Profiling and Grouping Users to Edge Resources According to User Interest Similarity

Pengyuan Zhou
University of Helsinki, Finland
pengyuan.zhou@cs.helsinki.fi

Jussi Kangasharju
University of Helsinki, Finland
jussi.kangasharju@helsinki.fi

ABSTRACT

Cloud computing provides a shared pool of resources for large-scale distributed applications. Recent trends such as fog computing and edge computing spread the workload of clouds closer towards the edge of the network and the users. Exploiting the edge resources efficiently requires managing the resources and directing user traffic to the correct edge servers. In this paper we propose to profile and group users according to their interest profiles. We consider edge caching as an example and through our evaluation show the potential benefits of directing users from the same group to the same caches. We investigate a range of workloads and parameters and the same conclusions apply. Our results highlight the importance of grouping users and demonstrate the potential benefits of this approach.

CCS Concepts

•**Networks** → **Cloud computing;** *Network resources allocation; Traffic engineering algorithms;*

Keywords

edge caching; profiling; user interest; grouping

1. INTRODUCTION

Cloud computing provides a shared pool of resources tackling large-scale distributed applications. With the introduction of SDN, NFV, CDNs, etc., controlling functions are becoming centralized while service and data are pushed towards the edge. These technologies allow for the provision of flexible and easily configurable control functions, and furthermore improve scalability and availability of data and services near the users. Recent approaches, such as fog computing [2,5,23] and edge computing [7], attempt to formalize the structure of how resources at the edge can be exploited for data and service provision.

A major concern is the efficiency of use of edge network resources. As argued in [3], clouds in cloud computing have different properties and users and applications have different requirements. Hence, it may be hard to profile the cloud with a single explicit resource provisioning policy; besides, most related researches focus on top-down solutions or ignore features and requirements of end users. For instance, SDN and NFV provide high level network control functions mostly in the cloud. CDNs distribute content towards end user without analyzing the characteristics and profiles of users. To realize the full potential of edge network resources, we also need to profile user requests. The analysis and profiling of user request can help us free up network resources for a more targeted provisioning policy. Although users have heterogeneous and dynamic quality of service requirements, their interests are likely to be relative steady. Users normally take a long time to develop an interest which would last a long time in most cases; these interests change very slowly, at least relative to the frequency of incoming requests. Nowadays the popularity of recommender system may also accelerate the formulation of user interest and steady it. As studied in [1] and [24], recommender system can influence user preferences. According to the anchoring theory, user choice can be heavily influenced by the first piece of information offered [21]. Based on the above, we believe user interests can act as a means of request profiling.

In this paper, we choose content delivery as an example of a cloud-backed service to illustrate the benefits of profiling users based on their interest preferences. According to the forecast of Cisco, IP video traffic will be 80 percent of all IP traffic by 2020 [20]. Also, the continued expansion of social networks brings lots of user-generated content. Content delivery is the main cloud-backed network services today and not only dedicated CDN providers like Akamai but also large companies like Google and Facebook are running CDNs.

CDNs are built around caches towards which user requests are directed. Research in the past has focused on caching hierarchies, cache replacement algorithms, and cache partitioning, but since these do not take any stand on how users are directed, they must act in a user-agnostic manner. Normally, users are directed to the closest cache in order to minimize network latency and traffic. However, users in the same area might have very different interest profiles, e.g., based on age, education, hobbies, and normal re-direction mechanisms are unable to capture these. All content requested by the users competes for space in the cache. From an overall system efficiency point of view, this approach has two shortcomings:

1. This may be unfair for non-mainstream users. Their requests have a smaller chance to be cached at the

CAN'16, December 12 2016, Irvine, CA, USA

DOI: http://dx.doi.org/10.1145/3010079.3010081

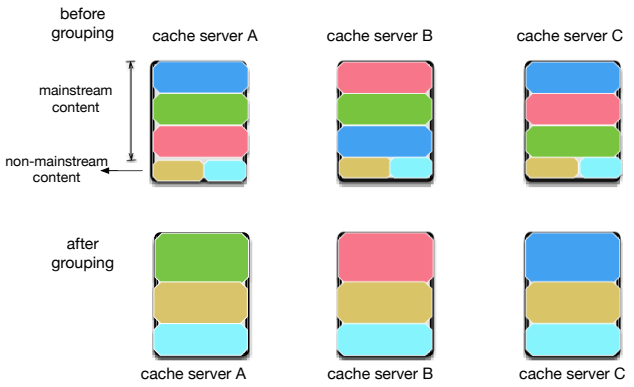


Figure 1: Difference of cache server with or without grouping users

edge because of lower popularity. The requested content may still be found in the parent cache with higher latency.

