

Exploiting Correlations in Request Streams: A Case for Hybrid Caching in Cache Networks

Adita Kulkarni, Anand Seetharam

Department of Computer Science, SUNY Binghamton, USA
akulka17@binghamton.edu, aseethar@binghamton.edu

Abstract—With the increasing popularity of cache networks, in recent years, multiple static and dynamic caching strategies have been proposed that seek to improve user-level performance. Most existing caching strategies rely heavily on assumptions such as content popularity following a well-known Zipfian distribution and request streams following an Independent Reference Model (IRM). In this paper, we consider multiple real-world user request stream traces to investigate the validity of these assumptions and observe that they do not hold true. We conduct a detailed factor analysis and observe that violation of the IRM assumption significantly impacts the performance of caching strategies. We identify the interplay between the skewness of the content popularity distribution and the request stream correlation among unpopular content as the key factor impacting performance. We identify that in the high popularity skewness-low correlation regime, static caching outperforms dynamic caching, while the reverse is true in the low popularity skewness-high correlation regime. For the high popularity skewness-high correlation regime, static and dynamic caching provide similar performance. For this scenario, we propose *Hybrid Caching* that effectively combines static and dynamic caching strategies. The main idea is to split the cache into two parts—a static cache that statically caches content based on popularity and a dynamic cache that exploits the correlation in request streams. We conduct experiments on multiple real-world networks (e.g., WIDE, GEANT, GARR) and demonstrate that *Hybrid Caching* outperforms static or dynamic caching alone in the high popularity skewness-high correlation regime.

I. INTRODUCTION

In recent years, content caching at storage-enabled network nodes has attracted widespread attention, with multiple static and dynamic caching strategies being proposed that seek to improve user-level performance. Benefits of caching content within the network has been demonstrated for content distribution networks, information-centric networks, heterogeneous cellular networks and MANETs [3], [8], [9], [10]. Existing caching strategies can be classified into two groups—*i*) static caching or content placement that determines the optimized/optimal set of content to be placed at network nodes, *ii*) dynamic caching where each network node decides what content to insert and evict from its cache based on content that passes through it.

Existing work on designing content caching strategies rely heavily on two underlying assumptions—*i*) content popularity follows a well-known Zipfian distribution, and *ii*) user requests are generated according to an Independent Reference Model (IRM). Additionally, most caching strategies have been primarily tested on synthetic request streams generated based on

the above assumptions, and hence, their performance on real-world request streams is unknown.

In this paper, we consider publicly available user request stream traces from Wikipedia [17], YouTube [20], and the WeBrowse project [2], and investigate the validity of the Zipfian distribution and IRM assumptions. By conducting the Kolmogorov-Smirnov goodness of fit test, we conclude that the Zipfian distribution assumption does not hold true. We also observe that the IRM assumption may not always be valid, with the degree of correlation in request streams varying from one trace to the other. In the traces considered, we observe a strong correlation between requests for the same piece of content in the YouTube and WeBrowse traces, while the correlation is minimal in the Wikipedia trace.

We then conduct a factor analysis on a simple three-node network to study the impact of the Zipfian and IRM assumptions on performance of static and dynamic caching strategies. Our investigation reveals that though the Zipfian distribution assumption has an impact on the absolute performance numbers, the relative ordering of static and dynamic caching strategies does not change. In contrast, the IRM assumption has a profound impact on performance. Depending on the extent of correlation present in the request stream, particularly among the unpopular content, static and dynamic caching strategies can outperform each other.

We identify the interplay between the content popularity skewness and request stream correlation among unpopular content as the key factor affecting performance. We define content popularity skewness as the asymmetry in the content popularity distribution and use the percentage of requests for popular content as its measure. We use the percentage of requests for unpopular content occurring in close proximity as a measure of correlation. We observe that if the number of requests for popular content is high and the correlation among requests for unpopular content is low (i.e., high popularity skewness-low correlation regime), static caching outperforms dynamic caching. On the other hand, in the low popularity skewness-high correlation regime, dynamic caching outperforms static caching. Interestingly, in the high popularity skewness-high correlation regime, we observe that static and dynamic caching provide similar performance.

For the high popularity skewness-high correlation scenario, we propose *Hybrid Caching*, a caching strategy that captures the best of both static and dynamic caching. The main idea is to split the cache into two parts—a static cache that statically

caches content based on popularity and a dynamic cache that exploits the correlation in request streams. From our experiments, we observe that a small dynamic cache efficiently takes advantage of the correlation in the request stream. Based on our investigation, we recommend that network caches adopt static, dynamic and *Hybrid Caching* in the high popularity skewness-low correlation, low popularity skewness-high correlation and the high popularity skewness-high correlation regimes, respectively.

We conduct experiments on multiple real-world topologies (e.g., GEANT, WIDE and GARR) on the Icarus simulator [16], a simulator designed primarily for cache networks. We demonstrate that our recommendations perform the best in their respective settings for different networks. We demonstrate that *Hybrid Caching* consistently outperforms both static and dynamic caching in the high popularity skewness-high correlation scenario.

II. BACKGROUND AND PROBLEM STATEMENT

In this section, we first discuss related work and outline its limitations. We then describe the key questions that this work seeks to answer.

A. Related Work

As mentioned in Section I, existing research on content caching can be divided into two main categories—*i)* static caching (also referred to as content placement) and, *ii)* dynamic caching. Static caching strategies focus on determining the set of content to be cached at network nodes so as to maximize performance [4], [8]. In [4], the authors consider a general cache network and propose Greedy Caching, an optimized caching strategy that maximizes performance. In [8], the authors consider hybrid networks consisting of femtocell caches and a cellular infrastructure, and design joint routing and content placement strategies to reduce delay. They show that the problem of data caching and routing is NP-hard and propose greedy approximation algorithms to this end.

