Query-load balancing in structured overlays[*]

Anwitaman Datta
School of Computer Engineering
Nanyang Technological University (NTU, Singapore)
anwitaman@ntu.edu.sg

Roman Schmidt, Karl Aberer
School of Computer and Communication Sciences (I&C)
Ecole Polytechnique Fédérale de Lausanne (EPFL, Switzerland)
roman.schmidt@epfl.ch, karl.aberer@epfl.ch

Abstract

Query-load (forwarding and answering) balancing in structured overlays is one of the most critical and least studied problems. It has been assumed that caching heuristics can take care of it. We expose that caching, while necessary, is not in itself sufficient. We then provide simple and effective load-aware variants of the standard greedy routing used in overlays, exploiting routing redundancy originally needed for fault-tolerance, to achieve very good query load-balancing.
Keywords: *Query-load balancing, Structured overlays*

1. Introduction

In recent years the concept of *structured overlays* (e.g., Distributed Hash Tables [DHTs]) has attracted a lot of attention because of its potential to become a generic substrate for internet scale applications. Structured overlays are used for applications as diverse as locating resources in a wide area network or a grid computing environment in a decentralized manner. It can also be used for address independent and robust and flexible (group) communication - e.g., application layer multicast and internet indirection infrastructure and content distribution network to name a few. The basic function of a structured overlay is to act as a decentralized index. To that end, for each resource, a globally unique identifier (called the key) is generated using some function suitable to the applications that are supposed to use the index. The codomain (loosely speaking, range) of this function is called the key-space. For example, the key-space may be the unit interval $[0, 1]$ or an unit circle $[0, 1)$, so that the keys can be any real number between 0 and 1. The key-value pair is stored at peers responsible for the particular key. Efficient search of this key helps the applications to access the resource. This fundamental abstraction can be used for diverse applications including searching and sharing files or computational tasks or group communication. A structured overlay can be more formally defined as follows.

Definition 1 *Structured overlay networks comprise of the three following principal ingredients:*
(i) Partitioning of the key-space (say an interval or circle representing the real number between the range $[0, 1]$*) among peers, so that each peer is responsible for a specific key space partition. A peer responsible for a particular key-space partition should have all the resources (or pointers) which are mapped into keys corresponding to the respective key-space partition.*
(ii) A graph embedding/topology among these partitions (or peers) which ensures full connectivity of the partitions, desirably even under churn (peer membership dynamics), so that any partition can be reached from any other partition - reliably and preferably, efficiently.
(iii) A routing algorithm which enables the traversal of messages (query forwarding), in order to complete specific search requests (for keys).

Balancing load among the participating peers is an important concern in structured overlays. Most existing load-balancing techniques for structured overlays deal with managing the partitions [1, 2, 4] and key-assignments, so that approximately equal number of keys are assigned to each partition and hence each peer. These however do not balance the query-related loads of (i) query forwarding and (ii) query answering (access of keys). Indeed, in real life

[*]The work presented in this paper was (partly) funded by NTU's start up grant and partly carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European project NEPOMUK No FP6-027705.

it is likely that different keys are accessed with different frequency. The relative popularity may vary by orders of magnitude, for instance in a Zipf distribution.

A caching based approach to alleviate hot-spots appeals to the intuition. In a recent work [3] we formally proved the intuition of best possible caching scheme for a large family of structured overlays. We showed that caching proportional to query frequency for individual keys minimizes search cost subject to the system's storage capacity constraint. This holds when the caches are placed at specific peers, exploiting the overlay's topology. If the key-space partitions are abstracted as leaves of a search tree, the caches should be placed in the nearest neighboring subtree, as if recursively coalescing the key-space partitions.

We also exposed experimentally that currently used (greedy) routing strategies in overlay networks inherently lead to high variation in resource usage at different peers, even if the best possible caching scheme is used. This is expected due to the variance observed in an uniform random distribution. Queries are issued from different parts of the network, and a typical greedy routing mechanism (e.g., longest prefix matching) oblivious of the load at peers may lead to disproportionate use of peers' resources, which may in turn in the extreme case lead to overlay network congestion even when there is free capacity available in the overall system.