2. There can be a lot of duplicates of same contents in different cache servers. It results in waste of cache servers resource.

This motivates us to propose profiling requests and re-directing users based on interest similarity. This solution profiles traffic on a higher level of abstraction, comparing with profiling based on individual specific contents. Figure 1 shows an example where servers A, B, C and D cache lots of same contents since users in different areas share same interests. With current content popularity based cache strategy, the edge servers are more likely to cache mainstream contents. Since the cache size is limited, non-mainstream users have less chance to utilize the edge cache servers.

Considering the reasons above, we propose grouping users according to interest similarity to improve cache performance and efficiency. As shown in the lower part of Figure 1, if we can group users with similar interest and redirect their requests to same cache servers, we can utilize cache space more efficiently. Note that Figure 1 shows the ideal case of perfect profiling, which is likely to be hard to implement in practice. In this paper we focus on this ideal case to highlight the potential benefit of grouping users and leave the exact grouping methods for future work. More specifically, we make the following contributions:

- We propose a novel profiling strategy, where requests are grouped according to interest similarity.
- Cache servers receive similar requests with higher probability. This increases cache hit rates and reduces redundant copies in caches.
- After grouping, cached popular content can serve more users which saves cache resources and additional resources could be offered to non-mainstream users.

In this paper, we only evaluate the improvement of cache performance brought by grouping. Defining the practical grouping mechanisms is left for future work. The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 describes the network architecture. Section 4 presents the simulation model. Section 5 presents the results and discusses their impact. Finally, Section 6 concludes the paper and presents directions for future work.

2. RELATED WORK

Researchers have proposed some work related with edge computing. Some work such as [3] also provides ideas regarding cloud computing resource provisioning. Work in [17] and [11] put forward elastic or dynamic resource scaling schemes. However, profiling edge computing resources based on user interest similarity is still an open issue. Although researchers have proposed user interest recognition for some years, most of related work still remain on the level of identifying individual users. For instance, [14] introduces identifying users' preferences based on click history. Work in [12] proposes an aggregation of user profiles from multiple domains on social web. Some work has already addressed large scale user interest analysis such as [24]. However, that work is more about relation between overall user behavior and popularity of videos. On the other hand, some work also propose analyzing user interest such as [25], which put forward to merge user profile for better recommendation system. Authors in [9] characterize user viewing behavior and use Mixed Integer Programming to place the segments of videos optimally. A more similar work was proposed in [6]. However, the authors use one abstract function to represent the user interest similarity. Similarly, [4] indicates user interest similarity with a variable. The most important difference between these and our work in this paper is that *we evaluate several concrete parameters of user interest similarity to better understand this problem.*

As to content delivery, CDNs such as Akamai use caching overlays to distribute content more efficiently [18]. The major feature of a CDN is distributed servers and cache hierarchy. Information-centric networking (ICN) proposes, among other aspects, to exploit in-network caching to enable more efficient content distribution by addressing content via a unique name.. Some work has been done focusing on reducing in-network caching redundancy and improving caching efficiency in ICN. ProbCache approximates the caching capacity of a path [13] . Guo et al. proposed a collaborative caching guided by content popularity rank [8]. In [15], the authors evaluate the performance of in-network caching and conclude that content popularity is one of the most important parameters.

As outlined above, Most of related proposals focus on utilizing caches based on content popularity. There is not much work referring to leveraging grouping users according to interest similarity for better cache performance in CDN, ICN, or other networks. Currently, we believe that the following, still unanswered, questions need attention:

1. Would grouping users provide benefits? In this paper, we show that (in the ideal situation) grouping users improves overall cache hit ratio markedly.
2. How to implement user grouping, including pattern recognition and clustering? We need find out the optimistic algorithm for recognizing user interest patterns and clustering them.
3. How to optimize cache deployment and redirection of user requests? How to balance of the benefit of grouping users and the latency brought by redirection.