In contrast to static caching, in dynamic caching, each network node decides whether to cache a particular piece of content or not as it passes through it. If a node decides to cache a particular piece of content, it has to determine what content to evict from the cache. Hence, dynamic caching strategies focus on both content insertion and eviction aspects. The simplest cache insertion policy is the Leave Copy Everywhere (LCE) policy, where nodes cache all content passing through them [13]. More sophisticated policies (e.g., Leave Copy Down [14], Cache Less for More [7] and ProbCache [15]) aim at improving the content diversity in the network. For example, Cache Less for More (CL4M) [7] leverages the concept of centrality to determine the node most suited to cache content en route between the source and user. Latency-aware Caching (LAC) [5] leverages local latency measurements, while CACS [3] takes network congestion into account while making caching decisions.

The most widely used cache eviction policies are First in First out (FIFO), RANDOM, Least Recently Used (LRU),

and Time-to-live (TTL). Several variations of LRU caching (e.g., p-LRU, where content is inserted into a cache with probability p , with the eviction policy being LRU) have also been proposed [12]. Analytical approaches for modeling the performance of TTL based caches and selecting appropriate TTL values to make TTL caches emulate the behavior of other cache eviction policies have also been proposed [11].

B. Limitations

Most existing work [4], [6], [9], particularly those related to static caching explicitly rely on the assumption that content popularity follows a well-known Zipfian distribution with skewness parameter α . Static caching strategies primarily exploit the heavy tail of the Zipfian distribution and cache the most popular content in the network. Therefore, static caching strategies cannot seamlessly adapt to changing content popularity. Additionally, prior work assumes that content requests are generated according to an Independent Reference Model (IRM) (i.e., there is no correlation in the request stream). As such, static caching strategies fail to take advantage of correlations that are likely to exist in a real-world request stream.

To overcome these limitations of static caching, dynamic cache insertion and eviction policies have been proposed. However, dynamic caching strategies possess a different set of drawbacks. As they do not explicitly take content popularity into account, dynamic caching strategies sometimes end up evicting popular content from caches, thus adversely affecting their overall performance. Moreover, existing static and dynamic caching strategies have been mainly tested on synthetic traces generated based on the Zipfian and IRM assumptions. Therefore, their performance on real-world traces is still a relatively unexplored problem.

C. Problem Statement

To investigate the impact of the above-mentioned assumptions on the performance of static and dynamic caching strategies, we ask the following questions.

- 1) Do the Zipfian distribution and the IRM assumptions hold in real-world request stream traces?
- 2) If these assumptions are not upheld in real-world request streams, what is the individual impact of these assumptions on performance?
- 3) Is it possible to discern attributes of real-world request streams that can identify regimes where one type of caching strategy (static or dynamic) is likely to perform the best?
- 4) Are there any regimes where neither static nor dynamic caching necessarily performs the best? In these regimes, is it possible to design a caching policy that captures the best of both worlds?

Our investigation in Section III answers the first question in the negative. Therefore, to answer the second question, in Section IV, we conduct a detailed factor analysis to study the individual impact of the two assumptions. Our analysis shows that while the Zipfian distribution assumption impacts the absolute performance of the caching strategies, the relative

ordering of strategies is preserved. Violation of the IRM assumption in request streams can drastically impact performance, with static and dynamic caching outperforming each other depending on the extent of correlation.

We identify the interplay between content popularity skewness and correlation among unpopular content as the main factor dictating performance. Our experiments reveal that high popularity skewness and low correlation favors static caching while the reverse setting favors dynamic caching. We also observe that in the high content popularity skewness-high correlation regime, both strategies show similar performance. Therefore, in this regime, we propose *Hybrid Caching* that captures the best of both worlds by effectively dividing the cache into static and dynamic components (Section V).

III. ZIPFIAN DISTRIBUTION AND IRM ASSUMPTIONS

In this section, we consider three real-world request stream traces to investigate the validity of the Zipfian distribution and IRM assumptions.

- 1) *Wikipedia Access Trace* [17] contains 10% of all user requests issued to Wikipedia from September 2007 to January 2008. For our experiments, we consider a portion of this trace from September 2007 consisting of 1,000,000 requests. This trace does not contain any client information.
- 2) *YouTube Access Trace* [20] from the University of Massachusetts Amherst was collected by monitoring YouTube traffic at the campus gateway router between June 2007 and March 2008. The trace has 1,467,700 requests from 39852 users.
- 3) *WeBrowse Access Trace* [2] was collected by the support of the mPlane project. The trace collection period is one hour long and contains logs of HTTP requests observed by the Tstat probe installed at the egress link of the campus network. This is a smaller trace consisting of 148,180 total requests from 1875 users.

As streaming videos currently account for majority of the Internet traffic [1], we consider the YouTube request stream trace. But, we also study the WeBrowse and Wikipedia traces because http traffic also consists of requests for static webpages and web objects and the performance of these webpages can be significantly improved by adopting in-network caching. We note that though the WeBrowse trace is considerably smaller in comparison to the YouTube and Wikipedia traces, the WeBrowse trace consists of significant number of requests and hence is useful in evaluating caching strategies and drawing meaningful conclusions.

We make several interesting observations from these real-world traces (Table I). First, in the traces considered, a significant portion of content is requested only once (between 54 - 85%). We note that for content requested only once, neither caching strategy can obtain any hits. Second, significant number of content is requested twice (between 8 - 18%) and thrice (between 2 - 13%). Third, we observe that some pieces of content are significantly more popular than majority of the content. Overall, we observe that the content popularity distribution has a long-tail.

