WYT: Optimized Consistency for Geo-Diverse Online Social Networks

Kévin Huguenin, Ionut Trestian, Vijay Erramilli, Nikolaos Laoutaris

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

WYT: Optimized Consistency for Geo-Diverse OSNs

Vijay Erramilli — Kévin Huguenin — Nikolaos Laoutaris — Ionut Trestian

N° 7295

July 2010

Distributed Systems and Services

ISSN 0249-6399 ISRN INRIA/RR--7295--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

WYT: Optimized Consistency for Geo-Diverse OSNs

Vijay Erramilli* , Kévin Huguenin , Nikolaos Laoutaris* , Ionut Trestian†

Theme : Distributed Systems and Services
Networks, Systems and Services, Distributed Computing
Équipe-Projet ASAP

Rapport de recherche n° 7295 — July 2010 — 12 pages

Abstract: Large scale Online Social Networks (OSNs) like Facebook and Twitter are hosted out of multiple geo-diverse data centers to provide low latency and high fault tolerance. Such geo-diversity creates large amounts of WAN traffic for maintaining the consistency of replicas at different locations. Despite the dropping price of WAN bandwidth, the growth rate of OSNs combined with the incorporation of media rich long tail content (including images and videos) makes WAN traffic costs an increasing concern for OSN operators.

At the heart of the problem lies a tradeoff between consistency and WAN bandwidth cost. In this paper, we propose the *"Wait Your Turn"; WYT* system that optimizes the tradeoff by leveraging: (*i*) knowledge of mapping between social relationships and geographic location, and (*ii*) knowledge of timing regularities in end user activity patterns. We quantify the benefits of such an OSN-aware update propagation strategy through a trace-driven analysis and show that it reduces WAN traffic by 55% compared to an immediate update of all replicas, while having minimal impact on consistency. Furthermore, for a given budget for WAN bandwidth, WYT increases consistency by several orders of magnitude compared to FIFO scheduling of updates.

Key-words: Data centers, Social Networks, Replication

* Telefonica Investigación y Desarrollo (Telefonica I+D), Barcelona
† Department of Electrical Engineering and Computer Science (EECS), Northwestern University

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

WYT: Optimisation de la cohérence dans les réseaux sociaux géograpiquement distribués

Résumé :

Mots-clés :

1 Introduction

Online Social Network (OSN) applications have become highly popular over the last few years, having large user bases, distributed across the entire planet [22, 21]. It is generally accepted that latency to the end-user is the key performance metric for most web applications including OSNs [11], therefore, in order to handle a geographically diverse user base, relevant data should be located as close as possible to the end-user, as this reduces the user perceived latency [7].

In light of this, OSNs have opened data centers that host content[1] close to their users. Facebook, for instance, is rumored to be hosting content in at least three geographically diverse locations (West coast US, East coast US and Europe) [20, 5]. Users are assigned to the geographically closest data center and their interactions with the OSN (*i.e.*, reading their friends' profiles and updating their own profile) are performed through this data center.

The performance gains, as well as the high availability, come at the cost of maintaining replica consistency across data centers. Geo-diverse data centers normally communicate by buying transit bandwidth from Telcos, whereas in rare cases they own direct links (*e.g.*, Google). The updates performed by users at a data center are replicated to other data centers. This creates large amounts of expensive and potentially useless (as some content may not be accessed from certain replicas) WAN traffic. Despite the dropping price of leased WAN bandwidth and networking equipment, the growth rate of OSNs combined with the incorporation of media rich long tail content (*e.g.* images and videos) makes WAN traffic costs a big concern for OSN operators [6]. It is worth noticing here that Facebook hosts more images than all other popular photo hosting websites such as Flickr and PhotoBucket [19], whereas they have recently started hosting videos as well [1].

The peak-based pricing policy used on WAN links raises an interesting trade-off between consistency and bandwidth cost. Specifically, the problem of finding a balance turns out to be a scheduling problem. Instantaneously transmitting content uploaded by users to replicas provides the best achievable consistency, however, as users' activity traditionally follows a diurnal pattern with a peak at the end of the day, the bandwidth costs can be high for the total volume of data sent. Delaying updates to replicas reduces the traffic peaks but comes at the price of inconsistency that degrades user experience. Note that inconsistency can be experienced and communicated to the end user in multiple forms including for example *"new content uploaded but not available yet"*, *"low-resolution version available, check later for full version"*, *etc.*

