node v, $T-1$ holdover edges are added between v_i and v_{i+1} with infinite capacity.

For step cost edges, we develop a new conversion that decomposes the edge into multiple intermediary vertices and edges.

The conversion is illustrated with an example in Figure 5. The first intermediary edge $(v_i, v_i w_0)$ represents the transit time τ_e. Then, for each step in the cost function, we add an intermediary vertex and two intermediary edges. The first intermediary edge $(v_i w_0, v_i w_1)$ represents the fixed cost c_0 that must be paid to ship at least one more unit of flow. The second intermediary edge $(v_i w_1, w_{i+\tau_e})$ constrains the amount



(a) Edge (v, w) in $\mathcal{N}$ with step cost function.



(b) Decomposed edge $(v_i, w_{i+\tau})$ in $\mathcal{N}^T$. Unlabeled costs are 0, and capacities are ∞.

Fig. 5: Decomposition of an edge with a step cost function. The costs in $\mathcal{N}^T$ are fixed costs that must be paid if at least a single unit of flow goes through the edge. The vertices in $\mathcal{N}^T$ are v_i, v_iw_0, v_iw_1, v_iw_2, v_iw_3, and w_{i+1}.

of flow that can be sent while paying the fixed cost to capacity u_0. In the next step the intermediary edges are (v_iw_1, v_iw_2) with cost c_1 and $(v_iw_2, w_{i+\tau_e})$ with capacity $u_1 - u_0$, and so on.

As a whole, the edges between the original vertices v_i and $w_{i+\tau_e}$ can hold flow originating at v in the original graph at exactly time i. All flow must arrive at the destination w by time $i+\tau_e$. Therefore, unlike the original vertices, the intermediary vertices cannot store flow, and so no holdover edges are drawn between them.

B. Solving the Static Network Problem

In literature, the static time-expanded network is solved using polynomial time minimum cost flow algorithms [17], [21]. However, these solutions do not apply to our problem because of its unique characteristics. Concisely, in our static problem some of our edges (depicted in Figure 5b) have fixed cost defined by a constant cost k_e that must be paid in full when at least one unit of flow goes through the edge:

$$c_e(f_e) = \begin{cases} k_e & \text{if } f_e > 0 \\ 0 & \text{if } f_e = 0 \end{cases}$$

Given the presence of these edges, we solve the static network problem exactly as a Mixed Integer Program (MIP). We define integer (binary) variables y_e defined on the set of fixed cost edges $e \in F$. The value of y_e is 1 when the edge is used for at least one unit of flow and 0 when it is not used at all. Formally, the problem becomes:

$$\begin{aligned} \text{Minimize} \quad & c(f) := \sum_{e \in A} c_e(f_e) \\ \text{s.t.} \quad & f_e \le u_e y_e && \text{for all } e \in A \\ & \sum_{e \in \delta^+(v)} f_e - \sum_{e \in \delta^-(v)} f_e = D_v && \text{for all } v \in V \\ & y_e \in \{0, 1\} && \text{for all } e \in F \\ & y_e = 1 && \text{for all } e \in A \setminus F \end{aligned}$$

Unfortunately, a solution for the MIP formulation may not be computable within a reasonable time because of two factors. First, the static T-time-expanded network has a size of $O(n \cdot T)$ edges, where T is a numeric value of the input. Thus, even if a polynomial time algorithm on the static network size existed, the network size itself may be exponential in the input of the flow over time problem. [1]

Second, solving the MIP can be time consuming even with a small T, for a large network. The existence of integer variables suggests that the problem is hard to solve. In fact, it is impossible to remove the integer variables (thus making the problem a Linear Program) because the problem is NP-Hard.

Lemma 3.1: Solving the min-cost flow on a static network with fixed cost edges is NP-Hard.

Proof: The proof is by reduction from the Steiner Tree problem in graphs. To solve a Steiner Tree problem, we can convert the original graph to a directed graph by replacing each undirected edge (u, v) with two directed edges $u \to v$, $v \to u$; each directed edge is given unlimited capacity and unit fixed cost. All but one of the terminal vertices are set as source vertices with unit demand, and the remaining terminal vertex is set as a sink vertex with negative demand equal to the number of source vertices.