We use the Kolmogorov-Smirnov goodness of fit test (i.e., KS test) to determine if these request streams fit a Zipfian distribution. The MATLAB function used for the test is *kstest2*, which is a variant of the classic KS test. In the *kstest2*, two data vectors are provided and the null hypothesis used is the following—both data vectors are generated from the same distribution. To understand how *kstest2* is used here, let us consider one of the request stream traces (say Wikipedia trace). We then synthetically generate a dataset that fits a Zipfian distribution considering the same content universe size as seen in the Wikipedia trace. We then provide both these data vectors to *kstest2* to determine whether the null hypothesis is accepted or rejected at any reasonable level of significance.

Trace	Unique Content	Content Requested Once	Content Requested Twice	Content Requested Thrice
Wikipedia	292332	81%	11%	3%
YouTube	581527	54%	18%	13%
WeBrowse	70948	85%	8%	2%

TABLE I: Trace Statistics

One of the issues encountered is that to generate a synthetic data vector that fits a Zipfian distribution, one needs to specify the skewness parameter α . As the value of α determines the synthetic distribution and thus impacts the result of the KS test, we generate different synthetic data vectors by varying the value of α from 0 to 2 in intervals of 0.01. We then compare each real-world trace to these different synthetically generated data vectors via the KS test. We observe that the KS test rejects the null hypothesis at 1% and 5% levels of significance for all values of α for all traces. From this experimentation, we conclude that though content popularity has a long-tail, it does not strictly follow a Zipfian distribution.

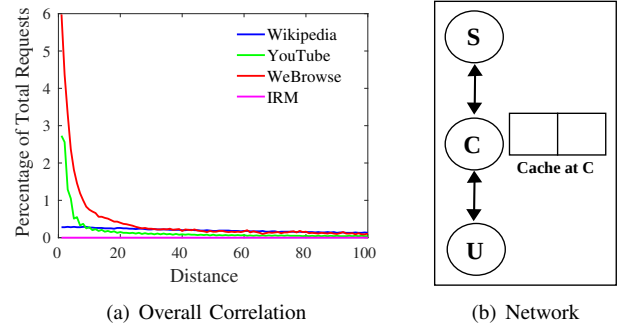


Fig. 1: Request Stream Correlation and Example Network

We next investigate the validity of the IRM assumption in these traces. For understanding this assumption, we only consider content that is requested more than once. In Figure 1(a), we study the percentage of total requests versus the number of requests after which the same content is requested (denoted as distance). We observe from the figure that the Wikipedia trace exhibits a low amount of correlation. In comparison, in the YouTube and WeBrowse traces, we observe that requests for the same piece of content occur in close proximity of one another, suggesting a high correlation in the

request stream. We note that an IRM trace results in a straight line parallel to the distance axis at value 0.

IV. FACTOR ANALYSIS

In the previous section, we observed that the Zipfian distribution and IRM assumptions do not necessarily hold true in real-world request stream traces. Having made this observation, in this section, we investigate the impact of these individual assumptions on the performance of static and dynamic caching strategies. We begin our study on a three-node network as shown in Figure 1(b) consisting of a single user U , a cache C and the custodian S . For this simple network, the optimal static caching policy is to statically cache the most popular content [9]. For dynamic caching strategies, we warm up the cache using approximately 20% of the total requests in each trace and use the remaining requests for actual performance measurement. For dynamic caching, we assume that the eviction policy is LRU. As there is only a single cache in the network, this setting helps us understand the key performance differences between static caching and LRU based dynamic caching. We vary the cache size (i.e., total number of content a node can cache) from 25 to 900. We note that studying this small network helps eliminate the impact of other factors such as routing, the quality of network links, the placement of custodians and cache insertion policy on performance.

A. Understanding Impact of Zipfian Assumption

To study the impact of the Zipfian distribution assumption, we first generate synthetic traces that fit a Zipfian distribution (say S_1). We next empirically compute the distribution of requests from the real-world request stream traces and generate an IRM request stream (say S_2) from the empirical distribution. We compare the performance of static and dynamic caching on S_1 and S_2 respectively. Note that this factor analysis isolates the impact of the Zipfian distribution assumption as both S_1 and S_2 obey the IRM assumption. To investigate the impact of the IRM assumption on performance, we compare the performance of static and dynamic caching on S_2 and the real traces (say S_3). Note that this comparison isolates the impact of the IRM assumption because both S_2 and the real traces have the same distribution and only differ in the degree of the correlation between requests.

Figures 2, 3 and 4 show the hit rate versus individual node cache size for static and dynamic caching for synthetic Zipfian IRM traces, empirically distributed IRM traces and real-world traces respectively. In Figure 2, synthetic Zipfian IRM traces are generated for α values of 0.6 and 1.0 respectively. Similar to previous work [4], we observe that static caching outperforms dynamic caching for a Zipfian distribution. From Figure 3, we also observe that static caching outperforms dynamic caching for synthetic IRM requests generated from the empirical distribution. This is expected because prior work [9] has shown that for the network considered in Figure 1(b), static caching provides optimal performance for IRM traffic generated from a known popularity distribution. Interestingly,

we observe that the actual performance numbers obtained for the empirical IRM traces differ significantly from those obtained for the Zipfian IRM traces.

