Moving Deferrable Big Data to the Cloud by Adopting an Online Cost-Minimization Approach

Baojiang Cui, Xiaohui Jin, Peilin Shi

School of Cyberspace Security, Beijing University of Posts and Telecommunications, China
Nation Engineering Laboratory for Mobile Network, China
cuibj@bupt.edu.cn, jinxiaohui@bupt.edu.cn, noahstone@163.com

Abstract

As cloud computing gets popular in recent years, the bandwidth cost of data centers becomes a hot research topic. For the analysis jobs based on MapReduce framework, locally generated big data usually does not need uploading immediately. Instead, certain delay is tolerable. Therefore, we can use the allowable delay time to optimize the bandwidth usage and minimize the cost.

In this paper, we discuss how to use the allowable delay window that a given workload has and propose two algorithm to reduce peak volume by increasing the maximum transmission of early stages. The experiments show that the peak value can be reduced by choosing a larger initial value.

Besides, we also discuss how to assign workloads among data centers in the cloud scenario. We point out that the total bandwidth cost of data centers will be minimal when the maximum transmission capacity of these data centers are generally equal to each other.

Keywords: Cloud computing, Bandwidth, Map reduce, Zoom-heuristic smoothing algorithm, Fast start heuristic smoothing algorithm

1 Introduction

There are many cloud service providers at present, providing us with a wide variety of cloud service. Among all these cloud service providers, Amazon [1], Google [2], Microsoft [3] and Alibaba [4] are the best-known vendors. Cloud computing is highly elastic and allows on-demand access to computing resources. It has already brought great convenience to users and is well accepted in many ways. A great deal of Internet applications are running on the cloud. Many medium and small-sized enterprises tend to build their online application systems by renting cloud services.

In addition to online application systems, data analysis is also an important category of cloud computing applications. As the fourth paradigm of scientific research [5] is eventually accepted, people are getting used to making decisions based on data analysis. To cope with the big data challenge, the cloud platform with highly flexible computing power is a natural choice. However, cloud users have to upload a great amount of local generated data to the cloud before executing their data analysis jobs.

As we know, a large number of data centers, which are typically geographically distributed, bear the cloud services. In order to transmit data from cloud users to data centers, we have to rent the transmission bandwidth offered by Internet Service Providers (ISPs). However, the bandwidth cost charged by ISPs is not cheap. Like the power consumption problem, the bandwidth cost is also not ignorable for the total cost of data centers. Therefore, in order to provide cost-effective cloud computing services to the users, researchers are exploring methods to reduce the bandwidth cost of data centers.

In general, Most ISPs do not use the total-volume based charge model or the flat-rate charge model. Instead, they have adopted a so-called θ-th percentile charge model. In the θ-th percentile charge model, the ISP divides the charge period, e.g., 30 days, into small intervals of equal fixed length, e.g., 5 minutes. Statistical logs summarize traffic volumes witnessed in different time intervals, sorted in ascending order. The traffic volume of the θ-th percentile interval is chosen as the charge volume. For example, let θ=95, the cost is proportional to the traffic volume sent in the 8208-th ($95\% \times 30 \times 24 \times 60/5 = 8208$) interval in the sorted list [6-8]. The MAX contract model is simply the 100-th percentile charge scheme. Such percentile charge models are perhaps less surprising when one considers the fact that the cost of infrastructure is more closely related to the peak than the average demand [9].

According to this kind of charge model, we can find that peak usage determines not only the network bandwidth of the month, but also the cost of data centers. Therefore, we should try to work out the optimization method by taking full advantage of the characteristics of data analysis jobs. Luckily, different from online applications, certain delay when uploading data is tolerable in the most of analysis jobs. In this

*Corresponding Author: Xiaohui Jin; E-mail: jinxiaohui@bupt.edu.cn
DOI: 10.3966/160792642018081904023

case, we can use the allowable delay window size to optimize the bandwidth usage and minimize the cost.

More specifically, we study the case when users use MapReduce framework to do data analysis. Google engineers originally propose the MapReduce framework [10], which quickly becomes one of the most popular computing models in the field of big data analysis. Recently, Google claims they have already abandoned MapReduce [11]. However, there is still a large number of real world applications using this framework. Moreover, MapReduce represents a general pipeline model in cloud computing. Other newly developed frameworks will continue to employ the design philosophy behind it. In our research, we assume that Mappers and Reducers are geographically distributed, and they connect with each other by employing the transmission services provided by different ISPs. Cloud users will generate plenty of local data, which needs to be uploaded to the data centers while certain delay is tolerable.

In this paper, we focus on minimizing the bandwidth cost when uploading deferrable big data. Deferrable here means that all of the generated data has to be uploaded to the cloud, but certain delay time is allowable. We improved the online algorithm provided in [12] to achieve a better result for reducing the bandwidth cost of data centers. We will discuss related work that other scholars have studied in Section 2, and especially focus on the research of uploading deferrable big data. Section 3 briefly introduces the MapReduce framework at first, and then presents the bandwidth cost model. We discuss two scenarios and propose algorithms that can further reduce the bandwidth cost. The evaluation results are in Section 4. Finally, we conclude the paper in Section 5.

