Missguided attempts at performance optimizations

Radu Ciorba {radu@devrandom.ro}

February 17, 2017

What is Borg Backup?

- Deduplicating backup program
- written mostly in python
- with some C code
- and some Cython code to tie everything together

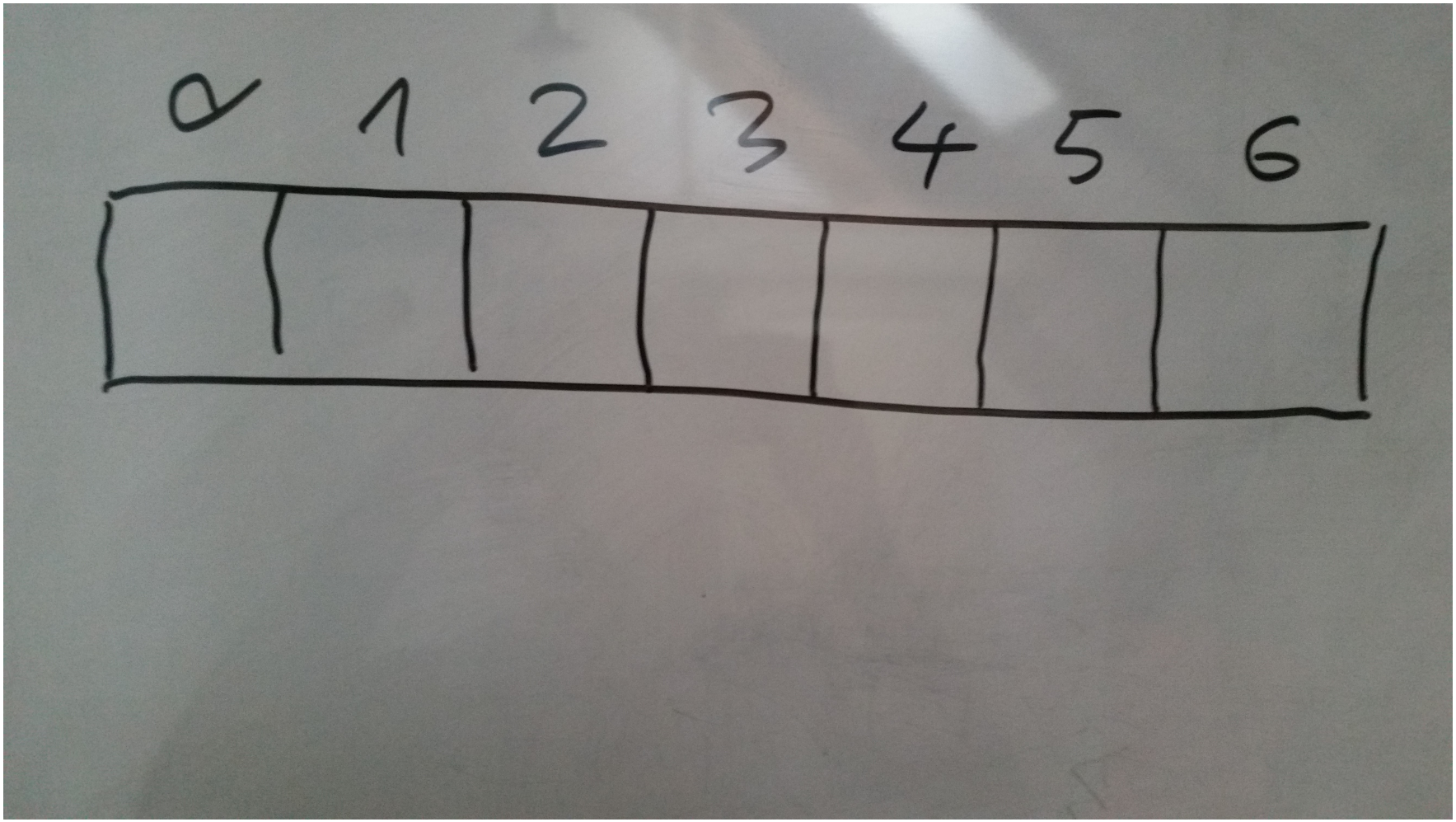
Cython you say?