In this paper, we propose the *"Wait Your Turn" (or WYT)* system that optimizes the aforementioned trade-off by scheduling the transmission of updates, leveraging: (*i*) knowledge of the mapping between social relationships and geographic location (friends are generally geographically close), and (*ii*) knowledge of timing regularities in the activity pattern of end users (user activity follows diurnal patterns shifted in time depending on the timezone of the data center). For example, WYT prioritizes the update of replicas that serve many of a user's friends, especially if these friends have a high probability of accessing the content. To achieve high consistency at low cost, WYT implements three optimizations: selective replication that reduces the replication traffic volume, social-aware scheduling that prioritizes content most likely to be accessed, and budget allocation that dynamically adjusts the bandwidth spent on updating each data center. Any replication-based system needs to deal with consistency issues as user experience can deteriorate if replicas are not updated on time. For the purposes of this paper, we define 'consistency' not in terms of the traditional *ACID* framework, but in terms of the 'staleness' of information that any user has access to.

We compare WYT against de facto policies used by most geo-diverse OSNs such as full replication everywhere and immediate FIFO transmission of updates at creation time. We quantify the benefits by a trace-driven analysis using a 8M-user Twitter dataset. Our extensive simulations show that WYT reduces WAN traffic by 55% without sacrificing consistency. Also, for a given budget for WAN bandwidth, our solution increases consistency by an order of magnitude compared to FIFO update scheduling.

[1] We are referring to actual content like images and videos and not to metadata associated with maintaining the social graph structure [13]

The rest of the paper is organized as follows. Section 2 introduces the system model. Section 3 describes the dataset. Section 4 states and motivates the problem. A formulation is given in Section 5. Our solution, WYT, is presented in Section 6 and our trace-driven evaluation of WYT in Section 7. Section 8 reviews related work and Section 9 concludes the paper.

2 Model of a geo-diverse OSN

Throughout the rest we assume the following model of operation for an OSN. Users of an OSN application are geographically distributed around the world. All users are assigned to data centers, and for each user, her data is first assigned to the geographically nearest data center, in order to provide high QoS. This is referred to as the master copy of the data for that user. Both reads and writes for a given user are handled by her master copy. Our underlying system model operates under full replication, user data is eventually replicated to all locations. However, the ideas can be extended to the partial replication case. Therefore, data pertaining to a user is replicated in other locations. For handling a read (for the data of the users he is friends with), the system would pull data from the user's master copy and poll the local copies of her friends on her master copy.

3 OSN dataset

To showcase our proposal we will use an existing dataset of 41.7M users with 1.47B edges [8] obtained through a massive crawl of Twitter between June - Sept. 2009. For the purpose of this work we need location information, as well as write patterns for users. We collected the location information by conducting our own crawl, using the user-ids given in the social graph described in [8]. Because the Twitter API rate limits the number of calls to a few hundred each hour [25], collecting write patterns for such a large number of users can take an impractical amount of time. As such, we proceed in the following way. Figure 1 shows the normalized number of social networking (Facebook, Twitter, MySpace) sessions active during a certain hour of the day. The data was collected from one of the access links of a large ISP [14]. We generate Twitter write patterns by assuming that for each time bin, a random number of users can generate a number of writes proportional to the value in Figure 1 (a single user can generate several updates). Twitter does not provide information pertaining to read patterns.

