Cloud-based Social Application Deployment using Local Processing and Global Distribution

Zhi Wang*, Baochun Li†, Lifeng Sun*, and Shiqiang Yang*
*Beijing Key Laboratory of Networked Multimedia
Department of Computer Science and Technology, Tsinghua University
†Department of Electrical and Computer Engineering, University of Toronto
{wangzhi04@mails., sunlf@, yangshq@}tsinghua.edu.cn, bli@eecg.toronto.edu

ABSTRACT

Social applications represent a paradigm shift on how the Internet is to be used, and have already changed the way we work, live, and play. When it comes to deploying social applications, cloud computing platforms are used to meet the Internet-scale, self-propagating, and fast-growing demands from these applications. Yet, to deploy social media applications in the most effective and economic fashion, we need to strategically design and follow a set of theoretical and practical principles. In this paper, we seek to design a set of new principles to guide social application deployment. Learning from large-scale measurement-based observations using a real-world social application, the gist of our principles is to detach the typically integrated "collection → processing → distribution" workflows in social applications into separate *local processing* and *global distribution* procedures, which can be effectively deployed using different cloud services. Moreover, based on a predictive model of regional propagation, we formulate the resource allocation problems in the processes of collecting/processing and distributing content as two optimization problems, which can be solved by efficient algorithms. Finally, based on our theoretical design, we have implemented an example social application on Amazon EC2 and Google AppEngine, where IaaS-based computation instances perform content collection and processing, and the PaaS-based platform is employed to distribute the contents that are widely propagating. Our PlanetLab-based trace-driven experiments have further confirmed the superiority of our design.

Categories and Subject Descriptors

C.2.4 [**Distributed System**]: Distributed Applications; H.4 [**Information Retrieval**]: Social Network

Co-NEXT'12, December 10–13, 2012, Nice, France.

General Terms

Measurement, Design

Keywords

Social application deployment, online social network, cloud computing

1. INTRODUCTION

Applications deployed in online social networks [18] have emerged as one of the most popular means for users to access multimedia contents in today's Internet [14]. This is due to a new development scheme in online social networks: by simply becoming *developers*[1] of large social networks like Facebook, social media companies can use user profiles and social relationships via Open APIs[2], and are able to develop applications for millions of potential users, without building a new social network. At the end of March 2012, over 9 million apps integrated with Facebook are using such a development paradigm[3].

In this paper, we focus on the problem faced by social media companies after new applications have been developed: how do we effectively and economically deploy these applications? The deployment of a social application is challenging due to a number of unique evolution characteristics: (1) It is potentially Internet-scale from the beginning, since a social application depends on an online social network to directly attract its global users, making the number of users grow much faster than traditional multimedia applications; (2) it is self-propagating, since the application can be recommended to users by their friends when they are using the social application; and (3) it is fast-growing, since the number of users can increase rapidly due to propagation caused by the social effects. As an example, the social application *WeChat* depends on Tencent Inc.'s social network services, and has hit a record of 200 million users in less than 14 months[4].

Since highly scalable and elastic network resources are required to deploy new social applications, it is promising to deploy social applications in the cloud for a number of reasons: (1) Small social application companies, which develop

[1]http://developers.facebook.com/
[2]http://en.wikipedia.org/wiki/Open_API
[3]http://newsroom.fb.com/content/default.aspx?NewsAreaId=137
[4]http://www.wechatapp.com

social applications for large social networks, can build their own global service by simply becoming customers of cloud providers; (2) the system can easily scale when the number of its users and the volume of its contents increase; and (3) social applications can be easily implemented in the cloud due to complete control of servers based on virtualization (*e.g.*, virtual machines (VMs) running different operating systems).

Cloud computing has been widely used to handle various traditional multimedia contents [15, 22], *e.g.*, Netflix has been delivering its movies to users based on the Amazon cloud infrastructure since 2010 [3]. Due to its unique propagation patterns, efforts have been devoted in the deployment of social media. Wang *et al.* [30] observed that information in an online social network can be used to predict content access in a standalone content sharing system, which can guide content deployment. Cheng *et al.* [9] have studied the partitioning schemes for social contents to achieve a balanced load at the servers and preserve social relationship. Wu *et al.* [31] have studied cost-effective video distribution in a social network by migrating videos in geo-distributed clouds. However, existing studies only solve the *content distribution* problem in social media; in this paper, we study the *deployment* of social *application*, which includes content collection, processing and distribution.

In a typical social application, user-generated contents (UGCs) are the dominant form of contents, *i.e.*, they are first generated by users, then collected and processed by the system, and finally distributed to other users through the social relationships. To deploy a social application, we take the characteristics of social media into account as follows: (1) Users are the sources of contents in social media. Instead of central content providers, users are the ones who generate contents for social media [11]; (2) social media is dynamically processed and aggregated, *i.e.*, contents generated by users are uploaded to the social media system, which performs various *processing* to these contents and distributes the processed contents to users [19]; and (3) the distribution of social media is severely affected by social propagation [29]. Propagation with social media is no longer random — it is determined by the social network topology and user sharing [8][27].

According to the properties of content collection, processing and distribution, we allocate computation, storage and network resources from the cloud to deploy a new social application as follows. (1) *Local processing* — contents are initially collected and processed by cloud servers that are geographically close to the user generating them. Since *content processing* varies from one application to another, we deploy the processing part using IaaS (Infrastructure as a Service)-based instances, *e.g.*, VM instances provided by Amazon EC2 (Elastic Compute Cloud) [1], where the application can be implemented in various programming languages. However, it is costly and difficult to build a highly scalable and global distribution platform to serve users over the world based on IaaS only. (2) *Global distribution* — processed contents are finally distributed by servers that are geographically close to users who receive such contents. Fortunately, cloud computing provides another resource allocation scheme, PaaS (Platform as a Service), *e.g.*, Google AppEngine [2], where resource is provided to users in an auto-scaled manner. In our design, we build the distribution platform using PaaS to distribute the contents processed by IaaS-based computation instances.

In our cloud-based social application deployment, we are presented with the following challenges: (1) How should we allocate IaaS-based computation instances to process contents generated by users located within different *regions*? (2) How should we choose contents to be replicated to a PaaS-based distribution platform to serve global users? and (3) How should we design efficient protocols to connect local content processing and global content distribution?

In this paper, we answer these questions by connecting the characteristics in social media propagation with its deployment design. Our contributions can be summarized as follows: (1) We conduct extensive measurements to study the propagation characteristics in social media and motivate our design; (2) We provide theoretical guidelines for social application deployment using local processing and global distribution; and (3) We implement an example social application to evaluate the effectiveness and efficiency of our design based on Amazon EC2, Google AppEngine and PlanetLab.

The remainder of this paper is organized as follows. We review related work in Sec. 2. We conduct large-scale measurements to study the characteristics of social media in Sec. 3. We present our detailed design and analysis in Sec. 4. We discuss our implementations in Sec. 5. We evaluate the performance of our design in Sec. 6. Finally, we conclude the paper in Sec. 7.

2. RELATED WORK

In this section, We discuss our work in light of the existing literature on social media deployment and cloud computing, respectively.

Online social applications. In a social media system, contents spread among users by users sharing them. A number of research efforts have been devoted to studying content propagation in social media applications. Kwak *et al.* [20] investigated the impact of users' retweets on information diffusion in Twitter. Social applications have greatly changed our assumptions in traditional content service deployment, *e.g.*, content distribution is shifted from a central-edge manner to an edge-edge manner, resulting in the massive volume of user-generated contents and a dynamically skewed popularity distribution [7]. In this paper, we not only focus on the distribution of contents already in an online social network, but also the collection and processing and contents generated by users in a social application. In particular, we explore the deployment of social applications based on cloud computing.