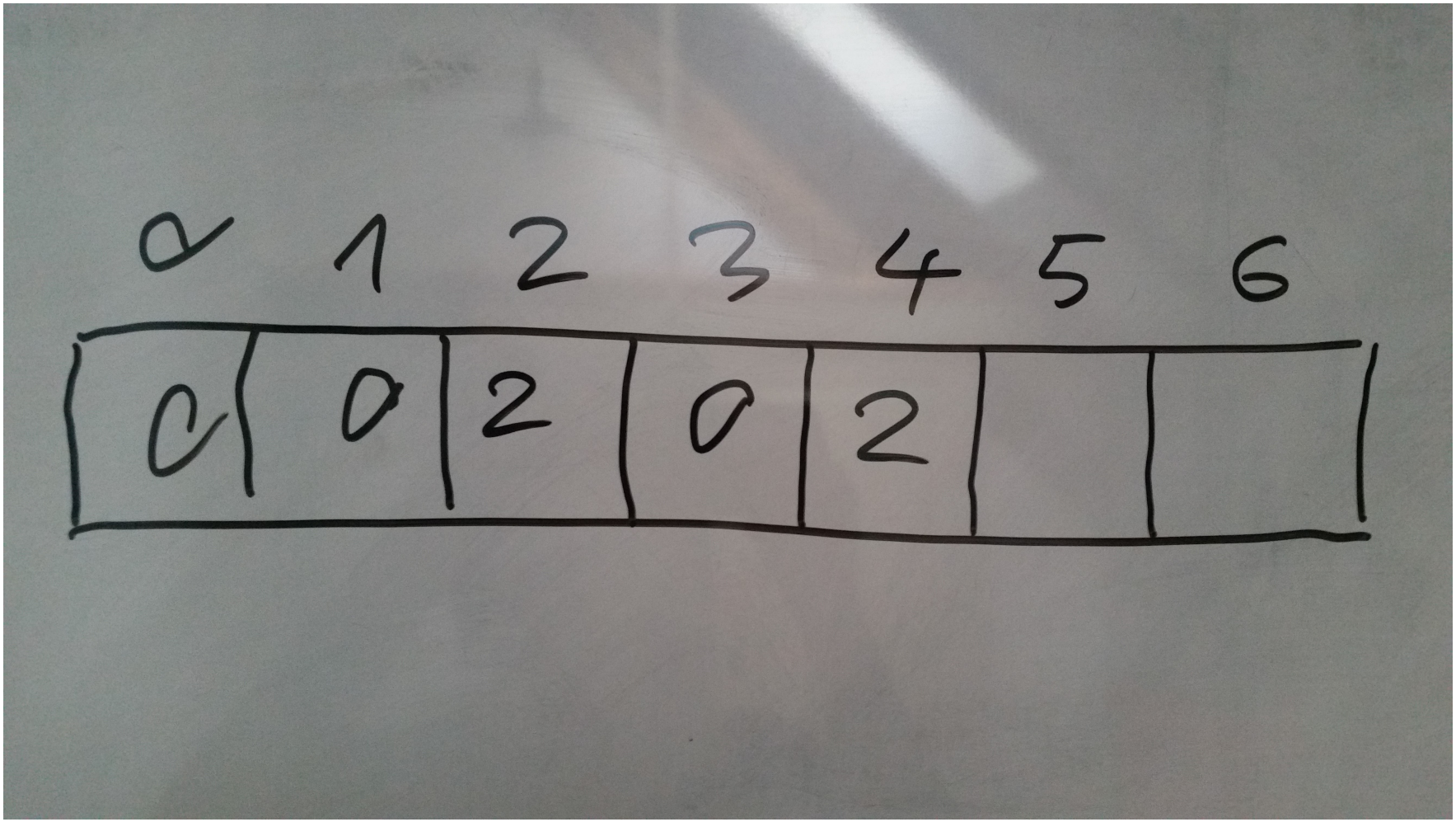
```
from cpython.bytes cimport PyBytes_AS_STRING

cdef extern from "_hashindex.c":
    ctypedef struct HashIndex:
        pass

    void benchmark_getitem(HashIndex *index, char *keys, int key_count)

def bench_getitem(ChunkIndex chunk_index, bytes keys):
    cdef int key_count = len(keys) // chunk_index.key_size
    benchmark_getitem(
        chunk_index.index, PyBytes_AS_STRING(keys), key_count)
```

