Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 423–436
computer communications
www.elsevier.com/locate/comcom

Scalable tag search in social network applications

Alberto Mozo [a,*], Joaquín Salvachúa [b]

[a] Department of Arquitectura y Tecnología de Computadores, Universidad Politécnica de Madrid (UPM), Madrid, Spain
[b] Departament of Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid (UPM), Madrid, Spain

Available online 2 September 2007

Abstract

Emerging social network applications for sharing and collaboration need a way to publish and search a big amount of social objects. Currently, social applications allow the association of a set of user defined keywords, named tags, when publishing these objects, in order to allow searching for them later using a subset of these tags. Commercial systems and recent research community proposals preclude a wide Internet deployment due to the emergence of scalability and hot spot problems in the nodes.

We propose T-DHT, an innovative hybrid unstructured–structured DHT based approach, to cope with these high demanding requirements, in a fully scalable, distributed and balanced way. The storage process allows attaching a set of user tags to the stored object and takes at most O(Log(N)) node hops. The tag information attached to the object is stored in a compact way into the node links using a bloom filter, in order to be used later in the search process. The search process allows searching for previously stored objects by means of a tag conjunction and also takes at most O(Log(N)) node hops. The search process is based on DHT typical search combined with an unstructured search algorithm using the tag information previously stored into bloom filters of node links. The simulation results show T-DHT performs in a fully balanced and scalable way, without generating typical hot spot problems even if unbalanced distributions of popular tags are used.

Although T-DHT has been devised to build a scalable infrastructure for social applications, it can be applied to solve the more general Peer-to-Peer keyword search problems.

Keywords: Distributed algorithms; Communication protocols; Peer-to-Peer; Very large database systems; Keyword search

1. Introduction

New applications and services are now conquering the audience attraction in Internet: the so-called social applications. The change of focus from the system to the people is something that allows people an easier way to collaborate in the web. Flickr, the social application for store, search, sort and share photos; Del.icio.us, the famous social bookmarking application; and YouTube the popular video sharing website where users can upload, view and share video clips, are paradigms of this new type of applications. In these applications, one of the methods to help people to classify and locate information in this so-called Web 2.0 paradigm is the use of folksonomies [1]. Here, instead of using static formal defined ontologies, the people associate a set of variable and user defined tags to different contents. The social applications usually need a way to publish and locate a big amount of so-called social objects and their attached tags. Nowadays, we must take into account that this Web 2.0 paradigm does not involve too many technical changes, because it is just a new way of using previously existent technologies, and so the storage for such contents is usually implemented using a relational database that is not tailored to this kind of usage, and so, it does not scale adequately. If we want to build a new infrastructure layer for these new applications we must look them from a completely different approach, and so, we should be able to support the exponential growth of user generated contents (like blogs, RSS sources and podcast contents).

* Corresponding author. Tel.: +34 91 336 5118.
E-mail addresses: amozo@eui.upm.es (A. Mozo), jsalvachua@dit.upm.es (J. Salvachúa).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.08.035

Currently, commercial solutions relay in database centric approaches, and they are dying of success because of their limited scalability. Trying to solve this problem, distributed database systems have been a hot topic of interest in the database research community during the last years. Some approaches have focused on how to make transparent the distribution of data behind a more or less SQL standard query language, sacrificing scalability. SDD-1 [3] and Mariposa [2] are examples of it, scaling at most a few hundreds of nodes.

Recent research community proposals are focusing on Peer-to-Peer storage infrastructures in order to provide a distributed, balanced and high scalable publish and search solution. Distributed hash tables (DHT) Chord [4], CAN [5], Pastry [6] and Tapestry [7], among others, provide a pretty good method to store and search data in a distributed, balanced and scalable way. However, the keyword search is difficult to implement due to the fact the search procedure needs to know the object key identifier (usually obtained using SHA-1 hash function on the object). This key identifier is needed in order to route adequately the search towards the node storing the object. However, in social networks the search application only has available a set of user defined tags describing the object, and does not have a direct way to obtain the object key identifier.

Several solutions have been devised ([8,25–27] among others) based on distributed inverted index. Besides publishing the object, a mapping between each keyword identifier and the object identifier is also published. Subsequently, the search procedure is achieved in two steps: (a) we locate and receive the object identifiers associated to each target tags; (b) we get the objects fulfilling the tag conjunction, joining the results obtained in the former step. This approach solves partially the problem but incurs in two inefficiencies. First, the popular tags defined by users will generate hot spots in the nodes responsible for storing the popular tags. These nodes will be responsible for storing and answering to a huge amount of search requests. Also the links and their neighbor nodes will suffer the unbalanced number of store and search requests directed to the former nodes. And second, the post-join process needed to get the subset of objects fulfilling the tag conjunction, forces to transmit into the network a huge amount of results that will be discarded subsequently, because they do not fulfill the tag conjunction, hence wasting bandwidth network resources. Some query-cache procedures are proposed [9] in order to mitigate the hot spot problem. Also [26] proposes additional publishing of two or three keyword combinations, instead of a single keyword, diminishing the number of results received when requesting for popular tags. Nevertheless, the scalability problem remains and precludes a wide Internet deployment.

We propose a new approach to achieve real scalability in the store and search procedure. We have developed an innovative technique modifying a typical DHT overlay in order to maintain the high quality properties when storing objects, but allowing a conjunctive tag driven search while maintaining the scalability properties of DHTs.

The storage process of an object allows attaching a set of user tags to the stored object. The storage process is based on typical DHT procedures, but slightly modified in order to also insert the object tag information. During the storage process, a publishing sub-phase is started where the object attached tag information is inserted into the node links, in order to adequately drive the later search procedure. Since the store and publish procedures are DHT based, the whole storage process takes at most $O(\mathrm{Log}(N))$ node hops.

The search process allows finding previously stored objects by means of tag conjunction. This process is based on typical DHT search combined with an unstructured locate algorithm using the tag information previously stored into the bloom filter of node links. Therefore, when a search procedure is started, the tag information previously inserted into the node links is consulted in order to determine the right path to reach the object that fulfills the tag conjunction. In this way, the locate procedure behaves similarly to an unstructured Peer-to-Peer overlay search procedure (e.g. Gnutella [10] without the scalability problems associated to this kind of systems [11]). Additionally, since the search and locate procedures are DHT based, the whole search process is bounded to $O(\mathrm{Log}(N))$ node hops.

In order to insert in a compact way and recover efficiently routing tag information in the links, we associate a bloom filter [12] to each node link. Since the bloom filter size is predetermined and constant at startup phase, we guarantee the system scalability, with a little tradeoff due to the probability of false positive generation, during the search routing process. These false positives are eliminated with a high probability in subsequent overlay hops and do not degrade substantially the system performance.

In T-DHT there is no need to generate expensive inverted indices in order to be able to search for tag conjunctions. Besides, no post-join operations are required in the source node to achieve the object subset fulfilling the tag conjunction constraints. The simulation results have shown that T-DHT performs storing and searching in a fully balanced, distributed and scalable way, avoiding the hot spots in nodes. Even if the tag distribution is composed by popular tags following an unbalanced Zipf distribution, the performance of conjunctive tag search achieves identical results to those obtained using a well balanced random tag distribution.

In the next section we introduce to the basis of T-DHT architecture and protocol. We describe the detailed T-DHT routing algorithm in the store and publish phase and subsequently the search and locate procedures. In Section 3, some protocol extensions are presented. Section 4 presents the results of the first working prototype and Section 5 shows the related works. Finally we conclude in Section 6.