Efficient Massive Sharing of Content among Peers

Peter Triantafillou, Chryssani Xiruhaki and Manolis Koubarakis
Department of Electronics and Computer Engineering
Technical University of Crete
Chania, 73100
Greece
peter@softnet.tuc.gr, xiruhaki@softnet.tuc.gr, manolis@ced.tuc.gr

Abstract

In this paper we focus on the design of high performance peer-to-peer content sharing systems. In particular, our goal is to achieve global load balancing and short user-request response times. This is a formidable challenge, given the requirement to respect the autonomy of peers, their heterogeneity in terms of processing and storage capacities, their different content contributions, the huge system scale, and the dynamic system environment. Our approach exploits the semantic categorization of published documents and constructs clusters of peers. We provide a formal formulation for the problem of load balancing in our setting and prove that it is NP-complete. We also present a greedy polynomial time algorithm that achieves nearly optimal load balancing as shown by our experimental results.

1. Introduction

The client-server model has been the dominant model for constructing distributed systems and services and the simplicity of its concept has played a key role in the successful commercial deployment of distributed computing for more than a decade. The emergence of internet computing and applications, however, have made prominent some of the model's inherent weaknesses: the requirement of central control of information and processing at specialized computing nodes (i.e., the servers) is too stringent and limiting for a variety of rapidly emerging internet computing application classes. Internet content sharing systems are an example of systems supporting such an application class, which has become very popular recently through systems like Napster, Gnutella, Kazaa, Audio Galaxy, [15,9,12,3] etc.

In general, content sharing systems consist of a (potentially very large) number of computing nodes, offering (computing and storage) resources and content ("documents") to the community. Thus, nodes belonging to users may contribute content to the rest of the community and, in addition, may permit the use of their own resources to store content contributed by others and allow access to it from other community members. Given these characteristics, the central control of information at special nodes is undesirable in order to avoid central points of failure and performance bottlenecks, to preserve the anonymity of users accessing content and services, and to fully utilize the available resources contributed by *all* member nodes. As a result, the *peer-to-peer* (P2P) paradigm for architecting distributed systems is recently becoming increasingly popular. P2P systems consist of a set of peers, which are nodes of equal stature, which have autonomy, and which can collaborate with each other, pulling together their resources, in order to either obtain services or jointly tackle large computing jobs.

1.1. A high level view of the problem – our goals

With the general goal of ensuring high performance in large-scale P2P content sharing systems in mind, in this paper we focus on two particular subgoals: (i) ensuring load balancing across all nodes of such a system, and (ii) ensuring short response times to user requests. In addition to the obvious high usefulness of achieving these goals, it should be stressed that achieving load balancing is of fundamental importance in a P2P system, given the equal stature of the nodes, and that ensuring low response times realizes one of the main promises inherent in the peer-to-peer paradigm. The load-balancing problem for P2P content sharing systems has not been addressed by related research and is an open problem.

Furthermore, typical systems such as Freenet and OceanStore [6,14] might face serious difficulties when it comes to ensuring low response times, since requests are passed from peer to peer, either until one is found that stores the desired document(s), or until a user-determined 'number-of-hops' count is reached and the system gives up. Our solution will not burden the user with such difficult decisions and will ensure a response time within only a few hops for the common case and an upper bound on the number of hops for the worst case.

1.2. A high-level view of the problem – the difficulties

Despite the appropriateness of the P2P paradigm, ensuring high performance in P2P content sharing systems is a formidable task. The associated difficulties stem from the desirable use of the P2P paradigm and from the application's features. Having autonomous nodes of equal stature introduces the need of complex distributed coordination algorithms. Furthermore, the system should be expected to scale to hundreds of thousands of nodes and millions of documents. In addition, different nodes make different and varying content contributions (e.g., from no contribution at all, to contributing several documents) and offer/possess heterogeneous processing and storage capacities, a fact which further raises significantly the level of complexity involved in managing content and providing efficient access to it. Finally, the system may operate in a highly dynamic environment in which the popularity of the stored content varies with time, nodes enter and leave the system at their free will, and content can be added and deleted at any time.

1.3. Summary of main results

We assume that the content in the system has (an initially static and) known popularity[1], with document popularities following the Zipf distribution, as is the case for Web objects [5,2] and as it is substantiated by [17]. In our envisioned system the contributed documents are accompanied by keywords characterizing their semantic content and we utilize tools (e.g., [16, 4]) that can classify the documents, given their associated keywords.