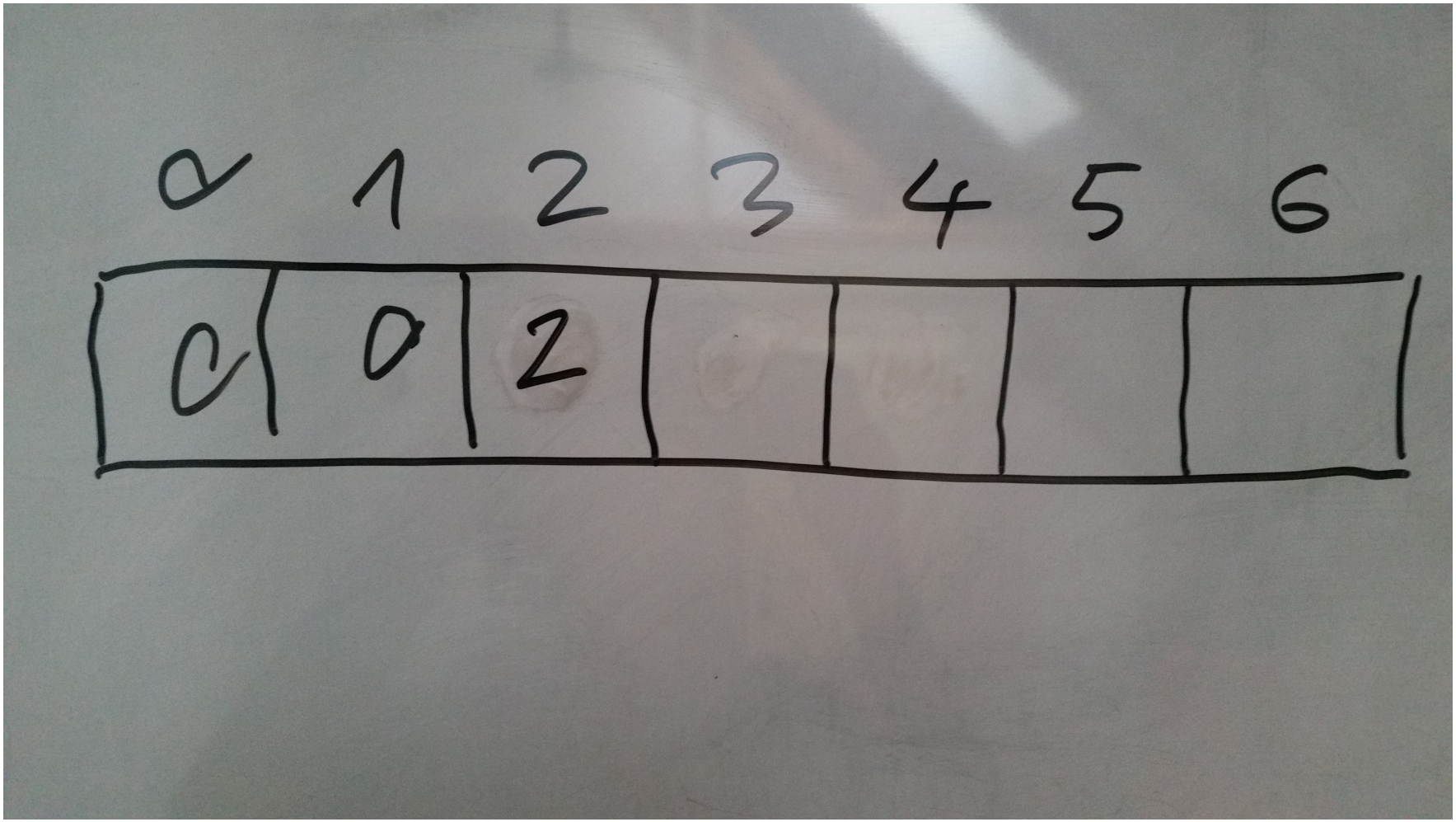
```
============================= test session starts ==============================
platform linux -- Python 3.5.2, pytest-3.0.2, py-1.4.31, pluggy-0.3.1 -- /home/rciorba/.venvs/borg/bin/python3
cachedir: .cache
benchmark: 3.0.0 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=5.00us max_time=1.00s calibration_precision=10 warmup=False warmup_iterations=100000)
Tests enabled: BSD flags, modes, atime/mtime, hardlinks, symlinks, root
Tests disabled: fuse
rootdir: /home/rciorba/repos/borg, inifile: setup.cfg
plugins: cov-2.3.1, benchmark-3.0.0
collected 23 items

src/borg/testsuite/hashindex.py::HashIndexTestCase::test_chunkindex FAILED
src/borg/testsuite/hashindex.py::HashIndexTestCase::test_chunkindex_merge PASSED
src/borg/testsuite/hashindex.py::HashIndexTestCase::test_chunkindex_summarize PASSED
src/borg/testsuite/hashindex.py::HashIndexTestCase::test_iteritems PASSED
src/borg/testsuite/hashindex.py::HashIndexTestCase::test_nsindex FAILED
src/borg/testsuite/hashindex.py::HashIndexTestCase::test_resize fish: "py.test src/borg/testsuite/hash…" terminated by signal SIGSEGV (Address boundary error)
```

