Functional Approach to Decentralized Search Engine for P2P-network Communities

Oleksandr Kuzomin, Illya Klymov

Informatics chair
Kharkiv National University of Radio Electronics
UKRAINE, Kharkiv, Lenina av. 14, office 437
kuzy@kture.kharkov.ua
illya.klymov@gmail.com

Abstract: In this paper we propose a simple powerful method to increase search performance in large p2p-network communities, especially, but not limited to Internet. Our approach is based on introducing common functional language methods "map" and "reduce" in distributed network environment. The method exploits different connectivity levels of p2p nodes, dividing them into two groups – master nodes, performing preliminary search processing and worker nodes, performing actual search. We show that approach could be effectively used in Grid network, built over common P2P-network utilizing benefits of routing in Grid environment. An experimental implementation of the method confirms the theoretical analysis of the method.

1 Introduction

An extreme amount of data available in an Internet requires an efficient tool in order to determine where we locate data matching our request. This process is not limited to web-search (solved by Google, Yahoo, etc.). In big P2P networks an efficient way for finding required chunk allows to significantly decrease overall bandwidth consumed by the network.

2 Problems of Using Common Search Engine Approaches in P2P-networks

A key idea of search engine functioning is an ability to retrieve links to new documents (let's speak about HTML pages for example) in process of indexing existing data. This is not applicable to P2P-networks since documents (files represented into network) have no interconnections between each other. On the other hand each node in P2P provides an interface to enumerate all documents, available on this node. In general the following problems renders common indexing algorithm almost unusable:

- Small peer TTL (time to live) – in normal indexing process we put recently discovered documents tot he end of queue, but when we end processing queue the peer, provided these documents, could already be offline. On other hand we could easy stuck on indexing one big, yet slow node, which is unacceptable for making quick searches
- Some peers could not be accessed directly (they have no white IP). This nodes still participate in P2P-network, they carry important network data, but we can't use „PULL" technique to get data from them - data must be directly pushed by that nodes
- P2P query consumes significant bandwith on the wide-area Internet; the total bandwith consumed by all queries (which should be performed in parallel if we want to provide an acceptable response time) must fit comfortably within Internet's capacity.

Another specific of P2P-search is that each document in network could be described in less than 1KB of data (including filename, metadata, node info) since we are not assuming performing full-text search on P2P-network. In other words we should pay significant attention to quick indexing each node, assuming that time of indexing document is not notable comparing to time for making connection and retrieving index of available node documents.

3 Search Engines in P2P-environments

3.1 Functional Approach to Search in P2P-networks

Any approach used for search in P2P-networks must be designed to meet the following goals: ability to operate in dynamic environment, performance and scalability and reliability. In order to start with functional approach in P2P-network let's define primary terms and operations over network.

We divide all nodes in two big groups – worker nodes and master nodes [MR09]. Worker nodes provide an index file of all its data on request: $\text{Index}(W_i) = L$, where W_i is a worker node and L – is an index in form of list, generated for quick search in it (consider it for example to be a binary search tree optimized for quick search or just a hashtable – selection of correct model greatly depends on specific of indexed data). Each worker node has at most one master node, to which it is connected.

Master nodes of 1 level store list of worker nodes, and store integrated index, which includes all data, provided by worker nodes, connected to this node (details of how worker nodes provide data to master ones are described in section 3.3):

$$\text{Index}(M_{i1}) = L \cup \text{Index}(W_1 \cdots W_k), W_1 \cdots W_k \in \text{Workers}(M_{i1})$$

Master node of n-th level is functioning as master node for all master nodes of (n-1)-th level and as worker node for at most one (n+1)-th level master node. Also each master node of n-th level contains a cached copy of all nodes of the same level, connected to (n+1)-th level master node, to which current node is connected.

Search operation could be implemented in two stages. First one is called "map". On this step each master node receives the query, spreads it to all its workers (except the source one) and transfer it to an upstream node. We guarantee that each node of level n has a node of level (n+1) connected if node of such level exists into network. The connection from worker nodes shown as dotted line because this nodes are not directly involved into search, but the data they provide is used for

Next step, called "reduce operation" is receiving results from all nodes and processing them into single instance. In order to generate a result for any node consider we have a function *Search* which returns results from an index. In such way results of node W_n could be described as follows:

$$\text{Search}(W_n) = \begin{cases} \text{Search}(\text{Index}(W_n)), & n = 1 \\ \text{Search}(\text{Index}(W_n)) \cup \text{Search}\,(\text{Workers}(W_n)), & n > 1 \end{cases}$$

Each node performs reduce operation before transferring query back in order to eliminate possible duplicates from response.

