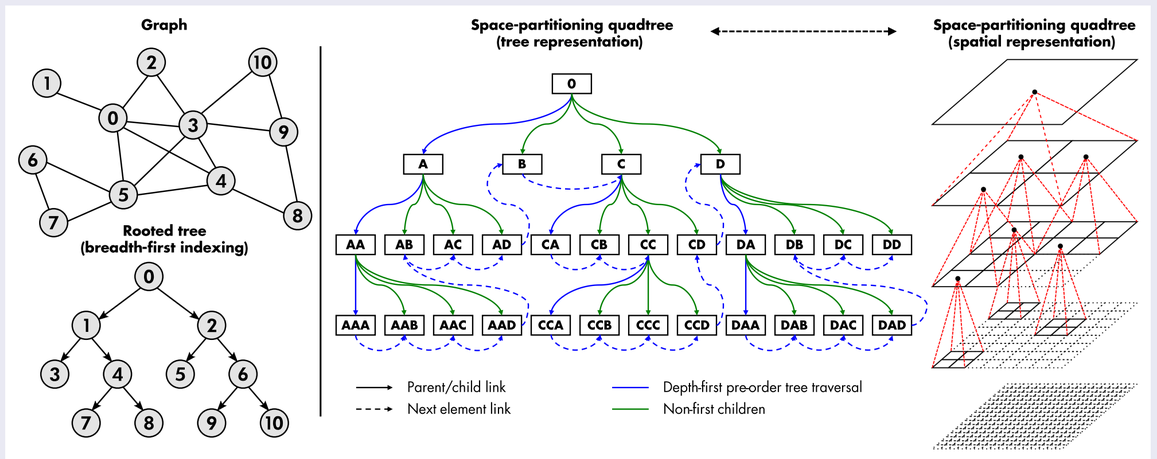
SI2-SSE STAMLA

Scalable Tree Algorithms for Machine Learning Applications

Abstract

The STAMLA project aims at designing efficient tree data structures and related algorithms for high performance machine learning applications. It finds its origins in computational astrophysics research where data structures and memory access patterns have become one the main bottleneck of simulation codes. Tree-based algorithms include, amongst many other examples, decision trees and Monte-Carlo tree search. This last one played a central role in the recent advances in artificial intelligence around the game of Go.

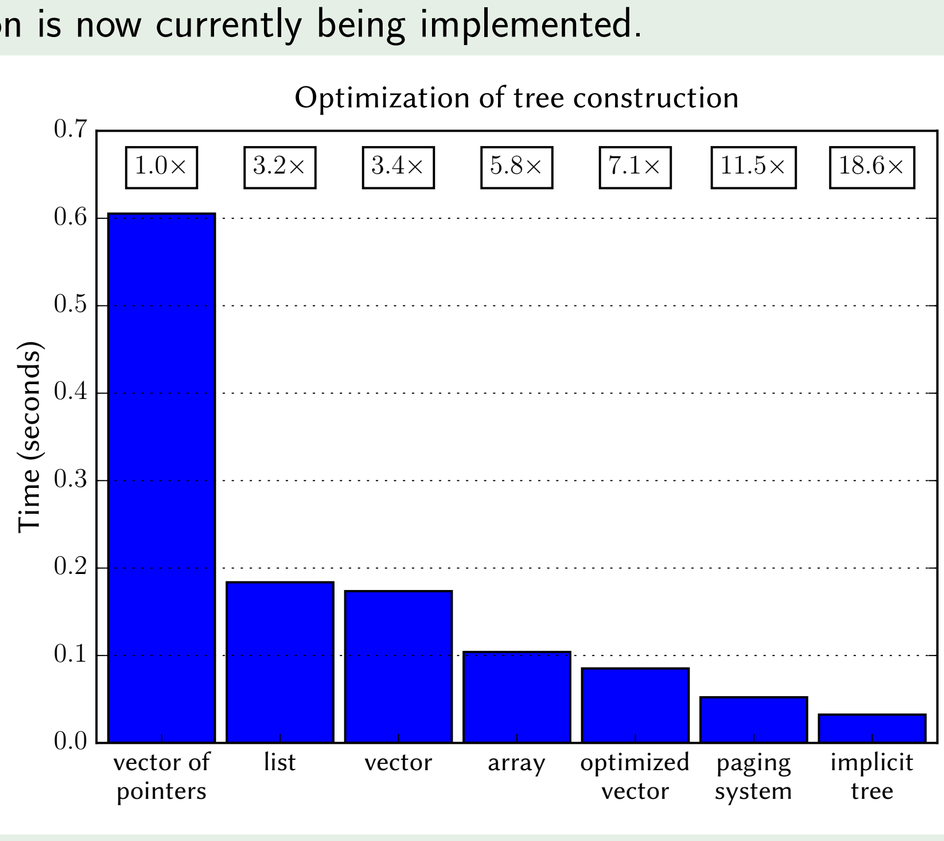
Trees and graphs



While graphs are generally stored as adjacency lists or matrices, rooted trees tend to be stored using explicit links between nodes. However, given a set of constraints at compile-time, like the arity of the tree, or its maximum depth, more efficient representations are available. In particular, implicit representations relying on indexing strategies demonstrate excellent cache-friendliness and vectorization properties. The STAMLA project aims at generating such representations at compile-time given a set of constraints depending on the application domain. Toward this goal, software architecture plays a central role to make optimizations as generic possible and impact as many application domains as possible.