Table 1: Network topologies used in the study

Topology	Source nodes	Routers	Receivers
Tiscali	44	160	1636
Garr	13	27	291
Geant	13	32	328
Datacenter	1	16	160

In this paper, we focus on identifying the *ideal benefit* of grouping users regarding cache hit ratio, i.e., the first of the three open questions above; others are left for future work. We consider situations of different user interest distribution and evaluate the difference in cache hit ratio. This paper is the first to consider explicitly the influence of different user interest profiles and how they impact caches near the edge of the network. Previous works have mainly focused on optimizing the performance of the cache but have not considered re-direction of users based on their interest profiles.

3. NETWORK TOPOLOGY

In this paper we use various network topologies to study the effects of edge cache deployment and user grouping on the edge caches. Edge cache deployment makes it easier to understand the different cache performance in situations with different workload and grouping methods. It also helps isolate the benefit of grouping user from other scheme such as in-network caching etc.

We use a self-defined topology "Datacenter" and several real topologies including "Tiscali", "Garr" and "Geant". "Datacenter" is a two-layer network which has one core node serving as origin content source, 16 edge routers serving as edge cache servers and 10 users connecting to each edge server. Garr and Geant are parsed from the Internet Topology Zoo dataset [10] and Tiscali is parsed from Rocketfuel dataset [19]. Some adaption is made to the topologies according to the assignment of simulation. For instance, we add 10 users to each edge servers.

The adaption helps us focus on the influence of grouping users on cache performance regardless of other factors such as different number of users connecting to cache servers. The properties of the topologies are summarized in Table 1. The cache servers are chosen based the centrality of the routers.

4. SIMULATION

We used Icarus as the simulator for evaluating the performance of grouping. Icarus is an ICN simulator for evaluating cache performance [16]. For the user interest distribution function, we use the widely accepted Zipfian distribution [22] as follows:

$$f(k;\alpha,N) = \frac{\frac{1}{k^{\alpha}}}{\sum_{n=1}^{N}\frac{1}{n^{\alpha}}} \quad (1)$$

where frequency of content k out of a population of N contents is $f(k;\alpha,N)$. N is the number of overall contents and k is the ranking of the content. We also use k as the ID of content here. Our simulation concerns several parameters as follows:

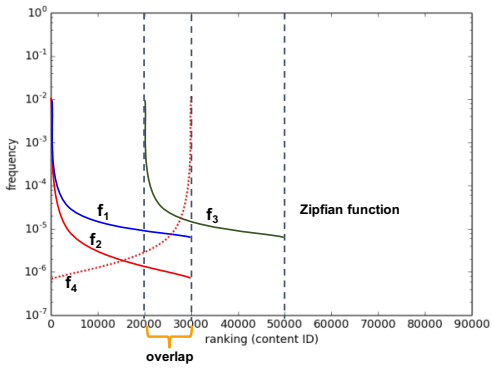


Figure 2: Example interest distributions

- α is the value of the exponent characterizing the interest distribution. In the simulations, we set α as 0.8,1.0,1.2.
- *rank* indicates the integer interval of content IDs that each user requests for. In the simulations, we set the interval as a population of $3 * 10^5$ content.
- *rank_similarity* indicates the overlap ratio of different ranks. In the simulations, we set the default *rank_similarity* as 0, so that users with different *rank* would have entirely different requests.
- *rank number* indicates the number of different popularity rankings. In the simulations, we set the default *rank_num* as 7, so that workloads before grouping have 7 different ranks of interest distribution. In other words, there are by default 7 different types of user profiles in the simulation.
- *cache_size* indicates the total size of network cache as a fraction of content population [16]. The default *cache_size* is set to 0.001.
- *workload* indicates the user request distributions with different grouping schemes.