Structured overlays typically use redundant routes for fault-tolerance. In this paper we provide simple and effective load-aware variants of greedy routing strategy exploiting such route redundancy, and evaluate the performance of the overlay with extensive simulations using the P-Grid [1] topology to demonstrate the very good quality of query load-balancing that is achieved with our scheme. The choice of simulation system has been partially guided by the fact that we integrate our schemes in the Java based ongoing P-Grid implementation, which is used as a substrate for diverse applications and is regularly deployed and tested in the PlanetLab [6] infrastructure.

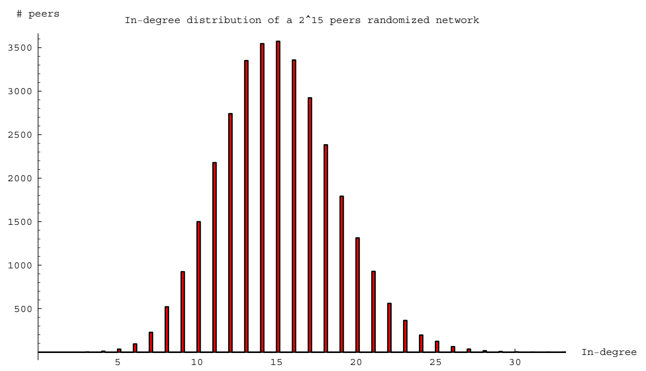
The paper is organized as follows. In Section 2 we recapitulate (from [3]) that in practice just caching is not good enough to achieve good query-answering load balancing. We also observe that the query forwarding load even in a network with good in/out-degree balance is again subject to high statistical noise. As a consequence of this noise, a system with sufficient capacity to meet the workload requirements will experience congestion and overload in some parts, while other peers will have their resources unexploited. This is because of the typical greedy route forwarding strategy used in current overlays which does not take into account load considerations. In Section 3 we propose and evaluate variants of greedy routing which exploit routing redundancy to dynamically choose least loaded of a set of potential routes. Our solution is simple still highly effective, nevertheless never used for the highly critical but neglected problem of query-load balancing in the last five years of extensive structured overlay research. In Section 4 we point out the few related works which provide partial solutions to query load balancing problems and then conclude in Section 5.

Before proceeding any further, we will like to point out that we make the following assumptions: (i) The number of keys a peer is responsible for is already balanced, which is more or less achieved under various settings - range-partitioned or DHTs - by [1, 2, 4]. (ii) All peers have same[1] and limited storage, part of which is dedicated to store the keys it is responsible for based on its role in the index, and the rest is used in order to dynamically cache some keys in order to alleviate load-balancing and improve search latency. (iii) The routing network itself does not lead to any systematic hot-spots because of large variations in degree distribution at each peer. This is a critical issue particularly for overlays with randomized routing networks [1, 10] (deterministic ones [12] already have good balance), which has not been studied in the literature in great detail, but can readily be achieved using standard techniques like power-of-two (or multiple) random choices [5]. In Figures 1(a) and 1(b) we show the in-degree distribution in a P-Grid network without and with such an in-degree balancing technique. We omit further discussion on balancing in-degree in randomized networks, and instead concentrate on query-load balancing at runtime in an already existing overlay.

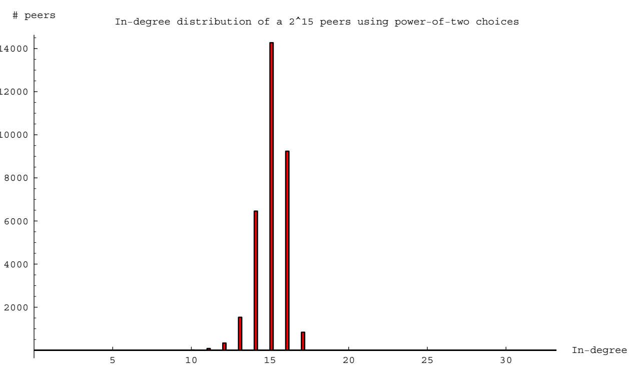
2. Is caching sufficient?