The key elements of our approach are that: (i) we exploit the semantics of documents, which permit their grouping into semantic categories, and (ii) we form clusters of peer nodes, based on the semantic category of the documents contributed by the peers. Given this, our load balancing task is then viewed as a two level problem: First, we ensure load balancing across the peer clusters (inter-cluster load balancing) through the appropriate assignment of the document categories to the clusters; second, we ensure the load balancing among the peers that belong in the same cluster (intra-cluster load balancing), for all clusters. Finally, our intra-cluster load balancing and response time goals are achieved through the exploitation of metadata associating clusters with semantic categories.

The inter-cluster load balancing problem is NP-complete thus we develop a polynomial time incomplete greedy algorithm, which utilizes ideas from number partitioning algorithms [11, 13, 8] to solve our problem.

In a companion paper [18] we take a different approach and utilize the *fairness index* of [10] as a metric for load balancing.

In that paper we also show that our solution is robust with respect to different distributions of the popularity of semantic categories, varying skew of access to documents, different scales (with respect to the number of documents, the number of nodes, the number of categories, and to the number of clusters), differing content contributions by nodes, differing node processing and storage capacities, and with respect to the dynamics of the environment.

We hold the view that internet content sharing systems are an exciting application class, indicative of the next wave of applications and the associated difficulties, which the distributed computing community will be called to face. Also, the P2P paradigm is rapidly emerging as the paradigm of choice for such applications. And, based on the difficulties mentioned above, P2P content sharing systems pose a number of difficult challenges to our community, especially if efficiency and high performance are sought. Despite the emergence of several applications for P2P content sharing recently, academic researchers have not paid the attention the subject deserves as of yet[2], especially to the performance aspects and particularly to the critical issues of load balancing and short response times. Thus, with this work we explore exciting new territory where we attempt to tailor and apply knowledge accumulated by the distributed systems community over the years.

2. Proposed architecture

[1] Initial popularities of documents can fairly easily be estimated in most applications. For example, music file popularities follows their position in popular charts; the popularities of book files in library applications can be estimated using check-out information at conventional libraries; popularities of video files can be estimated by information at video rental stores, etc.

[2] Notable exceptions are Gridella [1] and some other recent efforts.

The basic (technological and 'philosophical') principle of our approach is that in order to achieve high performance in such a system, we need to impose a logical system structure (as opposed to the 'chaos' that can emerge) that can facilitate the achievement of our performance goals. The challenge is to ensure that this structure respects the autonomic behavior of peers and the heterogeneity of their characteristics and that the overall solution successfully addresses the difficulties presented in the previous section.

2.1. System organisation

The nodes/peers of the system will be *logically* organized into a set *C* of clusters. All nodes belonging to the same cluster will be able to either serve the requests for documents contributed by all the nodes of that cluster (in the case the nodes store all documents), or find another node that can (because they maintain cluster metadata describing which documents are stored by which cluster nodes).

2.2. System and problem parameters

We denote with $|S|$, the number of elements of the set *S*. Our system will consist of a set *D* of *sharable documents*. The $|D|$ documents are contributed by a set *N* of *nodes*. Each node in the system is a user's computer. Nodes can contribute more than one document to the system. The $|D|$ sharable documents of the system belong to a set *S* of semantic categories. This is captured by a function $f: D \to S$ that uniquely maps each document to a semantic category. Each document *d* is associated with a *popularity*, $p(d) \in [0,1]$, that is the probability of access of *d*. We will denote with $p(s)$ the total popularity of semantic category *s*. This popularity is equal to the sum of the popularities of the documents it consists of. In other words, $p(s) = \sum_{\{d: f(d)=s\}} p(d)$. Each node *n* has a popularity $p(n) \in [0,1]$ that is equal to the sum of the popularities of its documents. Formally, $p(n) = \sum_{d \in D(n)} p(d)$.

2.3. User request processing

User requests are submitted to the system through the users' nodes. Requests can be either to ***publish*** (contribute) or to ***retrieve*** (download) documents. At each node:

1. the semantic categories of the associated documents will be first identified, through the keywords accompanying the request,
2. the metadata kept locally at the node: (i) will associate categories with the proper clusters of nodes and (ii) will identify either all or, if user storage space is scarce, only a representative subset (consisting preferably of 'nearby' nodes, for faster access) of the clusters' nodes, and
3. the request will be forwarded to the proper cluster(s) storing these categories.

Initially the request will be forwarded to one randomly chosen node of the cluster. This random node selection can ensure that the nodes get an equal share of the workload targeting this cluster. If that node is not responding (say, due to failures or because it does/could not store the requested document(s)), it will forward the request to some other node of the same cluster. This will be repeated until the desired document(s) are found. Note that, in the common case, the user request will visit only a very small number of nodes, and the response time in the worst case will be bounded from above by the (tuneable) number of nodes in the cluster. So, with this approach we can see that both (i) the load is balanced within a cluster and (ii) low response times are ensured.