Social application deployment based on cloud computing. Cloud computing is a new computing paradigm in which both hardware and software are provided to users over the Internet as services, in the form of virtualized resources [12]. Different cloud providers provide different types of services [26], including IaaS, PaaS, SaaS (Software as a Service), *etc.*, based on different pricing schemes [6], *e.g.*, by actual CPU cycles in Google AppEngine [2] or by the number of VM instances in Amazon EC2. Due to its elasticity, cloud computing has also been widely used by startup companies whose demands of resources grow over time [15]. Traditional systems, such as the Web [22] and video streaming [3], have been being successfully deployed in the cloud. Among multiple cloud providers, Li *et al.* [21] have proposed a service comparison methodology to compare the performance with

different cloud providers. Rehman *et al.* [25] have proposed a multi-criteria cloud service selection strategy, to determine the service that best matches the users' requirements from amongst numerous available services. Chohan *et al.* [10] have studied the extension of PaaS to facilitate the distributed execution of applications over virtualized cluster resources.

In the context of social applications, cloud computing has been explored for the social media distribution. Pujol *et al.* [24] have investigated the difficulties of scaling online social network, and designed a social partitioning and replication middleware in which users' friends can be co-located in the same server. Tran *et al.* [28] have studied the partition of contents in the online social network by taking social relationships into consideration. Cheng *et al.* [9] have studied the partitioning schemes for social contents to achieve a balanced load at the servers and preserve the social relationships. Wu *et al.* [31] have studied the problem of cost-effective video distribution in a social network by migrating videos in geo-distributed clouds.

Different from related works, in this paper, we study how content processing and distribution are jointly performed by the cloud. Particularly, we design deployment strategies for social applications by taking the characteristics in social media into account, based on measurement studies of real-world online social networks.

3. BACKGROUND AND MEASUREMENT STUDY

In this section, we explore the design principles of social application deployment and present the benefit of the cloud-based design using real-world measurements. We first show an example that has the general features of social applications. Then, based on an extensive measurement study of real-world online social networks and cloud systems, we show that contents in a social application can be processed locally and distributed globally.

3.1 Framework of a General Social Application

Though different social applications are designed to provide users with different contents and experiences, they share many common features: (1) Users contribute contents to the applications; (2) contents are aggregated by various approaches and provided to different users; and (3) contents propagate through social connections of the online social network.

To study the cloud-based deployment of social applications with these features, we use an example social application in our measurements and our system design. We design our example application to be as general as possible to capture most of the features in social applications. Our example social application is called *SICS*, a social and interest-based content sharing system. In SICS, besides contents shared by a user's friends as in Twitter-like systems, other contents are also recommended to the user, which is based on content processing, where computation resources are required to parse the contents and execute the processing algorithms. After content processing, SICS provides the user a set of enriched contents with the recommended ones. Fig. 1 illustrates the paradigm of SICS. a and b are the original contents generated by user A and user B; while a' and b' are the enriched contents after content processing, *i.e.*, $a' = \{a, a_1, a_2, \ldots\}$, where a_k is a recommended content based on the original content a, and $b' = \{b, b_1, b_2, \ldots\}$, where b_k is a recommended content based on content b. a' and b' are then provided to users C, D and E according to social propagation. As an example social application, SICS contains the general content generation, processing and distribution procedures. We observe that a general social application framework is similar to a microblogging system, *e.g.*, users have social relationships between them, contents are generated by users and processed by the system, and then distributed to other users who are socially connected. We next study how the social application can be effectively deployed, using a measurement-driven approach based on the traces from Tencent Weibo.

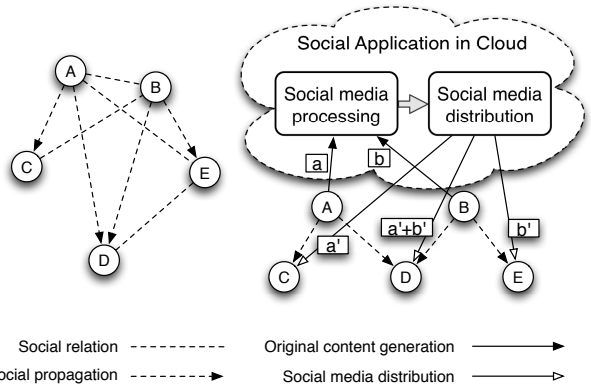
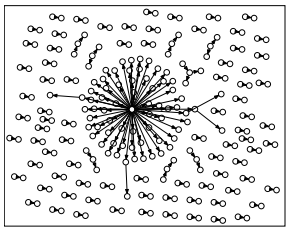


Figure 1: Content generation, propagation and distribution in a social application.

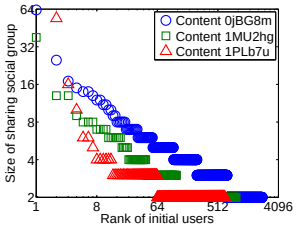
3.2 Regional Analysis for Social Application Deployment

Our measurement study is based on traces collected from the operation team of Tencent Weibo [4], which is a microblogging website, where users can broadcast a message including at most 140 characters to their friends. Tencent Weibo features several social activities in the system, *e.g.*, online chatting with friends who mutually follow each other. We obtained Weibo traces from the technical team of Tencent, containing valuable runtime data of the system in 20 days (October 9 – October 29) in 2011. Each entry in the traces corresponds to one microblog published (which will be regarded as an item of user-generated content), including the ID of the microblog, the IP address and geographic region of the publisher, the timestamp when the microblog was posted, the IDs of the parent and root microbloggers if it is a re-post, and contents of the microblog.

Since we are focused on how multimedia contents should be handled in a social application, in our measurement study, we have targeted at video contents which are imported from external websites. In particular, we have collected more than $300,000$ links from 5 popular video sharing sites. We then retrieve the microblogs which are related to these links, *i.e.*, the microblogs either include the links to these videos in the contents or they are re-shares of the ones that include the links. These links cover about 2 million microblogs in the time span, which are posted or re-shared by over 1 million users, from more than 100 regions in the propagation

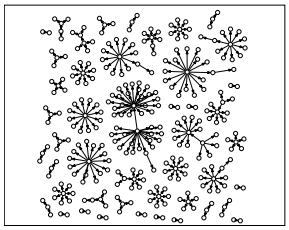


(a) Social groups sharing the same content.

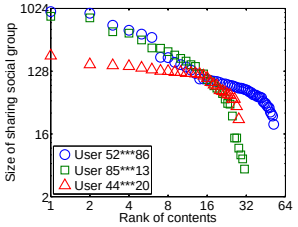


(b) Size of social group versus the rank of user initiating the sharing.

Figure 2: Social groups sharing the same content initialized by different users.



(a) Social groups initiated by the same user.



(b) Size of the social group versus the rank of contents.

Figure 3: Social groups initialized by the same user sharing different contents.

(each region is defined by Tencent as a city-level geographical area). In addition, we have also retrieved the social connections of these users. In our study, how contents are used by Weibo users will guide our design of social application deployment.

3.2.1 *Dynamics of Users*

The most important characteristic of a social application is that contents are propagated between users through their social connections. As a result, user influence is critical in social media application deployment. In Fig. 2(a), a circle represents a user, a directed edge represents the propagation between two users, and the connected components (trees) are social groups initiated by different users sharing the same content. We observe that while some users can attract a large number of friends to join the group, many others have much little influence. Particularly, in Fig. 2(b), each sample represents the size of the sharing group versus the rank of the user initiating the group. We observe that when the same content is shared by different users, the size of the sharing groups varies significantly.