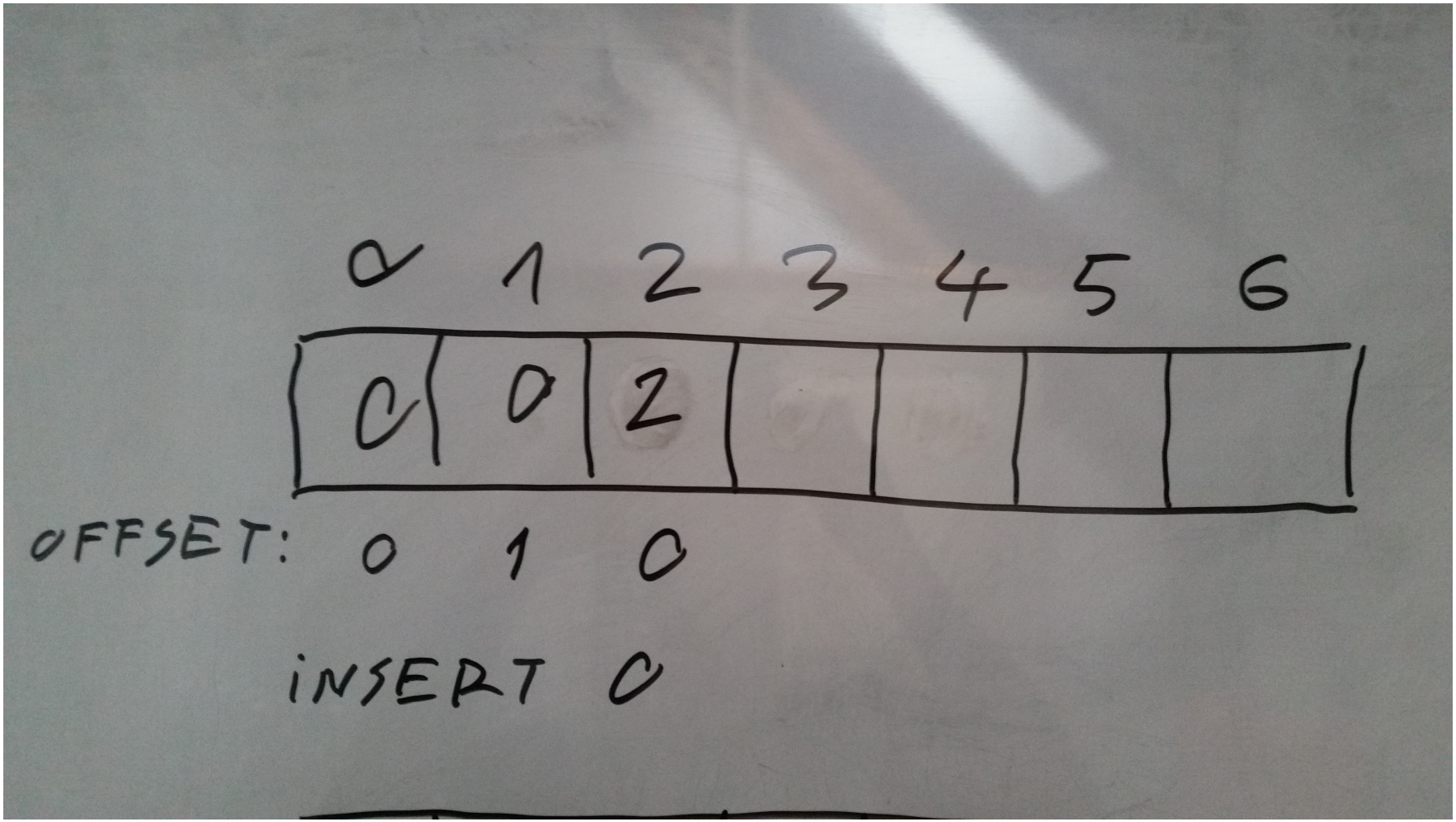
The goal: Robin Hood hashing

