CoShare: A Cost-effective Data Sharing System for Data Center Networks

Hao Zhuang, Imen Filali, Rameez Rahman, Karl Aberer
LSIR, École Polytechnique Fédérale de Lausanne (EPFL)
hao.zhuang@epfl.ch, imen.filali@epfl.ch, rameez.rahman@epfl.ch, karl.aberer@epfl.ch

***Abstract*—Numerous research groups and other organizations collect data from popular data sources such as online social networks. This leads to the problem of data islands, wherein all this data is isolated and lying idly, without any use to the community at large. Using existing centralized solutions such as Dropbox to replicate data to all interested parties is prohibitively costly, given the large size of datasets. A practical solution is to use a Peer-to-Peer (P2P) approach to replicate data in a self-organized manner. However, existing P2P approaches focus on minimizing downloading time without taking into account the bandwidth cost. In this paper, we present *CoShare*, a P2P inspired decentralized cost effective sharing system for data replication. CoShare allows users to specify their requirements on data sharing tasks and maps these requirements into resource requirements for data transfer. Through extensive simulations, we demonstrate that CoShare finds the desirable tradeoffs for a given cost and performance while varying user requirements and request arrival rates.**

***Keywords*—data sharing, cost-effective, small data center, trade-off management**

I. Introduction

In the era of Big data, heterogeneous data are generated at a huge scale from many different domains, such as online social networks (OSNs), sensor networks, medical images, scientific measurements (e.g., CERN [1]) and Internet-wide measurements. This brings new challenges on how efficiently share these big data for research community and industry. Hereafter, we consider two typical examples of applications that highlight big data sharing scenarios.

Example 1: The social data generated from online social networks such as Twitter, Facebook and FourSquare is rich in semantic content on almost every aspect of human life, which has a broad appeal for academia, industry and governments. Due to the openness of social network platforms, researchers can obtain the social data through Open APIs to verify their hypotheses, to mine the interesting patterns, and to develop business models for real applications. Invariably different organizations undertake such efforts in isolation. The generated datasets, ranging from hundreds of Megabytes to several Terabytes, are stored in local data centers and isolated, resulting in many small *data islands*. The value of these small data islands. e.g., in data analytics, is also limited, in that they only serve the immediate purpose of data collectors. If the data collected by different organizations is shared together, the community at large can have access to a larger data source allowing them to not only validate their hypotheses but also expedite wider collaboration and foster scientific progress.

Example 2: Similar to peer-produced systems like Youtube or Wikipedia, consider a collaborative sensor network system (e.g., SenseWeb in Microsoft [2]) to enable peer production of sensing applications. The contributors in the sensor network system deploy their own sensors or sensor networks for their own dedicated applications, such as environmental monitoring system for air pollution, surveillance camera network for security, heart rate monitor and step counter for runners, road safety for Intelligent Transportation Systems (ITS), etc. All these data are isolatedly stored in many Small Data Centers (SDCs) in diverse geographical areas. However, if these data can be efficiently shared together, many more applications can be developed. Thus, we need an effective data sharing system to coordinate data transfers among different applications and collaborators.

However, this data sharing process among different data islands incurs a large amount of data traffic as well as a huge cost on network bandwidth. These two main issues may discourage individual data centers from sharing their data. This process becomes more challenging because of the lack of a cost-effective data sharing system which is able to coordinate the data transfer. A client-server approach like Content Distribution Networks (CDNs) [3] for data sharing, requires servers that have high storage capacity and bandwidth to maintain a centralized repository. On the other hand, sharing data in a P2P manner can distribute the high cost of ownership to different peers [4]. Nevertheless, current implementations of P2P data sharing systems such as BitTorrent, focus on fast data dissemination based on incentive compatible data sharing strategies, e.g., Tit for Tat. Their main aim is to maximize the utilization of peer's bandwidth without considering the cost of data transfers [5].

What is needed is an approach that allows individual data centers to collaborate in a lightweight manner while providing cost and performance-effective data transfers, i.e., the system should find the most cost-effective solution given the performance requirements. Traditional P2P file swarming solutions [5] are lightweight, but delegate upload duties to all involved nodes. In a system that seeks to transfer massive amounts of data[1] between data centers ideally only those nodes that are less costly should be selected for upload duties. Furthermore, the scheme should respect users' requirements in terms of data transfer time. For example, if multiple data centers based in Switzerland, Poland and Tunisia are sharing data, and if the dataset to be replicated does not require to be shared urgently, then the most relevant solution would be choose data centers based in Tunisia as the uploading nodes (assuming cheap bandwidth cost is in Tunisia). On the other

[1]Which is of higher orders of magnitude in size than the smaller sized software updates and media files, usually served by P2P systems

	AWS			AZure			Google			
Location \Usage	1GB	10TB	50TB	5GB	10TB	50TB	1TB	10TB	10TB+	**Destination**
US/Europe	0	0.09	0.085	0	0.087	0.083	0.23	0.22	0.20	China
Asia (Tokyo)	0	0.14	0.135	0	0.138	0.135	0.19	0.18	0.10	Austraila
Sao Paulo	0	0.25	0.23	0	0.181	0.175	0.12	0.11	0.08	Others

TABLE I: Price Models of Three Cloud Providers (dollars per GB)

hand, if the data is required to be replicated crucially in the network, then data centers in Switzerland and Poland should also be included among the uploading nodes.