B. Understanding Impact of IRM assumption

We next study the impact of the IRM assumption on performance. Figure 4 shows the hit rate for the real-world Wikipedia, YouTube and WeBrowse traces. In comparison to Figure 3, we observe that while static caching outperforms dynamic caching for the Wikipedia trace, the performance is reversed for the YouTube trace. For the WeBrowse trace, both static and dynamic caching show similar performance, with static caching marginally outperforming dynamic caching. This result demonstrates that the IRM assumption in request stream traces can strongly impact the conclusions drawn about the performance of caching strategies.

We hypothesize the interplay between content popularity skewness and correlation in the request streams to be the main reason behind the disparate performance of caching strategies in Figure 4. Skewness of the content popularity distribution is likely to aid static caching as larger percentage of requests can be satisfied from the cache, while increased correlation in the request stream is likely to aid dynamic policies. Figure 5(a) shows the total number of requests for top ten popular content while Figure 5(b) shows the correlation in request streams for unpopular content for the different traces. We refer to content that is requested between 2 and 5 times as unpopular content.

Analyzing Figures 1(a) and 4 together, we observe that dynamic caching works well for traces that exhibit higher correlation. However, an important thing to note is that while correlation in the request stream favors dynamic caching, if this correlation is primarily contributed to by the popular pieces of content, then dynamic caching may not be able to outperform static caching because static caching will also be able to serve these requests. In contrast, if the correlation is mainly contributed to by relatively unpopular content, then while dynamic caching will be able to serve these requests, static caching will incur a miss for them. Therefore, it is necessary to study the correlation between unpopular content (Figure 5(b)).

We next explain the performance of static and dynamic caching for the Wikipedia, YouTube and WeBrowse traces in Figure 4. From Figures 5(a) and 5(b), we observe that the total number of requests for popular content is high and the correlation is low for the Wikipedia trace. This explains the superior performance of static caching for the Wikipedia trace. In comparison, low content popularity skewness and high correlation among unpopular content is responsible for the superior performance of dynamic caching over static caching for the YouTube trace. We observe that in the YouTube trace the individual most popular videos do not encounter more than 0.2% of the requests, which indicates why the performance is biased towards dynamic caching. For the WeBrowse trace, we observe that both the content popularity skewness and the unpopular content correlation is high, which results in static and dynamic caching having similar performance.