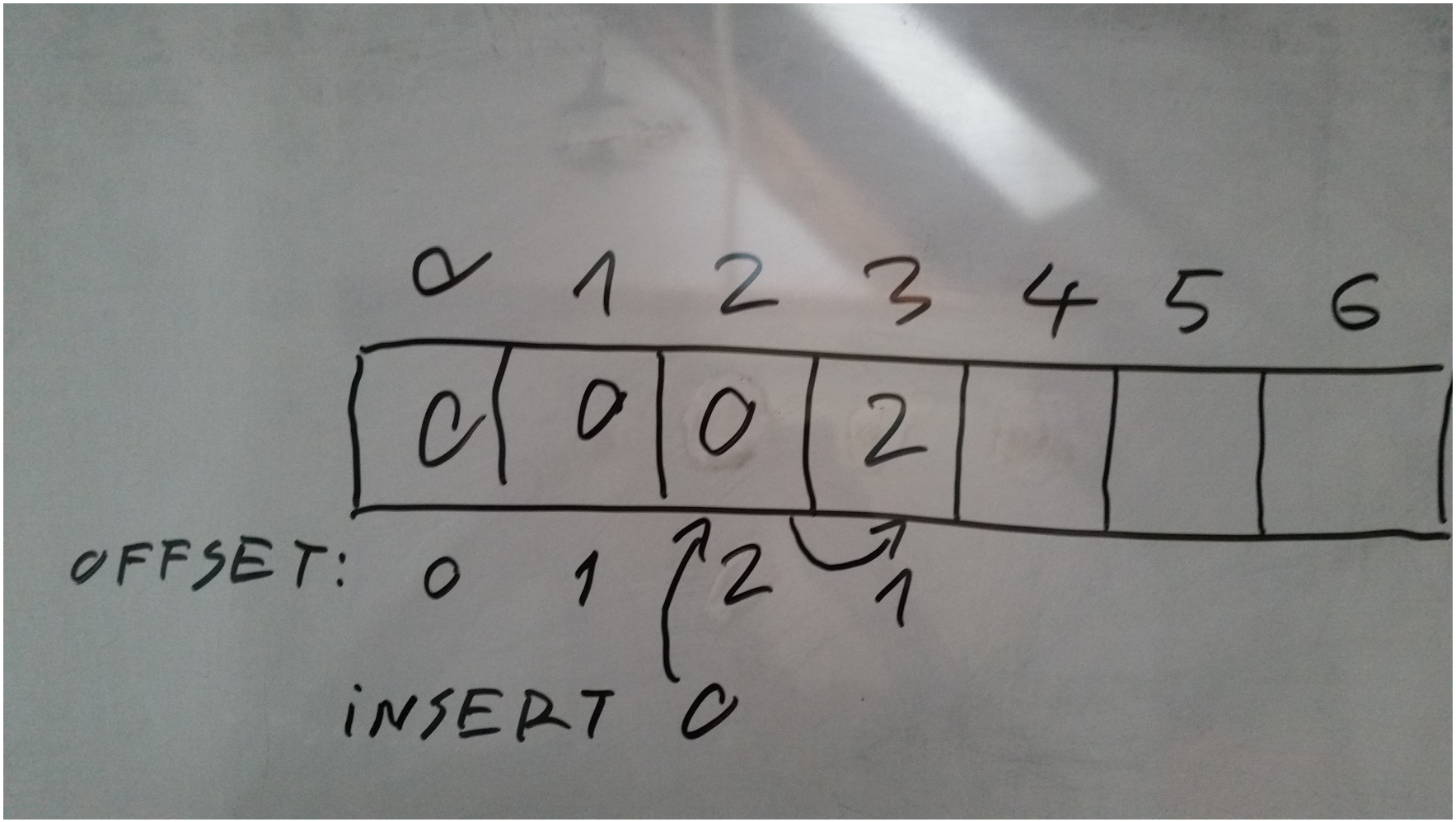
- Quick recap on hashmaps
- Open addressing
- Deletion