Distributing load randomly uniformly incurs high statistical variation. Balls into bins [7] is a commonly used analogy. Similar variation is also observed for storage load balancing in DHTs where uniform hashing is used to achieve storage load balancing in expectation, thus calling for more sophisticated load-balancing techniques [2]. Caching is popularly used to deal with hotspots and query-load imbalances [3, 8, 9]. However, just caching keys proportional to the access frequency but routing to any of these caches randomly is also expected to lead to high statistical variation of query-answering load. Similarly, even in overlay networks with well balanced degree distribution of peers, the routing process is expected to cause large variation in the actual forwarding load at peers. This intuition has however been largely ignored so far. One possible reason for such an oversight is that until recently there has not been any real deployed structured overlays, and most simulations were focused on key distribution (storage) workloads that cumulate over time, unlike bandwidth and CPU usage, which is temporal in nature. However, these are critical in forwarding

[1]Homogeneity can be achieved by using multiple virtual peers for resource rich peers.



(a) Randomized choice of routing table



(b) Power-of-two based random choice of routing table

Figure 1. Route in-degree balancing in randomized structured overlay topologies. All the results in this paper are based on overlay routing networks which already have a fairly well balanced in-degree distribution (like in Figure 1(b)), unless specified otherwise.

and answering queries in real time. Imbalance in query-load distribution may lead to congestion even when other peers have unused bandwidth capacity, or hotspots even when caching peers stay unused.

2.1. Simulation setup and workload

We demonstrate the validity of our intuition based on simulations. The consequence of such high statistical variation is that even if the system is provisioned for dealing with hotspots by providing extra storage capacity, and deal with the query traffic by providing sufficient aggregate bandwidth, the system will not be able to deal with the workload, simply because some peers will be overwhelmed while resource of many other peers will stay unused.

We simulated a randomized tree-structured network of 2^8 peers (specifically using the P-Grid routing topology). We considered that there was no route redundancy. So to say, for a given query, at a given peer there was only one potential routing entry, and this route is chosen greedily by the peer (longest prefix match in P-Grid). In real life, overlays need redundant routes for fault-tolerance, and we will show later how we can also use the same routing redundancy to achieve good query-load balancing. The sole routing entry was however chosen either (i) randomly or (ii) using the power-of-two random choices [5] to balance the in-degree (see Figure 1). Note that the power-of-two choices based routing table instantiation is somewhat similar to other mechanisms like proximity aware routing table construction in that the choice is made once during network construction process. While such a priori careful design is both useful in removing any systematic bias (high in-degree naturally will imply higher load) and is a good design practice, we see from the simulations that it does not provide mechanisms to cope dynamically with the system's workload at runtime.

Each peer initially held $D = 5$ unique data items, thus there were 1280 unique keys. Each peer had a capacity for storing a maximum of $R_{pot}.D$ data items (including the original and caches), where R_{pot} was chosen to be 10.

In the first experiment, queries with relative frequencies Zipf-distributed (parameter 0.8614) were originated at random peers chosen uniformly. Approximately 16700 queries were issued. We compared the system's performance with and without caching. For the former, queried objects were cached according to the optimal placement strategy [3] (proportional to the query frequency, and exploiting the topological characteristics of the overlay), with one additional cache created for each query received by any current cache. In case of lack of storage space, least recently queried object (locally perceived at individual peers) was removed in order to replicate a newly queried object. We also ran another set of experiments to isolate the effect of statistical noise of just the routing process. All objects were queried the equal number of times. We compared the effect on query-answering load with and without caching. The instance of the overlay network where routing tables were chosen based on power-of-two random choices was used.

2.2. Greedy routing in structured overlays

We show *cumulative distribution functions* in Figure 2 to summarize the load-balancing results, where two different measures of load are used: (i) the number of query messages forwarded by peers, as well as (ii) the number of queries actually answered by peers (possible when the peer has the corresponding key stored locally). The *cumulative distribution* plots are to be interpreted as follows: The x-axis represents the load and the y-axis the percentage of the peer population which has a load less than or equal to this specific (x-axis) load. Thus steeper ascent of the curve represent smaller variation of load among peers, while gradual ascent results from greater variation (poorer load-balance). As mentioned above, two sets of experiments were conducted, once with queries with relative frequency Zipf-distributed (parameter 0.8614), another where all keys were queried exactly the same number of times. Queries were issued at random peers.

As expected, there is reduction of the absolute number of forwarding messages per query with the use of caching (Figure 2(b)). In fact if the optimal caching strategy is used, the best reduction in search cost of an object with r replicas is from $log\,(N)$ to $log\,(N/r)$ where N is the number of key-space partitions (assuming that the overlay topology leads to logarithmic search cost) [3]. However, apart this reduction in search cost, there is also hidden costs of caching. We are thus not interested in the absolute load in this paper, but primarily on load-balancing aspects, and thus interested in the slope of the CDFs, the steeper the better load-balance.