It is easy to see that a demand-satisfying feasible flow in the converted graph defines a connected tree in the original graph (with same cost): each source is connected to the sink, and thus all terminals are connected. Likewise, each connected tree must be a demand-satisfying feasible flow (with same cost): there is a single path between a terminal and each of the other terminals, and this can be used as the path that flow goes from a source to the sink.

The min-cost flow in the directed graph must be a Steiner Tree (i.e., min-cost connected tree) in the original graph. Assume this is not true, and there is a connected tree with less cost; then there must be a corresponding flow with less cost, which is a contradiction. ■

To solve the MIP we use the branch-and-cut method provided in the GLPK MIP solver. We branch using Driebeck-Tomlin heuristics and backtrack using the node with best local bound [4].

[1] This is intuitively why the min-cost flows over time problem is NP-Hard despite the existence of polynomial time min-cost flow algorithms.

Finally, we describe how we re-interpret the solution in Step 4. When the static solution is solved, Pandora finally transforms the solution x back onto the original network. The transformation is straightforward: take the flow $x_{e=(v_i,w_{i+\tau})}$ going through an edge $(v_i, w_{i+\tau})$ in the static network; in the flow over time, this becomes the flow $f_{e=(v,w)}(i)$ that initiates at v at time $\theta = i$. For edges with step functions, we can take as $f_e(\theta)$ the amount of flow $x_{e'=(v_i,v_i w_0)}$ going through the first edge in the decomposition.

IV. Pandora Optimizations

The previous section showed how to convert the data transfer problem to a Mixed Integer Program (MIP) that can be solved by plugging into a general MIP solver. However, the NP-Hardness result and our evaluation in Section V show that processing time can become impractically long for larger problems. Therefore, we present four optimizations to the MIP problem, in order to improve computation time. The first two optimizations (A and B) are applied based on characteristics of the links that are converted into the static network, and do not affect the optimality of the final solution. In the third optimization (C), we make use of approximation in the form of Δ-condensed time-expanded networks. This allows us to systematically reduce the size of the static network while retaining an optimal minimum cost, but the resulting solution may overstep the deadline by a certain amount. Finally, we present an optimization (D) that allows Pandora to output compact transfer plans that do not contain unnecessary idle time.

A. Reducing shipment links

When shipping a package between sites, we have observed from our experiments with FedEx that there are a small number of possible package arrival times during a day. For example, an overnight package from UIUC sent anytime between noon and 4pm will arrive at Cornell the next day at 10am. Thus, it does not matter whether the package is sent at noon or four hours later.

We use this observation to our advantage. This allows us to reduce the number of shipment links represented in the MIP. When shipping has the same cost and arrival time for different send times, we only need to represent one of these shipment edges in the time-expanded graph. We can ensure this does not affect the deadline or cost by choosing the one with the latest send time. Intuitively, a flow that would have been sent earlier can be stored and sent at the chosen send time and still arrive at the same time. By reducing the number of shipment links present in the MIP, we crucially reduce the number of integer variables y_e.

B. Adding negligible amounts of cost to internet links

Our second optimization relies on the following observation: if data is to be sent over internet links, it makes sense to send data as soon as possible. In other words, there is no gain in storing data at the node. In contrast, when shipping disks we need to wait until a lot of data can be sent at once, because of the fixed costs involved. We use this as an assumption in formulating the MIP, by adding per each internet edge (v_i, w_i) in the time-expanded graph, a negligible cost proportional to the time i (e.g., $\frac{i}{T} \cdot 0.00001$ \$/GB in our experiments). This gives a hint to the MIP solver to send via internet edges as soon as data is available.

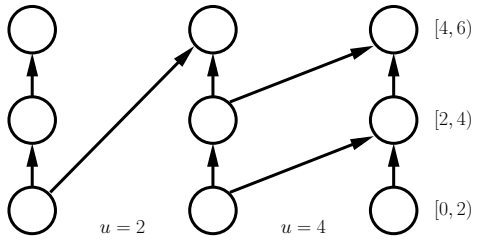


Fig. 6: The canonical Δ-condensed time-expanded network $\mathcal{N}^T/\Delta$ for network in Figure 4, with $\Delta = 2$. Intuitively, Δ consecutive time units at each vertex are compressed into a single vertex. The time ranges to the right reflect this.

