Subject: Questions about VectorMap
Posted by cbpporter on Wed, 10 Mar 2010 21:24:27 GMT
View Forum Message <> Reply to Message

I'm a newbie at VectorMap and friend implementations (AIndex) and never written a hash map implementation myself, though I have used a lot and read enough documentation about them so I could sit down and write a reasonable, yet probably somewhat naive one of the top of my head.

In my newest project I have a centralized string repository. Every entity that has a string like name refers to this repository. The repository must be indexable and streamable, so a constant index that survives the entire run time of the application during multiple read/writes in streams is desirable.

The most obvious solution is th have a Vector<String>. Items are added at the end, so index uniqueness is guaranteed. Problem is that lookup is slow on such vectors: O(n) worst case. Using binary vector is not possible because that would shift the indexes around.

So why not use VectorMap? Right now I'm using a VectorMap<String, int>. The int is the index in another Array<String> which preserves the order of indexes, and the VectorMap is used only for fast lookup of the index.

But VectorMap is also indexed? What guarantees are there regarding this index. As the hash grows, will it get invalidated? Storing pointers to the values is probably a bad idea.

The end result should be that before first streaming you are able to gather all strings, you can practically create a perfect hash function. With perfect hash function a VectorMap becomes practically a Vector.

For anybody experienced in this domain: how does the VectorMap + Array + items that require string having a pointer to the Array item sound? I'm interested in performance, memory is not a huge concern. A high very load is around a few thousand strings. Any better ideas?

PS: I also use a lot of Reserves. One of the big containers reserves around 40 MB. This is overkill, but the strange part is that while hitting Ctrl-Alt-Del, my program does not seems to grow in memory requirement. Is this some clever Windows trick of keeping unused pages out of RAM?