Figure 2 shows an example of how the different ranks and rank similarities are implemented. We show 4 example interest distributions, f1–f4. The x-axis shows the content items ranked according to their popularity (possibly different for each interest distribution). The y-axis shows the number of requests generated by the interest distribution for each content item. Since there are a total of 4 different popularity rankings, the *ranknumber* is 4. Curves f1 and f2 share the same popularity ranking and their *rank_similarity* is 1 since they cover the exact same range of content. However, they have different α values since the actual popularities of the objects are different. Likewise, f4 has a *rank_similarity* of 1 with both f1 and f2, but it counts as a different *rank* since the popularity of the objects is inverted. Requests from f3 overlap with f1, f2, and f4 by 33.3% so their *rank_similarity* would be 0.33. Since the shapes of f1 and f3 are identical, they share the same α. Using this kind of a model as a basis, we generate the various workloads used in the simulation.

Our goal is to investigate the effects of the various parameters (α, *rank_similarity*, and *rank number*) on the performance of the caches in the network. As a first step we focus

Table 2: Workload definitions

Workload	*rank number*	*Num_of_α*	*Num_of_distribution*
R	1,3,5 or 7	1	1,3,5 or 7
R-GR	1(locally)	1(locally)	1(locally)
AR	1,3,5 or 7	6	6,18,30 or 42
AR-GR	1(locally)	6	6
AR-GAR	1(locally)	1(locally)	1(locally)

on evaluating only cache hit rate, but other caching metrics, e.g., byte hit rate, latency, could also be used. The above model is flexible enough to support a variety of scenarios. The workloads we consider are as follows:

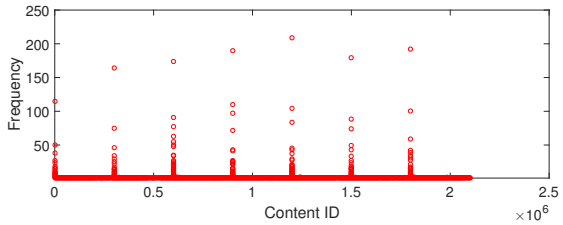
- **R**: Users have interest distributions with different *rank* but same α. *rank_similarity* is 0 which means user groups have entirely different interests. The number of groups is determined by the parameter *rank* but users are not grouped in any way.
- **R-GR**: Grouping users with same *rank* in **R** workload After grouping, the users connecting to same cache servers would have same interest distribution.
- **AR**: Users have interest distributions with different *rank* and different α, but no grouping of users is performed.
- **AR-GR**: Grouping users with same *rank* in **AR** workload, but a group may contain users with different values of α.
- **AR-GAR**: Grouping users with same *rank* and same α in **AR** workload, i.e., each group has a specific *rank* and α.

The default *rank_similarity* is set to 0. We also run simulations with different *rank_similarity* to identify the effects of it. Most of the simulations have different *rank number* which is at most 7. The *cache_size* is set as [0.001, 0.003, 0.005, 0.007, 0.009, 0.01]. The number of users connecting to each cache server does not change after grouping.

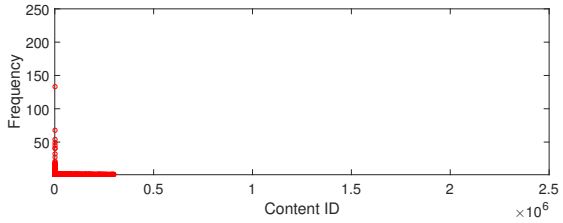
Table 2 shows the detail of the workloads. The word "locally" indicates the interest distributions of the users connecting to same cache servers. The different simulations aim at identifying the benefit of grouping users regarding cache hit ratio in different scenarios. For instance, in **R** and **AR** workload, we tried different numbers of *rank*. It can help us identify the factors influencing the benefit of grouping users. Since we keep each cache node serving the same number of users as before grouping, there are still some cache servers connecting to users with different interest after grouping. We think this is more close to reality due to the capacity of cache servers because it is hard to actually put all users with same interest under a same cache server and this reflects some inaccuracies in the grouping.