Motivated by this problem, we design a **Co**st-effective data **Shar**ing system, **CoShare**, which allows users to specify these high-level performance goals. As an example, for a data sharing task, users can attach a priority such as *high, medium, low* to the task.The priority indicates their preference on the data sharing such as completion time and monetary cost. The system will map the user's requirements on data sharing tasks into resource requirements and determine *which* and *how many* resources need to be assigned to the tasks. Existing systems [4], [6], [7] lack efficient mechanisms for translating users' requirements into resource requirements. CoShare provides such a mechanism for a data sharing network of collaborating data centers. The key idea of CoShare is to efficiently manage the tradeoff between cost and performance in such a network. Specifically, our contributions can be summarized as follows:

- We analyze the tradeoff curve between time and cost in a data sharing network based on different sharing plans and present a system interface that allows users to specify their requirements.
- We formulate the problem of finding a sharing plan with both cost and performance constraints and present our algorithm to find a sharing plan that matches user's preferences.
- Through extensive experiments based on our simulator, we demonstrate that CoShare is able to find the desirable tradeoffs between cost and performance given varying user requirements and request arrival rates.

II. Data Sharing Networks

In this paper, we consider a data sharing network of small data centers (SDCs) which represent local SDCs of different research organizations and companies [8]. Each SDC generates a dataset and then replicates this dataset to SDCs that already expressed their interest to it. This sharing process continues until all interested SDCs receive this dataset. In the following, we briefly present two typical network architectures for data sharing and discuss the tradeoff between conflicting cost metrics in a data sharing network.

A. Network Architecture for Data Sharing

Dropbox [9] is a representative of commercial cloud-based solutions that provide users with file synchronization services. As it has a centralized architecture, all the data needs to be uploaded to the cloud first and then other interested clients can download the data from the cloud. This client-server approach for data sharing requires dedicated servers and high network bandwidth, which leads to high cost for data sharing. This architecture is not suitable for our scenario since participant SDCs are self-organized, and with limited budgets for data sharing. On the other hand, P2P-based data sharing approaches distribute the large volume of data between peers directly without a central entity and thus there is no up-front cost on building the network. Nonetheless, current P2P swarm protocols such as BitTorrent and Avalanche fail to provide guarantees on the cost and performance of data sharing [4].

In this paper, we design CoShare inspired by P2P network architecture. To manage the tradeoffs between performance and cost in the data sharing network, we add a centralized entity called *sharing manager* which coordinates the data transfer among SDCs[2]. The coordination is performed in terms of the *sharing mode* which determines the direction of data exchange. There are three types of sharing modes between SDCs: 1) *sharing*: SDCs can upload/download from each other; 2) *downloader*: download only; and 3) *uploader*: upload only. The communication between the sharing manager and SDCs belonging to the data sharing network can be summarized as follows. After receiving the sharing request from a given SDC, the sharing manager will ask each interested SDC to report its current bandwidth usage and generate a *sharing plan* considering both performance and cost of data sharing. Here, the sharing plan is defined as a set of sharing modes corresponding to each SDC in the network. The sharing plan will be sent to all SDCs. Once it is received by SDCs, they can communicate with each other directly by using BitTorrent-like protocols. The shared dataset is divided into small data blocks and then indexed by block hashes. After establishing connections, interested SDCs will exchange their indexes and then synchronize the dataset by requesting missing blocks from others.

B. Cost Metrics in Data Sharing

The goodness of a data sharing network can be measured in terms of several metrics such as monetary cost, link utilization, network congestion, etc. In this paper, we discuss two main metrics: *monetary cost* and *completion time*.

We define the *monetary cost* of serving a sharing request as the total bandwidth cost of all SDCs that contribute to the data sharing. Greenberg *et al.* [10] have revealed that the network cost amounts to around 15% of operational costs to a data center. The bandwidth cost is usually charged based on the percentile-based charging scheme. For example, the common billing method is referred to as the "95th Percentile Rule" [1] in which an Internet Service Provider (ISP) measures the traffic volume that a data center generates every 5 minutes. At the end of a charging period, all 5-minute samples will be sorted and the highest 5% samples will be discarded. However, when it comes to the charging scheme between data center and users, the 100-percentile charging scheme is applied. This implies that data center providers will charge all the data traffic without removing the peak traffic. Table 1 summarizes the piece-wise linear price models for network bandwidth of

[2]We note that this is a lightweight component and need not be centralized. If nodes use gossip protocols or Distributed Hash Table (DHT) services to keep abreast of the state of the network, and the sharing node runs a lightweight tracker, then no central entity is needed.

three cloud data centers, namely Amazon, Google and Azure. For instance, for AWS data center in Asia, the price of data traffic within 1GB is free while that of traffic between 1GB and 10TB is charged at 0.14$/GB. Table 1 clearly shows that besides the total volume of data traffic, geographic location is also a crucial factor that influences the bandwidth cost. For example, the price of bandwidth in Sao Paulo is nearly two times than that in US/Europe area in Amazon data center. In particular, the price of bandwidth in Google also depends on the location of data traffic destination regardless of source location. Furthermore, it is worth to mention that all the cloud data centers only charge on upload links while downloading from external Internet is usually free. Thus, in our scenario, we assume that the participating data centers are located in different areas with various bandwidth price models. This setting gives us the opportunity to optimize the total cost of data sharing.

The other metric we consider is the *completion time* of serving a sharing request. For each sharing request issued by a SDC, the completion time is measured from the time that the source SDC starts to upload the dataset until the time that all interested SDCs finish downloading all the dataset. Many factors have an impact on the completion time. This includes the dataset size, bandwidth and monetary cost constraints.