C. Δ-condensed time-expanded network

While the last two optimizations can reduce the time needed to obtain a solution, the time-expanded network still grows with T. Thus we utilize Δ-condensed networks to find fast approximate solutions for the flow over time problem [18]. Intuitively, a Δ-condensed network compresses each group of Δ consecutive time units at each vertex in the time expanded network into a single vertex. Further, this is done synchronously across all vertices.

As illustrated in Figure 6, a canonical Δ-condensed T-time-expanded network $\mathcal{N}^T/\Delta$ is constructed of $\lceil T/\Delta \rceil \Delta$ copies of vertices in $\mathcal{N}$. The transit times in a condensed network are rounded up to the nearest multiple of Δ. An edge $e = (v, w)$ in $\mathcal{N}$ is now transformed into edge $(v_i, w_{i+\lceil \tau_e/\Delta \rceil})$. This edge represents all the flow through e originating during a Δ timespan, thus its capacity is set to $u \cdot \Delta$.

We define a modified Δ-condensed network conversion for step cost edges. An infinite capacity step cost edge is decomposed in the same way as in a T-time-expanded network, except that the destination vertex is $w_{i+\lceil \tau_e/\Delta \rceil}$. The capacities u_i do not change because they represent the cost function, rather than the actual capacity of the link (which is infinite).

By condensing the network, we are able to reduce the number of edges by a factor of Δ. However, to insure that we still get a transfer plan with minimum cost, the time expansion of the static network must be increased. The corresponding Δ-condensed network has an increased deadline $T(1+\varepsilon)$, where $\varepsilon = \frac{n \cdot \Delta}{T}$ as we prove in Theorem 4.1. The resulting network has $O(n^2/\varepsilon^2)$ vertices, which is less than the $O(n \cdot T)$ vertices in a regular time-expanded network when T is large and ε is not too small.

Thus, we substitute the time-expanded network in the transform step of the solution in Section III with a Δ-condensed network with time expansion $T(1 + \varepsilon)$. Concretely, step 2 becomes:

- **(Step 2: Transform*)** Transform $\mathcal{N}$ into a static Δ-condensed time-expanded network $\mathcal{N}^T/\Delta$, where $\Delta := \frac{\varepsilon}{n}T$; and time expansion $T' := T(1+\varepsilon)$.

Step 4 (re-interpret) is modified in the following way: Consider flow going through an edge $(v_i, w_{i+\lceil \tau_e/\Delta \rceil})$. Define τ'_e such that $\tau'_e + \tau_e = \lceil \tau_e/\Delta \rceil \Delta$. Notice that $\tau'_e + \tau_e = \lceil \tau_e/\Delta \rceil \Delta$ is the (rounded) transit time in the Δ-condensed network. For a linear cost edge, we convert to flow over time by holding the flow at v for τ'_e and sending $1/\Delta$ of the flow for Δ time units in interval $[\tau'_e + i\Delta, \tau'_e + (i+1)\Delta)$. Fixed cost edges are converted by holding the flow for $\tau'_e + \Delta - 1$ time units and sending the entire flow at once at $\tau'_e + (i+1)\Delta - 1$. These conversions obey flow conservation and capacity. The cost remains the same because the cost function and flow for the edge in $\mathcal{N}^T/\Delta$ and $\mathcal{N}$ remain the same.

Theorem 4.1: A solution f to the data transfer over time problem that finishes within time $T(1+\varepsilon)$ with a cost of C can be obtained from a flow x of the Δ-condensed network with cost C and time expansion $T(1+\varepsilon)$. When C is the minimum cost of the Δ-condensed network, it is also the minimum cost of the original flow over time with deadline T.

Proof: We must show that (a) given a feasible solution to the flow over time problem $\mathcal{N}$ with cost C and time T, there exists a solution to the static Δ-condensed time-expanded network in Step 2 with cost C and time $T(1+\varepsilon)$; and (b) a flow x in the Δ-condensed network can be modified to a flow over time f with the same cost and time.

The conversions in Step 4 prove (b), by showing such a modification.