5. RESULTS

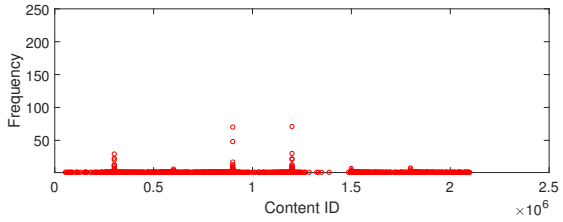
The results are shown in Figure 3 separately for each of the workloads. We show one figure per scenario (**R**, **R-GR**, **AR**, **AR-GR**, **AR-GAR**) and each figure shows the requests observed by *a single cache* in the scenario. The



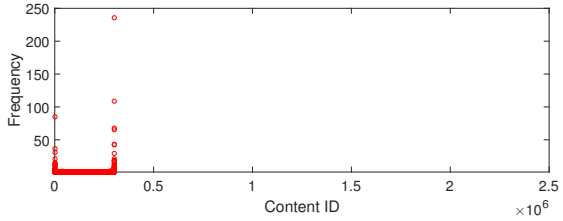
(a) Request distribution with workload **R**, in which there are 1 α and 7 *rank* as per Table 2



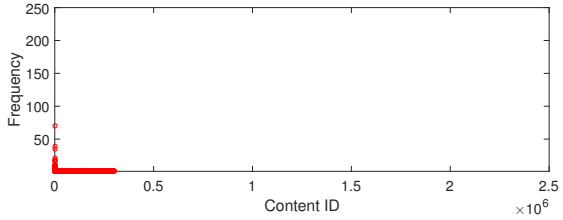
(b) Request distribution with workload **R-GR**, which is the request distribution for a cache server after grouping Figure 3a. There is 1 α and 1 *rank* as per Table 2



(c) Request distribution with workload **AR**, in which there are 7 α and 7 *rank* as per Table 2



(d) Request distribution with workload **AR-GR**, which is the request distribution for a cache server after grouping Figure 3c according to *rank*. There are 7 α and 1 *rank* as per Table 2



(e) Request distribution with workload **AR-GAR**,which is the request distribution for a cache server after grouping Figure 3c according to both α and *rank*. There is 1 α and 1 *rank* as per Table 2

Figure 3: Request distribution in different workload

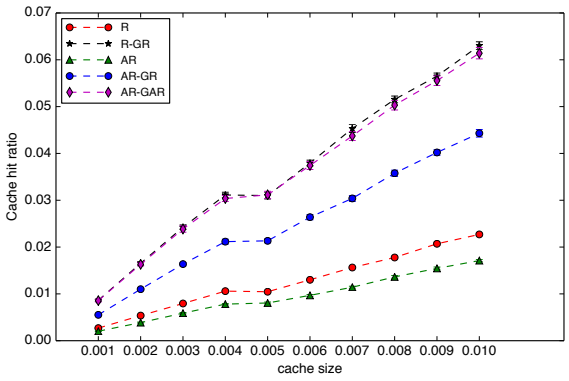


Figure 4: Cache hit rate vs. cache size.

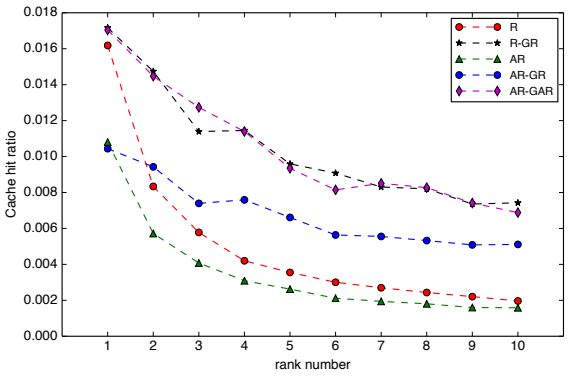


Figure 5: Cache hit ratio vs rank number.

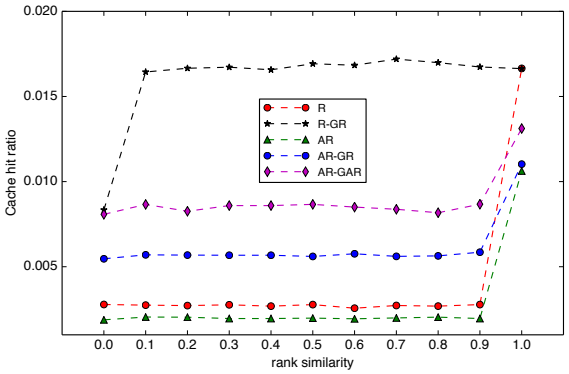


Figure 6: Cache hit ratio vs rank similarity.

