Subject: Re: Which is the biggest drawback of U++ "unpopuliarity"?
Posted by tvanriper on Sat, 26 Apr 2008 11:41:31 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 26 April 2008 02:11tvanriper wrote on Fri, 25 April 2008 20:36
If I have it right, your primary concern with std:: involves its relatively terrible performance,

Well, not really. If am to put it in a very simple way, the main problem with std:: is that it makes you wish the C++ had garbage collector....

I can appreciate that. I get frustrated sometimes, trying to write code that doesn't require 'new' when using std::. And I strongly agree with you that an object belongs somewhere.

luzr
tvanriper
If that's the concern, someone could potentially help you find a way to achieve the same performance you currently get with NTL, while using a more std::-like interface.

Well, what would be that? Something like these macros at the end of Core/topt.h?

(I only removed the quoted code to reduce the size of the message.)

Partially... but you also appear to introduce difference concepts for use with your collections than the standard. I would expect, if you were to incorporate this into boost::, you'd provide concept-checking mechanisms that would cause clear compiler errors if someone attempted to use an inappropriate data type... one that didn't support your collections' concept.

luzr
tvanriper
I could, of course, be mistaken. I'm not completely clear on why you feel these are so incompatible... as perhaps I'm not 100% clear on your design goals, or I'm ignorant of the fundamental problem you see in std::.

The real trouble starts with the fact that you cannot use std::string as map keys. You cannot use any concrete class defined in std:: as element of any Vector flavor U++ container.

So far, the main "incompatibility complaint" was that "U++ guys seem to define their own containers and string". This is not easy to fix

I do not view this, necessarily, as a problem.

I note that boost currently has specialized containers to solve specific sorts of problems. Your containers are no different in that respect.

luzr
tvanriper
I've read and re-read this page, but I still can't quite see how U++ and std:: can be so incompatible that there's no hope of improving the std::-style system to the point of matching U++ performance.

Ah, but you could fix std::. But it is not likely to happen.

Moreover, adopting all U++ tricks into std:: would change its semantics and break existing code.

Bad choice of words on my part... a consequence of trying to write quickly.

By 'improving the std::style system', I didn't necessarily mean 'replace std:: containers'. I meant that you could create another set of containers that feel std::-like, but have the qualities you prefer.

luzr
tvanriper
I only pose this idea because it feels to me like you and boost have similar goals. I could, of course, be wrong. I know, for example, that boost has less of an emphasis on performance and more of an emphasis on their idea of 'correctness', so you may differ significantly there. (This is certainly not to say you have no concern for 'correctness', but that you may have a slightly different idea of what is 'correct' from boost).

Oh, I have a very strong concern for 'correctness' - to the degree that I often rather break existing code by fixing some "incorrectness" in U++ Core.

Also, please, do not think I am not aware about boost or that I think these people are stupid. Of course not, boost is a very good effort and the code is pretty good.

I just feel U++ is not a good fit there. It is almost like suggesting boost to adopt Java

Well, firstly, I apologize if I seem pushy, or condescending. I suppose I just wanted this conversation to take place, just to fully explore the idea.

Secondly, while it might be true that this is like suggesting boost adopt Java, I wanted to at least explore the idea to see if that was really true. Other folks can eventually read all of this and see that we've covered all the points to cover on the subject.

At the very least, I thank you for your patience.

luzr
BTW: I mostly care about "optimality" with U++. If I would care about "popularity" more, I would certainly use another path and boost would be the part of it.

That is understandable.

luzr
tvanriper
Perhaps someone could submit an article to Dr. Dobb's Journal showcasing the use of Ultimate++; that's a fairly popular magazine, at least here in the United States (the CUJ folded to Dr. Dobb's a few years ago, sadly, or I would have recommended it instead).

I guess that would be much better idea:)

Hmmm... well, I need to buy some issues of Dr. Dobb's Journal, review the past articles, and see what sort of article they'd accept. I could probably write something... it's been a long time since I wrote anything for a journalistic venue, but I could probably do it, if they'd accept the article.

Oh, and many thanks to mr_ped for pointing out the other link. I had read that a long time ago, but couldn't remember where I saw it.