Subject: Re: Map implementation
Posted by Novo on Sat, 13 Apr 2019 19:25:22 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 10 April 2019 11:37About the only thing that is in question is how to deal with collisions. Some advanced hashmaps might e.g. use binary trees to resolve collisions. I believe that it is not an overal gain (and the fact that it is not widely used in industry makes it likely) and that we should rather invest time to investigate proper hashing techniques.

Anyway, the real benchmark would be to create real world scenario and test there. IMO U++ Index/VectorMap wins that easily.

AFAIK, STL can't use open addressing or other such techniques because it is specified to maintain stable key/value addresses. U++ Index doesn't support that. This is why it is possible to make it faster.

About collisions. There are many ways to deal with them. A classic approach is to store chain either explicitly (a list) or implicitly (you know how to calculate address of the next colliding slot). I saw a paper from Microsoft recommending an overflow area ...

Basically, what I want to say is that there is a million of different ways to design a hash table. IMHO, the best way is to split it into multiple policies, so you can easily replace one part of it without rewriting the whole data structure. :)
I personally couldn't figure out how to do that. :)