Subject: Re: Writing Bits object to disk
Posted by crydev on Thu, 27 Apr 2017 08:31:43 GMT
View Forum Message <> Reply to Message

Thanks Mirek!

I have been trying a thing or two myself. I wrote a version of the Set(int, bool) method that accepts four bool values at the same time. This version _probably_ has more pipelining capabilities, and is faster than the regular set method!

```
union BitBoolPipeline
{
 struct
 {
  bool b1;
  bool b2;
  bool b3;
  bool b4;
 };

 dword dw;
};

// Different function implementing Bits, but with a pipelined set method.
const int VectorBoolOrBitsetTestBitSetPipeline(Bits& buffer, const Vector<bool>& rand)
{
 const int count = rand.GetCount();
 for (int i = 0; i < count; i += 4)
 {
  // We can now set four bools at the same time. However, we must assume that a bool that
  // is set to true has value 0x1, and a bool set to false has value 0x0.
  BitBoolPipeline b;
  b.b1 = rand[i];
  b.b2 = rand[i + 1];
  b.b3 = rand[i + 2];
  b.b4 = rand[i + 3];

  buffer.PipelineSet(i, b.dw);
 }
 int alloc = 0;
 buffer.Raw(alloc);
 return alloc;
}

const dword PowersOfTwo[] =
{
 0x1, 0x2, 0x4, 0x8, 0x10, 0x20, 0x40, 0x80, 0x100, 0x200, 0x400, 0x800, 0x1000, 0x2000,
```

```
 0x4000, 0x8000, 0x10000, 0x20000, 0x40000, 0x80000, 0x100000, 0x200000, 0x400000,
0x800000,
 0x1000000, 0x2000000, 0x4000000, 0x8000000, 0x10000000, 0x20000000, 0x40000000,
0x80000000
};

// Testing pipelined version of Bits::Set(int, bool)
void Bits::PipelineSet(int i, const dword bs)
{
 // Check whether i is within the bounds of the container.
 ASSERT(i >= 0 && alloc >= 0);

 // Get the DWORD index for the internal buffer.
 int q = i >> 5;

 // Get the bit index of the next available DWORD.
 i &= 31;

 // Do we need to expand the internal buffer first?
 // Also check whether we can place 4 bits in the existing DWORD. If not, we should expand.
 if(q >= alloc)
  Expand(q);

 // Get integer bit values according to existing DWORD value and indices.
 // Assuming default value of bool is 0x1 if true!
 bp[q] = (bp[q] | bs & PowersOfTwo[1] | bs & PowersOfTwo[8] | bs & PowersOfTwo[16] | bs &
PowersOfTwo[24]);
}
```

This function assumes that a bool set to true has value 0x1, and 0x0 otherwise. I made the array of constant powers of two in advance. Replacing the array indices with the array constant itself doesn't improve any further, because the compiler propagates the constants by itself. The measurement is as follows:

I'm also working on a SIMD version that should be able to set 16 bools in parallel, having some assumptions about the input. :)

crydev

File Attachments

1) Capture.PNG, downloaded 772 times