Use Case: #1 Seller submits an offer

Scope: Marketplace

SuD: Marketplace Information System

Actors: Seller, Trade Commission

Main success scenario specification:

1. Seller submits item description.
2. System validates the description.

...

Figure 1: Fragment of a textual use case.

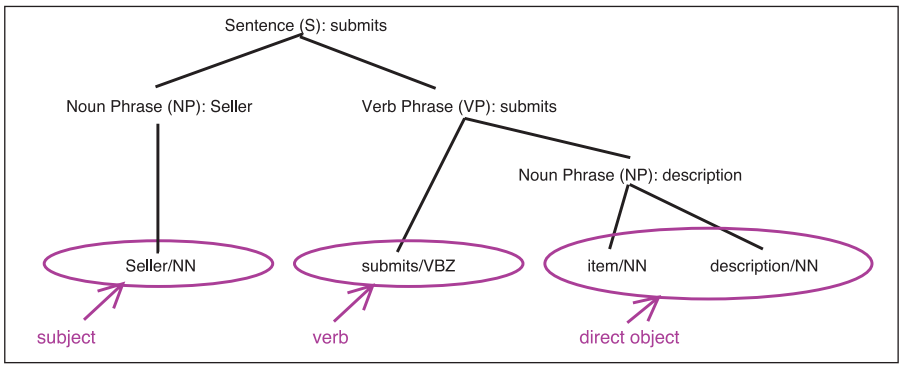

Figure 2: Parse tree of sentence "Seller submits item description".

In a subsequent stage, we convert the whole use case into a behavior protocol (IEEE Trans. Software Eng. 28(11), developed within our group), a formal method featuring, eg, a decidable compliance relation. We have implemented the conversion outlined here in the Procasor tool (Pro-case stands for Protocol use case). As the tool is employing a statistical natural language parser, the output is only an estimate of the behavior specification; nonetheless, it can prove useful in obtaining an initial design of interfaces of the future system. Our future work focuses on enhancing the transformation to produce matching event tokens for complementary actions; the long-term goal is to apply reasoning available for behavior protocols to detect inconsistencies in use case specifications.

Link:
http://nenya.ms.mff.cuni.cz/

Please contact:
Vladimir Mencl, Charles University Prague/ CRCIM,Czech Republic
Tel: +420 2 2191 4232
E-mail: vladimir.mencl@mff.cuni.cz

GHT*: A Peer-to-Peer System for Metric Data

by Michal Batko, Claudio Gennaro and Pavel Zezula

GHT* is a scalable and distributed similarity search structure that has been specifically designed to support metric space objects. The structure is based on the Peer-to-Peer (P2P) communication paradigm. It is scalable, which means that the query execution achieves practically constant search time for similarity range queries in data-sets of arbitrary size. Updates are performed locally and node splitting does not require sending multiple messages to many peers.

P2P communication paradigms are quickly gaining in popularity due to their scalability and self-organizing nature, forming the basis for building large-scale similarity search indexes at low cost. However, most of the numerous P2P search techniques proposed in recent years have focused on single-key retrieval. A good example is the Content Addressable Network (CAN), which is a distributed hash table abstraction over Cartesian space.

Our objective is to develop a distributed storage structure for similarity search in metric spaces that would scale up with (nearly) constant search time. The advantage of the metric space approach for data searching is its 'extensibility', allowing us to perform exact match, range, and similarity queries on any collection of metric objects. Since any vector space is covered by a metric space with an appropriate distance function (for example the Euclidean distance), even n-dimensional vector spaces are handled easily. Furthermore, there are numerous metric functions able to quantify similarity between complex objects, such as free text or multi-media object features that are very difficult to manage. For example, consider the edit distance defined for sequences and trees, the Hausdorff distance applied for comparing shapes, or the Jacard coefficient, which is often used to assess the similarity of sets. Much research is now focused on developing techniques to structure collections of metric objects so that search requests can be performed efficiently.

A convenient way to assess similarity between two objects is to apply metric functions to decide the closeness of objects as a distance, which can be seen as a measure of the objects 'dis-similarity'. For any distinct metric objects x, y, z, the distance must satisfy the properties of reflexivity, $d(x,x) = 0$, strict positiveness, $d(x,y) > 0$, symmetry, $d(x,y) = d(y,x)$, and triangle inequality, $d(x, y) \leq d(x, z) + d(z, y)$.