We now show (a). Because our flow over time network has non-negative costs, there exists an optimal flow f^* with no cycles. We show there exists a feasible flow $\hat{f}$ with cost at most C satisfying demands D by time $T^*(1+\varepsilon)$ in the network with transit times rounded up to the nearest multiple of $\Delta := \frac{\varepsilon}{n}T$; define a topological ordering $\{v_0, v_1, ..., v_{n-1}\}$ of V on the graph with edges in f^*, and for an edge $e = (v_i, v_j)$, set $\hat{f}_e(\theta) = f^*_e(\theta - i\Delta)$. The completion time of $\hat{f}$ is $T^* + \Delta(n-1) \leq T^*(1+\varepsilon)$, with same cost and demands (since flow travels on the same paths). Using storage, we make this a flow in the Δ-rounded network, by holding flow sent on e at v_j for an additional $\Delta(j-i) - \tau'_e \geq 0$ units of time. ■

D. Adding negligible amounts of cost to holdover edges

Our fourth optimization removes unnecessary storage of flow that increases the finish time i.e., when the final byte of data enters the sink. This compaction is done by adding a negligible amount of cost to all the holdover edges, except for those at the sink (e.g, 0.0001 $/GB in our experiments). This is especially useful for Δ-condensed networks, because it is quite possible that the finish time will be beyond the original deadline. While compacting the transfers does not guarantee that the deadline is met, it does cause the finish time to be as soon as possible given the sequence of transfers. Intuitively, at each time step that flow is stored at a vertex, if sending that flow to the sink does not worsen the solution, the cost at holdover edges causes this option to be chosen.

V. Experimental Results

We implemented Pandora and ran experiments to evaluate the system. We present two sets of experimental results. We first show in Section V-A the benefit of using Pandora's flexible graph formulation that produces a cooperative data transfer plan. Next in Section V-B, we present microbenchmarks to evaluate the optimizations made to the MIP formulation.

We evaluated Pandora using trace-driven experiments. The basic topology we used had a single sink at uiuc.edu and 9 additional sites (sometimes used as sources) at .edu domains as listed in Table I. The internet bandwidth between the sites was derived from PlanetLab available bandwidth traces measured using the Spruce measurement tool [27] by the Scalable Sensing Service (S^3) [30] (at 12:32 pm on Nov 15, 2009). We obtained real shipping cost and time data between all sites by using FedEx SOAP (Simple Object Access Protocol) web services [3], with site addresses provided by a whois lookup to the domains. For service charges at the sink, we used Amazon AWS's published costs.

We solve the MIP formulation of the static problem using the GNU Linear Programming Kit (GLPK) [4]. The execution times were taken on a machine with 4GB RAM and two Quad-core 1.86 Ghz Intel Xeon processors; GLPK only used one core.

A. Pandora Transfer Plans

To evaluate the quality of Pandora transfer plans, we ran Pandora on the topology of 10 PlanetLab sites. For the ith experiment, the sites used as source nodes are 1 through i. The total dataset was fixed at 2 TB. The data was spread uniformly over the source nodes. The results are shown in terms of time and cost in Figures 7 and 8.

We compare Pandora's transfer plan to two baseline plans that make independent choices at each node. In the first, called "Direct Internet," all the sites send their data to the sink over the internet. This incurs a total cost of $200 for the total data for all settings. To calculate the time required, we assume optimistically that there is no data bottleneck at the sink. Thus, the time required is equal to the amount of data at the slowest source, divided by the available bandwidth to the sink, as shown in Figure 7.

The second baseline plan, called "Direct Overnight," ships a disk overnight immediately from each source site. While this gives a very fast transfer time of 38 hours, the price of transfer grows increasingly with the number of sources, as shown in Figure 8. This is because the cost of sending a disk is incurred at each source.

Pandora differs from the above two approaches. The flexibility of using both internet and shipping data transfer gives Pandora a wide range of possible plans to choose from. We present results for deadlines of 48, 96, and 144 hours. For the 48 hour deadline, Pandora is slower than the direct overnight shipment, but it gives price savings that are significant. Relaxing the deadline to 96 hours gives us in all cases a cheaper alternative to direct internet transfer, while in most cases reducing the

Index	Site	BW	Index	Site	BW
Sink	uiuc.edu	-	5	rochester.edu	6.9
1	duke.edu	64.4	6	stanford.edu	5.3
2	unm.edu	82.9	7	wustl.edu	2.0
3	utk.edu	6.2	8	ku.edu	6.4
4	ksu.edu	65.0	9	berkeley.edu	7.1

TABLE I: Sites used in experiments. BW is the measured available bandwidth (Mbps) to the Sink.