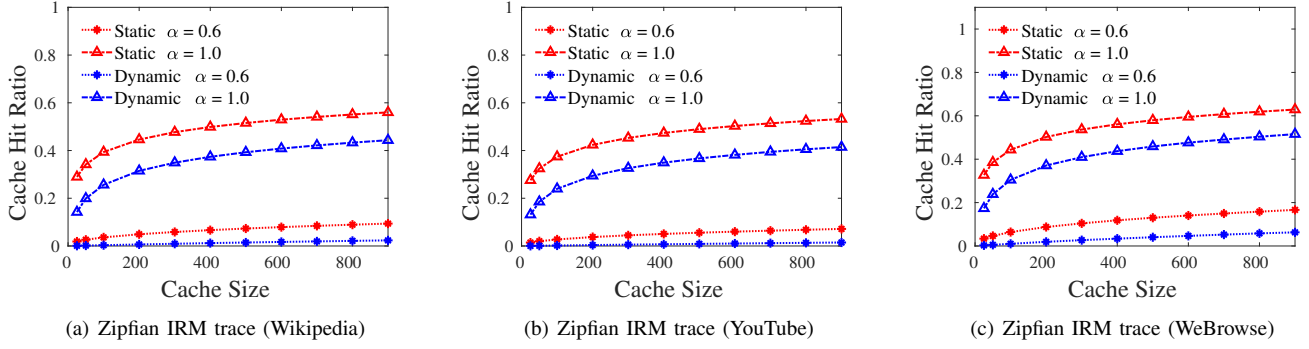


Fig. 2: Performance of IRM Traces generated from Zipfian distribution

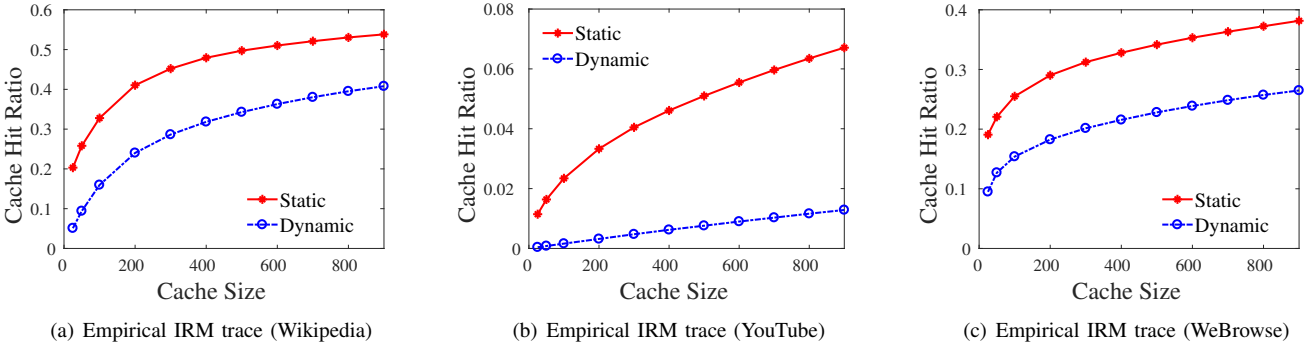


Fig. 3: Performance of IRM Traces generated from empirical distribution

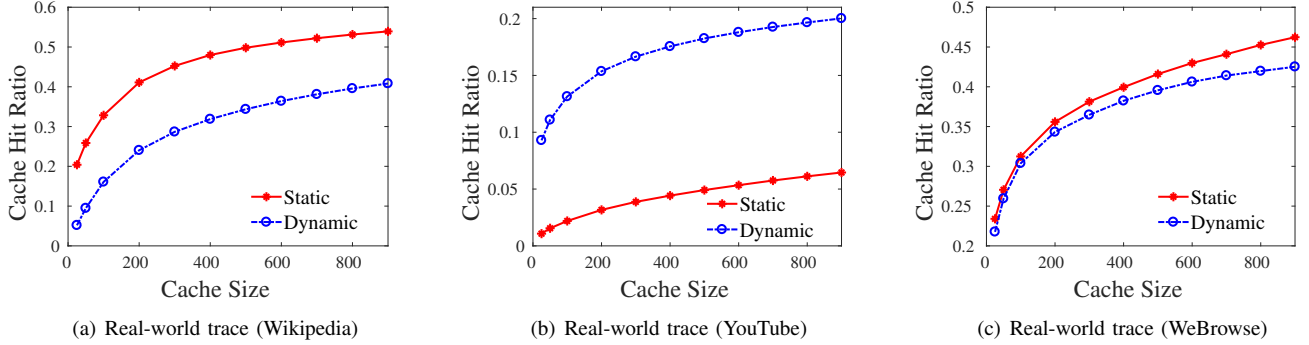


Fig. 4: Performance of Real-World Traces

To demonstrate that the arguments made above hold true, we next analyze the performance of dynamic and static caching with respect to unpopular, medium popular and popular content in Figure 6 for the network in Figure 1(b) for cache size 100. In Figure 6, we group content based on the total number of requests (say x) they obtain in the request stream. We sort these groups according to the decreasing order of x . In this sorted order, we consider those pieces of content falling in the top 5% as the most popular content. The subsequent two 5% content popularity brackets in this list fall in the 5% to 10% and 10% to 15% popularity range respectively. As mentioned earlier, we consider all pieces of content having value of x between 2 to 5 as unpopular content. We exclude groups of content with value of $x = 1$, as they never obtain a cache hit. We consider all remaining pieces of content to fall in the medium popularity range.

As there is low correlation in the Wikipedia trace, we observe from Figure 6(a) that dynamic caching obtains limited number of hits for the unpopular content. In comparison, we observe from Figures 6(b) and 6(c) that dynamic caching exploits the correlation present in request streams and obtains a significant number of hits for unpopular and medium popular content. As expected, static caching only secures hits for the popular content.

C. Recommendations

Based on discussions in the preceding subsection, we make the following recommendations as to which caching strategy is likely to perform the best depending on the popularity skewness and correlation for unpopular content (Table II). For the high popularity skewness-low correlation (i.e., Wikipedia trace) and the low popularity skewness-high corre-

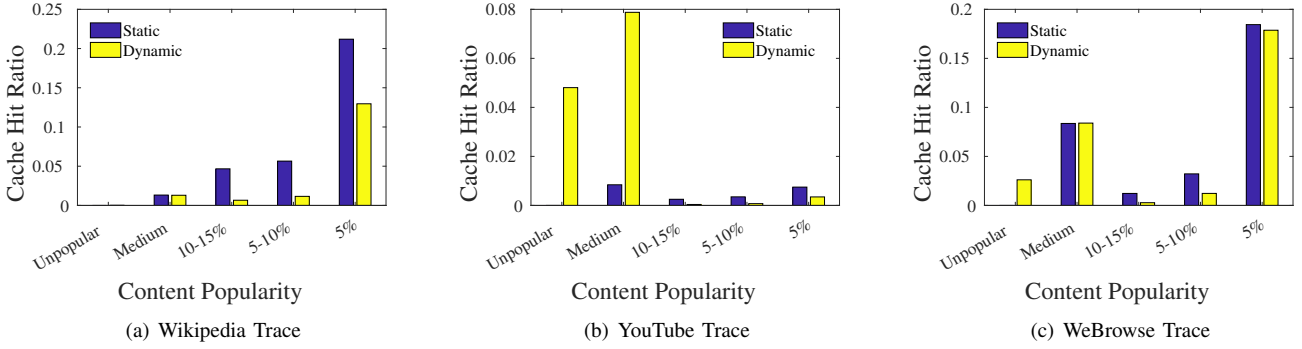


Fig. 6: Static Caching and Dynamic Caching Gain

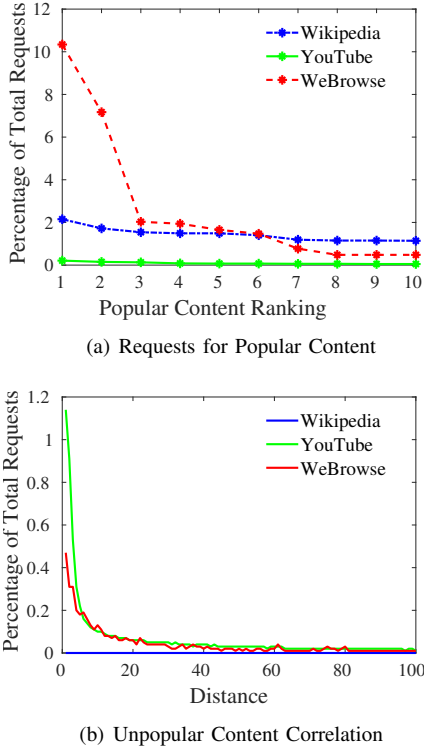


Fig. 5: Understanding Interplay of Popularity and Correlation

lation (i.e., YouTube trace) regimes, we recommend adopting static caching and dynamic caching respectively. In contrast, high popularity skewness-high correlation results in a strong competition between static and dynamic caching as evidenced in Figure 4(c) for the WeBrowse trace.