C. Tradeoff in Cost Metrics

As we mentioned in Section 1, there exists a tradeoff between cost and performance (in terms of transfer time) in a data sharing process. For better understanding, consider the scenario depicted in Figure 1 where we have a swarm of four SDCs with the same bandwidth capacity but different prices. The SDC 1 wants to share a dataset with the other three SDCs. To achieve this goal, different plans can be employed as is shown in Figure 1. Specifically, Figure 1 (a) shows that in the Client-Server plan (P1), three client SDCs (with *downloader* modes) only download the data from the server SDC 1 (with the *sharing* mode) while Figure 1 (b) depicts the Carry-Forward plan (P2) according to which SDC 1 shares the dataset with SDC 4 first and then SDC 2 and SDC 3 download the data from SDC 4. The benefit of P2 is that it can reduce the cost if the price of SDC 4 is lower than that of SDC 1. As we assume that the bandwidth capacity is the same, the completion time of plans P1 and P2 is also the same. Without the loss in completion time, the cost of P2 is lower than that of P1. In this sense, we say that P1 is *dominated* by P2. The plan

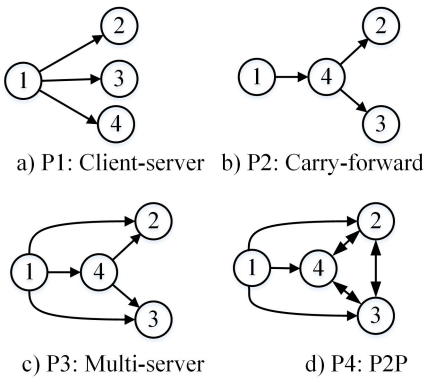

Fig. 1: Different sharing plans

P2 can even be the best plan in terms of cost if we assume that the price of SDC 4 is the lowest among all four SDCs. On the other hand, to reduce the completion time, we can add more upload capacities to transfer the data. This is depicted in Figure 1 (c) which shows the Multi-Server scenario (P3) in which two more links are added compared to P2 to accelerate the data sharing. Thus, SDC 2 and SDC 3 can download from SDC 1 and SDC 4 simultaneously. We can observe that P3 is better with respect to the completion time, whereas P2 is better in terms of cost. In this case, we say that P2 and P3 are *non-dominated* sharing plans, i.e., there is no single plan better than the other in both cost and performance. Figure 1 (d) depicts the P2P sharing plan (P4), in which the best completion time can be achieved since all SDCs' bandwidth are fully used; however this makes it costlier.

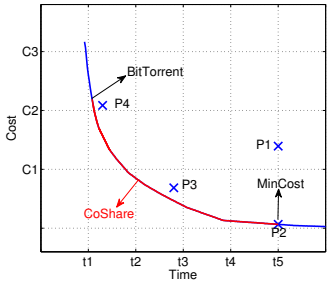

Fig. 2: Tradeoff management by CoShare in terms of cost and time. While BitTorrent and MinCost are two extremes on the tradeoff curve (e.g., the most fastest but expensive versus the slowest but cheapest), CoShare aims at finding a desirable tradeoff and adapting to bandwidth usage.

From Figure 1, we can see that each sharing plan represents a tradeoff between performance and cost. Given a sharing plan, we can plot its cost and performance on a 2-dimensional plane. In Figure 2, the four sharing plans with different cost and performance are presented. Given a target performance on the completion time, we define a non-dominated sharing plan as the one with *the minimal cost* for that target performance. We refer to the curve in Figure 2 as the '*tradeoff curve*', which is comprised of all the possible non-dominated sharing plans. In Figure 2, P1 is not on the tradeoff curve because the cost of P1 is not the minimal one given the completion time *t5*. P2 has the same completion time at a lower cost, and thus dominates P1. In this sense, only non-dominated plans are of interest. Selecting a sharing plan from all non-dominated plans largely depends on user's preferences. In CoShare, we design three priority levels, namely *high*, *medium* and *low*, for users to specify their preferences in order to manage their tradeoff on the data sharing. The priority indicates users' preference on the performance metric, i.e., completion time in our system. For example, if a user assigns high priority to a sharing request, CoShare will choose a non-dominated sharing plan with relatively faster completion time.

Thus, the key questions that we aim to answer can be reformulated as follows: *Given a swarm of SDCs with different bandwidth capacities and price models, 1) how can we find all the non-dominated sharing plans? 2) From these plans, which is the desirable sharing plan that satisfies the user's requirements?*

III. CoShare : Cost-effective Data Sharing

In this section, we present the design of CoShare. Firstly, we discuss how to generate all non-dominated sharing plans on the tradeoff curve. Then, we will evaluate the goodness of sharing plans from the tradeoff curves in terms of users' preferences. Finally, we will present the algorithm in CoShare that selects the desirable sharing plan.

A. Generating Non-dominated Sharing Plans

We consider a data sharing network composed of a set $\mathcal{P}$ of heterogeneous SDCs, which is denoted by $\mathcal{P} = \{p_1, \ldots, p_n\}$. Each SDC $p_i \in \mathcal{P}$, is characterized by its maximum upload and download capacities, denoted respectively by u_i and d_i. We adopt the 100-percentile price model with free charge on ingress traffic, i.e., the data center only charges on the egress links. Thus, the cost function can be defined as a piecewise linear nondecreasing function $\mathcal{C}(x)$, where x indicates the total volume of data. A sharing task is specified as a tuple $\{p_{src}, \mathcal{S}(p_{src}), Q\}$ where p_{src} is the SDC of data source, $\mathcal{S}(p_{src}) = \{p_j | p_j \in \mathcal{P} \wedge j \neq src\}$ is the set of SDCs which subscribe to the dataset from the p_{src} and Q is the size of the dataset. The data sharing process follows the one-to-many data transfer model, i.e., dataset is replicated from one SDC to the interested SDCs in the network. Suppose $p_s \in \mathcal{P}$ proposes a sharing task $\{p_s, \mathcal{S}(p_s), Q\}$. Given the current available bandwidth ratio ρ, we formulate the optimization problem to minimize the total sharing cost as:

$$min \sum_{i=1}^{n} \mathcal{C}_i(\sum_j x_{ij}) \tag{1}$$

subject to

$$\sum_i \sum_j x_{ij} = (|\mathcal{S}(p_s)| - 1) * Q \tag{2}$$

$$0 \leq \sum_{j \neq i} x_{ij} \leq u_i * \rho * T, \forall i, j \tag{3}$$

$$\sum_{i \neq j} x_{ij} = Q \leq d_j * \rho * T, \forall i, j \tag{4}$$

$$Q \leq \sum_{j \neq s} x_{sj} \leq u_s * \rho * T \tag{5}$$

$$\sum_i x_{is} = 0, \forall i \tag{6}$$

$$x_{ij} \geq 0, \forall i, j, i \neq j \tag{7}$$

where x_{ij} is the total amount of data uploaded from p_i to p_j and T is the expected transfer performance in terms of completion time. As we mentioned in Section II-B, we evaluate the performance of data sharing not only in terms of completion time but also in terms of the cost which refers to the monetary fees for bandwidth cost. The objective of problem 1 is to minimize the total cost with the performance constraints on completion time T. Constraint 2 guarantees that all interested SDCs have downloaded this data of size Q. Constraints 3 and 4 specify the bandwidth constraints for both upload and download capacities. With constraints 5 and 6, we ensure that the SDC of data source p_s uploads at least all the data without any data downloading. The parameter ρ indicates the mean available bandwidth ratio of the data sharing network, which reflects the current load in the network. Thus, given a target performance T, we can compute the optimized cost of data sharing by solving problem 1. By varying all possible values of T, we can derive the tradeoff curve. Considering this curve as an input, the remaining issue that we have to resolve is to determine the data sharing plan with desirable tradeoff that satisfies user's requirements.

B. Goodness of Sharing Plan

Through solving problem 1, we can obtain the tradeoff curve with all non-dominated sharing plans. No single plan on the tradeoff curve has both lower cost and lower completion time. Our system will evaluate the goodness of a sharing plan with respect to users' preferences. To map users' preferences to the sharing plan, we define an adjustable tradeoff factor tf as the *marginal utility* which indicates the marginal improvement in completion time by adding additional unit costs. For example, given two sharing plans sp_i, sp_j with different completion time and cost $< t_i, c_i >$ and $< t_j, c_j >$, the marginal utility of two plans is computed as absolute value of $tf = |(c_j - c_i)/(t_j - t_i)|$. In this sense, tf is the slope of two points, which evaluates the rate of change of sharing cost with respect to the sharing completion time. Taking the tradeoff curve given by Figure 2 as an example, a sharing plan with tf in the curve is the point where the slope of the curve becomes higher than tf when going from right to left. tf equals to 0 indicating a sharing plan with the lowest cost while higher value of tf means less completion time at the higher cost. Based on this analysis, different priorities of sharing tasks correspond to different values of tf. We have to note that the value of tf is influenced by the system configurations, such as the total available bandwidth, request arrival rate and dataset size.

C. Searching Desirable Sharing Plan

To determine the tradeoff factor tf, we firstly identify two main factors in problem 1, dataset size Q and mean available bandwidth ratio ρ, which exert great influence on the generation of tradeoff curves. The exact values of these two factors can only be known at the time of sharing while other factors such as upload/download capacities and bandwidth cost can be derived when the SDC joins the network. Given the varying available bandwidth in the system over time, the same task may generate different tradeoff curves while on the other hand, tasks of different sizes may also present similar performance. For example, if the tradeoff factor tf is set as a fixed value, the first task of size Q_1 can achieve this tradeoff as long as the available bandwidth reaches $\rho < 1$ while the second task of the larger size $Q_2 > Q_1$ can reach the same tradeoff only when all the bandwidth is available, i.e., $\rho = 1$. Thus, if we fix the value of tf for all tasks, some tasks have to wait until there is enough available bandwidth to satisfy the user-specified tradeoff. To avoid this issue, we relax the tradeoff factor into an interval and map user's preference to the corresponding tradeoff interval. For instance, if the tradeoff factor interval is $[tf_1, tf_2]$, the task can be served as long as the system can search a sharing plan with the tradeoff factor $tf \in [tf_1, tf_2]$. The final intervals for different priorities are derived based on the whole system configurations. We

Algorithm 1 Sharing plan search algorithm

Require: a sharing request $req = \{p_s, \mathcal{S}(p_s), Q\}$, request priority $l \in \{high, medium, low\}$, request arrival rate λ
Ensure: a sharing plan **sp**

```
1: t_start = max{Q/u_s, Q/min{d_i}}; //lower bound
2: t_exp = meanReqInterArrival(λ); //upper bound
3: peerDCStatus = getPeerDCStatus(); //obtain each SDC status
4: while t_min <= t_max do
5:    t_mid = midtime(t_start, t_exp);
      //apply linear programming solver
6:    solverResult = LPSolver(req, l, peerDCStatus, t_mid);
7:    if solverResult.accept < 0 then
8:       t_start = t_mid + 1
9:    else if solverResult.accept > 0 then
10:      t_exp = t_mid − 1
11:   else
12:      break;
13:   end if
14: end while
15: return solverResult.sp
```

will provide a comprehensive discussion regarding the tradeoff factor in Section IV.