The distributed environment is composed of network nodes (peers), which hold metric objects, execute similarity queries on stored data and communicate with other peers. Every peer is uniquely denoted by its identifier PID. Peers hold data in a set of buckets. Each bucket has a unique identifier within a peer, designated as BID. Each peer also maintains a tree structure called the address search tree (AST). This is used to route similarity range queries through

the distributed network. AST is based on the metric data partitioning principle known in the literature as the Generalized Hyperplane Tree (GHT). GHT is a binary tree with metric objects stored in leaf nodes implemented as buckets of fixed capacity. Inner tree nodes contain selected pairs of objects called pivots. Respecting the metric, the objects closer to the first pivot appear in the left subtree and those closer to the second are in the right subtree.

We start by building a tree with only one root node represented by a bucket B_0 When the bucket B_0 is full we must split it: we create a new empty bucket B_1 and move some objects (half of them if possible) into it to gain space in B_0. See Figure 1. The split is done by choosing a pair of pivots p_1 and p_2 from B_0 and moving into bucket B_1 all objects o that satisfy the condition $d(p_1, o) > d(p_2, o)$.

Pivots p_1 and p_2 are associated with a new root node and thus the tree grows one level. This split algorithm can be applied on any leaf node and is an autonomous operation (no other tree nodes need to be modified).

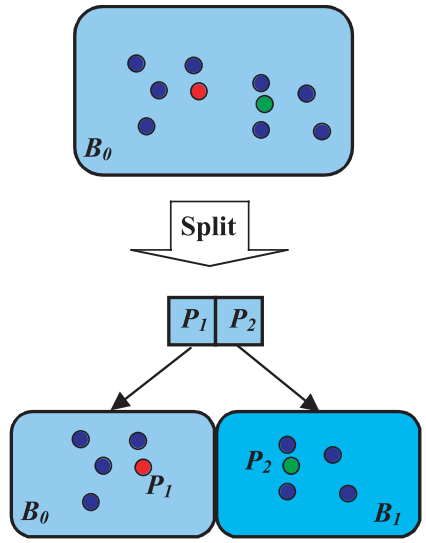

Figure 1: Split of a GHT bucket.

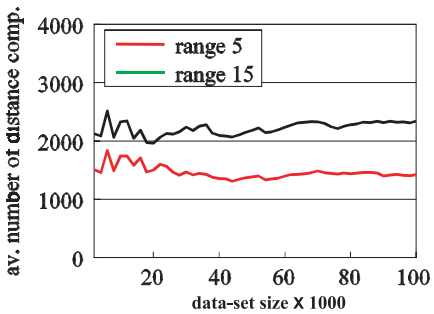

Figure 2:
Parallel response time of a range query.

The most important advantage of GHT* with respect to single-site access structures is its scalability through parallelism. As the size of a data-set grows, new server nodes are plugged in and both their storage and the computing capacity are exploited. Figure 2 shows the result of the parallel response time of a range query, determined as the maximum of the costs incurred on servers involved in the query plus the search costs of the AST. For evaluation purposes, we use the number of distance computations (both in the AST and in all the accessed buckets) as the computational costs of a query execution. This experiment shows how the parallel search time becomes practically constant for arbitrary data volume and the larger the dataset the higher the potential for interquery parallelism. This is the most important feature of GHT*.

Our future work will concentrate on strategies for updates (object deletion) and pre-splitting policies, plus more sophisticated mechanisms for organizing buckets.

Please contact:
Claudio Gennaro, ISTI-CNR, Italy
Tel: +39 050 315 2888
E-mail: claudio.gennaro@isti.cnr.it

CWI's FACS Cluster - Facilitating the Advancement of Computational Science

by Stefan Blom, Sander Bohte and Niels Nes

The Centre for Mathematics and Computer Science (CWI) performs a wide range of research. Consequently, its high-performance computing facilities must meet diverse requirements. This has led to the purchase of a cluster with an asymmetric core-edge design. Under the influence of this asymmetric structure, some cluster applications are evolving into Grid applications.

Many research groups at CWI share a need for high-performance computer facilities. In the FACS (Facilitating the Advancement of Computational Science) project, three of these groups combined their resources to set up a large shared cluster. Since each user group has its own specific demands on memory, CPU power and bandwidth, the FACS cluster deviates from the more usual symmetric design. The cluster's edge serves as a compute farm. It contains several groups of machines, whose main task is to provide CPU cycles. The core consists of a small number of machines, connected by a high-bandwidth low-latency network (Infiniband) and each equipped with a large amount of memory. Currently, we have 22 (mostly dual CPU) machines in the edge and two quad AMD Opterons with 16G each in the core.

The Database Architectures and Information Access group works on database technology and needs machines with both as much memory as possible and huge, fast storage. For this type of work, the bandwidth between memory and storage is critical. Furthermore, part of the research consists of designing highly efficient systems for extremely large databases. Since the research considers typical database hardware set-