Trace	Correlation unpopular content	Total requests for popular content	Caching
Wikipedia	Low	High	Static
YouTube	High	Low	Dynamic
WeBrowse	High	High	Hybrid
Uniform Distribution	Low	Low	Does not matter

TABLE II: Recommendations

For the high popularity skewness-high correlation regime,

we recommend a hybrid caching strategy that captures the best of dynamic and static caching by dividing each network cache into static and dynamic cache components. We describe the proposed *Hybrid Caching* in detail in Section V. We note that the low popularity-low correlation regime will correspond (in the extreme case) to content requests being generated identically and independently according to a uniform distribution. In such a setting, there may be limited advantage to caching content (either statically or dynamically).

V. HYBRID CACHING

After identifying and analyzing the key scenarios that work in favor of static and dynamic caching, in this section, we propose *Hybrid Caching*, a caching strategy that combines the best of both static and dynamic caching to provide superior performance in the high popularity skewness-high correlation regime. In *Hybrid Caching*, each network cache is divided into two parts—one part operates as a static cache and stores content based on popularity, while the other operates as a dynamic cache and evicts content based on the LRU policy. We note that content that is already statically cached is never inserted into the dynamic cache. Though prior work [18], [19] has explored a concept similar to hybrid caching, we note that past work focuses mainly on proactively caching content for mobile settings, in addition to caching the most popular content.

A. Greedy Caching and Static-Dynamic Cache Split

Before proceeding further, there are two important questions that need to be answered. The first question is—what is the optimal static caching strategy for a general cache network that maximizes performance (e.g., hit rate)? To the best of our knowledge, this is still an unsolved problem. In [4], the authors propose Greedy Caching, a static caching strategy that determines the optimized set of content to be cached at network nodes. Greedy Caching takes content popularity, the graph connectivity and the miss stream from downstream nodes into account while making caching decisions at upstream nodes. The authors show that Greedy Caching works well in a variety of different settings. Therefore, in this paper, the static caching strategy we consider for a general network is Greedy Caching. For a network operating under *Hybrid*

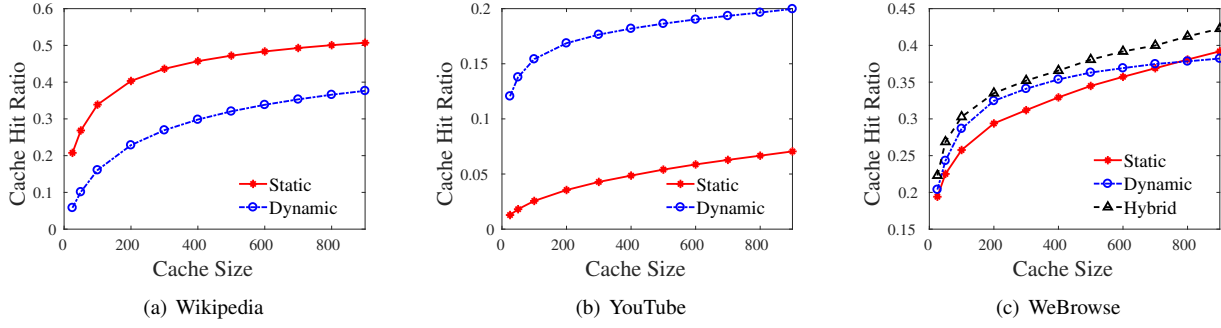


Fig. 7: Hit rate for GARR topology (single custodian)

Caching, we use Greedy Caching to determine the content to be cached in the static portion of the caches. The dynamic portion of all caches operate under the LRU caching eviction policy. The insertion policy studied in this paper is the Leave Copy Everywhere (LCE) policy.

The second question is—what is the static-dynamic cache split at network nodes for *Hybrid Caching*? To determine a cache's split into its dynamic and static components, we revisit Figures 1(a) and 5(b). We observe from Figures 1(a) and 5(b) that after a distance of 20, the correlation in the request stream decreases considerably and becomes constant. This means that a small dynamic cache can efficiently exploit majority of the correlation present in request streams. Therefore, to leverage the correlation in the request stream, we hypothesize that a dynamic cache size of 20 should be sufficient. We show in the next section that this is indeed the case and that *Hybrid Caching* can outperform static and dynamic caching alone in the high popularity skewness-high correlation regime. An important question that arises is—how to optimally split a cache into static and dynamic components that adapts seamlessly to network conditions? Determining the optimal split is challenging as it depends on a number of factors including the network topology, routing, exact correlation in the request stream, content popularity and its variation over time, and cache insertion and eviction policies, and is beyond the scope of this work.

We note that *Hybrid Caching* can be adapted to work seamlessly as static or dynamic caching based on network conditions. The recommendations in Table II can be fulfilled with *Hybrid Caching* by adjusting the static-dynamic cache split. To transform *Hybrid Caching* to purely static or dynamic caching, we only need to set the dynamic and static portions of the cache to size 0 respectively.