2.3. Global load balancing

In order to ensure the high performance of the system, we have to avoid bottlenecks and balance the load across *all* system nodes. *Load* in our case is the number of requests served by a data store node of the system. This goal can be partly achieved by associating the semantic categories with clusters of nodes, in a manner that ensures a uniform distribution of the document-category popularities to the clusters of nodes. We refer to this property as *inter-cluster load balancing*.

With the term *intra-cluster load balancing* we mean that, for each cluster, all the nodes that belong to it receive on average (approximately) the same number of requests from the total requests that target this cluster. With the term *global load balancing* we mean that the average load of all the nodes that belong to the system must be as uniform as possible. Global load balancing is achieved by independently achieving inter-cluster and intra-cluster load balancing. Since, intra-cluster load balancing is straightforwardly achieved as described in Section 2.3, henceforth we concentrate on the problem of inter-cluster load balancing.

2.3.1. Formal Problem Definition. We now formally define the problem of inter-cluster load balancing *(ICLB).* We initially assume that peers have the same processing and storage capacities and enough storage space to store all documents of the categories to which they contribute. We

do away with all these assumptions in the companion paper [18]. Our next definition is in the spirit of [7].

2.3.2. The decision problem ICLB. ***Instance:*** A set N of nodes (peers) and a set D of documents contributed by the nodes in N. Each document $d \in D$ has an associated semantic category $f(d)$ from a set of semantic categories S and a popularity $p(d) \in [0,1]$. Each node is assumed to contribute documents belonging to a single semantic category.

Question: Is there a partition of N into k clusters $N_1, N_2, \ldots, N_k$ such that the following two constraints are satisfied?

1. If two documents belong to the same semantic category, then the nodes that contributed/store these documents belong to the same cluster.
2. Clusters have equal normalized popularities. Formally, $\frac{p(N_i)}{|N_i|} = \frac{p(N_j)}{|N_j|}$, for all $1 \le i, j \le k$.

The load-balancing objective formalized by ICLB is to create k clusters of peers/nodes that will have equal total popularity and consequently equal load. But not all clusters are of equal size. So the popularity should be normalized with respect to the number of nodes in the clusters so that the average load faced by each peer/node in the system will be equal, facilitating global load balancing. Thus, we want the clusters that will be created to have equal *normalized cluster popularities* and this is captured by the second constraint above.

Proposition. ICLB is NP-complete.

Proof. Membership in NP is straightforward. To prove NP-hardness we use a transformation from the BALANCED PARTITION problem. This problem is a generalization of the PARTITION problem given in [7]. [3]

Obviously, in many situations we will not be able to come up with a solution to ICLB. In these cases, it is useful to consider partitioning the given set of nodes N into clusters with *nearly equal* normalized popularities. This can be done, for example, by minimizing the following quantity:

$$\min_{1 \le i \le k} \sum_{1 \le j \le k} \left| p(N_i) - p(N_j) \right|$$

We have experimented with a greedy algorithm, called MinDiff, which solves this minimization problem by using similar ideas as in known number partitioning algorithms [11, 13, 8].

[3] The NP-hardness result for BALANCED PARTITION is due to Apostolos Dimitromanolakis.

In MinDiff initially all clusters are empty. Then MinDiff considers each semantic category s in turn, and assigns it to the cluster of nodes which minimizes the *total difference* of balanced popularities among clusters. The assignment of a category s to a cluster is essentially the assignment of all nodes contributing documents with category s to that cluster. It is not difficult to see that MinDiff is incomplete (i.e., it might miss the optimal solution). However, MinDiff runs in polynomial time and achieves very good results as we show below.

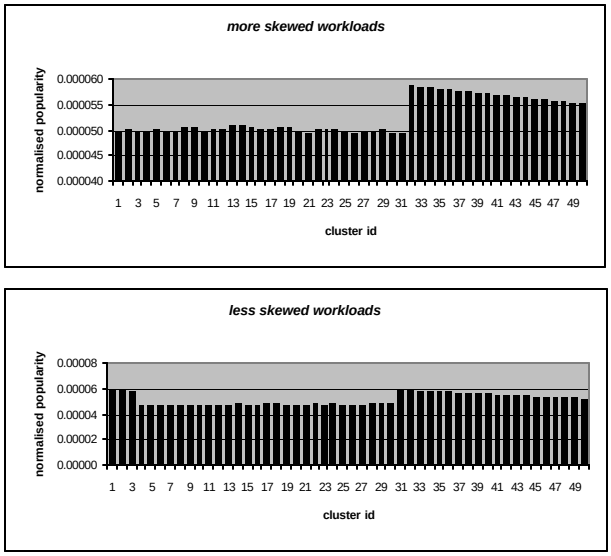

Figure 1. The MinDiff algorithm for more skewed (top) and less skewed (bottom) workloads