3.2.2 *Dynamics of Contents*

Further, we also observe that social media sharing is highly affected by the contents themselves. In Fig. 3(a), the trees are social groups initialized by the same user sharing different contents. We observe that different contents can attract quite different numbers of friends. In particular, in Fig. 3(b), each sample represents the size of the sharing group versus the rank of the content shared in the group. We have observed that when different contents are initially shared by the same user, the size of the sharing groups varies significantly as well.

The propagation of contents in social applications can be

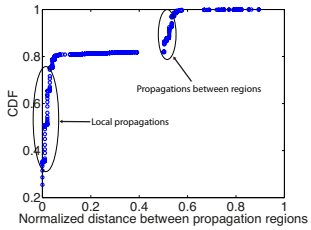


Figure 4: Normalized distances between the propagation region pairs.

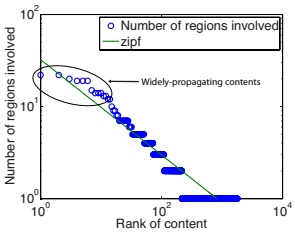


Figure 5: The number of regions involved in the distribution of a specific content.

dynamically affected by both users and contents. To effectively allocate cloud resources for deployment, in our study, we analyze propagation within a regional level, in which propagation statistics can be highly stable and predictable. Next, we will present the efficiency of local processing and global distribution, both of which are carried out using a regional analysis approach.

3.3 Efficiency of Local Processing

3.3.1 *Locality of Content Propagation*

First, we show the propagation locality between content generation and distribution. We define a normalized geographic distance to measure two regions in our study as $\frac{d_{ij}}{\max\{d_{ij}\}}$, where d_{ij} is the real great-circle distance between region i and region j. Fig. 4 illustrates the normalized geographic distances between the region where new content is generated and the regions where the content is distributed to according to the social connections. We observe that the normalized geographic distances for most of the media content propagations are very small, *e.g.*, more than 80% of the propagations are within a normalized distance of 0.1. The reason is that a dominant portion of the contents generated by users within a region will be served to the users mainly from the same region, according to social connections that determine how information flows in an online social network [8].

Furthermore, we also observe the locality in content distribution. We define a region involved in the content's generation (distribution) as a region where the content is generated from (to be distributed to). Fig. 5 illustrates the number of regions involved in the distribution of a specific content in one day. Contents are ranked in a descending order with respect to the number of regions involved in their distribution. Each sample represents the number of regions involved in the distribution versus the rank of the content. We observe that for most of the contents, only a few regions are involved in their distributions. The reason is that most of the contents are distributed locally, *i.e.*, many users are located within the same region. The observations indicate that in social application deployment, computation instances could be allocated at multiple regions so as to collect, process and distribute the contents locally, reducing inter-region traffic to deliver the contents across regions.

3.3.2 *Stability of Regions Involved Over Time*

Due to the locality of propagation, the social application system will allocate computation instances within different

regions to process the contents locally. However, how many and which regions should be selected to deploy the computation instances is still a question. To answer this question, we next study which regions are involved in content propagation.

Fig. 6(a) illustrates the number of contents generated by users from all the regions over time. Each sample represents the number of contents generated by users in a time slot (hour) versus the time. We observe obvious daily pattern — much more contents are generated during the peak hours (about 7000 per hour) than during the off-peak hours (about 500 per hour). We then study the regions involved in content generation.

Comparing to the number of generated contents, no evident daily pattern is observed in the number of regions involved in the generation, as illustrated in Fig. 6(b). Each sample in Fig. 6(b) represents the number of regions involved in content generation versus the time. We observe that the number of regions involved in content generation remains at a relatively high level over time (with the largest number 37 and the smallest number 33 per hour). Similar results are also observed in content distribution. Fig. 6(c) illustrates the number of regions involved in content distribution over time. We observe that the number of regions involved in content distribution also stays at a stable level.

These observations indicate that contents are always generated from and to be distributed to almost all the regions (the total number of regions is 41 in our traces).

3.3.3 Predictability of Regional Propagation

According to the observations above, we need to allocate computation instances at almost all the regions available to collect and process the contents locally. We next investigate how much network and computation resources within each region we should allocate, by studying the number of contents generated by users at each region.

Let the *content generation rate* of a region denote the number of contents generated by users within the region in a given time slot. In Fig. 7, regions are ranked according to their content generation rates. Each sample in this figure represents the content generation rate of a region versus the rank of the region. The popularity distribution of regions is not even — some regions can generate much more contents than other regions. Fig. 8 illustrates the distribution of the content generation rates at 4 different regions randomly selected (in Fig. 8, the content generation rates at level x are in $[40x, 40(x+1))$). We observe that the distributions of content generation rates at different regions are also quite different. However, in Fig. 9 which illustrates content generation rate of each region over time, we observe strong evidences of daily patterns for all regions. This observation indicates that the content generation rate of a region is highly predicable.

Based on the observations, we are able to design a predictive model to estimate the *regional* content generation rate, which will be utilized in local content processing. We will discuss our detailed design in Sec. 4.

3.4 Efficiency of Global Distribution

To handle the social contents generated at different regions locally, we allocate computation instances at different regions to collect, process and distribute the contents locally. In social media, some contents can be very popular with many requesting users. Such contents are referred to as *widely-propagating contents*, which can attract users from a large number of regions, as illustrated in Fig. 5. In the distribution of a widely-propagating content, a large fraction of users will experience low download performance if they all download the content from the computation instance where the content is originally collected and processed, since these users can be located at other regions far away from the original region, resulting in a low download bandwidth. To address this problem, a global distribution platform which can effectively distribute widely-propagating contents to users within many regions is needed. Due to dynamic social propagation, it is not always easy to predict the popularity of social media contents, which is affected by not only the social network topology but also the influence and preference of users.

In our measurements, we will show that contents processed by computation instances can be effectively replicated to a global distribution platform, which is able to significantly improve the distribution performance. We implement the computation instances in C++ on Amazon EC2 micro nodes and the distribution platform in Python on Google AppEngine. We choose the following different sizes for contents that can be generated by users: 1.1 MB, 160 KB and 50 KB. We allocate computation instances in the 7 regions provided by EC2: Virginia (US East), Oregon (US West), California (US West), Ireland (EU West), Singapore (Asia Pacific), Tokyo (Asia Pacific) and Sao Paulo (South America). Meanwhile, 57 PlanetLab nodes are implemented to upload and download contents as well, simulating the social application users. The detailed implementation is to be discussed in Sec. 6.

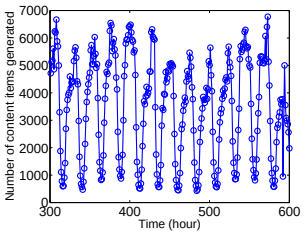
3.4.1 Connectivity Between Local Processing and Global Distribution

When using the distribution platform to deliver the widely-propagating contents, these contents have to be first replicated from the computation instances (EC2) to the distribution platform (GAE). We measure the overhead for such replication. As illustrated in Fig. 11, we compare the times computation instances at different regions spend on uploading the contents to the distribution platform, with the average time that the instances spend on directly uploading them to the PlanetLab nodes at different locations, in the case that the distribution platform is not employed.

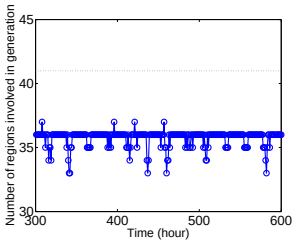
We observe that the time computation instances spend on uploading the processed contents to the distribution platform is much smaller than the average time computation instances spend on uploading the contents to the users directly. This observation indicates that the replication overhead is small, compared to the time the instances spend on uploading the contents to the users directly.