VI. EXPERIMENTAL RESULTS

In this section, our goal is to demonstrate via extensive experimentation that the recommendations made in Table II hold true in general cache networks. We first describe the setup and then present simulation results. We conduct experiments on the Icarus simulator [16], a simulator designed for cache network research. We perform experiments on multiple real-world network topologies—WIDE (Japanese academic network) consisting of 30 nodes with 17 users, GEANT

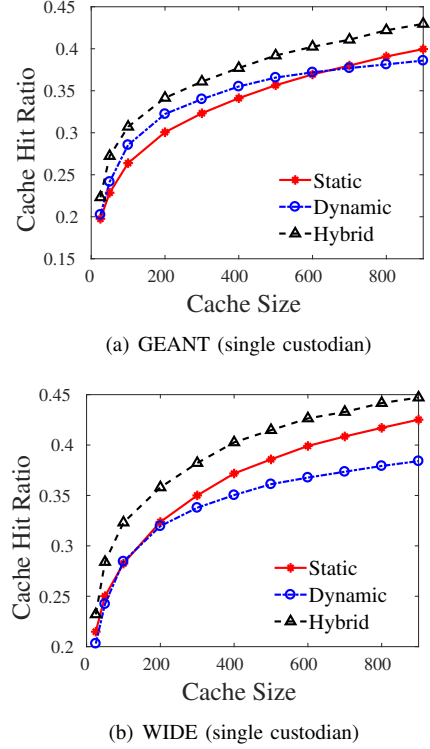


Fig. 8: WeBrowse: Hit rate for single custodian case

(European academic network) consisting of 39 nodes including 8 users and GARR (Italian national computer network for universities and research) consisting of 61 nodes with 21 users.

We once again consider the YouTube, Wikipedia and WeBrowse request stream traces. As the number of users in the YouTube and WeBrowse traces is significantly higher than the number of users in the topologies considered, we divide the entire trace into different subsets of users and map each subset to one user in the simulator. As the Wikipedia trace does not contain any client information, requests are randomly generated from users in the simulator.

We conduct experiments for the single custodian and multiple (2 and 5) custodian scenarios. In the simulator nodes with the highest degree are considered as custodians. If multiple nodes have same degree, we randomly select custodians from them. We select nodes with degree 1 as users. We warm up

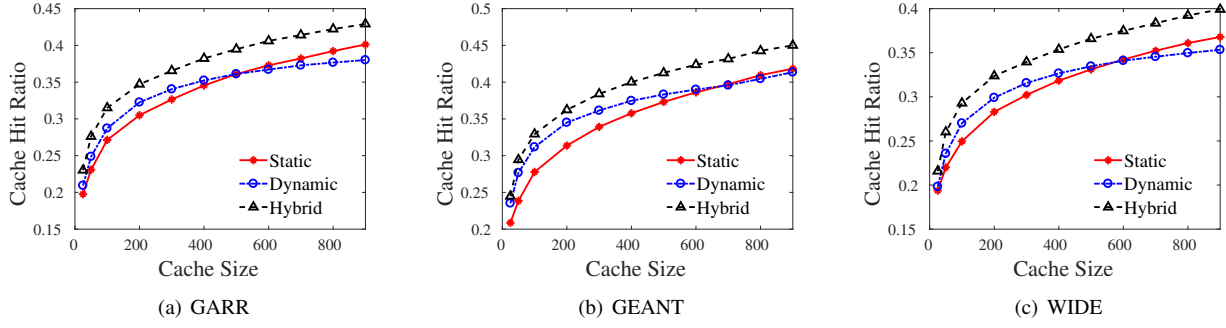


Fig. 9: WeBrowse: Hit rate for multiple (five) custodian case

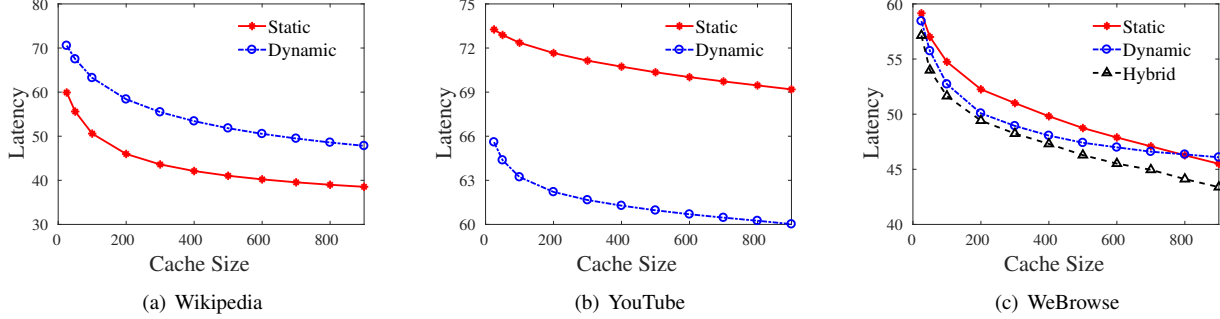


Fig. 10: Latency for GARR topology (single custodian)

all caches prior to our evaluation with approximately 20% of the total requests. As the real-world request traces are of a fixed and limited size, we use synthetic empirically distributed IRM traces to warm up the caches. We once again vary the cache size at individual nodes from 25 to 900. Cache sizes greater than 1000 are not realistic for the traces considered here, as one can readily cache most of the content universe apart from the relatively unpopular content. We consider the cache hit rate and delay as the primary performance metrics in our experiments. A request is considered to be a cache hit if it is served by any intermediate network cache apart from the custodian.

A. Cache Hit Ratio

Figure 7 shows the performance of the different caching strategies for the GARR topology. As expected, the cache hit rate increases as cache size increases. We observe that static caching outperforms dynamic caching for the Wikipedia trace while the reverse is true for the YouTube trace. We also observe that *Hybrid Caching* which captures the best of both static and dynamic caching outperforms them for the WeBrowse trace. Overall, we observe from the figure that our recommendations in Table II hold true. Note that we do not plot *Hybrid Caching* in Figures 7(a) and 7(b), because in line with our recommendations, it does not outperform both static and dynamic caching in the high popularity skewness-low correlation and low popularity skewness-high correlation regimes respectively.