To search a sharing plan that satisfies a desired tradeoff, we apply the binary search over the performance space, i.e., the space of completion time. Algorithm 1 shows the skeleton of sharing plan searching algorithm. The bandwidth Constraints 3 and 4 in problem 1 show that the completion time may be constrained either by the upload bandwidth of data source or the minimal download bandwidth of other SDCs. In other words, the completion time of data sharing satisfies $t_{start} \geq max\{\frac{Q}{u_s}, \frac{Q}{min\{d_i\}}\}$. This gives the start value of the completion time (line 1 in Algorithm 1). On the other hand, we expect that the current sharing request can be served before the next request arrives in the system. Thus, the expected completion time t_{exp} can be the average inter-arrival time of requests (line 2 in Algorithm 1). Given start time t_{start} and expected completion time t_{exp}, we can carry out the binary search over the time interval $[t_{start}, t_{exp}]$ (lines 4-14 in Algorithm 1). For each search, the system will apply a linear programming solver to solve problem 1. In practice, t_{exp} can be extended to a *hard* deadline by a scale factor if we cannot seek for sharing plan in the interval $[t_{start}, t_{exp}]$. After extension, if there is still no sharing plan with desirable tradeoff, the sharing request will put back into the request buffer to wait for more available bandwidth.

IV. Evaluation

In this section, we will introduce the evaluation methodology of our system by presenting the simulator, the workload as well as the performance metrics. Then, we will illustrate detailed experiments and results under different settings.

A. Simulator

We developed a discrete event simulator to simulate the interactions between entities forming the data sharing network. More specifically, two entities in CoShare are implemented: the sharing manager and SDCs. The sharing manager is responsible for coordinating the data sharing process among different SDCs. The incoming sharing requests generated by SDCs will be buffered in a queue and be processed by the sharing manager in a first-come-first-served manner. To serve the sharing request, the manager will firstly ask each interested SDC to report their current bandwidth usage. Based on this information, it searches for a sharing plan by solving problem 1. Without loss of generality, we assume the cost function to be linear, i.e., $c(x) = ax$ where x is total volume of egress traffic. To solve problem 1, the sharing manager employs MOSEK [11] to solve the linear programming problem and generate the tradeoff curve. After finding a desirable plan, it will send the sharing plan to each interested SDC in the network. Once the sharing plan is received, interested SDCs can start the data sharing process. From the network performance perspective, we assume that the bandwidth of each SDC is equally shared by all the links for both download and upload. Each SDC calculates the download/upload throughput as the ratio of the total size of data download/upload at a regular measurement interval, i.e., 100ms. As mentioned in Section II-A, the data is transferred in a unit of small data block and the size of small block in our simulator is set to 1 MB. To compute the transfer delay, the system will first find out the minimal rate between local upload rate and remote download rate. Then, the transfer delay is calculated as the ratio between the size of a data block and this minimal rate.

B. Simulation Setup and Metrics

Setup: We simulate a swarm of 50 SDCs with the download and upload bandwidth uniformly distributed in the intervals [80Mbps, 160Mbps] and [40Mbps, 80Mbps] respectively. The price of bandwidth is obtained from public cloud providers and CDN providers. The most expensive price is 0.25 $/GB from Amazon data center in south America (Sao Paulo) while the cheapest one is 0.06 $/GB from MaxCDN in Europe area. The average price of all 50 SDCs is about 0.153 $/GB. The size of datasets for each sharing request is uniformly distributed in the interval [300MB, 500MB]. For each dataset, we randomly generate the number of interested SDCs which belongs to the interval [16, 32] (i.e., the average size of interested SDCs is about 24). Unless otherwise stated, the default priority of the sharing request is *low*. We simulate a total of 200 requests with a total aggregated data size of nearly 2TB. Requests are generated according to a Poisson distribution with a mean arrival rate of λ. We tune the system bandwidth utilization by varying λ. For each experiment, we run the simulator 10 times with different random seeds and the variance in results is as low as about 5%.

Metrics: The efficiency of CoShare is evaluated through two groups of performance metrics: 1) *Sharing cost and completion time*: Sharing cost is the monetary fees on the

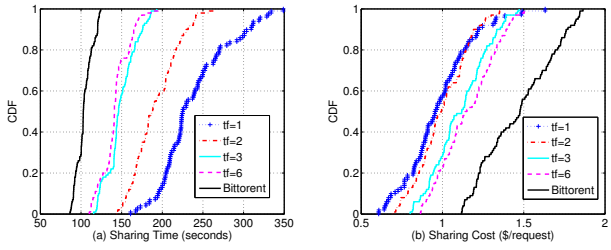

Fig. 3: CDF of tasks with fixed tradeoff factors

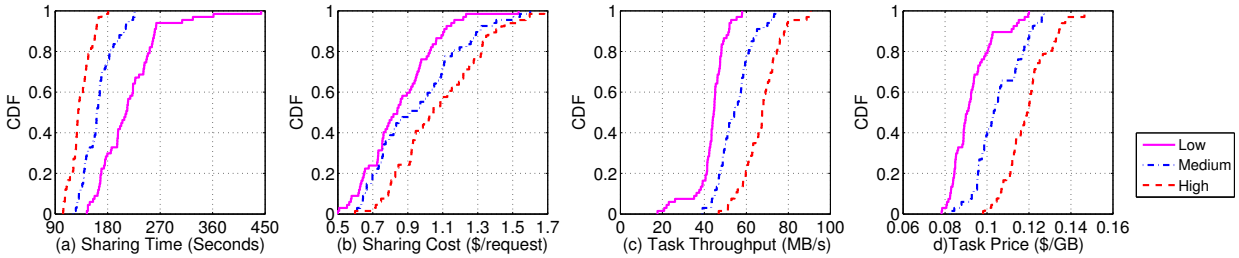

Fig. 4: Tradeoffs for tasks with high, medium and low priority.

bandwidth and completion time is the total time consumed by serving a sharing request; 2) *Task throughput and price*: Since individual sharing requests are of different sizes, i.e., each task replicates different amount of data with interested SDCs, we compute the task throughput as the ratio between the total amount of data transferred and the completion time. The task price is calculated as the ratio between the total sharing cost of a task and the total amount of data transferred.

