

libMarika – efficient implementation of set and map with integer keys and values

Aleksey Cheusov

vle@gmx.net

<https://github.com/cheusov/libMarika>

Kutaisi, Georgia, 2025

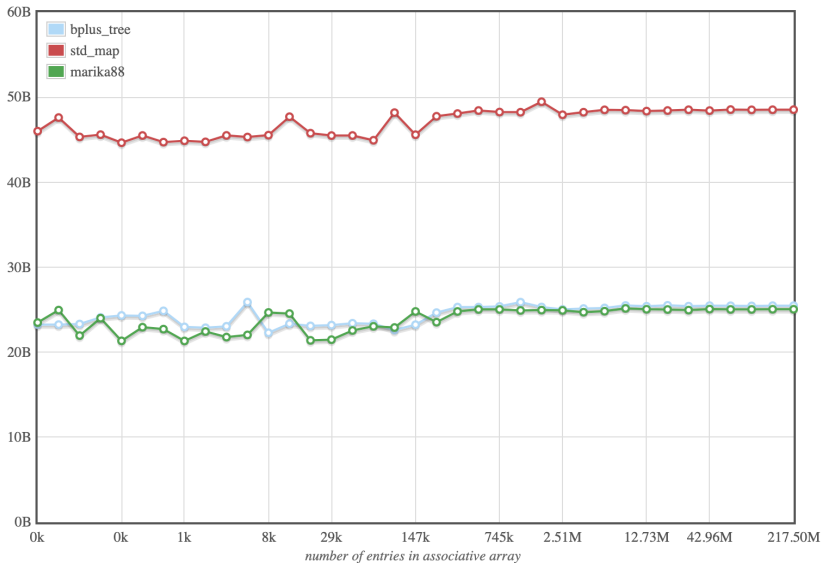
Features

- ▶ Internal data structure is a B+ tree in memory.
- ▶ Both leaf and internal nodes are aligned with cache line (usually 64 bytes). This is a compile-time option.
- ▶ Leaf and internal nodes are 256-byte in size. This may change in the future.
- ▶ The number of elements within the node is stored in lower bits of pointer to this node. As a result node contains more keys and values.
- ▶ The structure of tree node depends on size of key, value and pointer to minize a padding. That is, almost all bytes in the node are used for keys, values and pointers. This is true for 32/64-byte CPUs and 4/8-byte keys and values.

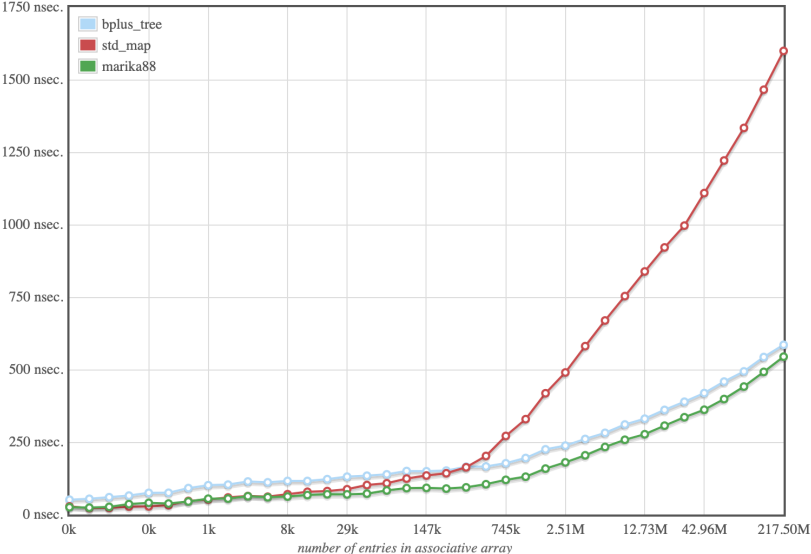
Features

- ▶ Top-level data structure is represented as a single pointer, thus allowing nested B+ trees.
- ▶ API for map is very similar to JudyL. API for set is very similar to Judy1 <https://judy.sourceforge.net/>.

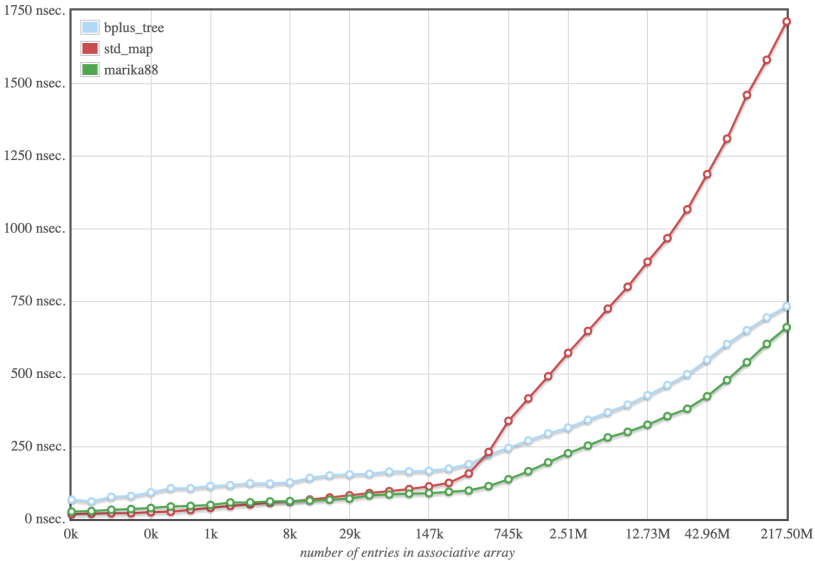
Benchmarks (AMD Ryzen 7 3700X), memory consumption



