Supplemental File: Social-Aware Replication in Geo-Diverse Online Systems

Stefano Traverso, Kévin Huguenin, *Member, IEEE,* Ionut Trestian, *Student Member, IEEE,* Vijay Erramilli, Nikolaos Laoutaris, and Konstantina Papagiannaki

1 Budget Allocation

Algorithm 1 Budget allocation across K PoPs

Input: B ▷ Total budget
Output: $\mathbf{b}$ ▷ Budget allocation

$b \leftarrow 0$ ▷ Total budget allocated
$\mathbf{b} \leftarrow (0, \ldots, 0)$ ▷ Budget allocation
while $b < B$ **do**
  $p_{\min} \leftarrow \infty$ ▷ Minimum penalty
  $k_{\min} \leftarrow \perp$
  for $k \in \{1, \ldots, K\}$ **do**
    $\mathbf{b}_k \leftarrow \mathbf{b}_k + \delta_b$ ▷ Allocates a budget increment to the k-th PoP
    $p \leftarrow \text{penalty}(\mathbf{b})$ ▷ Computes the penalty for a given budget allocation
    $\mathbf{b}_k \leftarrow \mathbf{b}_k - \delta_b$
    if $p < p_{\min}$ **then**
      $p_{\min} \leftarrow p$ ▷ Update minimum penalty
      $k_{\min} \leftarrow k$
    end if
  end for
  $\mathbf{b}_{k_{\min}} \leftarrow \mathbf{b}_{k_{\min}} + \delta_b$ ▷ Allocate the budget to the PoP s.t. the penalty is minimized
  $b \leftarrow b + \delta_b$
end while

Where B is the total budget to be allocated to the PoPs, $\mathbf{b}$ is the budget allocation (i.e., a vector of real values where the k-th element of $\mathbf{b}$ is the budget allocated to the k-th PoP), and δ_b is the budget increment for the iterative algorithm (i.e., the budget increment to be allocated at each iteration of the allocation algorithm). The variable b is the total budget allocated so far, and $p_{\min}$ denotes the minimum penalty obtained, at a given iteration, for all the possible allocations of the budget increment δ_b (and $k_{\min}$ denotes the PoP to which the budget increment is assigned in the allocation that achieves the minimum penalty).

2 Data details

We rely on a large dataset of 41.7M users with 1.47B edges obtained through a massive crawl of Twitter between June – Sept. 2009 [1]. We then collected the users' location information by conducting our own crawl and translated everything to latitude/longitude using Google Maps API. In the end, we extracted locations for 8,092,624 users from about 11M users that had entered location information. We use this social graph only between these nodes for our analysis in this paper. With regards to the user's locations in the dataset, we find that US has the most users (55.7%), followed by UK (7.02%) and Canada (3.9%).

2.1 Upload Activity

For the users who had entered their locations, we collected their tweets. We found that the mean number of tweets was 42 per user. Not all users had tweet activity; the number of active users (who tweeted at least once) was 6.3M users. For these 6.3M users, we ended up collecting approximately 499M tweets, until Nov. 2010.

This dataset is valuable for characterizing activity patterns of users. From these tweets, we extracted those that contain hyperlinks pertaining to pictures (plixi, Twitpic, etc.) and videos (Youtube, Dailymotion, etc.) that we consider as UGC[1], which gave us 101,079,568 links. Here, we consider that posting a URL pointing to a video is equivalent to uploading the same video as in a social network that offers the option to host multimedia content (e.g., Facebook for photos), a user would have uploaded her generated content directly instead of uploading it to a third party service and posting a URL pointing to it.

We recorded the size of each piece of content that is shared, resolving URL shorteners when needed. The largest file happened to be of a cricket match on YouTube, with a size 1.3G on 480p (medium quality). We collected the number of views for each link, wherever available, and the closest (Kullback-Leibler (KL) distance) fit was the log-normal distribution (parameters: (10.29,3.50)); we found around 30% of the content to be viewed less than 500 times. The most popular was a music video by Lady Gaga on YouTube, viewed more than 300M times.