2 Related Work

In the era of cloud computing, the cost of large-scale data centers has drawn a lot of attention. Many factors will affect the total cost of data centers, while the power consumption and the bandwidth occupation are the two major ones. Comparatively speaking, the research on how to reduce power consumption starts earlier. Yao et al. [13] initially provide an optimal offline algorithm, the YDS algorithm, to minimize the power consumption by scaling CPU speed. They assume that the power consumption is a convex function of the CPU speed. After that, researchers carry out several extension work from different aspects. For example, by adopting the total volume based energy charge, computed by integrating instantaneous power consumption over time, Albers et al. [14] design an online algorithm for multi-processor job scheduling without inter-process job migration. Adnan et al. [15] consider deadline constraints and analyze the energy minimization problem in a data center when available deadline information of the workload can be used to defer job execution for reduced energy consumption.

The initial research on the cost of big data transmission has a basic assumption that all the data has to be uploaded immediately. Cho et al. [16] work with total volume based charge model and design a static cost-aware planning system for transferring large amounts of data to the cloud provider via both the Internet and courier services. Considering a dynamic transfer scheme, Zhang et al. [17] propose two online algorithms to minimize the total transfer cost.

However, as mentioned in the previous chapter, ISPs tend to employ the percentile charge model. Therefore, Goldenberg et al. [6] study the multi-homing problem under 95-percentile traffic charge model. Grothey et al. [8] investigate a similar problem through a stochastic dynamic programming approach. They both leverage ISP subscription switching for traffic engineering to minimize the charge volume.

Golubchik et al. [18] carry out the first research pointing out the deferrable data transmission problem. The authors focus exclusively on the single point-to-point link case. They propose an online algorithm called Simple Smoothing Algorithm (SSA). This algorithm involves evenly smoothing every input across its window of tolerable delay for upload. Although SSA is really a simple algorithm, it can approach the offline optimum within a small constant under the MAX model.

In [9], the authors assume that ISPs adopt the MAX contract model and study how to minimize the bandwidth cost for uploading deferral big data to a cloud platform for processing by a MapReduce framework. They first analyze the single ISP case and then generalize to the MapReduce framework over a cloud platform. In the single ISP case, they design a Heuristic Smoothing Algorithm (HSA) whose worst-case competitive ratio is proved to fall between $2-1/(D+1)$ and $2(1-1/e)$, where D is the maximum tolerable delay. In the cloud scenario, they employ the HSA as a building block, and design an efficient distributed randomized online algorithm, achieving a constant expected competitive ratio.

Inspired by [9], our research team successively propose Dynamic Self-adaption Algorithm (DSA) [19] and Improved Dynamic Self-adaption Algorithm (IDSA) [20] for the cloud scenario, and compare the experimental results with the randomized online algorithm in [9]. On this basis, [12] extends the HSA to support multiply deferrable big data, which are produced by local cloud users and have its own delay window sizes. The paper studies a basic single ISP case at first and proposes a Multi-Heuristic Smoothing Algorithm (MHSA) for the single case. It is proved that the worst-case competitive ratio of the MHSA falls between $2(1-(1-D_{max})^{D_{max}})$ and 2, where D_{max} is the maximum delay window size. For the cloud scenario,

[12] designs the Multi-Dynamic Self-Adaption Algorithm (MDSA) to optimize the cloud scenario based on the Multi-Heuristic Smoothing Algorithm. The simulation experiments demonstrate that the total cost can be reduced by 12% when adopting the Multi-Dynamic Self-Adaption Algorithm.

3 System Model

In order to go on with the previous research work in [12], we assume that cloud users keep generating large amounts of local data at all hours and employ the MapReduce framework on the cloud computing platform to carry out their big data analysis jobs. The cloud computing platform is made up of geographically distributed data centers. Some of these data centers work as Mappers, while others work as Reducers. During a MapReduce processing job, the first thing that users have to do is uploading data to a data center, which works as a Mapper. After that, the intermediate data generated by Mappers will be sent to certain data centers which are responsible for the reducing jobs. The ISPs employed between users and Mappers, Mappers and Reducers are probably different, and have their own charge models.

Moreover, the most important hypothesis here is that the data generated by users is deferrable. In other words, a reasonable amount of uploading delay (often specified in service level agreement, or SLA) is tolerable by cloud users. In our research, we will not restrict the allowable delay time to the same value. Inside, we focus on the deferrable big data which has different delay window sizes.

Later in this chapter, we will define the mathematical model of bandwidth cost minimization problem. Two simplified scenarios will be discussed based on this model. First, we will consider the single ISP case which means there is only one Mapper and one Reducer with only one ISP between them. After that, we will extend the single ISP case to the cloud scenario that contains multiple Mappers and Reducers with different ISPs between these data centers. Figure 1 shows the network structure used to carry out experiments in the cloud scenario. There are five data centers in this cloud computing platform. As shown in the figure, DC1, DC2 and DC3 are Mappers, while DC4 and DC5 are Reducers.

But before all of these, we will make a brief introduction to the MapReduce framework.

3.1 MapReduce

MapReduce is a parallel architecture for large scale data processing [10]. A typical MapReduce program contains two critical functions called Map and Reduce.