A slightly steeper slope in Figure 2(a) for the caching based scheme shows that the deviation in the number of queries answered using the replication based strategy is lower than without replication, that is, replication leads to some improvement in query-answering load-balance. We also notice that balancing the in-degree based on power-of-two choices (Po2C) leads to improvement in load-balance. The improvements are discernable, but limited. We attribute it to the statistical noise. The huge effect of statistical noise becomes apparent in Figure 2(c) for the experiment where all keys are queried the same number of times. In this case, the query-answering load is perfectly balanced if no replication is done, since each peer receives all queries for its own keys and all keys are queried equally. However, with caching, as keys are replicated - the effect of statistical noise kicks in, thus in fact leading to load-imbalance.

This experiment where keys were queried equally was more to put in context the effect of statistical noise. Under realistic work-loads, we'll need the query-adaptive caching as a means to achieve load-balancing on an average. However, such a load-balancing is in itself inadequate unless complemented with other mechanisms to reduce the variance. Since real overlays need multiple route choices per level primarily for fault-tolerance, we next study whether such routing redundancy can be exploited to improve load-balancing.

3. Exploiting routing redundancy

Redundant routing table entries are used in structured overlays for fault-tolerance. So to say, any peer typically knows several peers to which it can forward a query. In a tree structured (prefix based routing) overlay, these choices are from all the peers belonging to the complementary subtree. Of-course, at certain levels, there may be fewer choices. We investigate whether such routing redundancy can be exploited for improving load-balancing. We repeat the experiments with same work load as above, i.e., Zipf distributed access frequency of keys, in a P-Grid network where peers had at least one (no redundancy) and at most five unique redundant route choices per level.

3.1. Greedy routing with blind choice from redundant routing entries

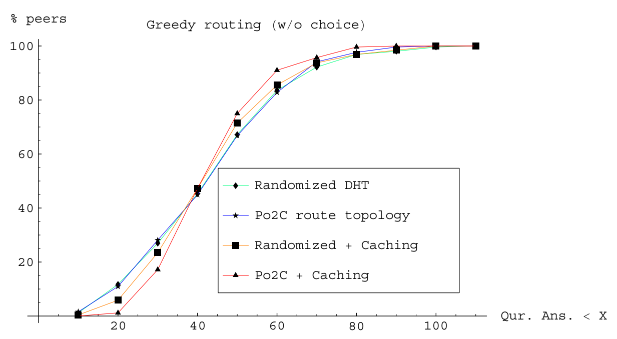
The actual route was randomly (blindly) chosen for each query at each hop, out of all the possible candidates as determined by greedy routing. In doing so, we obtain a somewhat steeper CDF for query answering, however just using multiple choices blindly does not realize good load-balancing for either query answering or forwarding load, as can be seen from Figure 3.

3.2 Greedy routing choosing redundant entry with least forwarding load

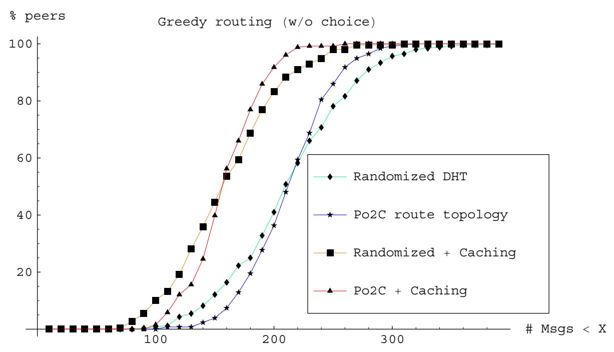
The high variance of both query forwarding and answering load are artefacts of the routing process. So we investigate next, what happens if we forward queries based on the forwarding load of peers, thus choosing dynamically the least loaded (at that time instant) candidate as the route. The idea is intuitive and requires simple modification of existing approaches, nonetheless, has not been employed in structured overlays, and we see next that extension of this basic intuition leads to very good query-load balancing even for very skewed load-distributions. In Figure 3 we note that a forwarding load aware routing process dramatically improves the forwarding load balance. However, even with caching and forwarding load aware routing, the query-answering load imbalance stays high. There is also reduction of absolute number of messages in the strategy where caching is used, as expected.

