Subject: Re: Question / Issue about Vector
Posted by alexn on Wed, 21 Nov 2007 06:25:05 GMT
View Forum Message <> Reply to Message

Mirek,

I really like a lot of attributes and the thought processes of Ultimate++ and I want to use as much of the provided classes as possible. One of the things that initially attracted me, was your approach of trying to address the performance issues of std yet provide many of the coding (methods) and runtime benefits.

Actually, I would think it would be a violation of the Vector class contract if it didn't observe the alignment requirements of the each element. I can not speak for compilers outside of MSVC, as it isn't easy for me to test, but with 7.1 declaring a simple array of an aligned elements delivers the alignment of each element. I struggle to see why the Vector class should be given a free pass.

Before using trying to use Ultimate++, I ended up writing my own "FastArray" class that had many of the positive attributes of the Vector class vs. the standard std::vector class (e.g. moveable), yet providing many of the same methods. I think I actually prefer the Ultimate++ provided methods.

The basic problem I didn't have time to work through was an instrinic way for the class to determine the alignment property of an element when allocating a new block of memory so I defined an optional alignment template attribute that was used by the internal memory allocator. In VC++ I usee the _aligned_memory function to allocate memory. It solved the problem of the address of the first element. sizeof(Element) as the basic increment of memory made sure following elements were aligned.

I think the Vector class is struggling between trying to use the general Ultimate++ heap allocation mechanism that is alignment unaware and adding additional info the the base Vector class to maintain awareness between the allocated address and the aligned address.

I would think that for performance oriented applications this could be an issue. Anyone that tries to take into account the cache alignment of their data structures with the Vector class could be affected by the internal heap manager that assumes an 8 byte alignment is okay for any data element.

IMO, cache and SSE2 alignment is not a low level issue in some number of applications. This becomes even more interesting when trying to move SSE2 based data between UI constructs and the rest of my program.

I can certainly move my old FastArray class over to my Ultimate++ application. I guess I was just hoping to leverage as much of your framework as possible.

Regards,
Alexn