The Map function maps a key/value pair (K1, V1) of the input data to a temporary key/value pair (K2, V2). Each key/value is independent of the operation and

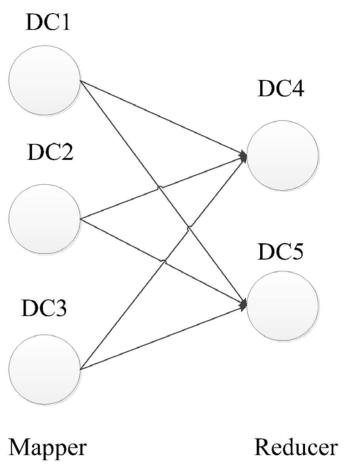

Figure 1. Data centers of MapReduce framework

does not modify the original data, so the Map operation can be highly parallel.

The Reduce function receives keys for specific K2 of a group of temporary key/value pairs, merging these results to obtain the final result (K2, V3), where key K2 is constant. It is relatively difficult to achieve a number of parallel reductions.

Besides these two functions, there are some more functions available. For example, there are some optional steps before the Reduce function, such as Combine, Shuffle and Sort. We will ignore these steps in our discussion. In addition, the time consumption of the Map function is also ignored in our following model.

3.2 Mathematical Model of Bandwidth Cost Minimization

Let M and R represent the set of Mapper data centers and Reducer data centers. ISP_m ($m \in M$) is the ISP employed between the user and Mapper m. The maximum traffic volumn of ISP_m in a charging cycle is denoted as V_m^{max}. Similarly, we use $ISP_{m,r}$ $(m \in M, r \in R)$ to represent the ISP employed between Mapper m and Reducer r, and the maximum traffic volumn of $ISP_{m,r}$ in a charging cycle is denoted as $V_{m,r}^{max}$.

We assume that all the ISPs employ the MAX contract model. Therefore, the cost function of ISP_m can be denoted as $f_m(V_m^{max})$, and the cost function of $ISP_{m,r}$ can be denoted as $f_{m,r}(V_{m,r}^{max})$. As discussed above, the two cost function f_m and $f_{m,r}$ are both non-decreasing and convex. So, our work now is to build a mathematical model to minimize the sum of all these f_m and $f_{m,r}$ under certain constraints.

When a workload is generated by the user, its maximum delay window size is also specified.We use D_i to represent one of multi-delay window sizes of workloads. Let $D_{max} = Max\{D_1, D_2, \ldots, D_i\}$. We use W_t^i to represent each workload released at the user location at time slot t with the delay window size D_i. Let $x_{t,d}^{m,i}$ be a decision variable indicating the proportion of the workload W_t^i assigned to the Mapper m at time slot $t+d$. According to the definition of $x_{t,d}^{m,i}$, we have:

$$0 \le x_{t,d}^{m,i} \le 1, \forall m \in M \quad \textbf{(1)}$$

We denote

$$n_m = \sum_{d=0}^{D_i} x_{t,d}^{m,i}, \forall t, m \quad \textbf{(2)}$$

To ensure all workloads can be uploaded to the cloud, we have:

$$\sum_m n_m = 1 \quad \textbf{(3)}$$

In time slot t, the traffic volumn V_m^t between the user and mapper is:

$$V_m^t = \sum_{D_i} \sum_{d=0}^{D_i} W_{t-d}^{D_i} x_{t-d,d}^{m,i} \forall m \quad \textbf{(4)}$$

V_m^{max} is the maximum traffic volumn of ISP_m, so:

$$V_m^{max} - V_m^t \ge 0 \ \forall t \quad \textbf{(5)}$$

Next, we calculate the traffic volumn between Mapper and Reducer. We assume that the ratio of the size of a mapper's output data to the size of a reducer's input data is β. Let $y_{m,r}^t$ represents the portion of the output of Mapper m that is transmitted to Reducer r at time slot t. So we have:

$$V_{m,r}^t = \beta V_m^t y_{m,r}^t \quad \textbf{(6)}$$

Similarly, we know that $V_{m,r}^t$ satisfies:

$$V_{m,r}^{max} - V_{m,r}^t \ge 0 \ \forall t \quad \textbf{(7)}$$

In general, the bandwidth cost of data centers can be calculated by adding the cost funtions of the user-to-mapper part and the mapper-to-reducer part. Therefore, the expression of the overall traffic cost is $\sum_m f_m(V_m^{max}) + \sum_{m,r} f_{m,r}(V_{m,r}^{max})$. Taking the constraints into consideration, we obtain the mathematical model of the overall traffic cost-minimization problem under the MAX contract charge model as:

$$min \sum_m f_m(V_m^{max}) + \sum_{m,r} f_{m,r}(V_{m,r}^{max}) \quad \textbf{(8)}$$

