Subject: Re: Impressive improvement in std::vector when dealing with raw memory.
Posted by Lance on Mon, 14 Nov 2022 16:24:16 GMT
View Forum Message <> Reply to Message

I have uploaded my (incomplete, simplified) std::vector implementation to GitHub. It's just a proof of concept. It's a refactoring of Upp::Vector, using std::vector interfaces and Upp::Vector memory management facilities and logic. It performs at par with(if not marginally faster than)Upp::Vector and should generate smaller executable size (which can be proved in theory and is tested true in practice).

Once I found it handles raw bytes slower than std::vector, I lose confidence/interests to continue. But it suffices to demonstrate my point: std::vector can be trained to handle trivially relocatable class object just as good as Upp::Vector (I expect it happen in not long future, if I can do that, why not all those a million times smarter guys), and more than that, in many situations, we can make the vectors work nicely with non-trivially relocatable objects, some times with tremendous performance gain.

I am able to relocate a Ctrl (even though that's really little point to do that --- in certain cases, it makes sense to put a lot of dynamically allocated child Ctrls of same [or similar, a different story] kind in a vector instead of Upp::Array for memory efficiency and less fragamentaion), and I make a std::basic_string relocatable, even though in this case I actually got a performance penalty: basic_string is too small to gain anything from move raw bytes then adjust affected pointers. But you can conceive there are lots of cases where moving raw bytes then adjust a couple of back pointers make sense.