2.2 Geo-distributed PoPs

To study the effects of geo-diversity on bandwidth costs, we use location data and assign users to PoPs

1. Note that the popularity distribution of such content exhibit a shorter tail [2], [3] than the content uploaded on highly-social platforms on which users upload content for a more limited audience, e.g., pictures of events with friends or family members.

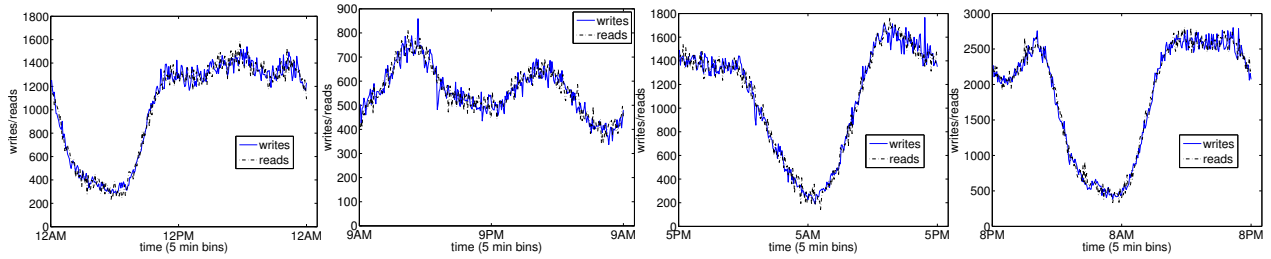

Fig. 1. Writes along with synthetic reads (a) London (b) Tokyo (c) LA (d) Boston.

distributed around the world. We assume the distributed architecture as described in the preliminaries. Because we simulate the actual effects of different PoP locations, for the social-aware scheduler we consider multiple placements. The number of PoPs is a free variable but the locations we choose are fixed as we do not address the PoP 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 PoPs will cover this region and this also explains the imbalance observed in the number of users for the 2 PoPs case. Note that for the above 3 PoPs, the number of users is relatively balanced across them.

City	2 PoPs	3 PoPs	4 PoPs	5 PoPs	6 PoPs	7 PoPs
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
LA		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 PoP for different layouts.

For TailGate, we assume there exist data centers in these four locations: Boston, London, LA and Tokyo. We assign users to locations by using a simple method: compute the distance of a user to a location, and assign the user to the nearest location (w.r.t. the Haversine distance). For the four locations, we observe the following distribution of users: (Boston: 3,476,676, London: 1,684,101, LA: 2,045,274, Tokyo: 886,573). The East Coast of the US dominates in our datasets. The relatively low number of users in Asia is due to the fact that most users in Asia prefer local versions of OSNs. However, we choose Tokyo precisely for this point – users in Asia comprise social contacts of users from around the world, sharing and requesting content, and adding to bandwidth costs. We find that, on average, a user has 19.72 followers in her own PoP and 8.91 followers in each of the other PoPs. It is well known that friends in social networks are located close together with respect to geographical distance [4].

2.3 Read Activity

Our systems rely on information about accesses, i.e., reads. The ideal information will be about *who* requests the content and *when*. We could not obtain direct read patterns from Twitter/FB, so we proceeded as follows.

To get an idea on *who* requests, we collected packet traces via TCPDump from an outgoing link that connected a university in northern Italy (9th Mar, 2011) to the rest of the Internet. The collection was carried out for two distinct periods: from 11AM to 4PM and from 10AM to 6PM. We extracted `HTTP-POST`, `HTTP-GET` requests that correspond to the Facebook domain. We further collected all links corresponding to pictures and videos that show up on users' wall feeds. The clicks on these icons lead to new `HTTP-GET` requests, that are counted as requests for the content. We found that, on average, 9.8% of the posted links were clicked on. It was hard for us to obtain a per-user metric, as the users were behind a NAT device. We looked at smaller set of 200 users that were not behind a NAT device, and observed a similar 10% click rate. Note that in some social networks, users might exhibit much more selective behaviors, varying from one relationship to another (as shown in [5], [6]) while in some cases (e.g., when using a thick client) the content might be downloaded automatically. Therefore, the click-through rate might be very different from one application context to another. We use the observed rate as the probability of a friend requesting content (ρ) for our evaluation, when needed.