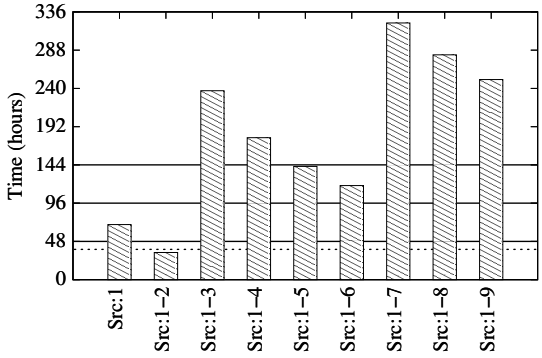


Fig. 7: Time required for Direct Internet transfers for each experiment. Lines are for comparison with Direct Overnight shipment (38) and Pandora deadline experiments (48, 96, 144).

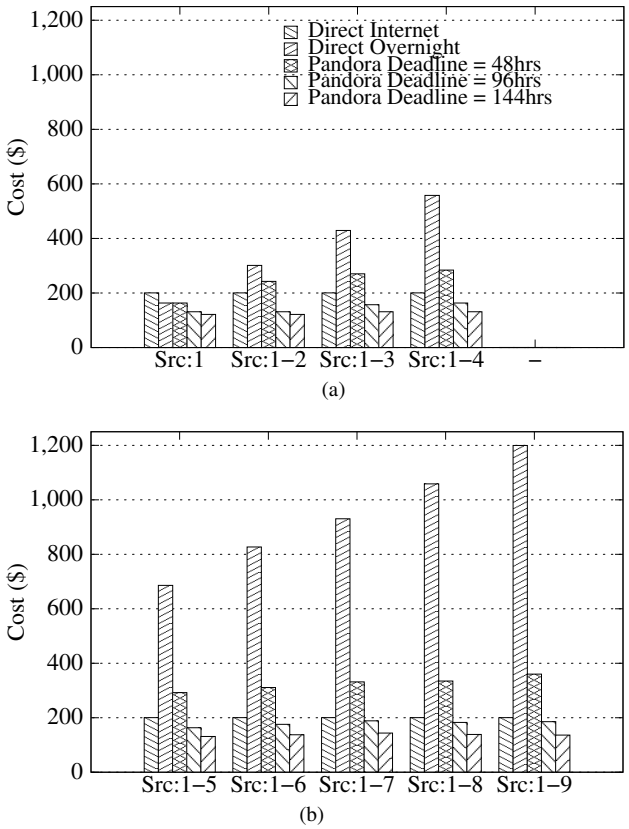


Fig. 8: Cost comparison of transfer plans. Pandora produces low-cost solutions under different conditions and deadlines. Direct transfer approaches are inflexible, and do not adapt well to data being transferred from many locations.

latency as well. A 144 hour deadline gives us even cheaper alternatives.

These results show that Pandora can significantly improve bulk data transfer on realistic topologies.

B. Pandora Optimization Microbenchmarks

We now investigate the computation time taken to create these solutions. It is critical that computation time remains low, so the transfer plans can be executed as soon as possible.

In Figure 9a, we show the computation time taken to solve the problem in the experiment with Sources 1 and 2. The computation times show a general trend of increasing as the deadline increases. This is expected because a deadline increase means the MIP problem size increases. The original MIP computation time increases to above an hour once the deadline is set to be beyond 220 hours. Using our shipment link reduction optimization (Section IV-A) decreases this computation time by a large amount, staying below 3 minutes for deadline up to 240 hours.