a
1
2
3
4
5
6


0 1 2 3 4 5 6
C 0 2 0 2


0 1 2 3 4 5 6
C
0
2


0 1 2 3 4 5 6
C O Z
OFFSET: 0 1 0
INSERT C


0 1 2 3 4 5 6
C
O
O
2
OFFSET: 0 1 2 1
INSERT C

Borg's current Hash

- key is 32 bytes, value is 12 bytes
- Open Addressing
- Tombstones
- hashindex_lookup will promote buckets on top of tombstones

insert actual implementation here ...

Cool story bro. Benchmarks?

```
def test_chunk_indexer_getitem(benchmark):
    max_key = 2**20
    index = ChunkIndex(max_key)
    keys = [sha256(H(k)).digest() for k in range(max_key)]
    for key in keys:
        index[key] = (0, 0, 0)

    def do_gets(keys=keys):
        for key in keys:
            index[key]  # noqa

    benchmark.pedantic(do_gets, rounds=200)
```

Cool story bro. Benchmarks?

master with MAX_HASH_LOAD=0.93; 2^23 keys (reference)

benchmark: 2 tests

Name (time in s)	Min	Max	Mean	StdDev	Median	IQR
test_chunk_indexer_setitem	2.5804 (1.0)	2.6710 (1.0)	2.6278 (1.0)	0.0326 (6.19)	2.6251 (1.0)	0.0330
test_chunk_indexer_getitem	4.4887 (1.74)	4.5013 (1.69)	4.4920 (1.71)	0.0053 (1.0)	4.4897 (1.71)	0.0039

robin_hood with MAX_HASH_LOAD=0.93; 2^23 keys

benchmark: 2 tests

Name (time in s)	Min	Max	Mean	StdDev	Median	IQR
test_chunk_indexer_setitem	4.8533 (1.0)	4.8696 (1.0)	4.8591 (1.0)	0.0062 (1.0)	4.8569 (1.0)	0.0057
test_chunk_indexer_getitem	4.9203 (1.01)	4.9791 (1.02)	4.9487 (1.02)	0.0211 (3.42)	4.9457 (1.02)	0.0218

What now?