Note that future reads are impossible to know, but due to strong regularity in user behavior [7], an OSN such as Facebook or Twitter can predict future accesses with high accuracy. For the purposes of this work – we make the assumption that read patterns follow a diurnal trend similar to write/upload patterns, as observed by other authors [8], [9]. In order to generate read patterns at a fine time scale (seconds), we use SONG [10], which we also describe in the Appendix.

For TailGate, we are interested in evaluating different scenarios, as well as studying the affect of quality of information on performance gains from TailGate. In order to do this, we generate *two* different types of

read patterns: the first type is when social information is available at a fine scale, for instance when Facebook is operating TailGate; and the second is when little or no social information is available, for instance when a CDN is operating TailGate.

Reads with fine grained information: Each user is assigned reads. For this, we make the assumption that the distribution of the number of reads per user follows the same distribution as the number of uploads per user; a log-normal distribution.

Reads with no social information: We assume we do not have enough information to predict reads accurately, but rather general trends, such as diurnal trends, are known. For this, once we generate reads over time bins, we normalize all bins with the total number of reads found across all bins. In other words, we create a diurnal trend for reads. In Fig. 1, we plot the update patterns given by the data and synthetic reads generated for the **day** dataset for all the four centers we consider. Note that the updates follow a diurnal trend and the synthetic reads follow a similar pattern.

Finally, we distinguish long-tailed contents as those which were requested less than 1,100 times (roughly corresponding to 70% of the contents available in our trace). This choice was driven by a modal shift we observed around this value in our dataset. We however evaluated the sensitivity of our results with respect to this choice, varying the threshold for long-tailed content from 500 to 4,000, but we could not appreciate any significant variation in TailGate performance.

2.4 Limitations and Assumptions

We note here that Twitter now has around ∼200M users[2] and our dataset is a relatively small sample. Hence all numbers presented in this paper should be interpreted in this context. An ideal dataset for evaluation would be the UGC that is uploaded and shared on an OSN, but such data is not available to us. Instead, we use the links collected above as proxy for the media, which can be shared over OSNs, and we recognize that activity patterns with regards to posting and sharing media content should follow similar diurnal patterns [9]. We assume read patterns follow a diurnal trend similar to write patterns [8], [9]. Note that we are more interested in time-of-day effects; more sophisticated read patterns where we consider content interest, quality of content, etc. can be used, but we do not have sufficient information to calibrate these effects. We also assume a simple homogeneous click through model in which all users click on the links shared by their friends with a fixed probability, constant across users. In addition, we assume that all the contents uploaded by the users are unique, which is not the case when users can re-share contents shared by their friends (e.g., on Facebook).

2. http://en.wikipedia.org/wiki/Twitter

	ALL		POP		LT	
	d_time1	d_time2	d_time1	d_time2	d_time1	d_time2
Boston	4.11 (6.63)	4.81 (8.51)	3.69 (6.61)	4.39 (7.67)	4.73 (6.67)	4.68 (8.35)
LA	2.94 (5.53)	2.70 (5.71)	1.59 (2.55)	1.72 (3.33)	4.70 (7.46)	3.49 (6.50)
Lon	5.15 (5.06)	4.56 (5.11)	4.57 (3.53)	4.06 (3.37)	6.43 (6.72)	4.88 (5.92)
Tokyo	6.18 (6.76)	5.09 (5.64)	5.36 (6.23)	5.10 (5.42)	6.88 (6.67)	4.65 (4.27)

TABLE 2
Average download times (in seconds) with standard deviations for buffering stages.

