Concurrency & Parallel Programming

Indexing and resource discovery in peer-to-peer networks

Authors: Tom Peerdeman &
René Aparicio Saez

Datum: 19-12-2012

1 Introduction

In the modern day, scientists all over the world produce lots of data. Even though lots of work can be processed by computers, the amount of unprocessed work keeps increasing. It is possible to let supercomputers keep up with all the data, however this is expensive and not everbody has acces to these kind of machines. Therefore grid-networks are widely used.

There are many ways to set up grids, and many ways to see what the grid has to offer in terms of resources. This review paper looks at some of the most recent research done in resource discovery. Two different types of new algorithms are discussed. Finally a special resource discovery method, made especially for social media is reviewed.

2 Event-based Resource Discovery

This specific algorithm makes use of publish/subscribe messages, which need three key elements. Subscribers, or resource discovery clients, who want specific information or resources. Publishers, or resource providers, who provide specific information or resources. And brokers, or broker networks. Brokers are used for matching subscribers and publishers and serves as a router between them.

What makes this resource discovery interesting is that it considers if the resource is static or dynamic. Besides this consideration it also is necessary to know if resources are used only once or that they are going to be used continuously. When resources are needed continuously the algorithm checks if in the time the subscriber is still busy, new resources are found that the subscriber might need.

The system makes use of a so called 'Modeltype'. For resources this Modeltype can have the value 'Static' or 'Dynamic'. When a Subscriber needs resources he can ask for a Modeltype. The four possible Modeltypes he can ask for are: 'Static', 'Dynamic', 'Static Continuous' and 'Dynamic Continuous'. Each publisher tells what resource he has to offer. Static resources are denoted with numerical operators ($<$, $\leq$, $>$, $\geq$, $=$), whereas dynamic resource don't have a numerical operator. For example, a static storage disk with 200GB could look like this: [disk,$\leq$,200]. For a dynamic storage disk this would look like [disk, 200].

2.1 Resource discovery

There is a difference in resource discovery when the resource is static and when the resource is dynamic. When the resource added to the network is static, the publisher of this resource floods his resource description (Modeltype, OS, storage memory etc.) to all the brokers in the network in an advertisement. The brokers then cache this resource information. When a subscriber wants a specific resource, all it has to do is send a message to the broker and the broker will reply with the resource information he got earlier.

When the resource is dynamic, the publisher will only send his resource description to a single broker. The broker will recognize this, because the Modeltype will be 'Dynamic'. When the resource description is updated, it only has to be changed in the broker. When a dynamic resource is needed, a message is sent to brokers that match the resources. These brokers are referred to as 'edge brokers'. These edge brokers return the latest resource description to the subscriber, as long as the description still matches the conditions set by the subscriber.

2.2 Continuous usage

When the subscribers wants a continuous use of resources, there are slight differences. When the subscriber wants to continuously use static resources, the broker remembers that the subscriber issued a 'Static Continuous' Modeltype request. The resource given to the subscriber will be put in a table, and will continuously be given to the subscriber. The subscriber therefore does not have to issue new requests.

When a subscriber wants to use dynamic resources continuously, the algorithm works practically the same as with the non-continuous algorithm. The only difference is that when the resource changes, this is updated and sent back to the subscriber. When a subscriber does not want to use the resource anymore, he can issue a message telling the broker that he is done using that resource.

3 Resource Discovery Tree

This algorithm proposes to use a hierarchic way for resource discovery, using a tree structure with bitmaps to represent resource information. All the information and attributes of a resource are transformed into bitmap representations. The algorithm makes a difference between quantitative attributes and qualitative attributes. quantitative represent attributes like memory size or CPU speed. For all these attributes a best fit must be found, so that all users do not interfere with each others requests. The qualitative attributes represents information like the operating system.

3.1 Bitmap representations

The bitmap representation of the qualitative attributes are rather easy. Each Qualitative attribute has a set of possible values. For example the OS attribute can have the set {Linux, Unix, Win-XP, Win-Vista, MacOS}. The bitmap of this attribute will have the length of the set. If a bit is set to 1 in the bitmap this means that the resource has the corresponding attribute value of the attribute set.

The Quantitative have a similar set, however this set consists only numbers. The set will look like $\{a_0, a_1, ..., a_x\}$ so that $a_0 < a_1 < ... < a_x$. If in the bitmap a bit is set to 1 at position k, then this means that there exists a resource with an attribute value V such that $a_(k-1) < V < a_k$. If a bit is set at position 0 it means that there exists a resource with an attribute value smaller than a_0. Sometimes a resource is requested with an attribute value outside the range of the attribute, to solve this problem another bit is added at the end of the bitmap. If this bit, which we shall call $x+1$ is set it means that there is a resource with an attribute value larger than a_x. For saving the bits 0 to x we need x + 1 bits, adding the final bit gives a bitmap size of x+2.