We next focus on the performance of *Hybrid Caching* for the WeBrowse trace. Figure 8 shows the performance of the

various strategies for the GEANT and WIDE networks for the single custodian case, while Figure 9 demonstrates their performance for the multiple custodian case. We once again observe from the figures that *Hybrid Caching* performs the best, outperforming static and dynamic caching individually for all cache sizes. We note that in figures corresponding to Figures 8 and 9 for Wikipedia and YouTube traces, static and dynamic caching perform the best respectively. We omit these figures due to lack of space.

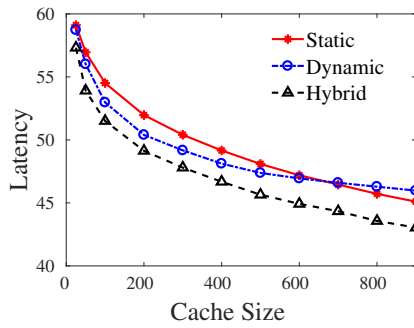
B. Latency

Figure 10 demonstrates the latency performance of the caching strategies for the different traces for the GARR topology. In Figure 11, we investigate their performance for the WeBrowse trace for the GEANT and WIDE topologies. Similar to hit rate, we observe that recommendations in Table II are valid for latency as well. We also observe that *Hybrid Caching* outperforms both static and dynamic caching for the WeBrowse trace for all network topologies.

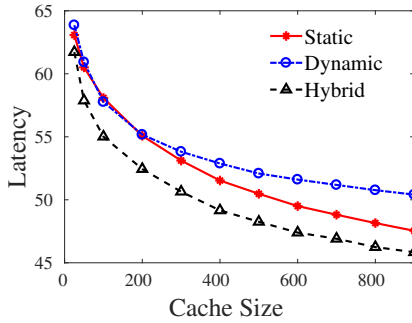
Overall, our experiments show that static and dynamic caching provide the best performance in the high popularity skewness-low correlation and low popularity skewness-high correlation regimes. More importantly, they show that *Hybrid Caching* outperforms both static and dynamic caching in the high popularity skewness-high correlation regime.

VII. CONCLUSION

In this paper, we demonstrated that widely accepted assumptions such as content popularity following a Zipfian distribution and request streams following an Independent



(a) GEANT (single custodian)



(b) WIDE (single custodian)

Fig. 11: WeBrowse: Delay for single custodian case

Reference Model (IRM) do not hold true in real-world request streams. We identified the interplay between the content popularity distribution skewness and the request stream correlation, particularly among unpopular content as the key factor impacting performance. We identified that in the high popularity skewness-low correlation regime, static caching outperforms dynamic caching, while the reverse is true in the low popularity skewness-high correlation regime. For the high popularity skewness-high correlation regime where static and dynamic caching provide similar performance, we proposed *Hybrid Caching* that effectively combines static and dynamic caching strategies. Via experimentation, we demonstrated that *Hybrid Caching* outperforms static or dynamic caching in the high popularity skewness-high correlation regime.

REFERENCES

- [1] "Cisco visual networking index: Forecast and methodology, 2016 - 2021," <https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf>.
- [2] "Webbrowse trace," <http://tstat.polito.it/traces-webbrowse.shtml>.
- [3] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-Aware Caching and Search in Information-Centric Networks," in *ACM ICN*, 2014, pp. 37–46.
- [4] B. Banerjee, A. Seetharam, and C. Tellambura, "Greedy caching: A latency-aware caching strategy for information-centric networks," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, June 2017, pp. 1–9.
- [5] G. Carofiglio, L. Mekinda, and L. Muscariello, "Lac: Introducing latency-aware caching in information-centric networks," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*. IEEE, 2015, pp. 422–425.
- [6] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1357–1370, Oct 2009.
- [7] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," in *IFIP Networking*, 2012, pp. 27–40.
- [8] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1635–1648, 2017.
- [9] M. Dehghan, A. Seetharam, T. He, T. Salonidis, J. Kurose, and D. Towsley, "Optimal caching and routing in hybrid networks," in *Military Communications Conference (MILCOM)*. IEEE, 2014, pp. 1072–1078.
- [10] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 147–158.
- [11] N. Fofack, P. Nain, G. Neglia, and D. Towsley, "Performance evaluation of hierarchical TTL-based cache networks," *Elsevier Computer Networks*, vol. 65, pp. 212–231, June 2014.
- [12] M. Garretto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 1, no. 3, p. 12, 2016.
- [13] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.
- [14] N. Laoutaris, H. Che, and I. Stavrakakis, "The lcd interconnection of lru caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.
- [15] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *ACM ICN*, 2012, pp. 55–60.
- [16] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a caching simulator for information centric networking (ICN)," in *SimuTools*, 2014, pp. 66–75.
- [17] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, vol. 53, no. 11, pp. 1830–1845, July 2009, http://www.globule.org/publi/WWADH_comnet2009.html.
- [18] X. Vasilakos, V. A. Siris, and G. C. Polyzos, "Addressing niche demand based on joint mobility prediction and content popularity caching," *Computer Networks*, vol. 110, pp. 306 – 323, 2016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1389128616303334>
- [19] F. Zhang, C. Xu, Y. Zhang, K. K. Ramakrishnan, S. Mukherjee, R. Yates, and T. Nguyen, "Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture," in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2015, pp. 1–9.
- [20] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network—measurements, models, and implications," *Computer networks*, vol. 53, no. 4, pp. 501–514, 2009.