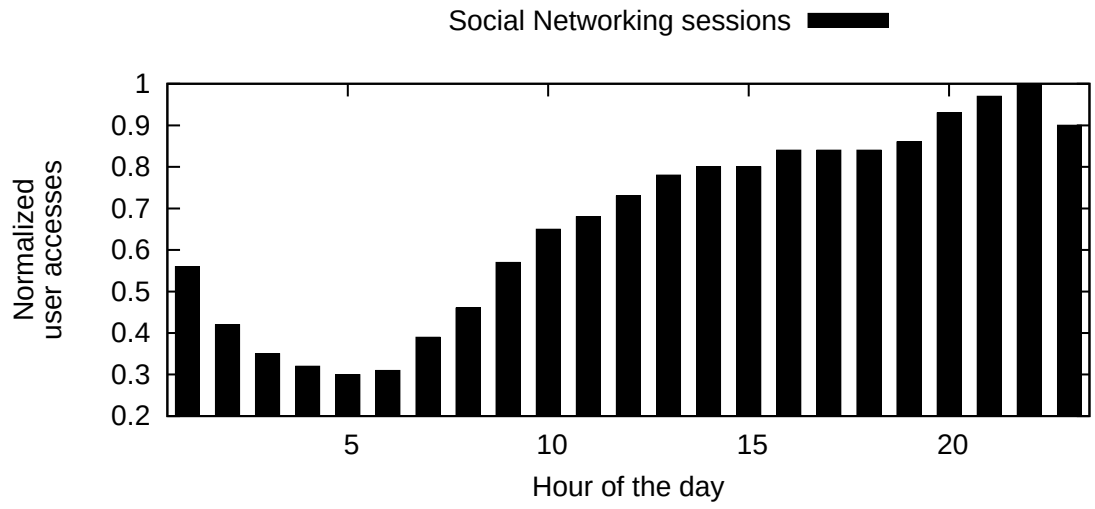

Figure 1: Social network user activity patterns function of time of day.

3.1 Extracting Location Information

Here we show how we enhanced our OSN dataset to include user location information. We use the fact that many Twitter users disclose their location in their respective profiles, in freetext. We first extract latitude/longitudes where available. Users who use Twitter extensively from their mobile devices provide this information. We then extract zipcodes, towns and cities and translate them to latitude/longitude, using Google Maps API. Some users (around 34%) disclose only their states or countries. For such cases, we assign them a latitude and longitude that falls roughly at the center of the state or country. In cases where there is some ambiguity, where there could be towns with the same name in different parts of the world; for e.g. there is a 'Rome' in Wisconsin, we assign the latitude/longitude of the most populous town, in this case Rome, Italy. We filter out junk information, where found. In the end, we extracted location for 8,092,624 users from about 11M users that have actually entered location information. We use this social graph, nodes and edges only between these nodes for our analysis in this paper. With regards to the location of the users in the dataset, we find that US has the maximum number of users (55.7%), followed by UK (7.02%) and Canada (3.9%). In terms of cities, New York has the highest number of users (2.9%), followed by London (1.7%) and LA (1.47%).

3.2 Clustering Users

Next we will show how we assigned users to the closest data center for QoS purposes. It is well known that contacts or 'friends' in social networks are located close together with respect to geographical distance [12, 10]. This fact can have large scale repercussions for assigning users to data centers in an OSN. In order to understand the distribution of users and their contacts around the world, we proceed as follows. We consider 7 locations around the world (in precisely this order): Boston, London, LA, Chicago, Tokyo and Delhi, where we can place data centers[2]. We assign users to locations using a simple greedy heuristic: compute the distance of a user to a location, and assign the user to the nearest location. For computing the distances we use the Haversine distance [15] as it takes Earth's sphericity into account. The number of data centers is a free variable but the locations we chose are fixed as we are not dealing with a data center placement problem here. The results are shown in Table 1. Because of the high number of users that are located in the US, several of the data centers will cover this region and this also explains the imbalance observed in the number of users for the 2 data centers case. One can note that for above 3 data centers, the number of users is relatively balanced across them.

City	2 Data Centers	3 Data Centers	4 DataCenters	5 Data Centers	6 Data Centers	7-Data Centers
Boston	5,608,894	3,476,676	2,187,867	1,318,388	1,318,388	1,318,388
London	2,483,730	2,274,409	2,274,406	2,274,374	1,684,098	1,492,839
Los Angeles		2,341,539	1,569,944	1,563,705	1,267,445	1,267,445
Houston			2,060,407	1,454,755	1,454,755	1,454,455
Chicago				1,481,402	1,481,366	1,481,366
Tokyo					886,572	533,166
Delhi						544,965

Table 1: Users per given data center for different layouts

4 The case for OSN-aware update propagation

In this section we will use the above mentioned dataset to motivate why an efficient propagation of updates has to be OSN-aware. Figure 2 shows the relationship between geographical proximity and the number of followers. It shows the average number of followers assigned to the same location as the user, and the average number of followers assigned to all other locations, and the maximum number of followers assigned to any other single location. We see that on average, the number of followers assigned to the same location is more than the maximum number of followers

[2]Note that data center operators such as Equinix [18] already have data centers in several of these locations