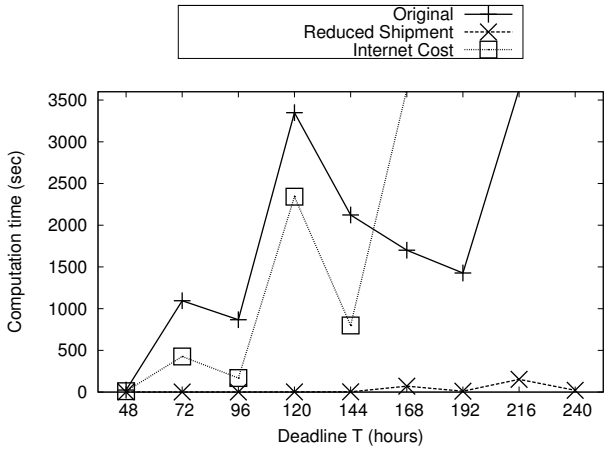
The results for adding costs on internet links (Section IV-B) are mixed. Below a deadline of 150 hours, the computation time is reduced from the original problem, however beyond 150 hours the computation times increase to over an hour. We believe the behavior can be explained due to two different factors. First, the added costs help the small problems find the optimal solution faster. However, the additional costs inflate the input size of the MIP formulation, and thus add more work. This added work is more evident as the problem gets larger, where each added input increases the running time (at worst exponentially, because this is an NP-Hard problem).

Figure 9b shows the computation time for the same experiment at larger deadlines. We see that the reduced shipment optimization keeps the computation time at a reasonable level. In addition, we apply the internet costs to this already reduced MIP and find that the computation time is reduced substantially, and remains below 10 seconds.

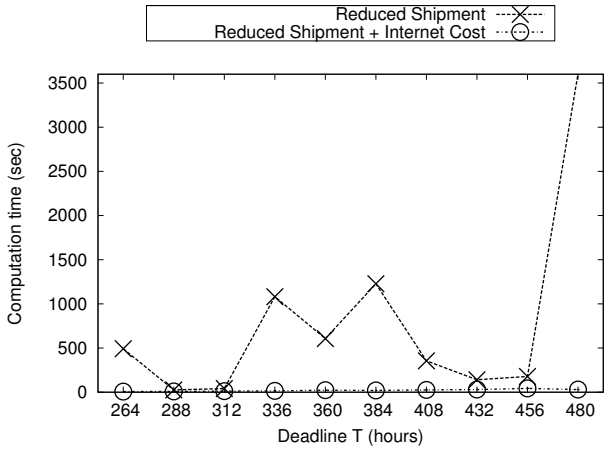
We show that the combined shipment and internet cost optimization continues to do well with larger problems in Figure 9c. In our largest setting, Source 1-9 with 9 sources, the computation time remains fast and stays below 300 seconds.

We next turn our attention to Δ-condensed networks (Section IV-C). Figure 10a compares the Δ-condensed MIP to the original MIP solution. As expected, the Δ-condensed MIP has a faster running time than the original solution.

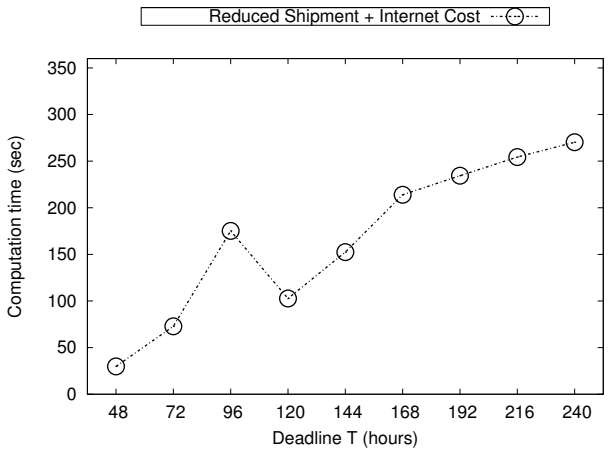
Given these results, we expected to get even greater gains by combining Δ-condensed optimization with the reduced shipment optimization. However, this turns out not to be the case, as we see in Figure 10b. We believe this can be explained by looking at the structure of the networks. Applying Δ-condensing on a network that has already reduced shipment edges to a minimum will not reduce shipment edges. In fact, by extending the time expansion to $T(1+\varepsilon)$ on the network, we may add a number of shipment edges, and thus integer variables.



(a) Computation times for the original MIP formulation, reduced shipment optimization, and internet cost addition optimization, under Source 1-2 settings.