3.4.2 Benefits of a Global Distribution Platform

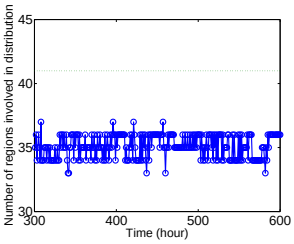
Next, we show that the GAE-based distribution platform outperforms the EC2-based computation instances in distributing widely-propagating contents to users at multiple regions. Fig. 12 compares the download times achieved by the computation instances and by the distribution platform. In this figure, each sample represents the average time that a PlanetLab node takes to download the content from the computation instance or the distribution platform. We observe that for most of the PlanetLab nodes, their download



(a) The number of contents uploaded by users over time.



(b) The number of regions involved in content generation over time.



(c) The number of regions involved in content distribution over time.

Figure 6: Social media generation and distribution.

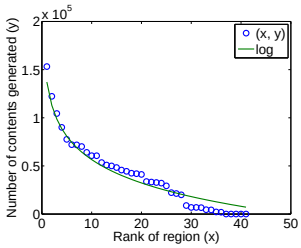


Figure 7: The number of contents generated within a region versus the rank of the region.

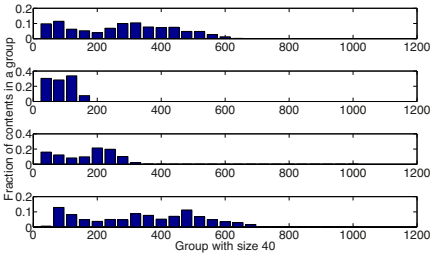


Figure 8: Distribution of content generation rate at four regions.

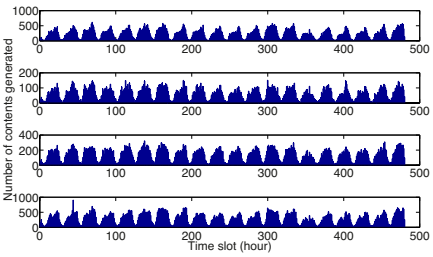


Figure 9: The content generation rate over time at four regions.

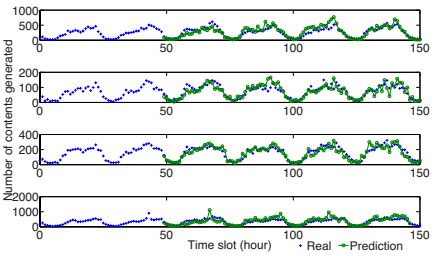


Figure 10: Prediction of content generation rates.

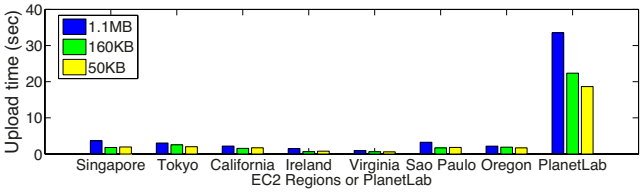


Figure 11: Delivery time comparison between the computation instances to the distribution platform and the computation instances to users.

times at EC2 are much larger than that at GAE. The reason is that GAE has already automatically replicated the contents to different locations, so that users can be redirected to servers that can best serve them. Though it is not the focus of this paper, interested readers are referred to the related works devoted to the distribution of social media contents [29][31]. The observation indicates that in social media distribution, a global distribution platform is needed when new content is supposed to attract users from many different locations. Motivated by this observation, we design a hybrid replication strategy to distribute the contents based on both local computation instances and the global distribution platform, where contents attracting more users from multiple regions will be replicated to the distribution platform.

4. DETAILED DESIGN OF THE SOCIAL APPLICATION DEPLOYMENT IN CLOUD

In our measurement study, we show that a social application can be effectively deployed based on local processing instances and a global distribution platform. In this section, we first present a new framework for social application deployment, and then describe our detailed design on how to collect contents generated by users, process them and distribute the processed contents to the users.

4.1 Framework

In a social application, though users are globally distributed within different regions when they generate contents for and download the contents from the system, content propagation is highly localized. To effectively handle the contents in a social application, we design a new framework as illustrated in Fig. 13: (1) IaaS-based computation instances are allocated to collect the contents generated by users within different regions and perform the content processing locally; and (2) a PaaS-based distribution platform is allocated to assist the distribution of widely-propagating contents. We next demonstrate the advantages of our new framework.

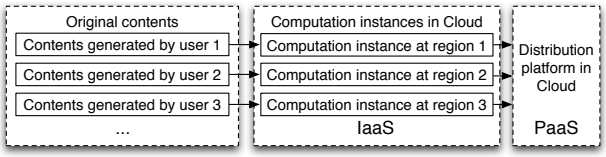
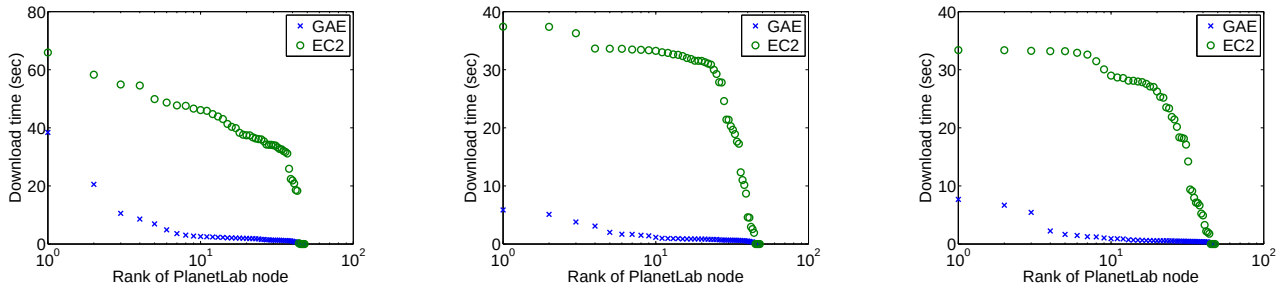


Figure 13: Framework of the social application deployment.

Local collection and processing. In content processing, allocating computation instances at multiple regions has the following advantages: (1) Allocating computation instances close to users can improve the performance for them to upload and download the contents; (2) we observe that the propagation is highly localized in our measurements, *i.e.*,



(a) file size 1.1 MB. (b) file size 160 KB. (c) file size 55 KB.

Figure 12: Comparison of download times users experience when downloading contents from the computation instances and the distribution platform.

contents generated within one region are likely to be requested by users within the same region. Deploying computation instances close to users can reduce the inter-region transmission cost [17]; and (3) the prices for computation instances at different locations can be different [6] — it is intriguing to investigate how to allocate cloud resource from different locations so that the generated contents can be efficiently processed with minimum costs.

Global distribution. After being processed by computation instances, users can directly download these contents from the instances. In our measurements, we observe that the PaaS-based distribution platform can achieve much smaller download times for users to obtain some popular contents, which are requested by the users at many regions. In our deployment design, we strategically select a set of such widely-propagating contents over time, and replicate these contents from the local computation instances to the global distribution platform. The social application system can dramatically improve the distribution performance when the distribution platform upload to users that are far away from the original computation instances. Due to the large number of contents generated and the limited budget for distribution, we need to strategically determine which contents should be replicated to the distribution platform and which ones are only served by the local computation instances. Table 1 summarizes important notations for ease of reference.