C. Simulation Results

Hereafter, we present the performance evaluation of CoShare. In particular, we are interested in answering the following questions: how to determine the value of tradeoff factors tf? What is the effect of priority levels specified by different users on the performance of tasks? What is the impact of the system workload on the average cost and aggregated throughput of the whole system? What are the benefits achieved by our design? To answer all these questions, we have conducted three groups of experiments.

Fixed Tradeoff Factors. In this group of experiments, all the tasks arrive with the same tradeoff factor tf. The request arrival rate is set to $\lambda = 12$ requests per hour, i.e., requests arrive about every 300 seconds. Under this configuration, we vary the tradeoff factor tf from 1 to 6. Figure 3 depicts the variation of the Cumulative Distribution Function (CDF) of tasks as function of time and cost under different tradeoff factors values. Note that for comparison, we also simulate the BitTorrent protocol under which all SDCs contribute their bandwidth to finish data sharing as fast as possible. As we can clearly see, BitTorrent has the shortest completion time, but at the *avoidable* high cost compared with our system. This is explained by the fact that SDCs using BitTorrent protocol aim to finish downloading datasets as fast as possible without caring about the transfer cost. A closer look to Figure 3(a) shows that BitTorrent enforces that all SDCs finish data sharing within about 120 seconds. After this period, they stay idle for about another 180 seconds before serving the next request, which makes it meaningless to get fast speed at the high cost as shown in Figure 3(b). From other side, Figure 3(a) and 3(b) also illustrate that tasks with lower tradeoff factors are served with less bandwidth in order to reduce the sharing cost, thus leading to longer completion time. Specifically, three groups of tasks with $tf = 2, 3, 6$ can also finish the task before the next request arrives at the much lower cost (by reducing about 25% compared with BitTorrent's cost). The tasks with $tf = 4, 5$ are omitted in the Figure 3 since their performance is between $tf = 3$ and $tf = 6$ and highly similar with $tf = 6$. The reason for the similar performance of tasks with different tradeoff values is that the tradeoff curve decreases slowly in this interval. For example, as it is shown in Figure 2 (cf. Section II-C), the sharing plans on the tail of the curve (e.g., time between *t3* and *t4*) shall present similar performance since the curve decreases slowly. Tasks with $tf = 1$ have the lowest cost reduced by about 20% than that of group with $tf = 3$ and by about 50% than that of BitTorrent. However, from Figure 3(a), we observe that only 85% of them can finish before the next request arrival, which means that about 15% requests will be scheduled on the SDCs with more expensive cost. As a result, we can notice that about 15% of requests with $tf = 1$ has much higher cost than others as it is reported by Figure 3(b).

User-specified Priority. Through this set of experiments, we aim to study the effect of the user-specified priorities on the task performance. In this scenario, all the tasks arrive in CoShare with different user-specified priorities. Based on the observations deduced from Figure 3, we map these priorities, i.e., high, medium, low to the tradeoff intervals [3,6), [2,3) and [1,2), respectively. One criteria for selecting suitable intervals is to find the turning points in the tradeoff curve where the curve decreases quickly. Other criteria such as task price and throughput can also be applied to find the interval that can differentiate sharing plans in terms of distinctive user-specified priorities. Then, we assign three priorities to tasks with the same proportion (i.e., nearly 33% for each priority). Figure 4 demonstrates the tradeoff of tasks with different priorities with respect to different metrics. It can be clearly seen that CoShare is able to apply users' preferable tradeoff into data sharing effectively. Specifically, for sharing completion time and cost depicted by Figure 4 (a) and 4(b), all tasks with high priorities can finish within around 180 seconds at the high cost (i.e., about 60% of the tasks with costs large than 1.0 dollars) while only 25% of tasks with low priorities can achieve completion within approximately 180 seconds with low sharing costs (i.e, 80% of them have costs less than 1.0 dollars). Furthermore, Figure 4(c) and Figure 4(d) present respectively the CDF of task throughput and price. We can observe that the average throughputs of tasks with high, medium, low priority are about 42, 50 and 65 MB/s at the price of about 0.093, 0.106 and 0.120 $/GB, respectively. Thus, based on these results, we can safely conclude that CoShare provides the *guarantee* on cost and performance for serving users' sharing requests. In particular, this guarantee is tunable by giving different values or intervals for tradeoff factors. For example, in order to increase/decrease the price of high priority task, we only need to adjust the corresponding tradeoff interval.

