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?

Subject: Re: Questions about VectorMap
Posted by mirek on Thu, 11 Mar 2010 13:20:51 GMT
View Forum Message <> Reply to Message

I think you definitely should learn Index semantics.

It works just like Vector for the most part, but is able to find index of element with given value really fast.

VectorMap is just a quite simple composition wrapper of Index and Vector. Index stores keys, Vector values.

Other than that, it really always behaves just like Vector.(Or two Vectors - imagine VectorMap as Vector<KEY> and Vector<VALUE>).

The only slightly problematic operation is Remove of element, not because it behaves differently than in Vector, but because it is slow (not only it has to move the memory, but also reindex hashtables). That is why there is Unlink that just "hides" the key, keeping the element in the place.

Mirek

Subject: Re: Questions about VectorMap
Posted by cbpporter on Thu, 11 Mar 2010 14:11:22 GMT
View Forum Message <> Reply to Message

I will definitely learn about Index and check out the code. for the easy tasks I had before, I could use it without deeper knowledge.

So if it behaves like Vector, if it is empty, I should have FindAdd("foo") = 0 and following FindAdd("bar") = 1 used in this order? And I can trust these indexes after a resize or hash reindex?

Also, if it behaves like a Vector<Key>, it will store the key, not just the hash. Pointer invalidation for both.

And what about ArrayMap. Does it behave like two Arrays or a Vector and an Array?

Subject: Re: Questions about VectorMap
Posted by mirek on Thu, 11 Mar 2010 22:24:01 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 11 March 2010 09:11I will definitely learn about Index and check out the code. for the easy tasks I had before, I could use it without deeper knowledge.

So if it behaves like Vector, if it is empty, I should have FindAdd("foo") = 0 and following FindAdd("bar") = 1 used in this order? And I can trust these indexes after a resize or hash reindex?

Yes.

Quote:
Also, if it behaves like a Vector<Key>, it will store the key, not just the hash.

Yes.

Quote:
Pointer invalidation for both.

Well, hash storage is implementation detail, not accessible by client code. Interface specifies invalidation for keys...

Quote:
And what about ArrayMap. Does it behave like two Arrays or a Vector and an Array?

Vector and Array. Simply because it is the most practical.

There is also ArrayIndex, which is Array counterpart of Index, but I do not remember I have ever used it. Theoretically, it would be possible and simple to create Array->Array map with it, but somewhat it is not ever useful. Maybe because most keys are simple types that store well into Vector.

Mirek