Table 1: Important notations

$\mathcal{R}^C$	Set of regions having computation instances
$\mathcal{R}^U$	Set of regions where users are located in
P^H	Unit price for content processing rate of computation instance
P^U	Unit price for upload capacity of distribution platform
P^S	Unit price for storage of distribution platform
λ_i	Content generation rate from region i
λ_{ij}	Content generation rate redirected from region i to j
Λ_j	Content generation rate redirected to region j
μ_j	Content processing rate allocated at region j
$M_c(\vec{\mu})$	Processing cost under allocation scheme $\vec{\mu}$
$\mathcal{D}^{(T)}$	Set of candidate contents for replication
$\mathcal{S}^{(T)}$	Set of processed contents served by the global distribution platform
$M_d(\mathcal{S})$	Distribution cost for the contents in $\mathcal{S}$

4.2 Computation Instance Allocation for Local Processing

Given that computation instances allocated in multiple regions can benefit social application deployment, the problem is to determine the allocation strategy, *i.e.*, the capacities of computation instances at different regions.

4.2.1 Allocation Scheme and Cost

A social media company has to perform instance allocation according to its budgetary constraints. In IaaS, the general pricing rules are as follows: (1) The more instances the social application allocates, the more the cloud provider will charge; (2) The prices vary with different regions, *e.g.*, the unit price of a VM instance in US West is higher than the price in US East in May 2012; and (3) The prices also vary over time.

The cost is determined by instance allocation. In our design, we let vector $\vec{\mu} = \{\mu_1, \mu_2, \ldots\}$ denote the cloud instance allocation scheme. Each entry $\mu_j, j \in \mathcal{R}^C$ in $\vec{\mu}$ determines the aggregate *content processing rate* of the cloud instances allocated at region j, and $\mathcal{R}^C$ denotes the set of regions where the computation instances can be allocated, *i.e.*, the cloud provider has deployed servers in data centers within these regions. Larger μ_j indicates that more contents can be processed in region j per time unit, resulting in a higher cost. In our design, the cost of an allocation scheme $\vec{\mu}$ can be estimated as follows:

$$M_c(\vec{\mu}) = \sum_{j \in \mathcal{R}^C} P_j^H \mu_j,$$

where P_j^H is the unit price of processing rate at region j. Note that we assume proportional upload and download bandwidths are also allocated at region j according to μ_j, so that the generated contents can be collected and distributed by the computation instances. The prices for bandwidths are included in P_j^H.

4.2.2 Prediction of the Content Generation Rate in Each Region

To efficiently allocate the computation instances within a region, *i.e.*, to determine the content processing rate, we can refer to the content generation rate in that region. The rationale is that it would be a waste if the content processing rate is much larger than the content generation rate;

while it takes too long for content to be processed when the content processing rate is much smaller than the content generation rate. The efficient content processing rates depend on the actual content generation rates. According to our measurements, the regional content generation rate is highly predictable.

Let $\lambda_i^{(T)}, i \in \mathcal{R}^U$ denote the content generation rate at region i in time slot T, where $\mathcal{R}^U$ is the set of all regions that users are located at (generally, $\mathcal{R}^C \neq \mathcal{R}^U$). In Sec. 3, we observe that the content generation rate at each region shows strong evidence of the daily pattern. It indicates that the content generation rates can be predicted using autoregressive models [5].

In our design, we predict $\lambda_i^{(T)}$ based on the historical content generation rates $\{\lambda_i^{(T-1)}, \lambda_i^{(T-2)}, \ldots, \lambda_i^{(T-M)}\}$, where M is the number of previous generation rates to refer to in the prediction. An ARIMA (AutoRegressive Integrated Moving Average) [23] model is used, *i.e.*,

$$(1 - \sum_{k=1}^{p} \Phi_k L_k) Y^{(T)} = (1 + \sum_{k=1}^{q} \Theta_k L_k) \varepsilon^{(T)},$$

where p is the order of autoregressive and q is the order of moving average. Φ_k and Θ_k are the parameters of the autoregressive and moving average parts, respectively. $\varepsilon^{(T)}$ is the white noise for the stationary distribution. $Y^{(T)}$ is defined as follows,

$$Y^{(T)} = (1 - L)^d \lambda_i^{(T)},$$

where L is the lag operation, *i.e.*, $L^d \lambda_i^{(T)} = \lambda_i^{(T-d)}$. In our design, the content generation rate at each region ($\lambda_i^{(T)}$) is recorded hourly. To capture the daily pattern, we choose the period of $d = 24$ hours, so that $Y^{(T)}$ can be regarded as wide-sense stationary. In our experiments, 48 hours of historical records are utilized to predict $\lambda_i^{(T)}$, by training the predictive parameters using a maximum likelihood estimation. Based on the implementation of ARIMA in R with parameters $p = 48$ and $q = 0$, we present the prediction results of the four randomly chosen regions used in our measurement in Fig. 10. We observe that the predictive model only needs a small learning window to give an relatively accurate estimate of the content generation rate.

4.2.3 Computation Instance Allocation

In the computation instance allocation, we regard each content generated from a region in $\mathcal{R}^U$ as a task for the computation instances to process. After a content is uploaded by a user to a computation instance, it is queued to be processed at the computation instance. A content will be available to users only after it has been processed. Our objective is to allocate computation instances strategically to minimize the average time for a content to be available to users. Fig. 14 illustrates the procedure of the computation instance allocation: (1) Historical content generation rates at different regions in $\mathcal{R}^U$ are collected; (2) The current content generation rates are estimated using the predictive model; (3) The predicted content generation rates are scheduled to different computation instances within regions in $\mathcal{R}^C$; and (4) According to the scheduled content generation rates, content processing rates are allocated. We will provide more details next.

Prediction and schedule of generated contents. For simplicity, we use λ_i to denote the predicted content generation rate from region i in time slot T, and λ_{ij} to denote the rate of contents generated at region i to be scheduled to the computation instances at region j in time slot T. The schedule is as follows:

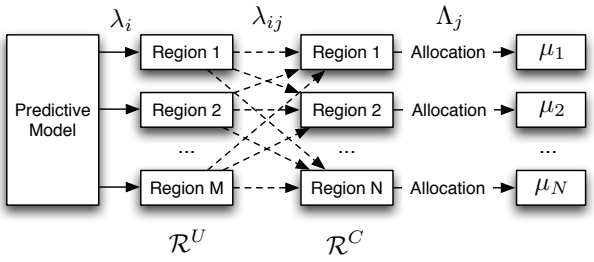


Figure 14: The allocation of computation instances.

$$\lambda_{ij} = \begin{cases} \lambda_i, & j = \arg\min_k d_{ik}, \\ 0, & j \neq \arg\min_k d_{ik}, \end{cases} \quad i \in \mathcal{R}^U, j \in \mathcal{R}^C.$$

The rationale is that we assign a user to the region that is closest to him, so that he can spend the minimum amount of time on uploading the generated contents to the system. Let Λ_j denote the rate of contents from all regions to be uploaded to instances allocated at region j, *i.e.*, $\Lambda_j = \sum_{i \in \mathcal{R}^U} \lambda_{ij}, j \in \mathcal{R}^C$.