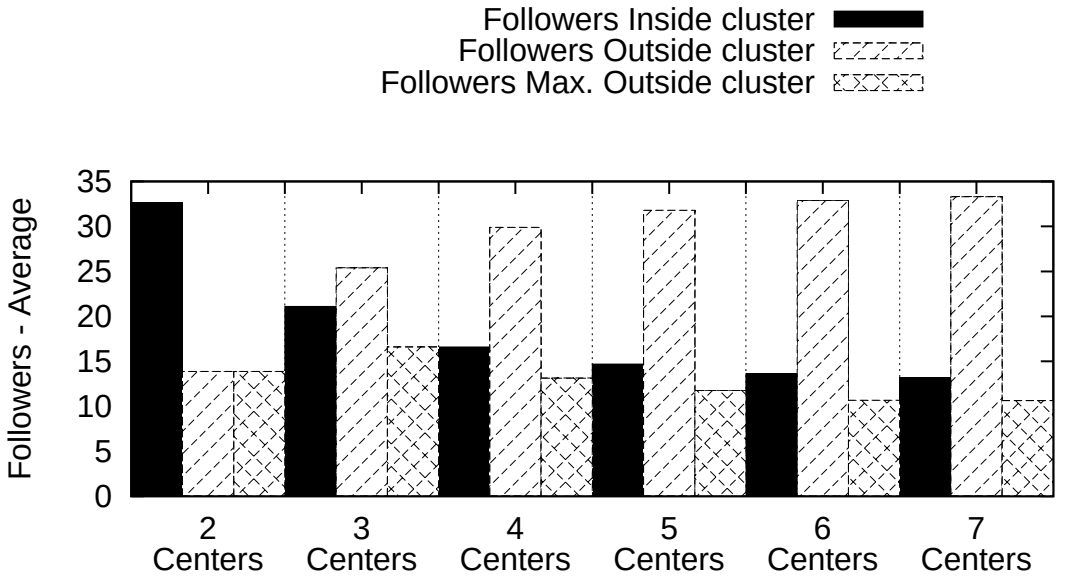

Figure 2: Average number of followers inside and outside user cluster.

assigned to any other single location which suggests that users are assigned to locations where they have the most followers.

Following from the above and in order to get an idea of the distribution of followers across different data center locations, we compute the mean number of locations a user has followers in. For the 2-7 locations we consider, we obtain values from 1.2 to 2.6 as shown in Figure 3. In other words, a user has followers in 60% of the locations on average in the case with 2 locations and has followers in 37% of the locations on average when we consider all 7 locations. This further suggests that instead of copying content in 7 locations the provider only needs to copy content in 2.6 locations. This result strongly suggests exploiting the notion of *selective replication*, where users are replicated in locations where their followers are present, rather than full replication.

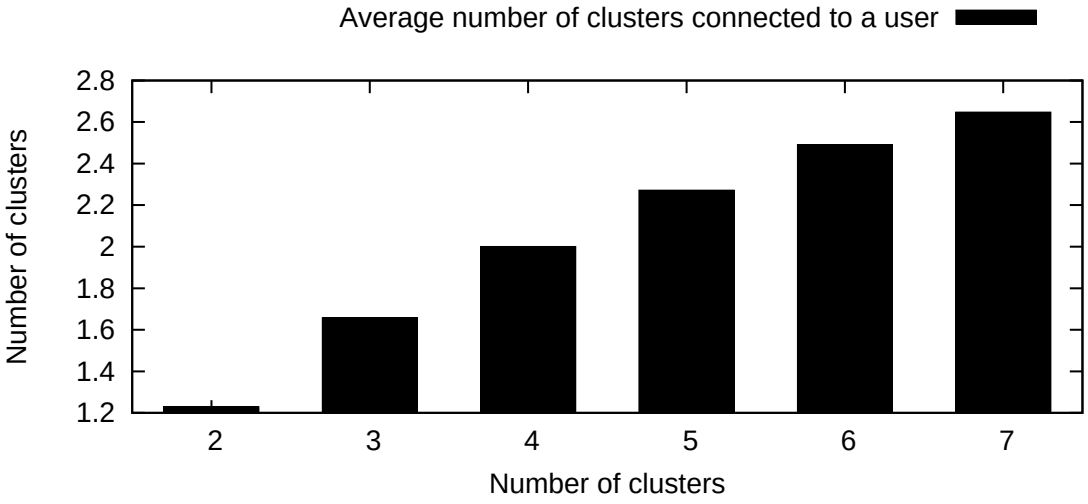

Figure 3: Average clusters connected to a user.

As noted in Figure 1, user social network activity manifests a strong diurnal pattern. Such diurnal patterns, combined with the 95_{th} percentile pricing scheme employed by ISPs have inspired recent work [9] that aims at using the free bandwidth available during off-peak hours. In our setting, reducing the peak usage by delaying updates and sending them during the off-peak hours in an effort to reduce WAN bandwidth costs for the provider reduces to a *scheduling* problem. Consider the following example: a US social network user that has followers both in Europe and Japan uploads content to the social network at 2PM local time. WYT would replicate the content to the Europe data center before the Japan data center, as the local time in Europe is 8PM compared to 4AM in Japan.

