Title : Efficient synchronization of remote files.

Co-Encadrants :
Andrei ROMASHCHENKO, `andrei.romashchenko@lirmm.fr`,
Alexander SHEN, `alexander.shen@lirmm.fr`

Keywords : communication complexity ; random hashing

Prerequisites : The candidate should have programming skills and some basic knowledge in communication complexity and/or coding theory.

Abstract : It is well known that some programs (e.g., `rsync`, or `btsync`, or some cloud services) can synchronize huge files in fraction of the time required to transmit the file — using the fact that the previous version of the file is available on the other side and only a few changes were made in the new version. These programs try to find the unchanged parts of the files, use some digital fingerprints to check they are indeed the same in both versions of the file, and avoid transmitting these parts in full.

Is there some theoretical model for this approach ? Can it be used to improve the existing tools ? The answer to the first question is "yes", there are several results in the theory of communication complexity that show how one could synchronize remote files with a small editing distance between them. Still, the existing theoretical approaches are quite limited and, on the other hand, the practical tools are often written *ad hoc* without using any theoretical groundwork.

So the question is twofold. Is it possible to extend the existing theoretical results and obtain faster algorithms for wider class of situations (types of changes between two file versions) ? Is it possible to improve existing practical tools using the known theoretical approaches or, possibly, new ideas ? A good starting point is to try random hashing instead of *ad hoc* fingerprints in existing tools and investigate if it helps in practice.

This internship can be followed by a longer research project, with possibly theoretical or more engineering aims.

More detailed description : We study the following natural question. Assume a `client` and a `server` contain different but *similar* versions of some long file. Denote X_C the version of the file known to the client and X_S the version known to the server. How can we synchronize these files in the most efficient way ? That is, how can the client upload X_C to the server (or, on the contrary, download from the server X_S) in the most economical way, avoiding resending the data that is already available both on the client and the server ?

Clearly, a synchronization of a pair of remote files can be done more efficiently if these files are "close" to each other, in one or another sense. Such a situation happens, for example, if one of the files was subjected to a few edit operations — adding and deleting letters — and the performed revisions are small compared with the entire file. There exist various utilities, open source and proprietary, for efficiently transferring and synchronizing files between networked computers. Arguably the most well known and well documented practical utility of this kind is the open-source tool `Rsync`, see [1,2].

In the literature there are quite a few theoretical publications suggesting different approaches to the problem of file synchrozination. Some of them make mostly theoretical contributions (e.g., [3]) ; the others propose rather practical protocols (e.g., [4,5]).

The usual way to implement an efficient synchronization of files employs the following natural idea : we split the files into small blocks, exchange hash values of these blocks to identify the *common* blocks that can be found in the files on both computers, and then transfer the remaining *non-common* blocks that are found on only one computer. In this procedure we have to compute hash values for a huge number of blocks. The efficiency of the protocol depends on a proper choice of the block size and of the hashing scheme. In this internship we are going to focus on techniques of *random hashing*.

The idea of randomization have been extensively studied in the theory of communication complexity ; it is well known that for some communication problems randomized protocols are much more efficient than deterministic ones, see, e.g., the classical textbook [6]. In this vain, we suggest to investigate (theoretically and experimentally) the efficiency of probabilistic techniques in approaches to the problem of file synchronization

Bibliography :

[1] Andrew Tridgell and Paul Mackerras. The rsync algorithm. (June 1996).

[2] Andrew Tridgell. Efficient Algorithms for Sorting and Synchronization. (1999).

[3] Alexandre V. Evfimievski, A probabilistic algorithm for updating files over a communication link. Theor. Comput. Sci. 233(1-2) : 191-199 (2000) PDF

[4] Hao Yan, Utku Irmak, and Torsten Suel. Algorithms for low-latency remote files synchronization. The 27th Conference on Computer Communications. IEEE, 2008. PDF

[5] Gupta, Deepak, and Kalpana Sagar. Remote file synchronization single-round algorithms. International Journal of Computer Applications 4.1 (2010) : 32-36. PDF

[6] E. Kushilevitz and N. Nisan, Communication Complexity (1997).