3.3. Greedy routing choosing redundant entry with least cumulative load

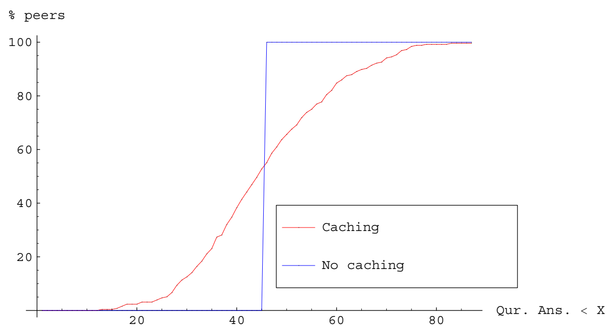
We next tried to decide the query-forwarding process based on a cumulative load measure. Typically, answering and forwarding queries require different amount of resources, both bandwidth and computation. While query forwarding involves a relatively small packet size, depending on the number and size of the query results, answering queries may be substantially more expensive. We define a parameter ζ, such that forwarding a query incurs ζ fraction of load as in answering a query. We simulated scenarios with various values of ζ, ranging between 1 to 0.1. Lower values of ζ imply answering a query is much more expen-



(a) Queries answered (Zipf)



(b) Messages forwarded (Zipf)



(c) Queries answered (Equal access)

Figure 2: Cumulative distribution of query answering and forwarding loads at peers. **(i)** Zipf distributed queries for random as well as power-of-two choice based route choices. **(ii)** Same number of queries per key, for only the power-of-two choices based route choices.

sive than forwarding a query. For these experiments, we first provide CDF of only the cumulative load in Figure 4.[2]

3.3.1 Is caching necessary?

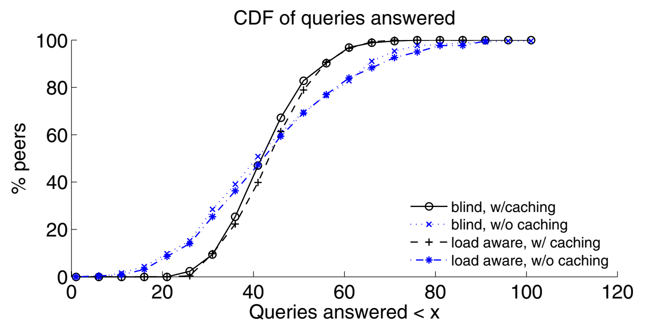
We observe that using such a cumulative load based routing significantly improves the load-balancing, particularly with the use of caching. Without caching, the load-balancing quality deteriorates for lower values of ζ. This is because, when the effort to answer a query is similar to the effort to forward a query, even if we do not use caching, the peers can be compensated with one kind of load or the other, but when the forwarding and answering loads are substantially different such a trade-off does not work. This in turn implies that if the effort for query-answering and forwarding are interchangeable ($\zeta \sim 1$), then effect of caching on load-balancing is marginal, while if query answering incurs significantly higher load than query forwarding ($\zeta \ll 1$) then caching is necessary. Of-course as already seen that caching in itself is not sufficient, and for query-load balancing we still need load-aware routing along with caching.

In Figure 5 we show separately the query answering and forwarding components of the cumulative load, and observe that even individually they are fairly well balanced based on the cumulative load based forwarding strategy (with the use of caching), though the balance of individual components is not as well as the balance of cumulative load. This is because the peers with higher answering load are compensated by lower routing load and vice-versa, and hence such imbalance in the component loads is in fact what the routing algorithm was designed to achieve.

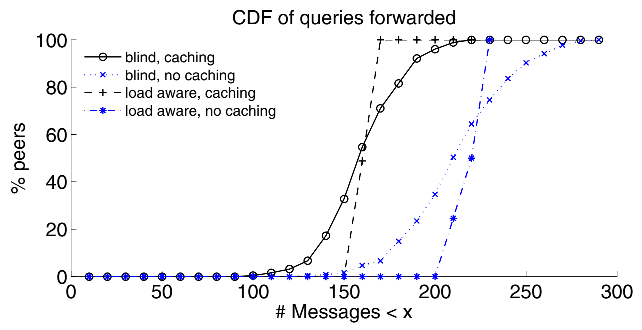
3.4 Impact of route redundancy

Finally, we show in Figure 6 that routing redundancy is a prerequisite by using the same cumulative load-aware routing algorithm for different instances of P-Grid with maximum routing redundancy per level per peer varying between 1 and 10. We notice as expected that for low route

[2]Ignore the reduction of the absolute value of loads for experiments with different ζ values, since the contribution from forwarding load decreases for lower ζ.



