```rust
#![feature(test)]
#[macro_use]
extern crate abomonation;
extern crate test;

use test::Bencher;
use abomonation::{Abomonation, encode, decode};

#[bench]
fn bench_populate(b: &mut Bencher) {
    b.iter(|| {
        Log::new()
    });
}

#[bench]
fn bench_serialize(b: &mut Bencher) {
    let log = Log::new();
    let mut bytes = vec![];
    unsafe { encode(&log, &mut bytes); }
    b.bytes = bytes.len() as u64;
    b.iter(|| {
        bytes.clear();
        unsafe { encode(&log, &mut bytes); }
        test::black_box(&bytes);
    });
}

#[bench]
fn bench_deserialize(b: &mut Bencher) {
    let log = Log::new();
    let mut bytes = vec![];
    unsafe { encode(&log, &mut bytes); }
    b.bytes = bytes.len() as u64;
    b.iter(|| {
        test::black_box(unsafe { decode::<Log>(&mut bytes) });
    });
}
```

```
bench_populate:              267 ns/iter (+/- 124)
bench_serialize:              54 ns/iter (+/- 4)  = 10148 MB/s
bench_deserialize:             8 ns/iter (+/- 1)  = 68500 MB/s
bench_deserialize_test: 112 ns/iter (+/- 25) =  4892 MB/s
```


This is really fast

It’s even faster than this

```rust
#![feature(test)]
#[macro_use]
extern crate abomonation;
extern crate test;

use test::Bencher;
use abomonation::{Abomonation, encode, decode};

#[bench]
fn bench_populate(b: &mut Bencher) {
    b.iter(|| {
        Log::new()
    });
}

#[bench]
fn bench_serialize(b: &mut Bencher) {
    let log = Log::new();
    let mut bytes = vec![];
    unsafe { encode(&log, &mut bytes); }
    b.bytes = bytes.len() as u64;
    b.iter(|| {
        bytes.clear();
        unsafe { encode(&log, &mut bytes); }
        test::black_box(&bytes);
    });
}

#[bench]
fn bench_deserialize(b: &mut Bencher) {
    let log = Log::new();
    let mut bytes = vec![];
    unsafe { encode(&log, &mut bytes); }
    b.bytes = bytes.len() as u64;
    b.iter(|| {
        test::black_box(unsafe { decode::<Log>(&mut bytes) });
    });
}
```

Projects

http://github.com/frankmcsherry/timely-dataflow

http://github.com/frankmcsherry/differential-dataflow

http://github.com/frankmcsherry/abomonation

```
    b.bytes = bytes.len() as u64;
    b.iter(|| {
        bytes.clear();
        unsafe { encode(&log, &mut bytes); }
        test::black_box(&bytes);
    });
}

#[bench]
fn bench_deserialize(b: &mut Bencher) {
    let log = Log::new();
    let mut bytes = vec![];
    unsafe { encode(&log, &mut bytes); }
    b.bytes = bytes.len() as u64;
    b.iter(|| {
        test::black_box(unsafe { decode::<Log>(&mut bytes) });
    });
}
```

```
bench_populate:          267 ns/iter (+/- 124)
bench_serialize:          54 ns/iter (+/- 4)  = 10148 MB/s
bench_deserialize:         8 ns/iter (+/- 1)  = 68500 MB/s
bench_deserialize_test: 112 ns/iter (+/- 25) =  4892 MB/s
```