3 Case Study: Long-Tailed Videos on YouTube

In this section, we study the case where TailGate has *little* access to social information (NI, IR), but can help with QoE in the case of long-tailed YouTube videos [11], [12], [13]. The entity controlling TailGate (e.g., CDN) can rely on publicly available information (e.g., Tweets), as we do here, and use TailGate to request or "pull" content to intelligently prefetch content before the real requests for the content, thereby decreasing latency for their customers. Towards this end, we develop a simple prototype of TailGate based on the design, and we deploy it on 4 PlanetLab nodes at the same locations: Boston, London, LA and Tokyo.

We proceed as follows: we rely on our dataset, where we assign the four sets of users to different "PoPs" as given by Planetlab nodes. We extract the set of links that correspond to YouTube videos from our dataset, along with the times they were posted. Note that the information related to the properties of YouTube videos accessed from Twitter is public: anyone can collect this information. We then provide this set of writes as input to TailGate, assuming no social information and that the expected reads in various locations follow a diurnal pattern. We get a schedule as output from TailGate, which effectively schedules transfers between the four locations. We take this schedule and instead directly *request* the YouTube videos from various sites, at a time given by TailGate, thus emulating transfers.

After this, we request the videos at the time of the "read", that is, we "emulate" users from each location issuing read requests for each video by sampling from the diurnal trend. Therefore each video is requested twice: the first time for emulating the transfer using a schedule given by TailGate, and the second time, for emulating a legitimate request by a user to quantify the benefit. Note that the first request would also be emulating a PULL, as we emulate a cold-miss. Hence any improvements we notice would be an improvement over PULL. We use `wget` with the `-no-cache` option for all our operations, to limit caching effects.

We focus on the Quality of Experience (QoE) for the end-user. In order to measure this, we first look at the proportion of a file that is downloaded during the initial buffering stage, after which the playout of the video is smooth. We say the playout is smooth

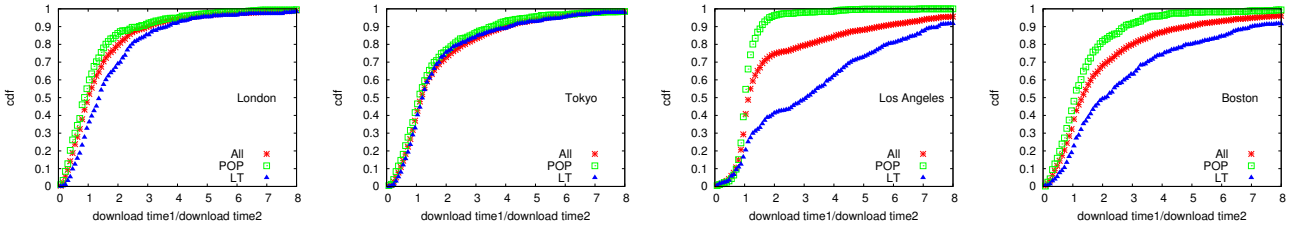

Fig. 2. Performance figures for YouTube videos, improvements for download times for buffering stage.