Allocation of processing rates. Let $W_c(\vec{\mu})$ denote the average waiting time for the original contents to be processed using the allocation scheme $\vec{\mu}$. According to the queuing model [13], we have the average waiting time for a content to be processed by the social application system as follows:

$$W_c(\vec{\mu}) = \frac{\sum_{j \in \mathcal{R}^C} \frac{\Lambda_j}{(\mu_j - \Lambda_j)}}{\sum_{j \in \mathcal{R}^C} \Lambda_j}.$$

To make the processed contents available to users as soon as possible, $\vec{\mu}$ is regarded as an optimization variable to minimize $W_c(\vec{\mu})$. We model the computation instance allocation as the following optimization problem:

$$\min_{\vec{\mu}} W_c(\vec{\mu}), \tag{1}$$

subject to:

$$\mu_j \geq \Lambda_j, j \in \mathcal{R}^C,$$

$$M_c(\vec{\mu}) \leq B^C,$$

where B^C is the budget for the allocation. We let $\mu_j \geq \Lambda_j, j \in \mathcal{R}^C$ so that the waiting time for a content to be processed is limited. $\mu_j \geq \Lambda_j, j \in \mathcal{R}^C$ indicates that we always allocate enough instances for the estimated volumes of contents generated by users, *i.e.*, $M_c(\{\Lambda_1, \Lambda_2, \ldots\}) \leq B^C$. The optimization is a convex programming, which can be efficiently solved by a general water-filling like algorithm: we iteratively allocate a small amount of resources to the computation instances within region k with the largest marginal time deduction, as follows:

$$k = \arg \min_{j \in \mathcal{R}^C} \frac{\partial W_c}{\partial \mu_j} = \arg \min_{j \in \mathcal{R}^C} \frac{-\Lambda_j}{\sum_{j \in \mathcal{R}^C} \Lambda_j (\mu_j - \Lambda_j)^2},$$

until the budget is used up. In Sec. 5, we will present how the algorithms are implemented to allocate computation instances dynamically.

4.3 Replication for Global Distribution

In the PaaS-based distribution platform, since both storage and upload capacity are charged according to the usage, our design is to determine which contents to be replicated to the distribution platform. When choosing processed contents to be replicated to the distribution platform, we select the widely-propagating contents that will be requested by users from many external regions, and let the computation instances serve other contents that are mostly requested by local users. The selection is based on not only the popularities of the contents, but also the social connections of the users generating these contents.

Let $\mathcal{S}^{(T)} = \{c_1, c_2, \ldots, c_S\}$ denote the set of processed contents served by the distribution platform in the time slot T, *i.e.*, users can download contents in $\mathcal{S}^{(T)}$ in time slot T. The distribution replication is then to determine the contents in $\mathcal{S}^{(T)}$.

In the content replication, there is also a budget B^D for the allocation of the distribution platform. Let $M_d(\mathcal{S}^{(T)})$ denote the cost when contents in $\mathcal{S}^{(T)}$ are served by the distribution platform. The cost includes both the storage and upload bandwidths, which can be formulated as follows:

$$M_d(\mathcal{S}^{(T)}) = P^S \sum_{c \in \mathcal{S}^{(T)}} A(c) + P^U \sum_{c \in \mathcal{S}^{(T)}} N(c),$$

where P^S is the unit storage price, P^U is the unit upload price, $A(c)$ is the size of the content c, and $N(c)$ is the amount of bytes to be uploaded to the users that are downloading the content from the distribution platform. $N(c)$ can be estimated as follows:

$$N(c) = v_c \sum_{i \in \mathcal{R}^U - \{R(c)\}} \left| \mathcal{F}_{u_c,i} \right|,$$

where u_c is the user who generate content c, $\mathcal{F}_{u_c,i}$ is the set of user u_c's friends that are located at region i, $R(c)$ is the region where content c is originally processed and served, and v_c is the average number of bytes served for the content. v_c can be estimated by an empirical value $\alpha A(c)$, where α is the average fraction of a video that users usually download [16]. The rationale of $N(c)$ is that the distribution platform will be in charge of uploading the content to u_c's friends that are located at external regions to reduce the download times.

The cloud distribution platform automatically replicates contents in $\mathcal{S}^{(T)}$ to different locations so that users can be better served. We design a content replication index $r(c)$ as follows:

$$r(c) = \sum_{i \in \mathcal{R}^U - \{R(c)\}} \left| \mathcal{F}_{u_c,i} \right| d_{i,R(c)},$$

where $d_{i,R(c)}$ is the geographic distance between region i and region $R(c)$. Larger $r(c)$ indicates that content c will be requested by more users from more external regions, and c should be replicated to the distribution platform for these users to download.

In our distribution platform allocation, we determine which contents to be replicated to the distribution platform by solving the following problem:

$$\max_{\mathcal{S}^{(T)}} \sum_{c \in \mathcal{S}^{(T)}} r(c), \tag{2}$$

subject to:

$$M_d(\mathcal{S}^{(T)}) \leq B^D,$$

$$\mathcal{S}^{(T)} \subset \mathcal{D}^{(T)},$$

where $\mathcal{D}^{(T)}$ is the set of candidate contents that can be downloaded by users in the future. In a social application, since users mainly request the contents recently generated by users, $\mathcal{D}^{(T)}$ can be formed from the contents recently processed by the computation instances.

The rationale of Eq. (2) is that we select the contents that can attract more users from more external regions. Such contents cannot be well served by only local computation instances. By replicating these contents to the distribution platform, which automatically replicates them to servers close to users, better download performance can be achieved for users that are not located closely to the original computation instances. The optimization can be heuristically solved by a dynamic programming algorithm in Sec. 5.

Next, we will discuss the implementation of the cloud-based social application deployment.

5. DISCUSSION OF SYSTEM IMPLEMENTATION

In this section, we discuss the details of our implementation. Our implementation is based on Amazon EC2 and Google AppEngine.

5.1 Computation Instance Allocation

We allocate the computation instances on Amazon EC2. The computation instance allocation algorithm is illustrated in Algorithm 1. Due to the limited number of regions where the computation instances can be allocated, the algorithm is carried out in a centralized manner periodically.

First, we collect the recent content generation rates from all the regions in $\mathcal{R}^U$, which are used to predict the current content generation rates $\lambda_i, i \in \mathcal{R}^U$. We assume the unit prices $P_j^H, j \in \mathcal{R}^C$ are also provided by the cloud provider. By solving the convex optimization problem in Eq. (1), we have the content processing rates $\mu_j, j \in \mathcal{R}^C$. According to the content processing rates, we allocate instances from EC2 — the processing rates determine the number and the model of the computation instances.

Second, the computation instances will receive and process the contents generated by users. At each region, a priority queue is utilized to store the contents uploaded by users. The contents are prioritized to be processed by the computation instances as follows: (1) Contents posted in the same region with the computation instance will be prioritized, and (2) contents are processed according to the timestamps they are uploaded.

5.2 Content Replication

According to our design illustrated in Sec. 4.3, the optimization can be solved using the dynamic programming algorithm, assuming that both the distribution price and budget can be regarded as positive integers. The replication procedure is illustrated in Algorithm 2. Contents in set $\mathcal{D}^{(T)}$ are the recently processed ones collected from all the computation instances, $M_d(\cdot)$ is the price function, and $r(\cdot)$ is the replication index function. We assume contents in $\mathcal{D}^{(T)}$ can be indexed from 1 to $|\mathcal{D}^{(T)}|$. Let $r(i,j)$ denote

Algorithm 1 Allocation of computation instances.

procedure ALLOCATION($\lambda_i^{(t)}, i \in \mathcal{R}^U, t = T-1, T-2, \ldots, T-M$, $P_j^H, j \in \mathcal{R}^C$)
  predict $\lambda_i, i \in \mathcal{R}^U$ using the predictive model
  $\mu_j \leftarrow \Lambda_j, j \in \mathcal{R}^C$
  while $M_c(\vec{\mu}) \leq B^C$ **do**
    $k \leftarrow \arg\min_{j \in \mathcal{R}^C} \frac{\partial W_c}{\partial \mu_j}$
    $\mu_k \leftarrow \mu_k + \Delta$
  end while
  allocate instances according to processing rates $\mu_j, j \in \mathcal{R}^C$
end procedure