Effect of request arrival rate. Our objective through this group of experiments is to investigate the impact of the request arrival rate on both throughput and price of tasks with different priorities. For this purpose, we generate two groups of tasks

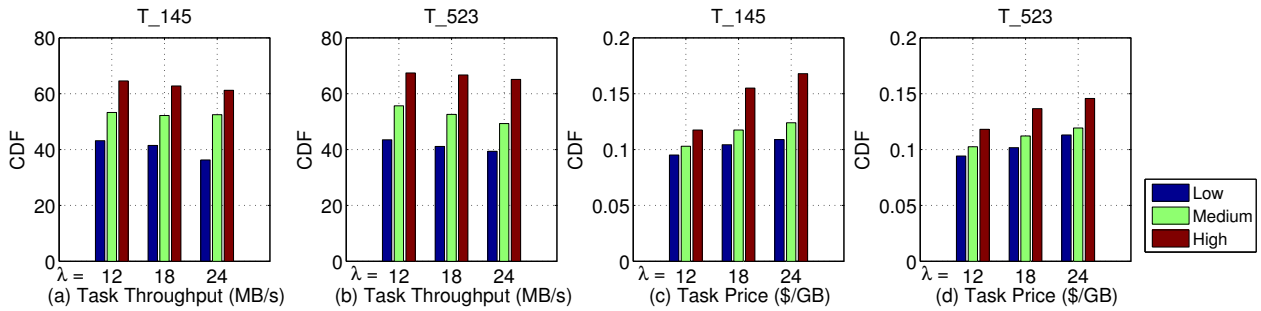

Fig. 5: Tasks with priorities in different proportions under various request arrival rates.

with various priorities in different proportions. In the first group, denoted by T_145, only 10% of generated tasks have a high priority while tasks with medium and low priorities represent respectively 40% and 50% of the total tasks. In the the second group, T_523, the ratio of tasks with high, medium and low priorities is fixed respectively to 50%, 20% and 30%. The difference between these two task groups is that the majority of SDCs in T_145 prefer to share the data at the low cost while those in T_523 are weighted in favor of sharing it in a fast way. We run the experiments by varying the request arrival rate $\lambda = 12, 18, 24$ requests per hour. There are two main findings that can be observed from Figure 5. First, SDC's assignments of different priorities in sharing requests exert little impact on the system performance with regard to the task throughput. As it is shown in Figure 5(a) and 5(b), we observe that task throughput of two groups of tasks is similar. Moreover, the system can also guarantee enough throughput for sharing requests of each priority under different request arrival rates. Second, keeping the same experimental settings, Figure 5(c) and 5(d) depict the variation of the price of both groups of tasks. As expected, we observe that the task price is greatly affected by the request arrival rates. Not surprisingly, we can observe that the task price increases with the increase of request arrival rate. This can be explained by the fact that more expensive bandwidth is utilized as more requests arrive at the system within the same time unit. We also observe that in Figure 5(c) where the most of tasks are with low priority, the variation in the task price of different priorities are much higher than these in Figure 5(d) where the majority of tasks have high priority. In the group of tasks T_523, and under the highest request arrival rate, most of the bandwidth resources with lower price are occupied by requests with high priorities. Under this situation, tasks with low priority are served by expensive bandwidth with high probability, which results in relatively higher task price. Thus, the difference in the task price of various type is diminishing. This also illustrates that the focus of CoShare is the tradeoff between the time and cost rather than minimizing the cost only. For each request, CoShare will find a sharing plan satisfying the corresponding tradeoff factor rather than the one with the lowest cost.

Discussion. Based on our extensive simulation of data sharing networks, we show that CoShare effectively helps users to manage the tradeoff between cost and performance with respect to three different priority levels. However, we may find that the advantage gained by CoShare will be decreased with the increase of the number of requests with high priorities. Table II summarizes the cost-savings of two groups of tasks under different request arrival rates compared with the cost of BitTorrent. We can observe that the relative cost-savings achieved by CoShare can decrease from 34% to 10.8% as more requests with high priority arrive at the highest request arrival rate ($\lambda = 24$).

	$\lambda = 12$	$\lambda = 18$	$\lambda = 24$
T_145	34.0%	18.6%	11.6%
T_523	29.1%	18.3%	10.8%

TABLE II: Total cost savings of CoShare compared with BitTorrent's cost.

On the other hand, we also observed that determining suitable tradeoff factors for a data sharing system is an iterative process. This suggests that the system designer will have to experiment in order to find the suitable tradeoff factors for her particular data sharing network. In the real network scenario, this process could take a very long time, which may incur time and monetary costs. Thus, our methodology provides an easy and economical way for system designers to speed up this process.

Finally, we discuss three major impediments that can limit the practicality of our model. Firstly, do SDCs, especially those that belong to areas of low bandwidth cost, have any interest on our globally optimized solution? Our global optimization can obtain the optimal cost-savings *overall*. The problem of how to distribute these cost-savings to individual members such that no single SDC has the incentive to leave the swarm is an orthogonal issue to ours. Secondly, do these SDCs report their bandwidth cost honestly? One way to solve this issue is to employ a trust model [12] for our data sharing system to discourage SDCs' dishonest behavior. Thirdly, since most researchers commonly store their data locally, can we extend our model to a more general setting that incorporates, besides SDCs, local hard disks? Currently, we only focus on the data center networks in which the number of peers is far smaller than that in traditional P2P network. Thus, optimization in a relatively small SDC network can be easily solved by using existing linear programming packages such as MOSEK. However, considering individual researchers with local hard disks will lead to a dramatic increase in the number of peers, which will in turn will increase the number of variables quadratically. When the number of variables is too large, no generic linear programming solver that we are aware of can be applied. In that case, we will have to develop a specific approximate algorithm to solve Problem 1 presented in Section III, e.g., a greedy algorithm. Another solution to include local disks of individual researchers is to assign the nearest SDC to each peer. For each instance of data sharing, peers can upload their data to the nearest SDC first and then the data will be replicated from this nearest SDC to the others

using the CoShare system directly.