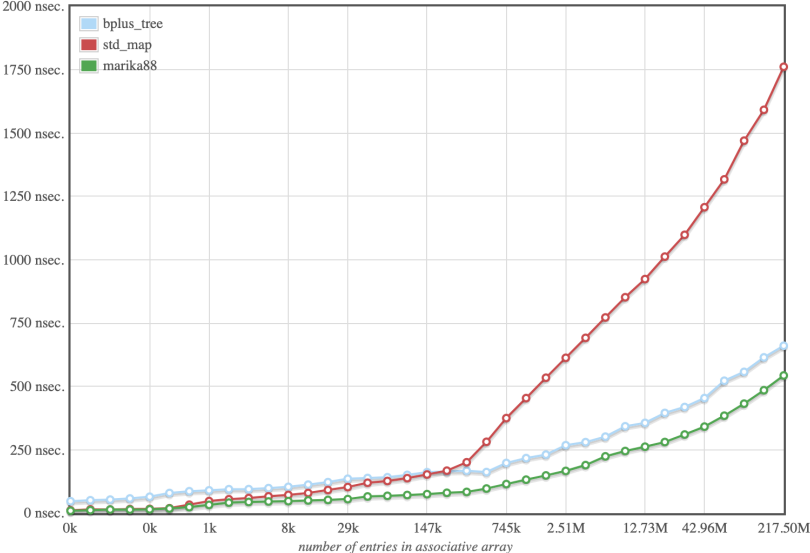
Benchmarks, insertion of new item



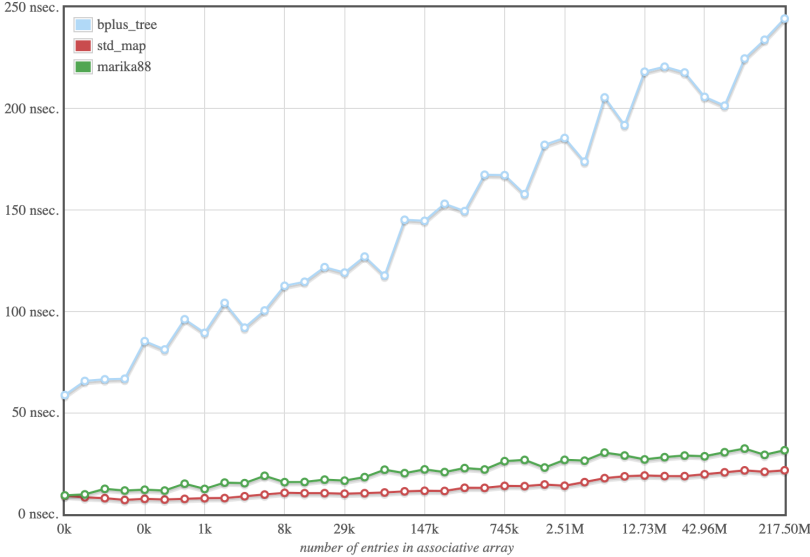
Benchmarks, insertion of existing item



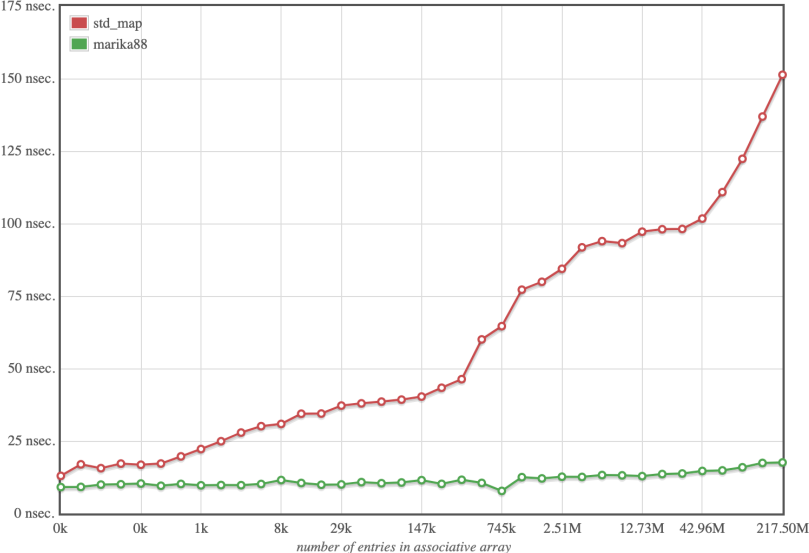
Benchmarks, search of existing item



Benchmarks, search of non-existing item



Benchmarks, iteration



Bugs, limitations, plans and ideas

- ▶ Removing of item is not implemented yet.
- ▶ Performance highly depends on **aligned_alloc(64,256)** which is Very inefficient on GNU libc (most Linuxes) and Darwin libc (maxOS).
- ▶ B* tree instead of B+ tree should be implemented.
- ▶ Unordered keys in leaf nodes.
- ▶ Wrappers for C++, Python, Julia, Ruby etc.

Literature

- ▶ "Algorithms for Modern Hardware" book by Sergey Slotin
<https://en.algorithmica.org/hpc/>
- ▶ Google's tcmalloc
<https://google.github.io/tcmalloc/overview.html>
- ▶ jemalloc
<https://jemalloc.net/>
- ▶ Microsoft's mimalloc
<https://github.com/microsoft/mimalloc>
- ▶ Tool for benchmarking
<https://github.com/cheusov/hash-table-shootout>

The End