Subject: Changes in hashing
Posted by mirek on Mon, 01 Jun 2020 13:43:45 GMT
View Forum Message <> Reply to Message

Up until now, hash codes in U++ very strictly 32bit dword. It turns out that using 64bit hash codes on 64bit CPUs is actually faster (more bytes can be processed at once), so I have introduced new type, hash_t, which is 32 bit with 32 bit CPU and 64 bit otherwise and changed the code to compute/use 64 bit hashes instead. Results in about 5% improvement in idmap benchmark...

Practical consideration for user types: If type supports hashing by dword GetHashValue() method, it will continue to work just fine, but might be improved by converting that to hash_t. Template specialisation GetHashValue needs to change the return type hash_t (compiler issues error if it is not).

Mirek

Subject: Re: Changes in hashing
Posted by Oblivion on Fri, 05 Jun 2020 10:28:37 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I'm somewhat confused about this.

Just to be clear: Does this mean that the client code can continue to use dword GetHashValue() variant on 64-bit machines?

I'm worried because, e.g, In Terminal ctrl I use a dword hash of incoming string data as unique ID for images and hyperlinks, in each cell.
Changing it to 64-bit would be very expensive (memory consumption will be significantly higher).

Truncating 64-bit to 32-bit probably won't do too much harm here, but still, it means information loss and I'd prefer to avoid that.

Or do I have to separately maintain the 32-bit hash funtions on 64-bit confiurations?

Best regards,
Oblivion

Subject: Re: Changes in hashing
Posted by mirek on Fri, 05 Jun 2020 10:40:47 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Fri, 05 June 2020 12:28Hello Mirek,

I'm somewhat confused about this.

Just to be clear: Does this mean that the client code can continue to use dword GetHashValue() variant on 64-bit machines?

I'm worried because, e.g, In Terminal ctrl I use a dword hash of incoming string data as unique ID for images and hyperlinks, in each cell.
Changing it to 64-bit would be very expensive (memory consumption will be significantly higher).

Truncating 64-bit to 32-bit probably won't do too much harm here, but still, it means information loss and I'd prefer to avoid that.

Or do I have to separately maintain the 32-bit hash funtions on 64-bit confiurations?

Best regards,
Oblivion

Well, first of all, using GetHashValue as unique ID is probably not a good idea, but I guess you mean something slightly different there.

Second, simply truncating to 32-bit bit would indeed, for memhash produced numbers, produce inferior hashes, just like taking lower bits in previous 32-bit incarnation, as fast hashing algorithms tend to accumulate entropy in highest bits. But exactly for this reason we have (for quite a long time) FoldHash function, which IMO seems ideal for you scenario. It takes hash_t and produces dword, while bringing entropy back to lowest bits (BTW, look at implementation, I think it is one of more clever ideas from me...

Mirek

Subject: Re: Changes in hashing
Posted by Oblivion on Fri, 05 Jun 2020 11:02:01 GMT
View Forum Message <> Reply to Message

Quote: first of all, using GetHashValue as unique ID is probably not a good idea, but I guess you mean something slightly different there.

Yeah, The IDs are not "really" meant to be unique. Bad wording. They are used in cache management, and dword is sufficient in my case.

Quote:But exactly for this reason we have (for quite a long time) FoldHash function, which IMO seems ideal for you scenario. It takes hash_t and produces dword, while bringing entropy back to lowest bits

Well, I did not know that, because, you know, lack of documentation... But this is good news. I'll test with FoldHash and look into the code ASAP.

Quote:

(BTW, look at implementation, I think it is one of more clever ideas from me...

You have a lot of clever ideas. The main reason why I prefer U++.

Thank you!

Best regads,
Oblivion