5 Problem formulation

In this section, we formulate the problem of scheduling transmission of *fixed size* updates. We note that our formulation is generic, and does not rely on any specific placement of replicas, only that a placement exists.

Let $\mathcal{U} = \{u_1, \dots u_N\}$ be the set of users and $\mathcal{S} = \{S_1, \dots S_M\}$ the set of replicas, distributed at different locations around the world. Each user is assigned to a master data center and the user is replicated on different data centers, following a placement strategy. This can be full replication (replicate all users on all data centers) or selective replication (replicate users only on a subset of data centers). We denote by $n \in S$ the fact that S is the master data center of user u_n. All writes are carried out on the author's master data center and updates are sent to other data centers where the author has data.

In order to incorporate the notion of consistency, we use a mechanism based on assigning penalties, where a penalty is assigned for every update not yet transmitted to the respective replicas every time the profile of the associated user is read through the replica. This captures the notion of 'staleness' of data, and hence captures the user-perceived quality of service. We use the number of followers a user has on each data center; $f_{n,m}$. We define $read_m[i]$ to be the probability of a read event happening on data center m at time interval i. We assume that this probability is independent from the couple author/reader. Then $p^{[i]}_{n,m} = \text{read}_m[i] \cdot f_{n,m}$. The parameter $\text{read}_m[i]$ can be obtained from ISP trace-driven analysis (see Figure 1).

For capturing bandwidth costs, we use the notion of a budget that will be used to purchase bandwidth on links to minimize the total penalty. Hence the problem is simply, minimize total penalty, with a fixed budget and is captured below. The notation we use for our formulation is given in Table 2.

Decision variables: replication traffic volume time series $R_{n,m} = \{r^{[i]}_{n,m} \mid 1 \leq i \leq I\}$.

Objective: minimize penalty

$$\min_{R_{n,m}} \left(\sum_i \sum_n \sum_{\{m \mid n \notin S_m\}} ({t'_n}^{[i]} - {r'_{n,m}}^{[i]}) \cdot p^{[i]}_{n,m} \right)$$

where $({t'_n}^{[i]} - {r'_{n,m}}^{[i]})$ represents the number of updates of user u_n that haven't been replicated on data center m yet.

Constraints:

$$\begin{aligned} {r'_{n,m}}^{[i]} &\leq {t'_n}^{[i]} \qquad \forall i, n, m \\ \sum_m c_m(v_m) &\leq c \end{aligned}$$

The first constraint is for causality. The second constraint is for the bandwidth costs, any allocation should not exceed a fixed budget. A dual problem would be to minimize the budget, (hence bringing down the costs) for a fixed level of penalty.

In the sequel, we propose a solution to this problem for updates of arbitrary sizes, a charging volume equal to the peak value (i.e., $v_m = \max\left(\max_i v_m^{\downarrow[i]}, \max_i v_m^{\uparrow[i]}\right)$) and a linear cost function c_m which slopes depend on the city where the data center is located. We leave the consideration of optimizing on the 5% of unbounded bandwidth time intervals when using 95th percentile pricing to future work.

I	Number of time intervals in a charging period, $i \in I$
$t_n^{[i]}$	Update traffic volume of user n during interval i. Note $t'^{[i]}_{n,m} = \sum_{i'=1}^{i} t_{n,m}^{[i']}$
$r_{n,m}^{[i]}$	Replication traffic volume of user n to data center m during interval i. Note $r'^{[i]}_{n,m} = \sum_{i'=1}^{i} r_{n,m}^{[i']}$
$p_{n,m}^{[i]}$	Penalty for inconsistency for user n per update that has not been replicated on data center m at instant i; $P_{n,m} = \{p_{n,m}^{[i]} \mid 1 \leq i \leq I\}$.
$v_m^{\downarrow[i]}$	Incoming traffic volume of data center m, i.e., $v_m^{\downarrow[i]} = \sum_{n \notin S_m} r_{n,m}^{[i]}$
$v_m^{\uparrow[i]}$	Outgoing traffic volume of data center m, i.e., $v_m^{\downarrow[i]} = \sum_{n \in S_m} \sum_{m' \neq m} r_{n,m}^{[i]}$
v_m	Charging volume of data center m, i.e., $v_m = v\left(V_m^{\downarrow}, V_m^{\uparrow}\right)$.
c_m	The cost function at data center m
c	Total budget dedicated to bandwidth cost