x-axis shows the content IDs in the whole workload and the y-axis shows the number of requests received by that cache in the simulation. As we can see, grouping users helps significantly narrow down the range of requests that a cache sees. For instance, in Figures 3a and 3b we clearly see the effects of various popularity rankings in Figure 3a as spikes in requests. When users are grouped according to their interests, the request pattern is much close to a single zipf-distribution, as shown in Figure 3b. Likewise, Figure 3e has a more concentrated request distribution than Figure 3d and Figure 3c, as expected. Given that we use *a priori* knowledge to assign groups perfectly, the results reflect an ideal situation, but they clearly illustrate the potential of this method.

The above results show the effect of grouping on the traffic workloads of the caches, and we now turn to evaluating the performance of the cache as a function of the various parameters described in Table 2. Although the figures below show results for all workloads in the same figure, the exact numbers can only be compared among (**R**, **R-GR**) and (**AR**, **AR-GR**, and **AR-GAR**) because of the different α values in the experiments. The exact hit rate numbers are not as important as the relative differences in performance in the different workloads.

The key findings can be summarized as follows:

1. *cache_size*: As Figure 4 shows, the cache hit rate increase with bigger cache size, as is to be expected. As we see, **AR-GR** and **AR-GAR** improve the cache hit rate compared with **AR**; **R-GR** likewise improves over **R**. This result shows that grouping users can benefit cache hit rate independent of cache size.

2. *rank number* : As Figure 5 shows, cache hit rates of all workloads decrease when *rank number* increases. This is expected since more ranks (i.e., more sets of interest distributions) will bring more different request distributions. However, grouping serves to alleviate the effect and remains effective even with a large number of different ranks. Note that the cache size used in Figure 5 is 0.001, i.e., the smallest in our study. Larger cache sizes exhibited the same kind of behavior.

3. *rank_similarity*: As mentioned before, the default *rank_similarity* is set to 0 in most simulations. We also investigated the effects of overlapping interests by varying the similarity from 0% to 100%. As Figure 6 shows, *rank_similarity* does not influence cache hit ratio much for most of the cases. Two notable observations are evident in the figure. First, all except **R-GR** show a jump at the end when similarity reaches 100%. The explanation is that when similarity reaches 100%, the distributions become identical (except for different α) and therefore all caches see the same traffic. Another is the jump in **R-GR** when similarity increases beyond 0%. While we do not have a full explanation for this phenomenon, we conjecture the reason to be related to way we have implemented similarity by sliding the request distributions over one another, as Figure 2 shows.[1] Understanding the reasons behind this and investigating it thoroughly are left for future work.

6. CONCLUSION AND FUTURE WORK

In this paper we have considered the problem of grouping users for more efficient request processing in edge caching scenarios. Our simulations show that grouping users can improve cache hit rate in many different scenarios. We have investigated various grouping strategies and results have consistently shown that when user interest profiles are different from each other, grouping users of one interest profile into one edge cache yields considerable benefits in terms of overall cache performance. Although our work was done under the assumption of an ideal distribution of user groups to caches, it highlights the potential for improvement in caching by this simple technique.

[1]We have verified the simulation code and ruled out errors in there.

Future work needs to tackle three main issues. First, we have assumed that any user can be re-directed to any cache, regardless of the locations of the two. While in principle this is feasible to do, it may result in significant increases in user-perceived latency. A more realistic look at which clients could be re-directed to which caches could be included as part of the work. Second, it is unlikely that the number of user groups is smaller than the number of edge caches, thus one cache may be forced to serve multiple user groups with different interest profiles. An obvious solution would be to attempt to assign user groups that have similar interest profiles in the same caches. Third, we have not considered the practical implementations of how the groups are formed. Various existing classification and clustering algorithms can most likely be used to achieve the groupings and evaluating their efficiency is part of our future work.

7. ACKNOWLEDGMENTS

The work for this paper was performed in the context of the EU FP7 Marie Curie Actions project Grant Agreement No. 607584 (the Cleansky project)

8. REFERENCES

[1] Adomavicius, G., Bockstedt, J. C., Curley, S. P., and Zhang, J. Do recommender systems manipulate consumer preferences? a study of anchoring effects. *Information Systems Research 24*, 4 (2013), 956–975.

