Subject: Re: Core 2019
Posted by Novo on Sun, 09 Jun 2019 19:39:32 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 09 June 2019 12:54Looking at peak profile, it looks like there are very little "small" blocks and most of memory is in those 80 "huge" (that means >64KB) blocks.

Can that be correct?

Mirek
It is hard to tell. I'm not controlling that.
Another problem is that all allocations/deallocations happen in CoWork's threads. I cannot call RDUMP(*PeakMemoryProfile()) inside of CoWork because it will be called at least 181363 times ...

The app is parsing Wikipedia XML dump. It is decompressing a bz2 archive and parsing chunks of XML. After that my own parser is parsing Mediawiki text.
As a first pass my parser is building a list of tokens organized as a Vector<> (I'm not inserting in the middle :))
My parser is avoiding memory allocation at all possible costs. I'm calling Vector::SetCountR and reusing these vectors. When I need to deal with String I'm using my own not owning data string class.
Unfortunately, I cannot control memory allocation with XmlParser. I have to relay on the default allocator.

Ideally, I'd love to see U++ allocator designed like this.
Related papers:
https://people.cs.umass.edu/~emery/pubs/berger-pldi2001.pdf
https://erdani.com/publications/cuj-2005-12.pdf
https://accu.org/content/conf2008/Alexandrescu-memory-alloca tion.screen.pdf

It doesn't have to be a complete implementation of everything. I just would like to be able plug into U++'s allocator in a similar fashion and extend/tune it.

mirek wrote on Sun, 09 June 2019 14:56I would like to get a list of allocations your code is doing so that I can hopefully replicate it and investigate whether there can be anything done to reduce the fragmentation.... I will post temporary changes to get the log tomorrow, if you are willing to help.

Yes, I'm willing to help. I even willing to implement this policy-based allocator. I just need the ability to integrate it into U++. It doesn't have to be a part of U++.