Subject to

$$\sum_{\Delta t=0}^{D_i} x_{t,d}^{m,i} = n_m, \forall t, i, m \quad \textbf{(8-1)}$$

$$\sum_m n_m = 1 \quad \textbf{(8-2)}$$

$$0 \le x_{t,d}^{m,i} \le 1, \forall m, i \in M \quad \textbf{(8-3)}$$

$$V_m^{max} - V_m^t \ge 0 \ \forall t \quad \textbf{(8-4)}$$

$$V_{m,r}^{max} - V_{m,r}^t \ge 0 \ \forall t \quad \textbf{(8-5)}$$

3.3 Single ISP Case

In the single ISP case, we assume that there is only one mapper and one reducer in the cloud, and all data centers employ a single ISP. In this case, the bandwidth cost minimization model can be simplified as:

$$min \ V \quad \textbf{(9)}$$

Subject to

$$\sum_{\Delta t=0}^{D_i} x_{t,d}^{m,i} = n_m, \forall t, i, m \quad \textbf{(9-1)}$$

$$\sum_m n_m = 1 \quad \textbf{(9-2)}$$

$$0 \le x_{t,d}^{m,i} \le 1, \forall m, i \in M \quad \textbf{(9-3)}$$

$$V_m^{max} - V_m^t \ge 0 \ \forall t \quad \textbf{(9-4)}$$

The workloads in this case are generated with multiple delay window sizes. Following the method in [12], we put the workloads with the same delay window size into a queue and try to smooth the peak volume of transmission in one charging cycle with respect to queues. We know from Theorem 4 in [12] that without considering whether V_i^{max} (which represents the peak volume for each queue i) is greater than V^{max}, V^{max} can be reduced to a smaller value of peak volume for the time slot τ if the principle proposed in [9] is used. Accordingly, we use Multi-Heuristic Smoothing Algorithm to cope with the multiple window delay sizes that every workload might have. The main idea of this Multi-Heuristic Smoothing Algorithm was inspired by the Simple Smoothing Algorithm in [18] and the Heuristic Smoothing Algorithm in [9]. The workload is divided into uniform parts to upload so that unreasonable peaks, such as scheduling all workloads to be transmitted in one or few time slots, can be avoided.

After observing the experimental results in [12], we can find out there is still space to reduce the peak of transmission. Let D represent the number of deferrable time slots in an entire charging cycle. If $V^{max} \times D$ is greater than the sum of all workloads generated in this charge cycle, it indicates that some capacity of transmission is wasted. For example, when the Multi-Heuristic Smoothing Algorithm is applied, there is just a few workloads need to be uploaded in the first time slot. Therefore, the maximum transmission of the first time slot is the lowest in the whole charging cycle. The following few time slots are generally lower than the latter slots, and the latter slots tend to have a greater value of transmission. This is because V^{max} is calculated based on both accumulation and the capacity of transmission. If the transmission capacity left can not guarantee to upload all the data on time, we have to increase V^{max} further.

Inspired by the analysis above, we try to enlarge the capacity of transmission of the first slot several times by simply multiplying a factor ψ, and design a new algorithm called the Zoom-Heuristic Smoothing Algorithm. The Zoom-Heuristic Smoothing Algorithm is shown below:

Algorithm 1. The Zoom-Heuristic Smoothing Algorithm (ZHSA)

```
1: $V^{max} = 0, V^{max,\lambda}, \lambda = 1, ..., i;$
2: $W_T^{\lambda} = 0, \forall \tau = T - D_{\lambda} + 1, ..., T; \lambda = 1, ..., i;$
3: $H_d^{\lambda} = 0, \forall d = 1, ..., D_{\lambda}; \lambda = 1, ..., i;$
4: for $\tau = 1$ to $T$ do
5:    for $\lambda = 1$ to $i$ do
6:       $V_{\tau}^{max,\lambda} = \frac{W_{\tau}^{\lambda} + \sum_{d=1}^{\tau} H_d^{\lambda}}{D_{\lambda}}$
7:    end for
8:    $V^{\tau} = Max\left\{ Max\{V_{\tau}^{max,\lambda}\} \frac{\sum_{\lambda} V_{\tau}^{max,\lambda} D_{\lambda}}{D_{max}} \right\}, \lambda = 1, ..., i$
9:    if $V^{max} < V^{\tau}$, then
10:       $V^{max} < V^{\tau}$
11:       if $\tau = 1$, then
12:          $V^{max} = V^{max} \times \psi$
13:       end if
14:    end if
15:    for $\lambda = 1$ to $i$ do
16:       Transfer the workload following Earliest Deadline
          First (EDF) strategy under the maximum volume
          $V^{max}$.
17:       update $H_d^{\lambda} = 0, \forall d = 1, ..., D_{\lambda}; \lambda = 1, ..., i;$
18:    end for
19: end for
```

Different from the smoothing strategy used in the former research, we now try to reduce the bandwidth cost by choosing a better initial value of V^{max}.

The Zoom-Heuristic Smoothing Algorithm shows that a larger initial value is able to improve the total transmission volume of an entire charging cycle. On the other hand, if the initial value is set too high, the bandwidth for transmission will eventually be wasted. Therefore, choosing a reasonable initial value is quite necessary.