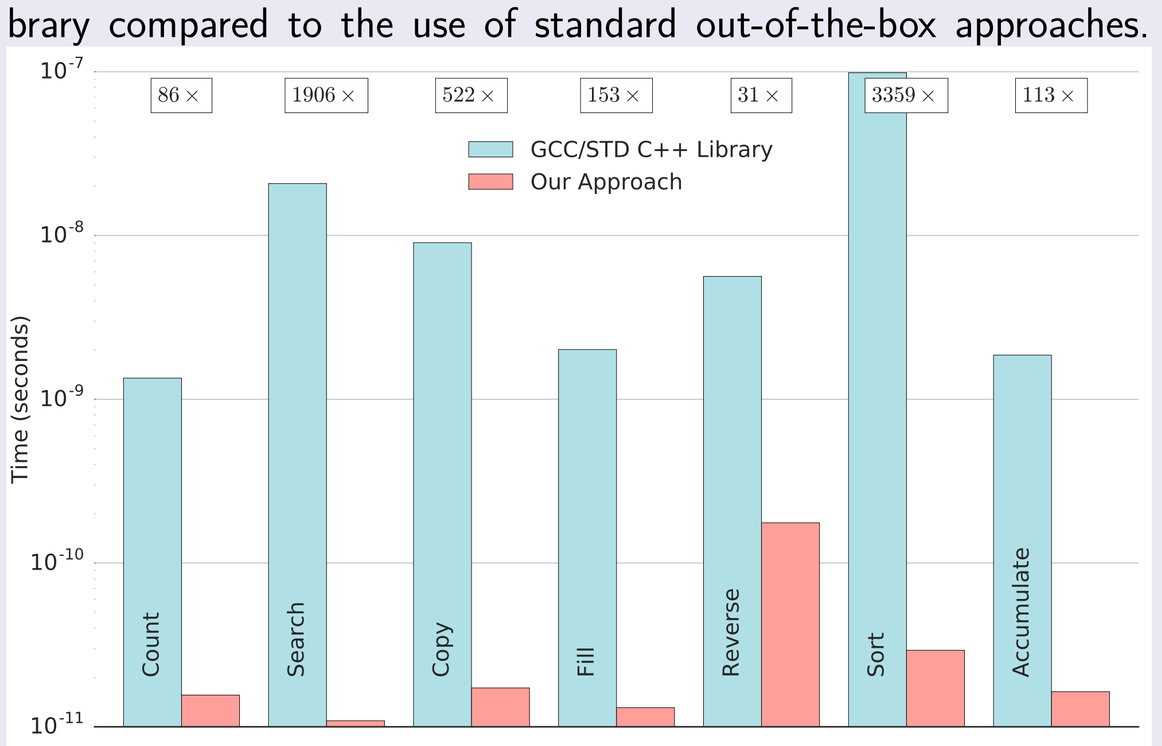
Example of optimization: kd-trees

Preliminary results have already been obtained on the optimization of kd-trees. A kd-tree is a type of space-partitioning tree to speed-up the search of elements within a space of a given dimension. They are widely used in high performance computational sciences, in geolocation applications and in clustering techniques. Although the algorithms are well known and many implementations already exist, better results have been obtained within the STAMLA project using advanced memory allocation techniques. By optimizing memory allocation, reusing already-allocated memory, and simplifying access patterns, significant speed-ups can be achieved as illustrated on the benchmark results below. As an extension of this work, a distributed version is now currently being implemented.



The bit library

The STAMLA project is closely related to the development of The Bit Library, a C++ library aiming at facilitating the design, implementation, and maintenance of high performance bit manipulation algorithms. After several years of work, the library is under formal review by the ISO C++ Standards Committee to become a part of the next version of the language and its standard library. As such, these new bit manipulation tools will be available on any computer system equipped with a recent compiler. The STAMLA project makes use of the library and in particular of bit iterators to speed up hashing and indexing techniques for implicit trees. The figure below illustrates speed-ups that have been obtained on several basic bit manipulation algorithms when using The Bit Library compared to the use of standard out-of-the-box approaches.



The conceptification of trees

The C++20 language will introduce new programming approaches, and in particular concept-based programming. The related tools, already available as experimental features in compilers, allow to constrain types and facilitate generic programming. The STAMLA project explores these new approaches, in particular to rigorously define and constrain tree types, ranging from binary rooted trees, to tries, and including suffix trees, trees for Monte-Carlo tree search, and Abstract Syntax Trees. The idea is to provide a robust and generic software architecture to provide users with means to implement any tree data structure given behavioral and performance requirements. Concept-based programming allow to clearly identify customization points to provide advanced users with ways to finely tune memory access patterns.

Current work and future directions

Current work focuses on finishing the standardization of The Bit Library, on the conceptification of trees, and on real-world applications as a way to test the library. The Bit Library has been approved by the C++ Library Evolution Working Group last year, and the final wording is currently being refined. As a consequence it should be a part of the Library Fundamentals v3 Technical Specification that will be released in 2019. The conceptification of trees is also on the way to achieve at the same time genericity, performance, and ease-of-use. Current work focuses on memory layouts aspects and especially on the design of customization points to allow users to choose between storing or re-computing node information. Finally, on the application side, manipulation of symbolic equations in physics has been taken as a real-world use case of decision trees, abstract syntax trees, and Monte-Carlo tree search. This example will allow to benchmark many aspects of the trees that are being developed within the STAMLA project.