if the download rate[3] for a file drops at 70% of the original rate. We tested other values and had similar results. We found that, on average, the playback is smooth after 15% of a file is downloaded. Therefore we noted the delay in terms of the time it takes for the first 15% of a file to be downloaded. As we download each video twice, once at a time given by TailGate and the second as representing the actual read request, we measure both and plot the cdfs of ratios (dload_time1/dload_time2) in Fig. 2. We plot for three different cases: "all" is the entire dataset; "pop" stands for only those videos that are popular ($\geq$ 500K views); and "LT" stands for long-tailed videos ($\leq$ 1,100 views– we observed a mod shift around this value in our dataset). First, we note that there is an improvement of a factor of 2 and it is higher for at least 30% of the videos for all locations. Second, this improvement is even more pronounced for "LT" videos, highlighting that TailGate aids long-tailed content. Indeed, we remark that popular content has in general a larger probability of being present at the PoP/cache at the moment of the first request, thus lowering de facto the benefits of adopting TailGate. This does not hold for long-tailed videos that show a lower probability of being replicated at many PoPs, thus increasing the potential gain of TailGate. Observe that given the bulky nature of video content, downloading a small fraction of a video may take some tens of seconds, so that the gain in time achieved by TailGate is significant for the QoE perceived by users. In Table 2, we report the mean and standard deviations of download times for all considered scenarios. For some videos, we see a decrease in performance (dload_time1/dload_time2 < 1). This could be due to load-balancing. In fact for Tokyo, we found that the closest PoP for YouTube seems to be relatively far (Korea) in the first place.

APPENDIX

We briefly review the model we use for generating synthetic reads. Let $X_i(t)$ denote the number of reads produced by user i, $1 \leq i \leq N$ with N the total number of users at a time instant t, where $X(t) = \sum_{\forall i} X_i(t)$. The time can vary from seconds to weeks. The description $X(t), \forall t$ gives the time series aggregated over all users. We need to account for two different time-scales - the first time scale spans multiple hours or days and we note the presence of diurnal trends. The second time scale spans seconds to a couple of hours where the mean and the variance are fairly stable. For the first time scale, we can have a model for the mean m_t of the time series that varies with time in a predictable way. For the second time scale, we can have a stochastic component. The model then is

$$X(t) = m_t + \sqrt{am_t}W_t \quad (1)$$

where m_t is function of time and W_t is a stochastic component which can be a zero-mean, finite variance process and a is a parameter called 'burstiness' (with the units:reads-secs) that accounts for magnitude of fluctuations.

The main method for generating synthetic reads is as follows: First we generate m_t using a Fourier series by first extracting the largest Fourier coefficients of the write time-series, then we add appropriately scaled noise, by estimating a, to the diurnal trend at each time bin (in our case, seconds). For our **day** and **week** datasets, we use the top 5 Fourier coefficients to generate the diurnal pattern and used WGN with an appropriate scale parameter to generate the reads. The generated time series contains the number of reads in a given time bin.

3. To measure the download throughput, we employed the download rate estimator embedded in `wget`.

REFERENCES

[1] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a Social Network or a News Media?" in *WWW*, 2010.
[2] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *IMC*, 2007.
[3] N. R. Sastry, "How to tell head from tail in user-generated content corpora," in *ICWSM*, 2012.
[4] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins, "Geographic Routing in Social Networks," *PNAS*, vol. 102, pp. 11 623–11 628, 2005.
[5] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Eurosys*, 2009.
[6] H. Chun, H. Kwak, Y.-H. Eom, Y.-Y. Ahn, S. Moon, and H. Jeong, "Comparison of online social relations in volume vs interaction: a case study of cyworld," in *IMC*, 2008.
[7] S. A. Golder, D. M. Wilkinson, and B. A. Huberman, "Rhythms of Social Interaction: Messaging Within a Massive Online Network," in *C&T*, 2007.
[8] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing User Behavior in Online Social Networks," in *IMC*, 2009.

[9] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding Online Social Network Usage from a Network Perspective," in *IMC*, 2009.
[10] V. Erramilli, X. Yang, and P. Rodriguez, "Explore what-if scenarios with SONG: Social Network Write Generator," http://arxiv.org/abs/1102.0699, 2011.
[11] highscalability, "YouTube Architecture," http://highscalability.com/youtube-architecture, visited June 2013.
[12] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao, "Dissecting Video Server Selection Strategies in the YouTube CDN," in *ICDCS*, 2011.
[13] "YouTube CDN Architecture," Private Communication, Content Delivery Platform, Google.