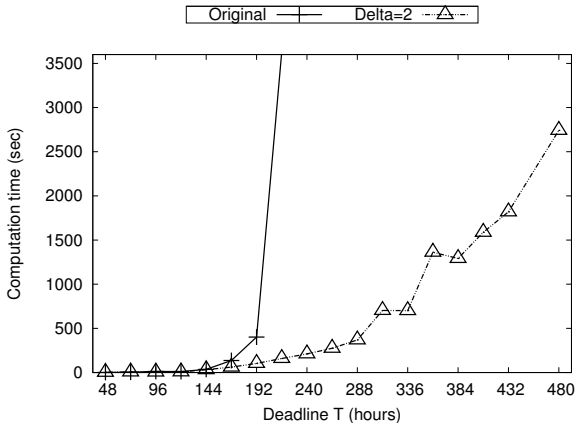


(b) Computation times when only the reduced shipment optimization is applied, and when both the reduced shipment and internet cost addition optimization are applied together, under Source 1-2 settings with relatively large T.

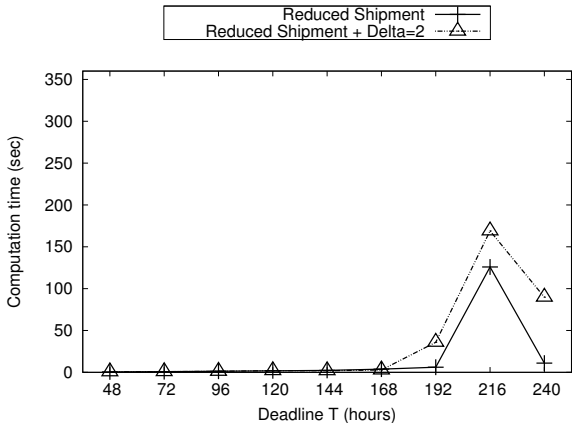


(c) Computation times when both reduced shipment and internet cost addition optimizations are applied, under Source 1-9 settings. The y-axis is scaled to 1/10$^{\text{th}}$ of previous experiments.

Fig. 9: Pandora Optimization Microbenchmarks



(a) Computation times of original MIP and Δ-condensed MIP with Δ=2 using Source 1 settings.



(b) Computation times of reduced shipment optimization and, when the optimized MIP is additionally Δ-condensed, using Source 1 settings. The y-axis is scaled to 1/10$^{\text{th}}$ of previous experiment.

Fig. 10: Δ-condensed network microbenchmarks.

Deadline	Solution
48 hrs	43 hrs
72 hrs	55 hrs
96 hrs	61 hrs
120 hrs	78 hrs
144 hrs	85 hrs

TABLE II: The deadline and finish time of the transfer plan solution, given $\Delta = 2$ in the Sources 1-2 setting. In our results the Δ solutions managed to stay within the deadline.

Finally, we present in Table II the finish time of the solutions for Δ-condensed MIPs. These experiments were run with negligible cost on holdover edges (Section IV-D). We were able to meet all time T deadlines, even though the worst case time is $T(1+\varepsilon)$.

VI. Related Work

Today's cloud services include: (1) industry clouds such as Google AppEngine [5], Microsoft Azure [10], IBM BlueCloud [8], and Amazon AWS [1] which charge customers for each GB transferred and stored, as well as CPU hours used, and (2) academic clouds such as the Google-IBM cluster [16],

and the OpenCirrus [11] cloud computing testbed (CCT) [9] at the University of Illinois. Using Pandora should benefit users with distributed data working on a single cloud, or even those that wish to combine data from multiple distributed clouds.

Previous work has looked at bulk network data transfers in the Grid [6], and on PlanetLab [25]. Yet, the scale of cloud data in the TBs challenges researchers to look for new approaches.

Likewise, the shipping of physical media for data transfer is not new. Jim Gray [22], the PostManet project [29], DOT [28] and others, have explored the feasibility of writing datasets onto a storage drive and shipping it as a means of data transfer. Amazon AWS [1] gives end users the option to use disk shipments to upload data, using the Import/Export service.

At the same time, the economic and performance trade-offs of cloud services, and data transfer in particular, has been gaining interest in the research community [14], [20]. Our work is the first to combine both data networks and shipping physical media, and to devise a single transfer plan that is optimal in respect to economic and performance goals.

Many algorithms for network flows over time using time-expanded networks have been studied since the seminal work of Ford and Fulkerson [19]. [18] introduces Δ-condensed networks that can be used to approximate time-expanded networks in polynomial time. Our work builds on the theoretical literature by applying it to a novel domain. At the same time, we solve a variation of the problem that has not been previously studied.