Algorithm 2 Replication of processed contents.

procedure REPLICATION($\mathcal{D}^{(T)}$, $M_d(\cdot)$, $r(\cdot)$)
  for d **from** 0 **to** B^D **do**
    $\mathcal{S}(0,d) \leftarrow \Phi$
    $r(0,d) \leftarrow 0$
  end for
  for i **from** 1 **to** $|\mathcal{D}^{(T)}|$ **do**
    for j **from** 0 **to** B^D **do**
      if $j \geq M_d(\{c_i\})$ **and** $r(i-1,j) < r(i-1, j-M_d(\{c_i\}))$ **then**
        $\mathcal{S}(i,j) \leftarrow \mathcal{S}(i-1,j) \bigcup \{c_i\}$
        $r(i,j) \leftarrow r(i-1, j-M_d(\{c_i\})) + r(c_i)$
      else
        $\mathcal{S}(i,j) \leftarrow \mathcal{S}(i-1,j)$
        $r(i,j) \leftarrow r(i-1,j)$
      end if
    end for
  end for
  $\mathcal{S}^{(T)} = \mathcal{S}(|\mathcal{D}^{(T)}|, B^D)$
end procedure

the optimized replication gain of deploying the candidate contents indexed 1 to i under the budget j, $\mathcal{S}(i,j)$ denote the contents selected for replication. Using the dynamical programming algorithm, $\mathcal{S}(i,j)$ and $r(i,j)$ are iteratively updated, and the solution to our replication problem in (2) is then $\mathcal{S}^{(T)} = \mathcal{S}(|\mathcal{D}^{(T)}|, B^D)$.

After replication, for a content in $\mathcal{S}^{(T)}$, users are able to download it from either the computation instance where it is processed, or the distribution platform, to achieve the best download rate. Stale contents in the distribution platform are removed to make room for new ones in an LFU manner. After a content is removed from the distribution platform, users can still download it from the computation instance.

6. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our design based on a prototype of the example social application implemented on Amazon EC2 and Google AppEngine.

6.1 Experiment Setup

Social application system. Amazon EC2 provides computation instances at 7 regions given in Sec. 3. We have launched one micro VM instance[5] at each region, where we implement the content collection and processing modules using C++. Since we are using the free-tier micro VM nodes which have low computation capacities, to evaluate different content processing rates in Algorithm 1, the content processing in the prototype is simplified so that the number of processed contents is directly determined by the processing rates without actual computing load. The implementation can be easily replaced by other content processing algorithms for different social applications. The prices for Amazon EC2 micro instances are following the latest prices provided on the website[6]. For the content distribution, we implemented a Python-based distribution platform on Google AppEngine with 5GB storage capacity and 1GB outbound bandwidth per hour. In our experiments, the distribution budget B^D is determined by the storage and bandwidth limitations, *i.e.*, we replicate contents to the distribution platform under the storage and bandwidth capacities. The distribution uses the data storage APIs provided by Google AppEngine to accept contents uploaded from the computation instances and serve users.

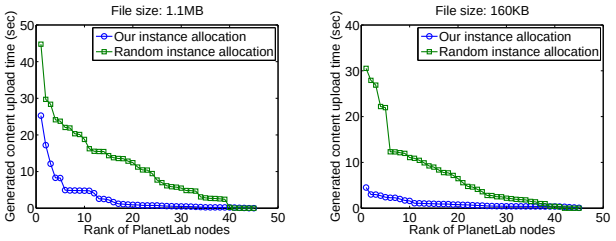
Users. We employ 57 PlanetLab nodes to upload and download contents according the traces from Tencent Weibo as follows: (1) In each round of the experiments, a set of 41 PlanetLab nodes are randomly selected and mapped to the 41 regions in $\mathcal{R}^U$; (2) The content generation rate of each PlanetLab node is determined by the traces, as used in our measurement in Sec. 3.3.3; and (3) After the contents are processed by the social application system, the nodes simulate to download the processed contents: followers of the users who have generated the contents will download the processed contents. The rationale is that it is highly possible for these followers to download the contents, and we use them to estimate the actual downloaders, though the number of total followers can be larger than the number of users who actually download the contents in real systems (*e.g.*, some users are never online to receive the contents).

Protocols. We present the practical protocols used to connect the users and the social application system. Contents are transferred as follows. (1) A user can upload a content to one of his local computation instances over TCP using private protocol; (2) If a processed content should be served by the distribution platform, the computation instance requests an uploading URL from the distribution platform, which is generated by the data storage API provided by Google AppEngine; (3) Using the upload URL, the content instance can upload the processed content by posting it to the given URL over HTTP; and (4) When downloading contents, a user is first provided with a XML file indicating where the contents can be downloaded, *i.e.*, either from a computation instance or the distribution platform. The user then downloads these contents from the computation instances or the distribution platform.

Records. In our experiments, we simulate two different content sizes for users to upload and download: 1.1 MB and 160 KB. Each PlanetLab node will record the time spent on uploading and downloading the contents. At each Amazon EC2 instance, we also implement the instance to record the time spent on processing each content. Based on these records, we evaluate the performance of content collection, processing and distribution in terms of the time spent on each task.

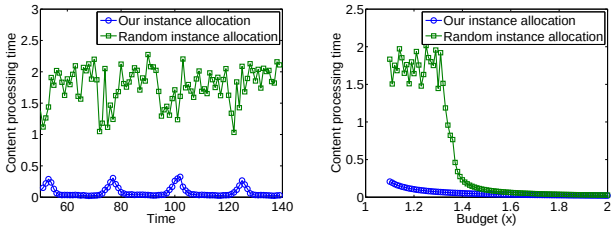
[5]http://aws.amazon.com/ec2/instance-types/

[6]May, 2012: http://aws.amazon.com/ec2/pricing/. California 0.025, Virginia 0.02, Oregon 0.02, Singapore 0.025, Ireland 0.025, Tokyo 0.027 and Sao Paulo 0.027 (USD per hour)



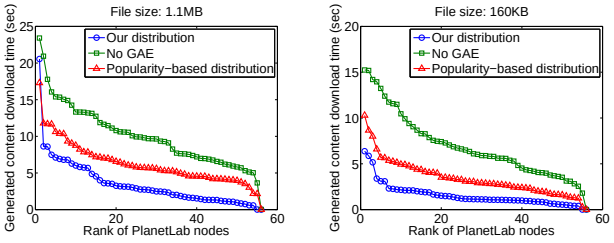
(a) file size 1.1 MB. (b) file size 160 KB.

Figure 15: Comparison of upload time in content collection.



(a) Processing time over time. (b) Processing time versus budget.

Figure 16: Comparison of processing time in content processing.



(a) file size 1.1 MB. (b) file size 160 KB.

Figure 17: Comparison of download time in content distribution.

6.2 Performance Evaluation

Content collection. First, we evaluate the performance for users to upload the generated contents to the social application system. We compare our instance allocation with a random instance allocation scheme, where processing rates at the instances are allocated randomly. The budget for both strategies is the same, $1.2\sum_{j\in\mathcal{R}^C} P_j^H \Lambda_j$. The rationale is that the social media company spends 20% more than the estimated demand on the allocation. We collect the average time each PlanetLab node spends on uploading the contents to the computation instances. Fig. 15 compares the upload times in the two allocation schemes. Each sample represents the average upload time at a node versus the rank of the node. We observe that it is much faster for users to upload contents in our design than in the random scheme — in our design, about 2/3 of the nodes can upload the contents with size 1 MB in less than 1 second, while most of the nodes have to spend more than 5 seconds to upload the same contents in the random scheme. The reason is that by taking the regional content generation rates into consideration, bandwidths can be allocated efficiently at the computation instances to satisfy users' uploading requests.