Table 2: Notation for our formulation

6 WYT system design

In the previous sections, we described the system model we operate under and the problem we want to address; minimizing the perceived inconsistency under cost constraints. We now present the *"Wait Your Turn"; WYT*, a social-aware scheduling system for OSNs. WYT operates in two steps. First it solves the problem of scheduling updates under a given budget allocation among data centers. Second, it uses a greedy heuristic to optimize the budget allocation. We evaluate WYT over a time period of two days and compare its performance to the solution used in practice assuming a system where updates are photos and videos. The sizes of contents are chosen according to the distributions observed in popular video and photo hosting systems such as Fickr and YouTube. The bandwidth prices for the cities where data centers are located have been obtained from a large ISP and are summarized in Table 3.

City	**DS-3 (155Mbps)**
London	$30
Chicago	$29
Houston	$30
Los Angeles	$29
Boston	$27
Tokyo	$80
Delhi	$370

Table 3: Bandwidth rates (2007)

6.1 Scheduling

As mentioned in the previous section, perceived inconsistency is caused by the reads that occur before the transmission of updates. Therefore, a solution that minimizes perceived inconsistency (captured by the notion of penalty introduced in the previous section) is to prioritize updates that are more likely to be imminently read and thus may incur penalty. WYT assigns a priority to each update based on the penalty and the size of the update: $\text{pr} = p_{n,m}^{[i]}/s = \text{read}_m[i] \cdot f_{n,m}/s$ for an update of size s to be replicated on server S_m where the author u_n has $f_{n,m}$ followers. The priority

is dynamically recomputed as the probability of being read on a remote data center depends on the time of the day. In WYT, updates are replicated by decreasing priority. If an update cannot be replicated because the uplink of the master data center or the downlink of the replica has reached the maximum budget assigned, the next update in the priority queue is examined and so on. The priority queue can be maintained by one specific data center that organizes the replication between all data centers. A decentralized solution is to make each data center upload its local updates by decreasing priority. Similarly, each data center downloads remote updates originating from the other data centers by decreasing priority. Note that due to the priority function used in WYT, selective replication is performed transparently: if a user has no followers attached to a given data center, the priority of her updates to be replicated to this data center falls to 0 and thus the updates are not replicated. In the case of 3 servers, our analysis shows that selective replication reduces the total replication traffic volume by 55%.

6.2 Budget allocation

Since the number of users varies from one cluster to another, assigning evenly the budget between data centers is sub-optimal. WYT optimizes the budget allocation among data centers using a greedy algorithm. The budget allocation algorithm is run on a small learning period (say a day) and recomputed periodically (say every week) to self-adapt to potential changes of user upload behaviors, new users joining, etc. Initially, all data centers are assigned a budget of 0. At each step of the iteration we compute the total penalty when a fixed extra budget is allocated to the first data center, to the second and so on. For instance at the first step of the iteration, we run the scheduling algorithm for the following budgets allocations (for 3 clusters): (1000, 0, 0), (0,1000,0) and (0,0,1000). The allocation is set to the one that brings the minimum penalty and a new step is performed. For instance is the allocation (1000, 0, 0) gave the minimum penalty, the next step tries the following allocations: (2000, 0, 0), (1000,1000,0) and (1000,0,1000) and so on until the whole budget is allocated.

7 Evaluation

We compare WYT to a FIFO scheduling algorithm where updates are replicated by decreasing age. For the sake of fairness, we complement the FIFO algorithm with selective replication: updates are not replicated where the author has no followers. In Figure 4, we plot the penalty as a function of the total budget distributed among the data centers using greedy allocation: each data center receives the same bandwidth budget. We vary the total budget from the budget needed to replicate all the update traffic over one time period and the budget needed to replicate all updates instantaneously. It can be seen that WYT outperforms FIFO by several orders of magnitude.

8 Related Work

The main contribution of this paper is a system to maintain consistency of data items in an OSN that are geo-replicated across data centers, at a minimum cost. In this section we compare and contrast our work with related work.

A lot of work has been done on WAN optimizers [3, 4], including commercial products [17, 24, 23] starting with the work of Spring et al [16]. These papers focus on removing duplicate content or on compression schemes to reduce network traffic, bringing down WAN bandwidth costs in enterprise networks [3] as well as inter-data centers [23, 17]. The solution presented in the paper can be thought of as a complementary approach as we focus on *scheduling* of content/updates to minimize bandwidth costs.

