Subject: Re: BufferPainter::Clear() optimization
Posted by Tom1 on Fri, 15 May 2020 23:59:37 GMT
View Forum Message <> Reply to Message

Hi,

I've worked on optimizing the new_memsetd() operation through various buffer sizes up to 8M and here's the best I can come up with (at least this night...). With CLANG it seems to be beneficial to use the Mirek's new MemSet() for buffer sizes above about 1M, but below that and also with MSBT19 / MSBT19x64 the result is better without. (This algorithm is especially efficient with small fills and therefore should work well as a BufferPainter filler too.) For best results, there are separate versions for 32-bit and 64-bit code. (The '#ifdef WIN64' obviously only works on Windows, but I think there was some other flag on Linux for detecting a 64-bit environment. Please apply that flag, whatever it is, if you test on Linux, etc...)

```
#ifdef WIN64

inline void new_memsetd(dword *t, dword data, int len){
#ifdef COMPILER_CLANG
 if(len>1024*1024){
  MemSet(t,data,len);
  return;
 }
#endif
 if(len&1) *t++=data;
 len>>=1;

 uint64 *w=(uint64*)t;
 uint64 q=data;
 q = (q << 32) | data;

 switch(len) {
  default:{
   uint64 *lim = w + len;
   while(w < lim) *w++ = q;
   break;
  }
  case 16: w[15] = q;
  case 15: w[14] = q;
  case 14: w[13] = q;
  case 13: w[12] = q;
  case 12: w[11] = q;
  case 11: w[10] = q;
  case 10: w[9] = q;
  case 9: w[8] = q;
  case 8: w[7] = q;
  case 7: w[6] = q;
  case 6: w[5] = q;
```

```
  case 5: w[4] = q;
  case 4: w[3] = q;
  case 3: w[2] = q;
  case 2: w[1] = q;
  case 1: w[0] = q;
 }
}

#else

inline void new_memsetd(dword *t, dword data, int len){
#ifdef COMPILER_CLANG
 if(len>1024*1024){
  MemSet(t,data,len);
  return;
 }
#endif
 switch(len) {
  default:{
   dword *lim = t + len;
   while(t < lim) *t++ = data;
   break;
  }
  case 16: t[15] = data;
  case 15: t[14] = data;
  case 14: t[13] = data;
  case 13: t[12] = data;
  case 12: t[11] = data;
  case 11: t[10] = data;
  case 10: t[9] = data;
  case 9: t[8] = data;
  case 8: t[7] = data;
  case 7: t[6] = data;
  case 6: t[5] = data;
  case 5: t[4] = data;
  case 4: t[3] = data;
  case 3: t[2] = data;
  case 2: t[1] = data;
  case 1: t[0] = data;
 }
}

#endif
```

The updated benchmarking code:

```
 RGBA c = Red();
```

```
 int bsize=8*1024*1024;
 Buffer<RGBA> b(bsize,(RGBA)Blue());

 String result="\"N\",\"Fill()\",\"new_memsetd()\",\"MemSet()\"\r\n";
 for(int len=1;len<=bsize;){
  int maximum=100000000/len;
  int64 t0=usecs();
  for(int i = 0; i < maximum; i++) Fill(~b, c, len);
  int64 t1=usecs();
  for(int i = 0; i < maximum; i++) new_memsetd((dword*)~b, *(dword*)&(c), len);
  int64 t2=usecs();
  for(int i = 0; i < maximum; i++) MemSet(~b, c, len);
  int64 t3=usecs();
  result.Cat(Format("%d,%f,%f,%f\r\n",len,1000.0*(t1-t0)/maximum,1000.0*(t2-t1)/maximum,1000.
0*(t3-t2)/maximum));
  if(len<32) len++;
  else len*=2;
 }

 SaveFile(GetHomeDirFile("Desktop/memset.csv"),result);
```

Again, I suggest you plot your results using a log-log chart to clearly see the performance with all different block sizes.

If you have some time to spare, please let me know how this works for you.

Best regards,

Tom