According to different research purposes and assumptions, several models have been developed to describe the variation of network traffic. Since our focus in this paper is on minimizing the cost in the charging cycles of ISPs, we assume the network traffic varies corresponding to the charging cycle. The total transmission volume in a charging cycle is periodic. According to the Zoom Algorithm, the initial value is calculated as V^{max} multiplied by a factor. However, this algorithm does not take the variation of network traffic into account. Therefore, we take $V_{\tau}^{\lambda} \times \omega$ as the initial value for charging cycles, and fine-tune the factor ω based on the features of input workloads. According to this idea, a new algorithm is designed, which is called Fast Start Heuristic Smoothing Algorithm. The new algorithm is shown below:

Algorithm 2. The Fast Start Heuristic Smoothing Algorithm (FSHSA)

```
1: $V^{max} = 0, V^{max,\lambda}, \lambda = 1, ..., i;$
2: $W_T^{\lambda} = 0, \forall \tau = T - D_{\lambda} + 1, ..., T; \lambda = 1, ..., i;$
3: $H_d^{\lambda} = 0, \forall d = 1, ..., D_{\lambda}; \lambda = 1, ..., i;$
4: for $\tau = 1$ to $T$ do
5:    for $\lambda = 1$ to $i$ do
6:       $V_{\tau}^{max,\lambda} = \frac{W_{\tau}^{\lambda} + \sum_{d=1}^{\tau} H_d^{\lambda}}{D_{\lambda}}$
7:    end for
8:    $V^{\tau} = Max\left\{ Max\{V_{\tau}^{max,\lambda}\} \frac{\sum_{\lambda} V_{\tau}^{max,\lambda} D_{\lambda}}{D_{max}} \right\}, \lambda = 1, ..., i$
9:    if $V^{max} < V^{\tau}$, then
10:       $V^{max} < V^{\tau}$
11:       if $\tau = 1$, then
12:          $V^{max} = V_{\tau}^{\lambda} \times \omega$
13:       end if
14:    end if
15:    for $\lambda = 1$ to $i$ do
16:       Transfer the workload following Earliest Deadline
          First (EDF) strategy under the maximum volume
          $V^{max}$.
17:       update $H_d^{\lambda} = 0, \forall d = 1, ..., D_{\lambda}; \lambda = 1, ..., i;$
18:    end for
19: end for
```

3.4 Cloud Scenario

Now we extend the single ISP case to the cloud scenario.

In the real world, cloud computing platforms usually have multiple Mappers and multiple Reducers, with multiple different ISPs employed between these data centers. To simplify the discussion, we assume that all the ISPs between cloud users and Mappers use the same cost function f_m. Similarly, we assume that all the cost function between Mappers and Reducers are also identical. We further assume that $f_m = f_{m,r}$. Since the cost function is non-decreasing and convex, we choose the cost function as $f_m = f_{m,r} = x^2$.

After the charging function is fixed, we can further improve the algorithm in [12] by using a variation of the Cauchy inequality. The Cauchy inequality is shown below:

$$\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2 \geq (\sum_{i=1}^{n} a_i b_i)^2 \tag{10}$$

Let $b_i = 1$, then we have $(a_0^2 + a_1^2 + \cdots a_n^2) \geq$ $\geq \frac{(a_0 + a_1 + \cdots a_n)^2}{n}$. The condition of equality is $a_0 = a_1 = \cdots = a_n$.

Let a_i represent the final V^{max} of each data center. Then, when the V^{max} of these data centers are generally equal to each other, the total cost on bandwidth of data centers will be minimized.

Based on the analysis above, we use a two-step dynamic programming to optimize the transmission cost in the cloud scenario. First, we decide the workload assigned to each Mapper. Then, the method in the single ISP case is used, which eventually helps to improve the entire effect.

In order to decide the workload assignment in the first step, a function Φ(ϕ) is defined.

$$\Phi(V^{max}, D) = \frac{(V^{max,m} \times D - \sum H_d)}{(V^{max,m} \times D)} \tag{11}$$

According to the conclusion of Cauchy inequality, which claims that the total cost on bandwidth of data centers will be minimized when the V^{max} of these data centers are generally equal to each other, we use a greedy algorithm and assign workload to the data center whose current transmission capacity is relatively lower than others with higher priority. In the second step, we use the Fast Start Heuristic Smoothing Algorithm provided in the former section to optimize the peak transmission volumn for each data center.

The updated algorithm for cloud scenario is shown below:

Algorithm 3. The Updated Muti-Dynamic Self-Adaption Algorithm

```
1: Workload queue is generated: $E_m = n$
2: while( $E_m$ )
3:    $\chi = 1, \Delta_{max} = 0$
4:    for $M = 1$ to $m$
5:       $\Phi(V^{max,m}, D_{max}) = \frac{(V^{max,m} \times D_{max} - \sum\sum H_d^{\lambda})}{(V^{max,m} \times D_{max})}$
6:       if $\Delta_{max} < \Phi(V^{max,m}, D)$
7:          $\Delta_{max} = \Phi(V^{max,m}, D)$
8:          $\chi = M$
9:       end if
10:   end for
11:   For each ISP $\chi$, apply the single ISP algorithm, e.g.,
      FSHSA to schedule the traffic.
12:   update $E_m$
```

4 Performance Evaluation

We implement the experiment to simulate the peak trend. The simulation environment built in [12] is updated according to the new algorithms proposed in this paper. The default input is generated uniformly at random, as shown in Figure 2, where all data are normalized. We set the charging cycle to have 130 time slots and the max delay size to be 30 time slots.