VII. Conclusion

In this paper, we have presented a solution to the group-based data transfer problem where multiple source sites wish to send disjoint datasets to a sink site. Our solution is the first ever to account for both internet as well as shipping edges. Our transfer planning algorithms outperform both internet-only and shipping-only transfers. In other words, our algorithms satisfy deadlines while simultaneously minimizing dollar costs. Our experimental results involving PlanetLab traces and real data from FedEx show the benefits of our algorithms. We also found that significant improvements can be achieved by using additional optimizations that make reasonable assumptions, or trade off cost optimality for improved computation time.

References

[1] "Amazon Web Services," Website, http://aws.amazon.com/.

[2] "BitTorrent," Website, http://www.bittorrent.com/.

[3] "Federal Express Developer Resources Center," Website, http://fedex.com/us/developer/.

[4] "GNU Linear Programming Kit (GLPK)," Website, http://www.gnu.org/software/glpk/.

[5] "Google App Engine," Website, http://code.google.com/appengine/.

[6] "GridFTP," Website, http://www.globus.org/toolkit/docs/4.0/data/gridftp/.

[7] "Hadoop," Website, http://hadoop.apache.org/.

[8] "IBM Blue Cloud," Website, http://www-03.ibm.com/press/us/en/pressrelease/22613.wss.

[9] "Illinois Cloud Computing Testbed (CCT)," Website, http://cloud.cs.illinois.edu/.

[10] "Microsoft Azure Services Platform," Website, http://www.microsoft.com/azure/.

[11] "Open Cirrus," Website, http://www.opencirrus.org.

[12] "United States Postal Web Tools," Website, http://www.usps.com/webtools/.

[13] "UPS Online Tools," Website, http://www.ups.com/e_comm_access.

[14] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," University of California at Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proc. of USENIX OSDI*, 2004.

[16] K. A. Delic and M. A. Walker, "Emergence of the academic clouds," *ACM Ubiquity*, vol. 9, Aug 2008.

[17] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[18] L. Fleischer and M. Skutella, "Quickest Flows Over Time," *SIAM J. Computing*, vol. 36, no. 6, pp. 1600–1630, 2007.

[19] L. R. Ford and D. R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows," *Operations Research*, vol. 6, no. 3, pp. 419–433, 1958.

[20] S. Garfinkel, "An evaluation of Amazon's Grid computing services: EC2, S3 and SQS," Tech. Rep. TR-08-07, Aug 2007.

[21] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by canceling negative cycles," *J. ACM*, vol. 36, no. 4, pp. 873–886, 1989.

[22] J. Gray and D. Patterson, "A conversation with Jim Gray," *ACM Queue*, vol. 1, no. 4, pp. 8–17, 2003.

[23] B. Klinz and G. J. Woeginger, "Minimum Cost Dynamic Flows: The Series-Parallel Case," in *Proc. of IPCO*, 1995, pp. 329–343.

[24] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: A Not-So-Foreign Language for Data Processing," in *Proc. of ACM SIGMOD*, 2008.

[25] K. Park and V. S. Pai, "Scale and performance in the CoBlitz large-file distribution service," in *Proc. of USENIX NSDI*, 2006, pp. 29–44.

[26] K. Rangan, "The Cloud Wars: $100+ billion at stake," Merrill Lynch, Tech. Rep., May 2008.

[27] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. of ACM SIGCOMM IMC*, 2003, pp. 39–44.

[28] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil, "An architecture for internet data transfer," in *Proc. of USENIX NSDI*, 2006, pp. 19–19.

[29] R. Y. Wang, S. Sobti, N. Garg, E. Ziskind, J. Lai, and A. Krishnamurthy, "Turning the postal system into a generic digital communication mechanism," in *Proc. of ACM SIGCOMM*, 2004, pp. 159–166.

[30] P. Yalagandula, P. Sharma, S. Banerjee, Sung-Ju.Lee, and S. Basu, "S^3: A scalable sensing service for monitoring large networked systems," in *Proc. of ACM SIGCOMM INM (Workshop)*, Sep. 2006.

[31] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, "DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language," in *Proc. of USENIX OSDI*, 2008.