- RH at least faster for missing items
- apply more coffe + time
- found a bug in hashindex_set, wasn't ever doing the position swap

Repeatable benchmarks

```
------------------------------------------------------------------------------ benchmark: 2 tests -------------------------------
Name (time in ms)                      Min                 Max                Mean            StdDev              Median
---------------------------------------------------------------------------------------------------------------------------------
test_chunk_indexer_setitem     112.1436 (1.0)      130.6243 (1.0)      116.1420 (1.0)      2.6196 (1.13)     115.1328 (1.0)
test_chunk_indexer_getitem     201.1404 (1.79)     216.5813 (1.66)     207.1470 (1.78)     2.3098 (1.0)      206.7306 (1.80)
---------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------ benchmark: 2 tests -------------------------------
Name (time in ms)                      Min                 Max                Mean            StdDev              Median
---------------------------------------------------------------------------------------------------------------------------------
test_chunk_indexer_setitem     107.8818 (1.0)      122.5610 (1.0)      111.2042 (1.0)      2.3189 (1.0)      110.3162 (1.0)
test_chunk_indexer_getitem     197.3264 (1.83)     244.0130 (1.99)     203.2360 (1.83)     6.8099 (2.94)     202.0415 (1.83)
---------------------------------------------------------------------------------------------------------------------------------
```

Repeatable benchmarks

- Use same inputs
- Use same machine
- Make sure nothing else is running
- CPU frequency scaling?
- Isolate your code as much as possible

Isolated your code for testing

```
def test_chunk_indexer_getitem(benchmark):
    max_key = 2**20
    index = ChunkIndex(max_key)
    keys = [sha256(H(k)).digest() for k in range(max_key)]
    for key in keys:
        index[key] = (0, 0, 0)

    def do_gets(keys=keys):
        for key in keys:
            index[key]  # noqa

    benchmark.pedantic(do_gets, rounds=200)
```

Isolated your code for testing

```
static void
benchmark_getitem(HashIndex *index, char *keys, int key_count)
{
  char *key = keys;
  char *last_addr = key + (32 * key_count);
  while (key < last_addr) {
    hashindex_get(index, key);
    key += 32;
  }
}
```

Repeatable benchmarks

benchmark 'test_chunk_indexer_c_setitem_update': 6 tests

Name (time in ms)	Min	Max	Mean	StdDev	Median
test_chunk_indexer_c_setitem_update[0.3]	102.7014 (1.0)	102.9814 (1.0)	102.8013 (1.0)	0.0882 (1.0)	102.7865 (1.0)
test_chunk_indexer_c_setitem_update[0.5]	173.7228 (1.69)	15,212.7702 (147.72)	2,462.7556 (23.96)	4,914.7807 (>1000.0)	235.4065 (2.29)
test_chunk_indexer_c_setitem_update[0.85]	299.2544 (2.91)	27,044.4509 (262.61)	4,073.4831 (39.62)	8,542.5512 (>1000.0)	400.0225 (3.89)
test_chunk_indexer_c_setitem_update[0.75]	324.0385 (3.16)	1,168.0475 (11.34)	549.7782 (5.35)	285.6380 (>1000.0)	420.7876 (4.09)
test_chunk_indexer_c_setitem_update[0.93]	335.1741 (3.26)	4,270.4711 (41.47)	1,017.9627 (9.90)	1,459.4260 (>1000.0)	335.2918 (3.26)
test_chunk_indexer_c_setitem_update[0.95]	344.4262 (3.35)	6,411.0975 (62.25)	951.2685 (9.25)	1,918.3883 (>1000.0)	344.6020 (3.35)