First, we make a comparison between the Zoom-Heuristic Algorithm and the Multi-Heuristic Smoothing Algorithm. Figure 3 to Figure 6 show the trend of V^{max} when the number of deferrable upload queues equals to 3, 6, 9 and 15.

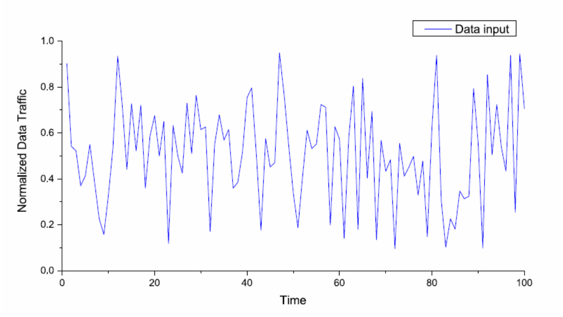

Figure 2. Data input of Gaussian distribution

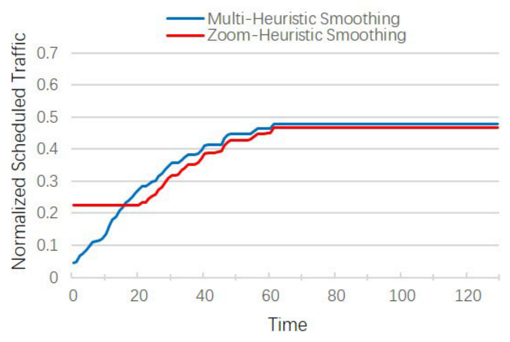

Figure 3. Zoom-Heuristic Smoothing vs. Multi-Heuristic Smoothing num=3

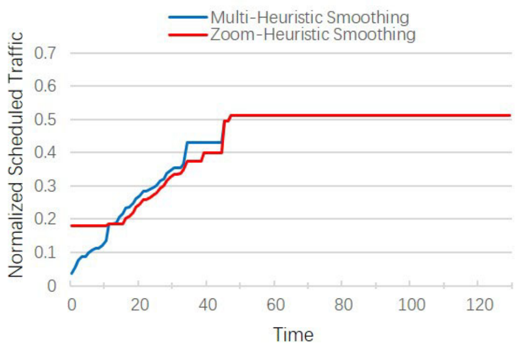

Figure 4. Zoom-Heuristic Smoothing vs. Multi-Heuristic Smoothing num=6

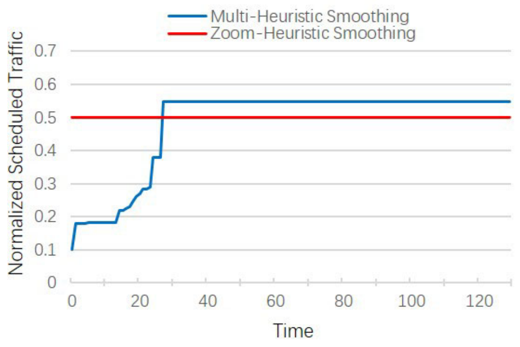

Figure 5. Zoom-Heuristic Smoothing vs. Multi-Heuristic Smoothing num=9

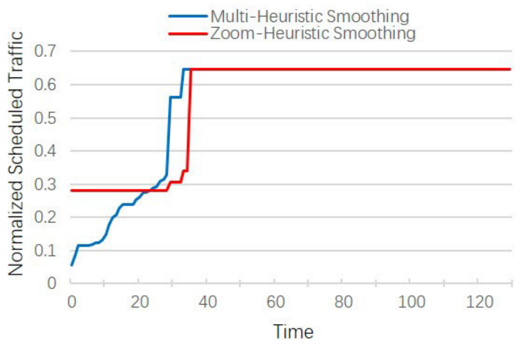

Figure 6. Zoom-Heuristic Smoothing vs. Multi-Heuristic Smoothing num=15

We find that the performance optimization achieved varies according to different numbers of queues. The best optimization result of 9.3% is achieved when the number of queues is 9. However, the optimized result is only 2.5% in the worst case. These results show that the initial value plays an important role during a charging cycle. As a user's uploading job proceeds, the peak volume of each queue has to be increased accordingly in order to guarantee that all the data uploading tasks can be finished in the allowable delay window sizes. The initial value for the difference queue is generally not the same, but in the Zoom-Heuristic Smoothing Algorithm, the initial value is positively related to the task volume.

For the Fast Start Heuristic Smoothing Algorithm, we implement the experiment where the number of window size queues is 3, 6, 9 and 15 in the single ISP case, and we compare these results with Multi-Heuristic Smoothing Algorithm. We carry out experiments and fine-tune the factor ω. The experiment shows that the best optimization result can be achieved when ω equals to 0.43. The comparison between Fast Start Heuristic Smoothing Algorithm and MHSA are shown in Figure 7 to Figure 10.