(a) There is marginal improvement in query answering load because of caching when using either blind or forwarding load aware routing. There is no discernible improvement because of forwarding load based routing.



(b) Expectedly, caching leads to a lower message forwarding load since query latency in terms of overlay hops and the absolute number of messages is also reduced. There is a significant improvement in the forwarding load-balancing with the use of a forwarding load aware routing in comparison to blind routing (both w/ and w/o caching).

Figure 3: Forwarding load aware greedy routing: CDF of query (a) answering and (b) forwarding loads at peers for Zipf distributed queries in a network where each peer had a maximum of 5 redundant route choices per level. The route is chosen dynamically (i) blindly, (ii) with least query-forwarding load.

redundancy, the quality of load-balancing stays poor. However the quality of load-balancing with caching is very good for even moderate level of redundancy (at most 5 entries), and the improvement there onwards is marginal. This is a good news since the moderate number of redundant routes required for fault-tolerance in structured overlays will in itself suffice to achieve good query load-balancing without any separate provisioning.

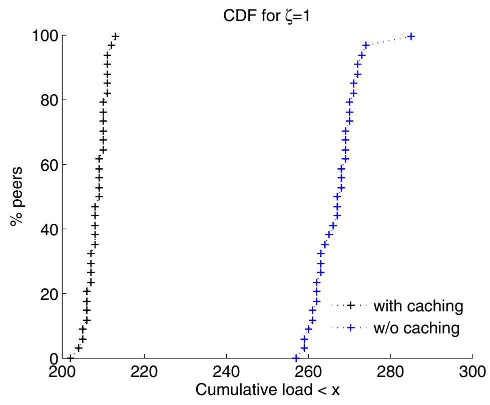
Discussion: From all these empirical results, we conclude that each of (i) exploiting route redundancy in structured overlays, (ii) load-aware routing, along with (iii) (more intuitive) access adaptive caching are important, complementary and necessary to achieve good query load-balancing, avoid congestions and deal with hotspots in structured overlays.

4. Related work

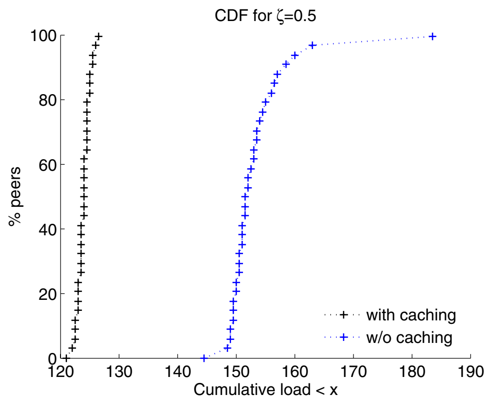
While there exists numerous work on balancing storage load in structured overlays, including [1, 2, 4], there are very few works on query-load balancing. A recent work [11] is the closest to our approach in that the authors propose to change the routing tables themselves in order to exploit the trade-off between query forwarding and answering loads. Such a scheme leads to change in the routing network itself, which means, some peers with less queries to answer may end up having high in-degree and will need to forward most of the traffic. This not only exposes the overlay to faults, but also leads to longer delays. Moreover the scheme has been shown to work with moderate success only for the special case when query forwarding and answering loads are completely interchangeable ($\zeta = 1$). Their limited success concurs with our results since no caching was used there. Another recent related work [13] proposes a mechanism to balance load (without differentiating types of load like: storage, forwarding or answering). The essential result in that paper has been two fold. (i) To show that heterogeneous peers need to provide resources proportional to their capacity. This in itself is both intuitive, and while necessary not sufficient because of the high variance observed in uniformly distributed random variables. (ii) Use aggregation of load-history and using topology aware caching to redistribute load. The topology aware caching is on similar lines as our previous work [3]. Aggregation of load-history and caching accordingly is a reactive strategy, and caching in itself does not solve the load-imbalance problem, as we observed in this paper. Our work tries to balance load dynamically at run time, complementing the caching techniques, by exploiting route redundancy, using routing heuristics which require minor changes in the current routing process.