benchmark 'test_chunk_indexer_c_setitem_update': 6 tests

Name (time in ms)	Min	Max	Mean	StdDev	Median
test_chunk_indexer_c_setitem_update[0.3]	102.7539 (1.0)	102.9468 (1.0)	102.8343 (1.0)	0.0602 (1.0)	102.8319 (1.0)
test_chunk_indexer_c_setitem_update[0.5]	173.7920 (1.69)	15,279.5163 (148.42)	2,473.3088 (24.05)	4,937.4798 (>1000.0)	235.5947 (2.29)
test_chunk_indexer_c_setitem_update[0.85]	299.0782 (2.91)	27,075.4213 (263.00)	4,077.7778 (39.65)	8,552.8954 (>1000.0)	400.2995 (3.89)
test_chunk_indexer_c_setitem_update[0.75]	327.6863 (3.19)	1,167.8112 (11.34)	550.0308 (5.35)	285.1877 (>1000.0)	420.8317 (4.09)
test_chunk_indexer_c_setitem_update[0.93]	334.9304 (3.26)	4,272.6914 (41.50)	1,018.8431 (9.91)	1,461.2820 (>1000.0)	335.2077 (3.26)
test_chunk_indexer_c_setitem_update[0.95]	344.5489 (3.35)	6,424.4459 (62.41)	952.8873 (9.27)	1,922.5098 (>1000.0)	344.7731 (3.35)

We need a better way to prezent these benchmarks

- Robin Hood hashing first try
- Robin Hood, only check missing item shortcut every N buckets
- Robin Hood, shift entire chunk in one memmove
- Now let's replace modulo on master as well for apples to apples

My takeaways

- MEASURE EVERYTHING!
- sometimes a 'worse' algorithm might be better
- integer division is slow

Useful stuff

- Scott Meyers: Cpu Caches and Why You Care
- Andrei Alexandrescu: Writing Fast Code I
- pytest-benchmark
- kcachegrind + google-perftools + https://pypi.python.org/pypi/yep
- A raspberry pi for running the tests

```
-> touch src/borg/hashindex.pyx ; pip install -e . ; py.test src/borg/testsuite/hashindex.py  -v --capture=no -k test_chunkindex
Obtaining file:///home/rciorba/repos/borg
Requirement already satisfied: msgpack-python>=0.4.6 in /home/rciorba/.venvs/borg/lib/python3.5/site-packages (from borgbackup==1.1.dev182+ng65ba75c.d20170216)
Installing collected packages: borgbackup
  Found existing installation: borgbackup 1.1.dev182+ng65ba75c.d20170216
    Uninstalling borgbackup-1.1.dev182+ng65ba75c.d20170216:
      Successfully uninstalled borgbackup-1.1.dev182+ng65ba75c.d20170216
  Running setup.py develop for borgbackup
Successfully installed borgbackup
============================================================================================================ test session starts ============================================================================================================
platform linux -- Python 3.5.2, pytest-3.0.2, py-1.4.31, pluggy-0.3.1 -- /home/rciorba/.venvs/borg/bin/python3
cachedir: .cache
benchmark: 3.0.0 (defaults: timer=time.perf_counter disable_gc=False min_rounds=5 min_time=5.00us max_time=1.00s calibration_precision=10 warmup=False warmup_iterations=
Testing BSD-style flags: yes
Testing fuse: no
fakeroot: no (>=1.20.2: no)
rootdir: /home/rciorba/repos/borg, inifile: setup.cfg
plugins: cov-2.3.1, benchmark-3.0.0
collected 22 items

src/borg/testsuite/hashindex.py::HashIndexTestCase::test_chunkindex FAILED
src/borg/testsuite/hashindex.py::HashIndexTestCase::test_chunkindex_merge python3: malloc.c:2392: sysmalloc: Assertion `(old_top == initial_top (av) && old_size == 0) ||
(unsigned long) old_end & (pagesize - 1)) == 0)' failed.
fish: “py.test src/borg/testsuite/hash…” terminated by signal SIGABRT (Abort)
(borg)rciorba@void [~/r/borg] [key_bucket_split]
```

Thanks

You can find the code here: https://github.com/rciorba/borgbackup

The slides are available at https://devrandom.ro/talks