Subject: Re: BufferPainter::Clear() optimization
Posted by Tom1 on Tue, 19 May 2020 10:35:28 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 19 May 2020 10:49Also, a little note about your testing code: You loop over the same "len" many times and measure that. The problem is that first pass setups branch prediction so all other passes are predicted. If "len" is changing, prediction fails and you might get different results....

Which explains why my tests, which feeds random lens, shows a bit different picture... :)

All in all, I think in the end we will just need to test this with Painter....

Mirek

I wish I came to think of this benchmarking pitfall... I mean the branch prediction. Well, I agree that we need to put it in the BufferPainter environment for real test.

Meanwhile, as you worked on 7, 7a and 8, I prepared 3T2, which avoids the switch and uses ifs instead. Funnily, your 7a does the same, but with table offsets. :)

```
void inline Fill3T2(dword *b, dword data, int len){
 if(len<4){
  if(len&1) *b++ = data;
  if(len&2){ *b++ = data; *b++ = data; }
  return;
 }

 __m128i q = _mm_set1_epi32(*(int*)&data);
 __m128i *w = (__m128i*)b;

 if(len >= 32) {
  __m128i *e = (__m128i*)b + (len>>2) - 8;
  if(len > 4*1024*1024 / 4 && ((uintptr_t)w & 3) == 0) { // for really huge data, bypass the cache
   _mm_storeu_si128(w, q); // Head align
   int s=(-((int)((uintptr_t)b)>>2))&0x3;
   w = (__m128i*) (b + s);
   do {
    _mm_stream_si128(w++, q);
    _mm_stream_si128(w++, q);
    _mm_stream_si128(w++, q);
    _mm_stream_si128(w++, q);
    _mm_stream_si128(w++, q);
    _mm_stream_si128(w++, q);
    _mm_stream_si128(w++, q);
    _mm_stream_si128(w++, q);
   }while(w<=e);
   _mm_sfence();
```

```
 }
 else
  do {
   _mm_storeu_si128(w++, q);
   _mm_storeu_si128(w++, q);
   _mm_storeu_si128(w++, q);
   _mm_storeu_si128(w++, q);
   _mm_storeu_si128(w++, q);
   _mm_storeu_si128(w++, q);
   _mm_storeu_si128(w++, q);
   _mm_storeu_si128(w++, q);
  }while(w<=e);
}

 if(len & 16) {
  _mm_storeu_si128(w++, q);
  _mm_storeu_si128(w++, q);
  _mm_storeu_si128(w++, q);
  _mm_storeu_si128(w++, q);
 }
 if(len & 8) {
  _mm_storeu_si128(w++, q);
  _mm_storeu_si128(w++, q);
 }
 if(len & 4) {
  _mm_storeu_si128(w, q);
 }
 _mm_storeu_si128((__m128i*) (b + len - 4), q); // Tail align
}
```

I really like the w++ incremental pointer logic over the Set4(pointer+offset). This approach seems to give a small improvement on my system.

Next, I will test your 7 + 7a and report against 3T2.

But seriously, we need to put an end to this madness! The bang for the buck is rapidly decreasing as working hours are increasing... :)

Best regards,

Tom