5. Conclusion

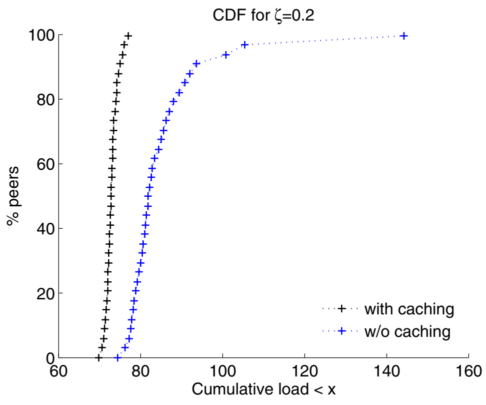
There is déjà-vu in what we discover from an objective look at query-load balancing. Initial research in overlay design [9, 10, 12] hoped to achieve good balancing of key-distributions among peers by using uniform distribution. The effect of statistical noise [7] was recognized only later, and had to be fixed using further load-balancing mechanisms [2]. One can nevertheless find in literature that in order to deal with hot-spots, caching is proposed (which is necessary!), and presumed sufficient, which is not the case. In some sense, it is unfortunate that despite dealing with the effect of randomization for storage load balancing, the same effects of randomization for more critical resources - bandwidth and peer's answering capacity under hot-spot condi-



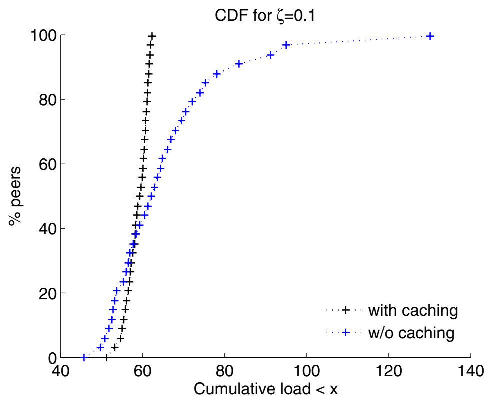
(a) Cumulative load ($\zeta = 1$)



(b) Cumulative load ($\zeta = 0.5$)



(c) Cumulative load ($\zeta = 0.2$)



(d) Cumulative load ($\zeta = 0.1$)

Figure 4: Cumulative load aware greedy routing: CDF of cumulative load at peers for Zipf distributed queries in a network where each peer had a maximum of 5 redundant route choices per level, and the routing process chooses the candidate with least cumulative load.

tions, were totally ignored, presuming caching itself will solve the problem. One may speculate several reasons for overlooking such an important thing: (i) Initial work based on simulations could observe the imbalance of key distribution, since it accumulates over time. Bandwidth consumption is however temporary, and if only the average is measured (as has often been reported in most results), the imbalance goes unnoticed. (ii) It is only recently that some structured overlay implementations have matured enough to be deployed and is dealing with moderate query loads, and hence the effects of imbalance has not been observed. But as the volume of traffic in structured overlays increase, balancing of query-load will become critical, since otherwise it'll cause congestion (and IP layer congestion control mechanisms will not be useful if the overlay systematically causes the congestion at end-nodes) even while other peers would have their resources under-utilized.

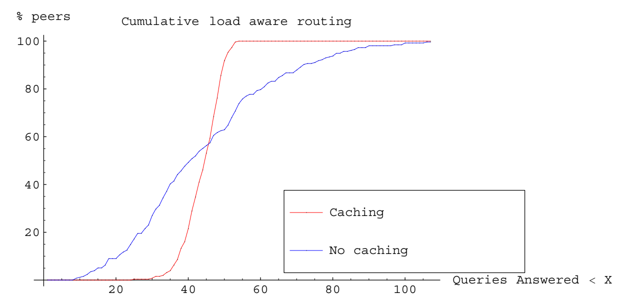
In that context, our work rediscovers the ghost of statistical noise. The way to reduce the variance is straightforward, requiring slight modification of the existing greedy routing mechanism used in most structured overlays, and provides excellent load-balancing. It is simple yet a powerful solution for a critical but long ignored problem of query-load balancing in structured overlays.

