Subject: Re: Writing Bits object to disk
Posted by mirek on Wed, 03 May 2017 07:53:03 GMT
View Forum Message <> Reply to Message

crydev wrote on Wed, 03 May 2017 09:00mirek wrote on Tue, 02 May 2017 23:55
I still think there is a glitch somewhere benchmarking this. Would it possible to post a whole package here?

Mirek

The testing application is on Bitbucket here (only runs on Windows):
https://bitbucket.org/evolution536/cry-performance-test

You still have to add the following functions to the Bits class:

```
void PipelineSet(int i, const dword bs);
void VectorSet(int i, const unsigned char vec[16]);
void VectorSetAVX2(int i, const unsigned char vec[32]);
```

crydev

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);
```