[2] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of Workshop on Mobile Cloud Computing* (2012), pp. 13–16.

[3] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience 41*, 1 (2011), 23–50.

[4] Chen, Z., and Kountouris, M. Cache-enabled small cell networks with local user interest correlation. In *2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications* (2015), IEEE, pp. 680–684.

[5] CISCO. Fog computing and the internet of things: Extend the cloud to where the things are (whitepaper).

[6] ElBamby, M. S., Bennis, M., Saad, W., and Latva-Aho, M. Content-aware user clustering and caching in wireless small cell networks. In *2014 11th International Symposium on Wireless Communications Systems* (2014), IEEE, pp. 945–949.

[7] Garcia Lopez, P. et al. Edge-centric computing: Vision and challenges. *SIGCOMM Comput. Commun. Rev. 45*, 5 (Sept. 2015), 37–42.

[8] Guo, S., Xie, H., and Shi, G. Collaborative forwarding and caching in content centric networks. In *International Conference on Research in Networking* (2012), Springer, pp. 41–55.

[9] Hwang, K. W., et al. Leveraging Video Viewing Patterns for Optimal Content Placement. *Springer Berlin Heidelberg* (2012), pp. 44–58.

[10] Knight, S., Nguyen, H. X., Falkner, N., Bowden, R., and Roughan, M. The internet topology zoo. *IEEE Journal on Selected Areas in Communications 29*, 9 (2011), 1765–1775.

[11] Kosta, S., Aucinas, A., Hui, P., Mortier, R., and Zhang, X. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *IEEE INFOCOM* (2012), pp. 945–953.

[12] Orlandi, F., Breslin, J., and Passant, A. Aggregated, interoperable and multi-domain user profiles for the social web. In *Proceedings of the 8th International Conference on Semantic Systems* (2012), pp. 41–48.

[13] Psaras, I., Chai, W. K., and Pavlou, G. Probabilistic in-network caching for information-centric networks. In *Proceedings of ICN Workshop on Information-centric Networking* (2012), pp. 55–60.

[14] Qiu, F., and Cho, J. Automatic identification of user interest for personalized search. In *Proceedings of the 15th International Conference on World Wide Web* (2006), pp. 727–736.

[15] Rossi, D., and Rossini, G. Caching performance of content centric networks under multi-path routing (and more). *Tech. Report, Telecom ParisTech* (2011).

[16] Saino, L., Psaras, I., and Pavlou, G. Icarus: a caching simulator for information centric networking (icn). In *Proceedings of ICST Conference on Simulation Tools and Techniques* (2014).

[17] Shen, Z., Subbiah, S., Gu, X., and Wilkes, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of ACM Symposium on Cloud Computing* (2011), ACM, p. 5.

[18] Sitaraman, R. K., Kasbekar, M., Lichtenstein, W., and Jain, M. Overlay networks: An akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services* (2014), 305–328.

[19] Spring, N., Mahajan, R., and Wetherall, D. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review 32*, 4 (2002), 133–145.

[20] Systems, C. White paper: Cisco vni forecast and methodology, 2015-2020. White Paper, 2016.

[21] Wikipedia. Anchoring, 2016. https://en.wikipedia.org/wiki/Anchoring [Online; accessed 6-September-2016].

[22] Wikipedia. Zipf's law, 2016. https://en.wikipedia.org/wiki/Zipf%27s_law [Online; accessed 6-September-2016].

[23] Yannuzzi, M., Milito, R., Serral-Gracia, R., Montero, D., and Nemirovsky, M. Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing. In *IEEE CAMAD Workshop* (2014), pp. 325–329.

[24] Yu, H., Zheng, D., Zhao, B. Y., and Zheng, W. Understanding user behavior in large-scale video-on-demand systems. In *Proceedings of EuroSys 2006*, pp. 333–344.

[25] Yu, Z., Zhou, X., Hao, Y., and Gu, J. Tv program recommendation for multiple viewers based on user profile merging. *User modeling and user-adapted interaction 16*, 1 (2006), 63–82.