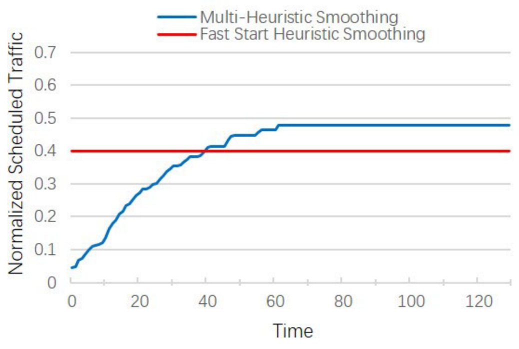

Figure 7. Fast Start Heuristic Smoothing vs. Multi-Heuristic Smoothing num=3

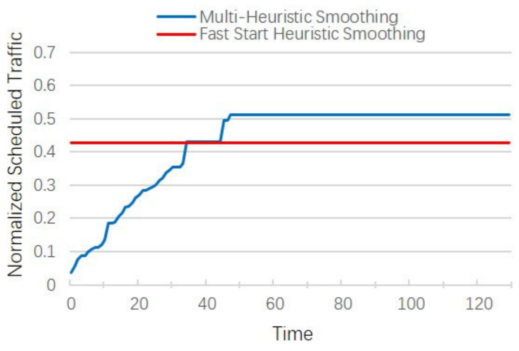

Figure 8. Fast Start Heuristic Smoothing vs. Multi-Heuristic Smoothing num=6

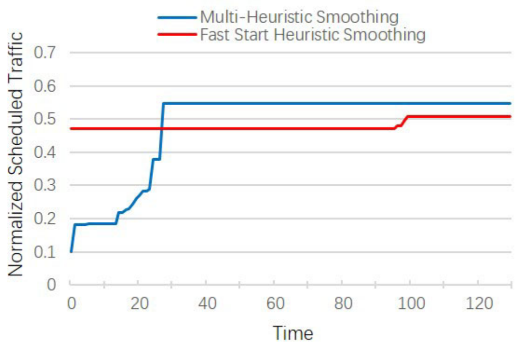

Figure 9. Fast Start Heuristic Smoothing vs. Multi-Heuristic Smoothing num=9

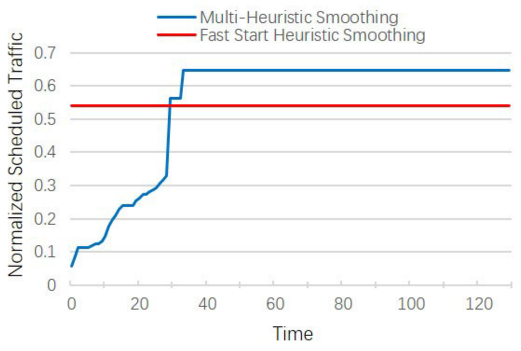

Figure 10. Fast Start Heuristic Smoothing vs Multi-Heuristic Smoothing num=15

The improvement of Fast Start Heuristic Smoothing Algorithm is obvious. Since it tends to choose a larger initial value, the bandwidth in the earlier time slots are used efficiently. The bandwidth saved in the early stage can reduce the maximum value needed to get the whole data uploading job done.

For the cloud scenario, we assume there are five data centers with the MapReduce framework applied, as shown in Figure 1. The Updated Multi-Dynamic Self-Adaption Algorithm is implemented. The charge function that we adopt is $f_m(x) = f_{m,r}(x) = x^2$, and we assume β = 1. The normalized scheduled traffic of five data centers is shown in Figure 11.

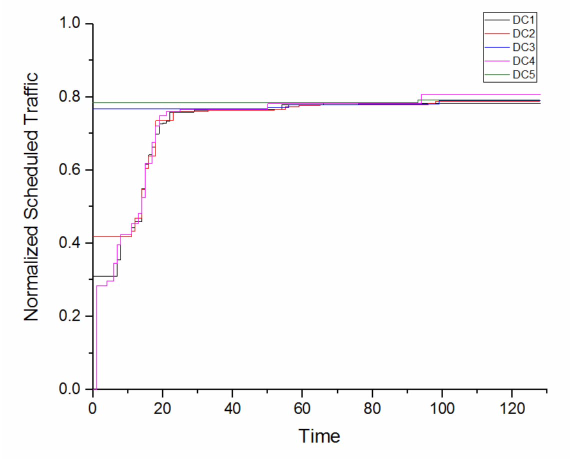

Figure 11. The normalized scheduled traffic of 5 data centers with Multi-ISPs using the Updated Muti-Dynamic Self-Adaption Algorithm.

As illustrated, after about 20 time slots, the traffic of the five data centers become almost identical, which means the total cost on bandwidth is minimized.

5 Conclusion

This paper focuses on minimizing the bandwidth cost of data centers in the cloud environment and tries to reduce the cost by leveraging the deferrable characteristic of big data. Based on the previous work, we further study the influence of initial maximum transmission capacity to the bandwidth cost of data centers. We propose the Zoom-Heuristic Smoothing Algorithm and the Fast Start Heuristic Smoothing Algorithm. Both algorithms try to increase the initial transmission capacity to leverage the space in the earlier time slots. The experiments show that the peak value can be reduced by choosing a larger initial value.