Overlays often maintain multiple routes for fault-tolerance. Using piggy-backed messages on control traffic or other means, load at the routing table entry peers can be estimated locally. This information can be utilized in dynamically choosing a proper route. We show the need and effectiveness of such load-aware routing strategies, complimenting caching in order to balance query-related loads in structured overlays. Integration and experimentation of these mechanisms in the actual P-Grid implementation is ongoing work.

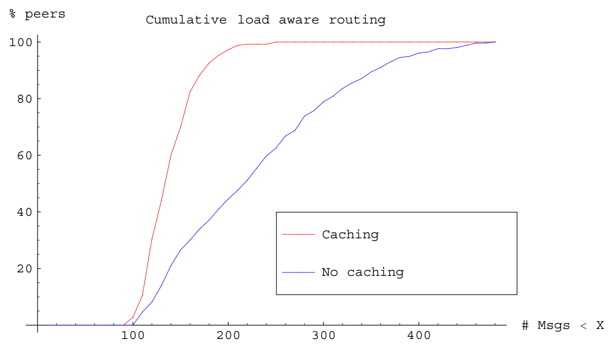
References

[1] K. Aberer, A. Datta, M. Hauswirth, and R. Schmidt. Indexing data-oriented overlay networks. *VLDB*, 2005.

[2] J. Byers, J. Considine, and M. Mitzenmacher. Simple load balancing for distributed hash tables. In *IPTPS*, 2003.

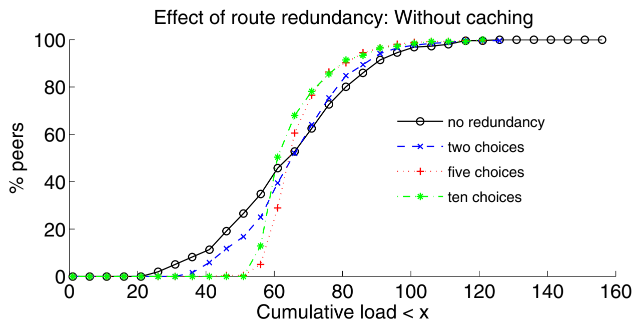


(a) Queries answered

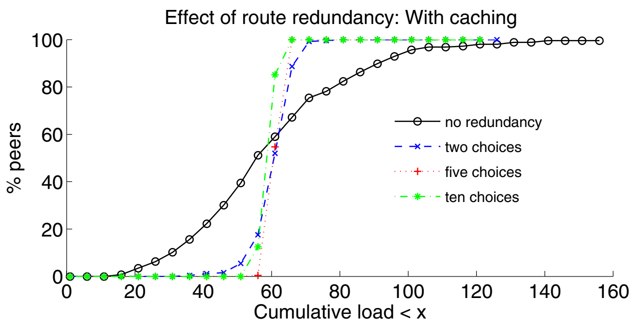


(b) Messages forwarded

Figure 5: Cumulative load aware greedy routing: CDF of components - query answering and forwarding loads - at peers for Zipf distributed queries in a network where each peer had a maximum of 5 redundant route choices per level, and the routing process chooses the candidate with least cumulative load.



(a) Without caching



(b) With caching

Figure 6: CDF of cumulative load at peers for Zipf distributed queries in various networks with different maximum redundant route choices per level, and the routing process chooses the candidate with least cumulative load ($\zeta = 0.1$).

[3] A. Datta, W. Nejdl, and K. Aberer. Optimal caching for first-order query load-balancing in decentralized index structures. In *The 4th International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, 2006.

[4] P. Ganesan, M. Bawa, and H. Garcia-Molina. Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. In *VLDB*, 2004.

[5] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12(10):1094–1104, 2001.

[6] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *SIGCOMM Comput. Commun. Rev.*, 2003.

[7] M. Raab and A. Steger. Balls into Bins - A Simple and Tight Analysis. In *RANDOM '98: Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 159–170, London, UK, 1998. Springer-Verlag.

[8] V. Ramasubramanian and E. Sirer. Beehive: O(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays. In *NSDI*, 2004.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM SIGCOMM*, 2001.

[10] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[11] S. Serbu, S. Bianchi, P. Kropf, and P. Felber. Dynamic load sharing in peer-to-peer systems: When some peers are more equal than others. In *Montreal Conference on eTechnologies (MCETECH'06)*, 2006.

[12] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM*, 2001.

[13] Z. Xu and L. Bhuyan. Effective load balancing in p2p systems. *CCGrid*, 2006.