3. Experimental results

Figure 1 presents the results of MinDiff for 19,800 documents, 300 semantic categories and 50 clusters of nodes. We consider two different semantic category popularity distributions: both are Zipf-like with ?=0.3 (top diagram) and ?=0.7 (bottom diagram). In both diagrams the 10% of the semantic categories are represented by a number of nodes proportional to their popularity, while the rest of the semantic categories share the rest of the nodes equally. In both cases we notice that the MinDiff algorithm achieves very good load balancing as you can see by inspecting the normalized popularities on the y-axis (the differences are in the order of 10^{-4} to 10^{-6}). Differences are smaller in the first diagram since the popularity distribution of the semantic categories is less skewed. More detailed experiments are given in the long version of this paper, which is available from the authors.

4. Summary

In this paper we studied the problem of achieving global load balancing and short user-request response times in peer-to-peer content sharing systems. This is a formidable challenge, given the requirement to respect the autonomy of peers, their heterogeneity in terms of processing and storage capacities, their different content contributions, the huge system scale, and the dynamic system environment. We presented an approach that exploits the semantic categorization of published documents and constructs clusters of peers. We provided a formal formulation for the problem of load balancing in this context and proved that it is NP-complete. We also presented a greedy polynomial time algorithm that achieves nearly optimal load balancing as shown by our experiments.

5. Acknowledgment

This work was supported in part by project DIET (IST-1999-10088) funded by the IST Programme of the European Commission, under the FET Proactive Initiative on "Universal Information Ecosystems". We would like to acknowledge the contributions of all partners of DIET (BTexact Technologies, Universidad Carlos III de Madrid, DFKI) to this work.

6. References

[1] K. Aberer, M. Punceva, M. Hauswirth and R. Schmidt, "Improving Data Access in P2P Systems", *IEEE Internet Computing*, January-February 2002.

[2] V. Almeida, A. Bestavros, M. Crovella and A. de Oliveira, "Characterizing reference locality in the WWW", *Proc. of PDIS'96*, 1996.

[3] Audio Galaxy web site: http://www.audiogalaxy.com

[4] Autonomy web site: http://autonomy.com

[5] Lee Breslau, Pei Cao, Li Fan, Graham Philips, Scott Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications". *IEEE INFOCOM*, 1999 pp. 126 –134

[6] Ian Clarke, O. Sandberg, B. Wiley and T. W. Hong, "Freenet: A Distributed, Anonymous Information Storage and Retrieval System", *Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[7] M. R. Garey and D. S. Johnson, "Computers and Intractability – A Guide to the Theory of NP-Completenes", *W.H. Freeman and Co.*, 1979.

[8] Ian P. Gent, Toby Walsh, "Analysis of Heuristic for Number Partitioning". *Computational Intelligence*, 1998, Vol. 14, number 3, pp. 430-451.

[9] Gnutella web site: http://gnutella.wego.com

[10] R. Jain D-M. Chiu, W.R. Hawe, "A Quantitative Measure of Fairnsess and Discrimination for Resource Allocation in Shared Computer Systems", *DEC-TR-301*, 1984.

[11] N. Karmarkar, R. Karp, J. Lueker, A. Odlyzko, "Probabilistic Analysis of optimum partitioning", *Journal of Applied Probability*, Vol.23, pp. 626-645, 1986.

[12] Kazaa web site: http://www.kazaa.com

[13] R. Korf, "From approximate to optimal solutions: A case study of number partitioning", *Proceedings of the 14th IJCAI*, 1995.

[14] John Kubiatowicz et. Al., "OceanStore: An Architecture for Global-Scale Persistent Storage", *Proceedings of ASPLOS 2000*.

[15] Napster web site: http://napster.com

[16] Semio web site: http://www.semio.com

[17] K. Sripanidkulchai, "The popularity of Gnutella queries and its implications on scalability", *Featured on O'Reilly's www.openp2p.com website*, February 2001.

[18] P. Triantafillou, C. Xiruhaki and M. Koubarakis, "Towards High-Performance P2P Systems". *Submitted*.