V. Related Work

Many related works focused on scheduling data transfers among data centers with different optimization objectives. The store-and-forward strategy adopted by [1], [13], [14], [15] aims to schedule the data transfer in order to minimize the cost [1], [14] and to readjust to the resource fluctuations [13], [15]. To increase the throughput and thus minimize the transfer time, StorkCloud [6] integrated multi-protocol transfers aiming at optimizing the end-to-end throughput while considering the link capacity, disk rate and the CPU capacity. NetStitchter [13], employed a network of storage nodes to schedule the data transfer in a store-and-forward manner based on predictions on the available leftover bandwidth at access and backbone links. In order to decrease the cost, Postcard [14] formulated an online optimization problem to minimize costs on inter-datacenter traffic by exploiting the price discrepancy in different locations. Other works such as [7] designed a set of transfer strategies to readjust to different network conditions (e.g., network latency, available bandwidth) between cloud data centers whereas [16] focused on the evaluation of different data transfer protocols for big data. However, none of these works considers the tradeoff between the time and the cost during the data transfer process. Unlike previous presented works, our work manages this tradeoff through system interfaces, which allows users to specify their preferences, and provides an efficient mechanism to find this tradeoff. Though Wu *et al.* [17] proposed scheduling approaches for bulk data transfers with different urgency levels. In their work, the priority assigned to a data transfer only concerned the job scheduling without taking into account the time-cost tradeoff.

Our work is not alone in managing the tradeoff between time and cost. Works presented in [18] and [3] optimized the time and cost performance for online services (e.g., search engine) and content multihoming (e.g., CDNs), respectively whereas our work focused on the data sharing networks. Tudoran *et.al* [19] proposed transfer as a service for multi-site cloud with considering the cost in both computation and network. They did not consider the tradeoff between the cost and time. Moreover, they also failed to provide interfaces for users requirements. As for data sharing in P2P networks, most of the works focus on maximizing the throughput [20], [21], [4] using the BitTorrent protocol. Capota *et al.* [20] formulated a maximum flow optimization problem while [21] improved neighbor selection based on traffic locality. Peterson *et al.*[4] presented the response curve of P2P swarm, which represents the swarm bandwidth as a function of seeder bandwidth. Their system assigns the seeder resources to the swarms with the steepest response curves. However, their optimization still did not consider the bandwidth cost. To the best of our knowledge, ours is the first work that manages the time-cost tradeoff for data sharing networks.

VI. Conclusion

This paper serves as a first step towards studying the tradeoff between cost and performance for massive data sharing among small data centers. We proposed CoShare, a cost-effective data sharing system that maps users requirements into resource requirements for managing the time-cost tradeoff. Through extensive experiments, we demonstrated that CoShare is effective in this mapping and guarantees the desirable cost and performance tradeoff for the data sharing process. Future efforts will focus on implementing a real data sharing system based on CoShare. We would also like to incorporate a semantic component for data sharing. For instance, this component can be able to manage the data deduplication in transferred datasets (e.g., Twitter datasets), which further reduces the total amount of data that have to be transferred.

References

[1] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram, "Delay tolerant bulk data transfers on the internet," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, pp. 229–238, ACM, 2009.

[2] A. Kansal, S. Nath, J. Liu, and F. Zhao, "Senseweb: An infrastructure for shared sensing," *IEEE multimedia*, no. 4, pp. 8–13, 2007.

[3] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, "Optimizing cost and performance for content multihoming," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, ACM, 2012.

[4] R. Peterson and E. G. Sirer, "Antfarm: Efficient content distribution with managed swarms.," in *NSDI*, vol. 9, pp. 107–122, 2009.

[5] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, pp. 68–72, 2003.

[6] T. Kosar, E. Arslan, B. Ross, and B. Zhang, "Storkcloud: Data transfer scheduling and optimization as a service," in *Proceedings of the 4th ACM workshop on Scientific cloud computing*, pp. 29–36, ACM, 2013.

[7] R. Tudoran, O. Nano, I. Santos, A. Costan, H. Soncu, L. Bougé, and G. Antoniu, "Jetstream: Enabling high performance event streaming across cloud data-centers," in *Proceedings of the 8th ACM International Conference on Distributed Event-based Systems*, ACM, 2014.

[8] H. Zhuang, R. Rahman, and K. Aberer, "Decentralizing the cloud: How can small data centers cooperate?," in *Peer-to-Peer Computing (P2P), 14-th IEEE International Conference on*, pp. 1–10, Ieee, 2014.

[9] "Dropbox." https://www.dropbox.com/.

[10] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[11] "Mosek lp solver." https://www.mosek.com/.

[12] L.-H. Vu, M. Hauswirth, and K. Aberer, "Qos-based service selection and ranking with trust and reputation management," in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pp. 466–483, Springer, 2005.

[13] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 74–85, ACM, 2011.

[14] Y. Feng, B. Li, and B. Li, "Postcard: Minimizing costs on inter-datacenter traffic with store-and-forward," in *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pp. 43–50, IEEE, 2012.

[15] Y. Wang, S. Su, A. X. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," *Computer Networks*, vol. 68, 2014.

[16] B. Tierney, E. Kissel, M. Swany, and E. Pouyoul, "Efficient data transfer protocols for big data," in *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pp. 1–9, IEEE, 2012.

[17] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. Lau, "Orchestrating bulk data transfers across geo-distributed datacenters,"

[18] Z. Zhang, M. Zhang, A. G. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing cost and performance in online service provider networks.," in *NSDI*, pp. 33–48, 2010.

[19] R. Tudoran, A. Costan, and G. Antoniu, "Transfer as a service: Towards a cost-effective model for multi-site cloud data management," in *SRDS*, 2014.

[20] M. Capota, N. Andrade, T. Vinkó, F. Santos, J. Pouwelse, and D. Epema, "Inter-swarm resource allocation in bittorrent communities," in *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pp. 300–309, IEEE, 2011.

[21] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in bittorrent via biased neighbor selection," in *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pp. 66–66, IEEE, 2006.