3.2 The tree

As mentioned, the algorithm makes use of a tree structure. "Each node inside the tree represents a grid site" [2]. To make communication simple, each node only has to remember the IP addresses of their parent and, if they have any, their children. All the nodes that are not leaves are called "Index Servers" or IS-nodes. Every IS-node must know the resource information of their children, using the bitmaps provided by them. Every resource also has a status, free or occupied. When a request is made for a resource, the query is sent to a tree node. This can be either a IS-node or a leaf node. The node looks if it matches the request. If not the message is forwarded to the parent node. If an IS matches a request to resource information of one of his children nodes, the request is sent down the tree toward the child. When finally the resource is found, the resource will be reserved. The status then changes to occupied, until the user releases the resource. This is updated in the parent IS-node.

3.3 Data structure and Updating

Each IS-node has two bitmaps, one local resource and one index bitmap. It also has a counter array, to store the resource information in. Leaf nodes only have a local resource bitmap. The index bitmap stores the resource information of its children. The counter array represents the amount of each resource the IS-node knows it has in its children. When a resource is occupied, the counter array will update, so that the IS-node knows it cannot use that resource anymore. This is done by lowering the correct counter value with one. When a resource is reclaimed, it has to update all the changed values. When there was a change in disk space for example. The node therefore sends the new information up to its parent node. When any update happens, the parent node is locked, to prevent race conditions. When the node is unlocked, the update is completed. Some attributes change more frequently. Therefore these attributes are given a threshold. If this threshold is not passed, no update will occur.

3.4 Failed Nodes

Failed nodes are detected by two measurements. Firstly, if the user request is not acknowledged in a specific time period, the node that did not received the acknowledgement considers the node as down. Secondly, parent nodes and children nodes send each other messages once in a while, to inform that everything is ok. To make sure that no information is lost if a IS-node is fails, a backup is saved in one of its children. If the backup node fails, a new backup is made. If the IS-node fails, the child will inform all the former children of its failure, and will take its place as parent node.

4 Social peer-to-peer

In social networks people often know contacts that have knowledge about resources they need. Peer to peer nodes don't have this knowledge about other peers available. This makes resource discovery hard because the peers don't know where to forward their query's to. The social peer-to-peer system tries to solve this by mimicking human behaviour in social networks.

4.1 The algorithm

In human networks people tend to remember their gained knowledge from social interactions. The social p2p nodes don't differ in this, the nodes keep an index of topics and the peer nodes that know more about this subject. When a search is conducted for information this index is updated by new entries which contain the nodes that responded to the search. Also the index is updated by removing data that has become invalid.
The maintaining of this index has the purpose of focusing the future search. If a node searches for a topic that resembles to a topic in its index, it is probably that the associated nodes also know something about this new topic. Also if these nodes don't know anything about this topic, they can probably use their index which is probably filled with topic relevant to the requested topic to find peers that do know anything about it.

When a node gets receives a query it can co trough 3 phases. The first phase is searching it's local index to peers associated with the exact topic. Here lies the preference in peers by the time it was last updated. However this phase is not very likely to give a lot of peers, the peer then goes into the second phase.
The second phase also searches the local index, however this time it searches for peers with topics in the interest area of the requested topic. The category's used are given by the Open Directory Categories [1]. The preference of the second phase lies in the amount of relevant topics in the category. Also if two peers have the same amount of relevant topics the preference of these topics is determined by their peers response time. The query is then forwarded to the relevant nodes, for each node where the query could be successfully forwarded a counter is increased. When the counter reaches a constant threshold the algorithm is done and terminates. However if the threshold is not reached yet and there are no more relevant nodes in the index the peer goes into the last phase.
The third phase is simply picking random peers from the index and forwarding the query to them.

The problem with this is that the modern day computers don't have infinite data storage, therefore the index is finite. In the social p2p this is solved by using the Least Recently Used (LRU) algorithm. When the index reaches it maximum capacity and more data needs to be added, the algorithm will drop the item that was last used before all other items.

4.2 Network forming

The connection between peers are build according to the searches. When a peer searches for some topic and it gets a response from another peer, the other peer initiates a connection to exchange the information. This connection is kept open while the peer is alive, the only moment when a connection is closed is when the peer removes the peer from it's local index. The result of this is that a peer is most likely to connect to peers with the same interest as the peer itself. Due to this principle virtual community's are formed with peers that have globally the same area of interest, which we see also in the human behaviour.

References

[1] The open directory project. `http://dmoz.org`.

[2] R.S. Chang and M.S. Hu. A resource discovery tree using bitmap for grids. *Future Generation Computer Systems*, 26(1):29–37, 2010.

[3] L. Liu, N. Antonopoulos, and S. Mackin. Social peer-to-peer for resource discovery. In *Parallel, Distributed and Network-Based Processing, 2007. PDP'07. 15th EUROMICRO International Conference on*, pages 459–466. IEEE, 2007.

[4] W. Yan, S. Hu, V. Muthusamy, H.A. Jacobsen, and L. Zha. Efficient event-based resource discovery. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, page 19. ACM, 2009.