Reducing latency for the end-user at a global scale has been a fertile area of research. CDNs and smart placement of replicas to reduce latency and bandwidth costs have been successfully used. Volley [2] for instance dynamically moves data between different locations in the world with the objective of reducing latency for the end-users. However, they do not consider the existence

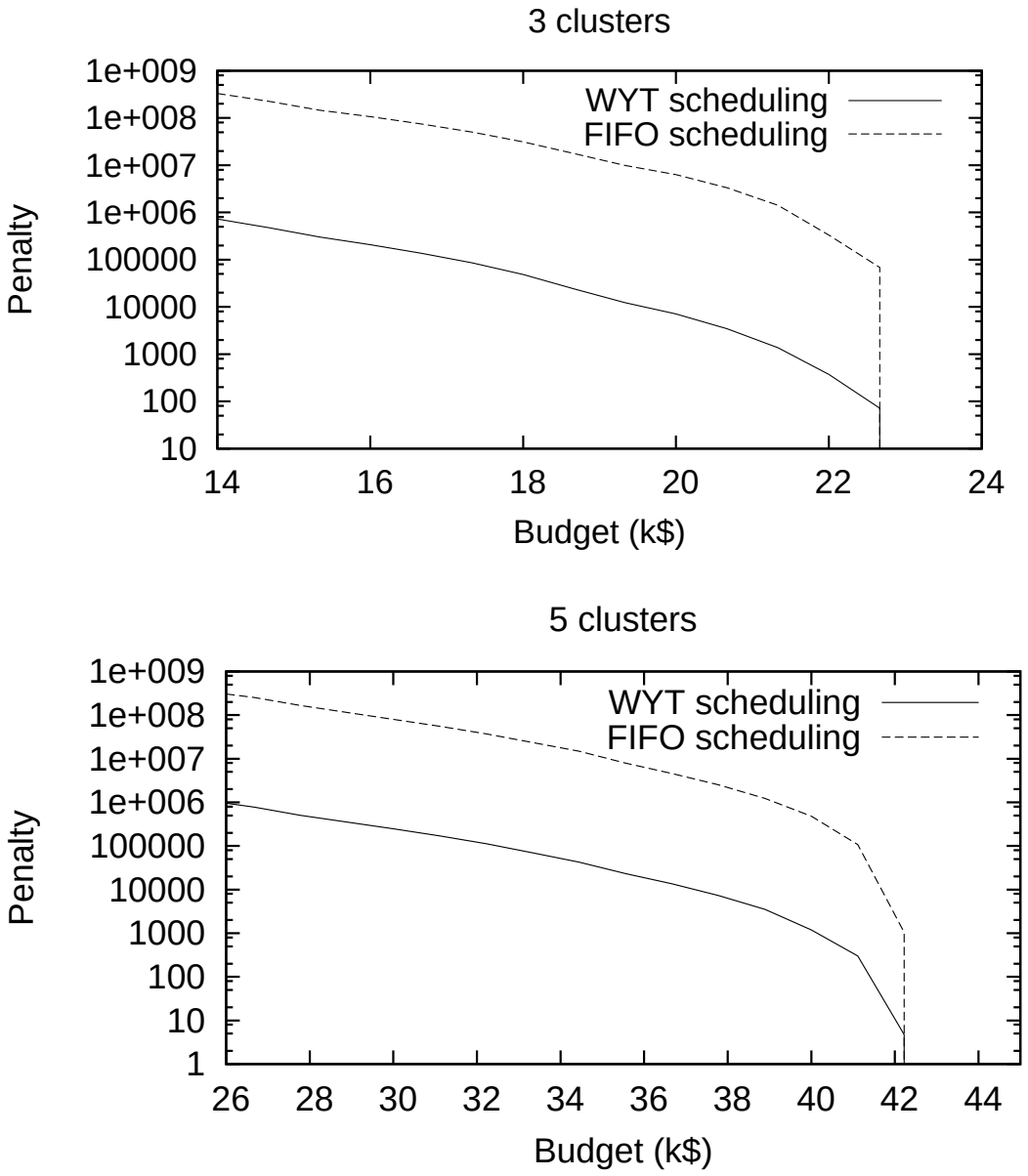

Figure 4: Penalty function of the budget.

of replicas, hence the question of maintaining consistency does not arise. Our problem is different - we assume the placement of replicas to be given, and we need to keep these replicas up-to-date at the lowest cost.

