Subject: Re: Writing Bits object to disk
Posted by crydev on Wed, 03 May 2017 09:12:54 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 03 May 2017 09:53
Well, it is what I thought:

```
const int VectorBoolOrBitsetTestBitSetVectorized(Bits& buffer, const Vector<unsigned char>&
randVec)
{
 const int count = randVec.GetCount();
 for (int i = 0; i < count; i += 16)
 {
  // Use the vectorized set method.
  buffer.VectorSet(i, &randVec[i]);
 }
 int alloc = 0;
 buffer.Raw(alloc);
 return alloc;
}
```

This is not the proper benchmark. The large part of work is already done by grouping input bits into randVec, which is not included in the benchmark time IMO.

If you wanted the proper benchmark, you could e.g. set randVec to random values and then set bits in Bits for those values that satisfy some condition - that IMO will benchmark scenario closer to the real use. E.g. (for original Bits and random 0..100):

```
for(int i = 0; i < randValue.GetCount(); i++)
    bits.Set(i, randValue[i] > 50);
```

Why is it the case? The random vector is precomputed, and the only thing both testing functions do is set the random values to the underlying Bits:

```
buffer.Set(i, rand[i]);

// vs.

buffer.VectorSet(i, &randVec[i]);
```

The problem is that the regular Set function only accepts one bool as parameter. The difference is

that I can put a pointer to a buffer of bools directly into the Set function, which is exactly my use case (bulk set)

crydev