Content processing. Next, we evaluate the performance of content processing, in terms of the average processing time, which is defined as the average delay for a content to be available to users after it has been uploaded to the system. Fig. 16(a) compares our instance allocation with the random scheme over time, under the same budget of $1.2\sum_{j\in\mathcal{R}^C} P_j^H \Lambda_j$. The contents to be processed at a computation instance are the contents uploaded by users. We observe that the processing time in the random scheme is about 10 times larger than that in our design. The reason is that many contents have to wait a long time to be processed when being queued at a computation instance with a small processing rate in the random scheme. We also observe that in our design, the processing time is correlated with the content generation rate, *i.e.*, it takes longer for a content to be processed during the peak hours; while in the random scheme, the processing times are randomly distributed.

We next investigate the content processing performance under different budgets. Fig. 16(b) illustrates the processing time versus the budget $x\sum_{j\in\mathcal{R}^C} P_j^H \Lambda_j$. We observe that the processing time decreases when the budget is increased in both algorithms, since more computation resource is allocated for the generated contents; however, the processing time is much smaller in our design than in the random scheme when the budget is small (*e.g.*, $x < 1.5$), indicating that our design can benefit the social media companies when they have a limited budget. Both algorithms can achieve a small processing time when enough resource is allocated (*e.g.*, $x > 1.8$).

Content distribution. We also evaluate the performance for the distribution of the processed contents. Similarly, the PlanetLab nodes record the times spent on downloading the contents from both the computation instances and the distribution platform. We compare our design with two simple schemes: (1) No GAE strategy in which users only download the contents from the original computation instances; and (2) Popularity-based distribution where contents in $\mathcal{D}^{(T)}$ are replicated from the computation instances to the distribution platform according to only the popularities of the users who generate the contents, *i.e.*, a content is more likely to be replicated to the distribution platform if the user has more friends. Fig. 17 illustrates the download time versus the rank of the PlanetLab node. Again, we observe that our distribution replication achieves the lowest download times for almost all the PlanetLab nodes. We also observe that popularity-based replication achieves better distribution performance than the no-GAE scheme.

The experimental results indicate that by only allocating limited storage and outbound bandwidth (*e.g.*, the free-tier GAE platform in our experiments) at the distribution platform, widely-propagating social contents can be well served to global users.

7. CONCLUDING REMARKS

Large online social networks are providing Open APIs for developers to implement different social applications. It is promising for small social media companies to obtain user relationships and social actions without building a new social network. In this paper, we explore an efficient and economical cloud-based social application deployment after a social media company has developed their application. With measurement studies, we show that even if the social application

is attracting users globally, the propagation can be quite localized; and even if the propagation is highly dynamical for users and contents, the regional propagation patterns can be highly predictable. A local processing and global distribution design principle can be effectively used in cloud-based social application deployment. We developed a theoretical framework to design our algorithms for computation instance allocation and content replication, which are implemented in our prototype of an example social application on Amazon EC2 and Google AppEngine. The superiority of our design is confirmed by trace-driven experiments.

Acknowledgment

This work has been partially supported by the National Basic Research Program of China (973) under Grant No. 2011CB302206, the National Natural Science Foundation of China under Grant No. 60933013/61272231, the National Significant Science and Technology Projects of China under Grant No. 2012ZX01039001-003, and the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

8. REFERENCES

[1] http://aws.amazon.com/ec2/.

[2] http://code.google.com/appengine/.

[3] http://techblog.netflix.com/2012/01/auto-scaling-in-amazon-cloud.html.

[4] http://t.qq.com/.

[5] H. Akaike. Fitting Autoregressive Models for Prediction. *Annals of the Institute of Statistical Mathematics*, 21(1):243–247, 1969.

[6] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, 2010.

[7] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing User Behavior in Online Social Networks. In *Proc. of ACM IMC*, 2009.

[8] M. Cha, A. Mislove, and K. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proc. of ACM WWW*, 2009.

[9] X. Cheng and J. Liu. Load-Balanced Migration of Social Media to Content Clouds. In *Proc. of NOSSDAV*, 2011.

[10] N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman, and R. Wolski. AppScale: Scalable and Open AppEngine Application Development and Deployment. *Cloud Computing*, 34(2):57–70, 2010.

[11] N. Ellison et al. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[12] B. Furht and A. Escalante. *Handbook of Cloud Computing*. Springer-Verlag New York Inc, 2010.

[13] B. B. V. Gnedenko and I. Kovalenko. *Introduction to Queueing Theory*. Birkhauser, 1989.

[14] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. Zhao. Analyzing Patterns of User Content Generation in Online Social Networks. In *Proc. of ACM SIGKDD*, 2009.

[15] P. Hofmann and D. Woods. Cloud Computing: the Limits of Public Clouds for Business Applications. *Internet Computing*, 14(6):90–93, 2010.

[16] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang. Challenges, Design and Analysis of a Large-Scale P2P-VoD System. In *Proc. of ACM SIGCOMM*, 2008.

[17] B. Huffaker, M. Fomenkov, D. Plummer, D. Moore, and K. Claffy. Distance Metrics in the Internet. In *Proc. of IEEE International Telecommunications Symposium (ITS)*, 2002.

[18] A. Kaplan and M. Haenlein. Users of the World, Unite! the Challenges and Opportunities of Social Media. *Business horizons*, 53(1):59–68, 2010.

[19] I. Konstas, V. Stathopoulos, and J. Jose. On Social Networks and Collaborative Recommendation. In *Proc. of ACM SIGIR*, 2009.

[20] H. Kwak, C. Lee, H. Park, and S. Moon. What Is Twitter, a Social Network or a News Media? In *Proc. of ACM WWW*, 2010.

[21] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Comparing Public Cloud Providers. In *Proc. of ACM IMC*, 2010.

[22] M. Miller. *Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online*. Que, 2008.

[23] T. Mills. *Time Series Techniques for Economists*. Cambridge Univ Pr, 1991.

[24] J. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The Little Engine(s) That Could: Scaling Online Social Networks. *ACM SIGCOMM Computer Communication Review*, 40(4):375–386, 2010.

[25] Z. Rehman, F. Hussain, and O. Hussain. Towards Multi-Criteria Cloud Service Selection. In *Proc. of IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2011.

[26] B. Rimal, E. Choi, and I. Lumb. A Taxonomy and Survey of Cloud Computing Systems. In *IEEE International Joint Conference on INC, IMS and IDC*, 2009.

[27] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft. Track Globally, Deliver Locally: Improving Content Delivery Networks by Tracking Geographic Social Cascades. In *Proc. of ACM WWW*, 2011.

[28] D. A. Tran, K. Nguyen, and C. Pham. S-CLONE: Socially-Aware Data Replication for Social Networks. *Computer Networks*, 56(7):2001 – 2013, 2012.

[29] Z. Wang, L. Sun, X. Chen, W. Zhu, J. Liu, M. Chen, and S. Yang. Propagation-based Social-aware Replication for Social Video Contents. In *Proc. of ACM Multimedia*, 2012.

[30] Z. Wang, L. Sun, C. Wu, and S. Yang. Guiding Internet-Scale Video Service Deployment Using Microblog-Based Prediction. In *Proc. of IEEE INFOCOM Mini-Conference*, 2012.

[31] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. Lau. Scaling Social Media Applications Into Geo-Distributed Clouds. In *Proc. of IEEE INFOCOM*, 2012.