In such scheme the main weak point is first master node, which performs overall search. If this node goes offline during search requests it leads to all search results lost, wasting a huge amount of computing time and network bandwidth. In order to reduce losses in that case each master node provides a list of B backup nodes (where amount of B may vary and should increase with network growth). This backup nodes is an ordered list which is known both to workers and upstream nodes and describe which node should receive data in case current node goes offline unexpectedly. The main idea of this approach is shown on Figure 1:

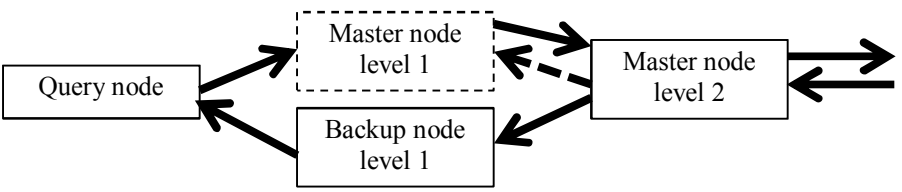

Figure 1: Fault-tolerance P2P search in action

On this figure it is shown how search reply from master node level 2 tries to reach master node level 1, but data transfer fails, for example due to timeout, the data is sent to the first backup node, which was provided in request, sent to master node level 2. This backup node notifies a data, which was not requested directly by that node, and analyzing the body of search response reveals that this data should be transferred to query node. In that manner we ensure that no results of the search would be lost during search process.

The main problem of such approach is that returning result of node is delayed until all downstream node formed it's response or timeout for waiting response is reached. This could be solved by selecting nodes, which are not heavily loaded in P2P-exchange as master nodes and decreasing timeout to low values, to make search quickly react on changing network structure [ZLX04]

3.2 Concept of P2P Search Engine using Proposed Approach

Now we are one step away from building a prototype of P2P search engine using proposed approach. First of all, it is time to decide how we can choose master nodes in the real grid network. Since master nodes would receive all requests, they should be not heavily loaded in real P2P-data exchange. This is a situation where the term "node temperature" becomes useful. We will measure node temperature as:

$$\theta_s = \frac{\text{Messages in buffer}}{\text{Buffer size}}, 0 \leq \theta_s \leq 1 \text{ [LU09]}$$

If node temperature is below some threshold θ_t this node could be considered as 1-level master node. All nodes with temperature lower than θ_t^2 will become master nodes of level 2 and so on.

Each master node of level *n* contains data about:

- all master nodes of n-1 level
- indexes received fron all master nodes os n-1 level (not necessary directly), with a TTL (time to live) mark attached to each
- master node of n+1 level to which it is connected.

In other words we are building a tree like system on top of existing grid mesh. Each node is functioning according to following algorithm:

1. Each X seconds requests if master node (node of higher level to which this one is connected) is alive. If not – all nodes which were using same master perform a voting process – node with least temperature becames a new master node (forced promotion)
2. If current temperature of node is lower than θ_t^k, where k>n (assume n is current level of node) for time more than Y seconds – node is promoting self to n+1 level. A new master is obtained from previous one, and all nodes, which were connected to previous master are notified and could change master

3. Each Z seconds a current index of all files is pushed to the upstanding master node. Master node merges all received indexes in one, includes own file index and provides this as new master node index
4. If current node has no master node privileges (worker node) it searches for any master node using blind search, receives list of all master nodes and using ant algorithm determines which node is most suitable to connect. After that it publishes own list to found master node.

Varying (X,Y,Z,S) we can vary intensity of updates in order to find an optimal solution between speed of reaction of overall search network to changes and an amount of service traffic, generated by network. Values of this parameters greatly depends on bandwidth between nodes, node lifetimes and how often node data is changed.

4 Conclusion and Future Work

In this paper, a method of building search engine in P2P-environment using map-reduce approach is introduced. This approach is provides a strong fault-tolerant backend for performing searches in a dynamic, decentralized system, avoiding common flooding approach widely spreaded in P2P networks. Moreover, by using this algorithm it is possible to significantly decrease a speed of most common searches in this environment, using cached map-results before performing reduce step.

In the future work, the algorithms must be studied more deeply to determine exactly the importance of different parameters of map-reduce search in real network environment. Additional research should be performed on determining sufficient and needed amount of master nodes for effective request processing depending on network size. An additional speed up could be obtained using effectively distributed master nodes all over the grid network (currently they are grouped around query node, which is not always effective). Another promising method is a possibility of integrating this approach with some parts of common blind search, granting benefits of quick reacting on new node appearance to proposed method.

References

[LU09] Lertsuwanakul L; Unger H.: A Thermal Field Approach in a Mesh Overlay Network: Proc. Of 5th National Conference on Computing and Information Technology, Bangkok, Thailand, May 22-23 2009

[MR09] Magharei N., Rejaie R.: PRIME: Peer-to-Peer Receiver-driven Mesh-based Streaming: Proc. Of IEEE/ACM Transaction on Networking (TON) Volume 17, Issue 4 (august 2009) ISSN 1063-6692

[SWU04] Sakaryan M.W.G.; Unger H.: Search methods in P2P networks: a Survey": Proc. Of Innovative Internet Community Systems (I2CS 2004), June 2003, Guadalajara, Mexico

[ZLX04] Z.Zhuang, Y. Lio, L. Xiao: Dynamic layer management in super-peer architectures.: Proc. Of the 2004 internation conference on parallel processing (ICPP 04), Montreal, Quebec, Canada, August 2004