```
#![feature(test)]
 1
 2
    #[macro_use]
 3
    extern crate abomonation;
 4
    extern crate test;
 5
 6
    use test::Bencher;
 7
    use abomonation::{Abomonation, encode, decode};
 8
 9
    #[bench]
    fn bench_populate(b: &mut Bencher) {
10
11
        b.iter(|| {
12
             Log::new()
        });
13
14
    }
15
16
    #[bench]
17
    fn bench_serialize(b: &mut Bencher) {
18
        let log = Log::new();
19
         let mut bytes = vec![];
20
        unsafe { encode(&log, &mut bytes); }
21
        b.bytes = bytes.len() as u64;
22
        b.iter(|| {
23
             bytes.clear();
24
             unsafe { encode(&log, &mut bytes); }
25
             test::black_box(&bytes);
26
        });
27
    }
28
29
    #[bench]
30
    fn bench_deserialize(b: &mut Bencher) {
        let log = Log::new();
31
32
        let mut bytes = vec![];
33
        unsafe { encode(&log, &mut bytes); }
        b.bytes = bytes.len() as u64;
34
35
        b.iter(|| {
36
             test::black_box(unsafe { decode::<Log>(&mut bytes) });
37
        });
38
    }
```

```
bench_populate:
                           267 ns/iter (+/- 124)
                            54 \text{ ns/iter} (+/- 4) = 10148 \text{ MB/s}
bench serialize:
                             8 \text{ ns/iter} (+/-1) = 68500 \text{ MB/s}
bench deserialize:
bench deserialize test: 112 ns/iter (+/- 25) = 4892 MB/s
```

## This is really fast

## It's even faster than this

```
#![feature(test)]
 1
 2
    #[macro_use]
 3
    extern crate abomonation;
 4
    extern crate test;
 5
 6
    use test::Bencher;
 7
    use abomonation::{Abomonation, encode, decode};
 8
 9
    #[bench]
    fn bench_populate(b: &mut Bencher) {
10
11
        b.iter(|| {
12
             Log::new()
        });
13
14
    }
15
16
    #[bench]
17
    fn bench_serialize(b: &mut Bencher) {
18
        let log = Log::new();
19
         let mut bytes = vec![];
20
        unsafe { encode(&log, &mut bytes); }
21
        b.bytes = bytes.len() as u64;
22
        b.iter(|| {
23
             bytes.clear();
24
             unsafe { encode(&log, &mut bytes); }
25
             test::black_box(&bytes);
26
        });
27
    }
28
29
    #[bench]
30
    fn bench_deserialize(b: &mut Bencher) {
        let log = Log::new();
31
32
        let mut bytes = vec![];
33
        unsafe { encode(&log, &mut bytes); }
        b.bytes = bytes.len() as u64;
34
35
        b.iter(|| {
36
             test::black_box(unsafe { decode::<Log>(&mut bytes) });
37
        });
38
    }
```

## Projects

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

```
b.iter(|| {
23
            bytes.clear();
            unsafe { encode(&log, &mut bytes); }
25
            test::black_box(&bytes);
        });
29
    #[bench]
30
    fn bench_deserialize(b: &mut Bencher) {
31
        let log = Log::new();
        let mut bytes = vec![];
32
        unsafe { encode(&log, &mut bytes); }
33
34
        b.bytes = bytes.len() as u64;
35
        b.iter(|| {
            test::black_box(unsafe { decode::<Log>(&mut bytes) });
36
37
        });
38
```

```
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
```