Bitstrm

Chris@DUrso.org

Association of C/C++ Users - San Francisco Bay Area

Palo Alto: Workshop and Discussion Group

November 29, 2017

https://github.com/chrisdursoorg/bitstrm/

Rational

- codecs, serde, packed integers … oh my!
- multiple implementations
- non trivial to do correctly

Motivation

- for problems that exceed preferred memory availability
- improve performance
- adding additional data
- increasing data density
- reducing secondary/tertiary access

Bitstrm Features

- adaptable to various hardware
- *lightweight accessor* behavior (e.g. for std algorithms)
- integrate endian transform
- two's compliment fully implemented
- simple codec primitives
  - run length specified encoding
  - run length prefixed encoding
  - count leading zeros (unary encoding)

Architecture

- **reg/ureg**
- ***bref, alloced_bref***
- **bit_int_itr**

Example

```
// example, storage and retrieval of 3 bit integer

alloced_bref example_buf(c_at_least_3_and_internally_stored_on_full_64_bit_boundry);
bref begin = example_buf;
example_buf.iwrite(min_bits(-4), -4);
bref end = example_buf;
// now, encoded as a single signed integer, [begin, end) -> -4
assert(begin.read_reg(end-begin) == -4);
```