Another recent work of interest is the work by Laoutaris et al [9] who propose to exploit already paid-off bandwidth (specifically, the valleys in bandwidth demand patterns) to send bulk data for delay-tolerant applications, thereby decreasing bandwidth costs. The approach we propose is subtly different – we exploit different activity patterns and intentionally *delay* some updates (by assigning them lower priority) to decrease bandwidth costs.

9 Conclusions

In this paper we have presented a novel system (WYT) that addresses the increasing WAN bandwidth costs faced by OSN providers. The key idea is (under a certain budget) to schedule content replication based on the probability of access on the remote server. We have developed and evaluated scheduling algorithms that take into account social characteristics such as the number of followers or probability of access on different data centers.

Our findings are as follows: (*i*) WYT manages to reduce the total WAN traffic by 55% by not sending updates to data centers where they are not accessed (selective replication), (*ii*) under the same monetary budget WYT improves consistency by several orders of magnitude over FIFO scheduling of updates. Given that WYT operates in a distributed manner, it shows great promise for adoption in an Online Social Network system.

References

[1] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, and A. Wolman. Volley: Automated Data Placement for Geo-Distributed Cloud Services. In *NSDI*, 2010.

[2] B. Aggarwal, A. Akella, A. Anand, P. Chitnis, C. Muthukrishnan, A. Nair, R. Ramjee, and G. Varghese. EndRE: An End-System Redundancy Elimination Service for Enterprises. In *NSDI*, 2010.

[3] A. Anand, V. Sekar, and A. Akella. SmartRE: An Architecture For Coordinated Network-Wide Redundancy Elimination. In *SIGCOMM*, 2009.

[4] J. Hamilton. Geo-replication at facebook. http://perspectives.mvdirona.com/2008/08/21/GeoReplicationAtFacebook.aspx.

[5] J. Hamilton. Inter-datacenter replication and geo-redundancy. http://perspectives.mvdirona.com/2010/05/10/ InterDatacenterReplicationGeoRedundancy.aspx.

[6] C. Huang, A. Wang, J. Li, and K. W. Ross. Measuring and Evaluating Large-Scale CDNs. In *IMC*, 2008.

[7] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a Social Network or a News Media? In *WWW*, 2010.

[8] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram. Delay Tolerant Bulk Data Transfers on the Internet. In *SIGMETRICS*, 2009.

[9] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic Routing in Social Networks. *Proceedings of the National Academy of Sciences*, 102:11623–11628, 2005.

[10] G. Linden. Marissa mayer at web 2.0. http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html.

[11] J. M. Pujol, V. Erramilli, and P. Rodriguez. Divide and Conquer: Partitioning Online Social Networks. http://arxiv.org/abs/0905.4918, 2009.

[12] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutris, P. Chhabra, and P. Rodriguez. The Little Engines that Could: Scaling Online Social Networks. In *SIGCOMM*, 2010.

[13] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger. Understanding Online Social Network Usage from a Network Perspective. In *IMC*, 2009.

[14] R. W. Sinnott. Virtues of the Haversine. *Sky and Telescope*, 68:159, 1984.

[15] N. T. Spring and D. Wetherall. A Protocol-Independent Technique for Eliminating Redundant Network Traffic. In *SIGCOMM*, 2000.

[16] Cisco wide area application acceleration services. http://www.cisco.com/en/US/products/ps5680 /Products_Sub_Category_Home.html.

[17] Equinix. http://www.equinix.com/.

[18] Facebook Hosts more Photos than Flickr and Photobucket. http://www.tothepc.com/archives/facebook-hosts-more-photos-than-flickr-photobucket/.

[19] Facebook: 20$ million a year on datacenters. http://www.datacenterknowledge.com/archives/2009/05/18/ facebook-20-million-a-year-on-data-centers/.

[20] Facebook user statistics. http://www.facebook.com/press/info.php?statistics.

[21] Growing around the world. http://blog.twitter.com/2010/04/growing-around-world.html.

[22] Infineta Systems: High Speed Replication. http://www.infineta.com/solutions/.

[23] Riverbed Networks: WAN Optimization. http://www.riverbed.com/products/compare/compression.php.

[24] Twitter: Rate limiting. http://apiwiki.twitter.com/Rate-limiting.

[25] Facebook Hits One Billion Video Views. http://news.cnet.com/8301-17939_109-10294085-2.html.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399