We also discuss how to assign workloads among data centers in the cloud scenario. We point out that the total bandwidth cost of data centers will be minimal when V^{max} of these data centers are generally equal to each other. Besides, we propose an Updated Multi-Dynamic Self-Adaption Algorithm for the cloud scenario and carry out validation experiments.

In order to do our work, we make many assumptions to the model. Some of these assumptions are too idealistic and inappropriate for real world scenarios. In future work, we will extend the model and continue to study how to reduce the bandwidth cost of data centers.

Acknowledgment

This work was supported by the Fundamental Research Funds for the Central Universities (2014ZD03-03).

References

[1] Amazon Cloud Front, http://aws.amazon.com/cn/cloudfront/.

[2] Google Datacenter, http://www.google.com/about/datacenters.

[3] Windows Azure, http://www.microsoft.com/windowsazure/.

[4] Alibaba Cloud Service, https://www.aliyun.com/.

[5] T. Hey, S. Tansley, K. Tolle, *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Microsoft Research, 2009.

[6] D. K. Goldenberg, L. Qiuy, H. Xie, Y. R. Yang, Y. Zhang, Optimizing Cost and Performance for Multihoming, *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Portland, OR, 2004, pp. 79-92.

[7] H. Wang, H. Xie, L. Qiu, A. Silberschatz, Y. R. Yang, Optimal ISP Subscription for Internet Multihoming: Algorithm Design and Implication Analysis, *IEEE INFOCOM 2005: 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, FL, 2005, pp. 2360-2371.

[8] A. Grothey, X. Yang, Top-percentile Traffic Routing Problem by Dynamic Programming, *Optimization and Engineering*, Vol. 12, No. 4, pp. 631-655, December, 2011.

[9] L. Zhang, Z. Li, C. Wu, M. Chen, Online Algorithms for Uploading Deferrable Big Data to the Cloud, *The 33rd IEEE Conference on Computer Communications*, Toronto, Canada, 2014, pp. 2022-2030.

[10] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, *The 6th Conference on Symposium on Operating Systems Design & Implementation-Volume 6*, San Francisco, CA, 2004, pp. 10-10.

[11] J. L. U. Gonzalez, S. P. T. Krishnan, *Building Your Next Big Thing with Google Cloud Platform*, Apress, 2015.

[12] B. Cui, P. Shi, W. Qi, M. Li, Uploading Multiply Deferrable Big Data to the Cloud Platform using Cost-effective Online Algorithms, *Future Generation Computer Systems*, Vol. 67, pp. 276-285, February, 2017.

[13] F. Yao, A. Demers, S. Shenker, A Scheduling Model for Reduced CPU Energy, *IEEE 36th Annual Foundations of Computer Science*, Milwaukee, WI, 1995, pp. 374-382.

[14] S. Albers, F. Muller, S. Schmelzer, Speed Scaling on Parallel Processors, *The 19th ACM Symposium on Parallelism in Algorithms & Architectures*, San Diego, CA, 2007, pp. 289-298.

[15] M. A. Adnan, R. Sugihara, Y. Ma, R. K. Gupta, Energy-optimized Dynamic Deferral of Workload for Capacity Provisioning in Data Centers, *2013 International Green Computing Conference*, Arlington, VA, 2013, pp. 1-10.

[16] B. Cho, I. Gupta, New Algorithms for Planning Bulk Transfer via Internet and Shipping Networks, *IEEE 30th International Conference on Distributed Computing Systems*, Genova, Italy, 2010, pp. 305-314.

[17] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, F. C. M. Lau, Moving Big Data to The Cloud: An Online Cost-Minimizing Approach, *IEEE Journal on Selected Areas in Communications*, Vol. 31, No. 12, pp. 2710-2721, December, 2013.

[18] L. Golubchik, S. Khuller, K. Mukherjee, Y. Yao, To Send or not to Send: Reducing the Cost of Data Transmission, *IEEE INFOCOM 2013: The 32nd IEEE International Conference on Computer Communications*, Turin, Italy, 2013, pp. 2472-2480.

[19] B. Cui, P. Shi, H. Jin, A Dynamic Self-Adaptive Algorithm for Uploading Deferrable Big Data to the Cloud Cost-Effectively, *The 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Blumenau, Brazil, 2015, pp. 292-295.

[20] B. Cui, P. Shi, J. Yang, Y. Hao, Uploading Deferrable Big Data to the Cloud by Improved Dynamic Self-Adaption Algorithm, *The 10th International Conference on P2p, Parallel, Grid, Cloud and Internet Computing*, Krakow, Poland, 2015, pp. 116-120.

Biographies

Baojiang Cui received B.S. in Hebei University of Technology, China, in 1994, M.S. in Harbin Institute of Technology, China, in 1998 and Ph.D. in Control Theory and Control Engineering in Naikai University, China in 2014. He is a Professor in the School of Computer Science at Beijing University of Posts and Telecommunications, China. His main research areas are detection of software, cloud computing and big data.

Xiaohui Jin is a Ph.D. candidate in the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research mainly focuses on machine learning, cloud computing and big data security.

Peilin Shi now is studying for a master’s degree in Computer technology at the Beijing University of Posts and Telecommunications in China. His work mainly involves cloud computing and big data.