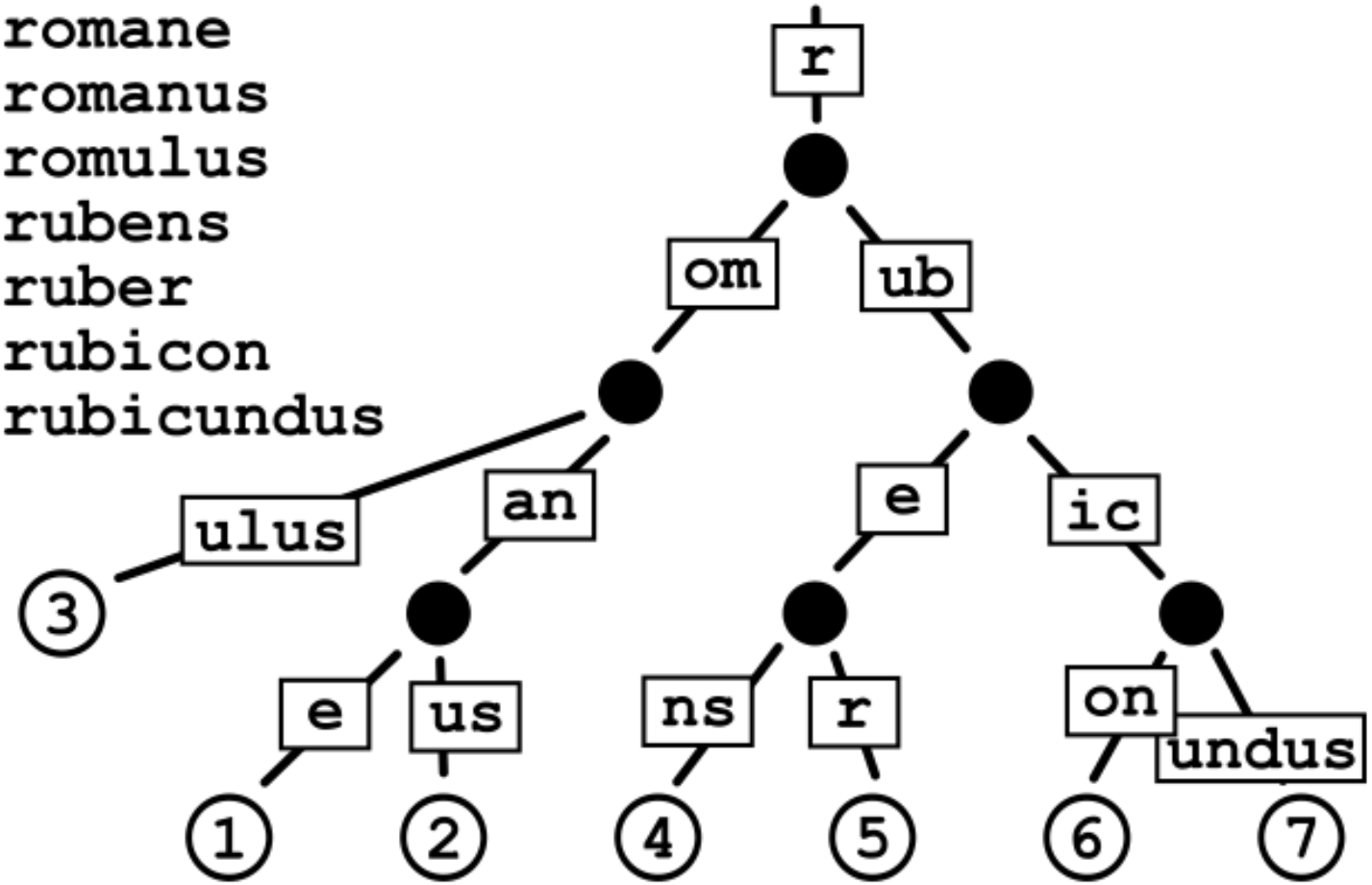
Rax, Listpack and safe contexts

@antirez

RAX

Redis radix tree implementation

1 romane
2 romanus
3 romulus
4 rubens
5 ruber
6 rubicon
7 rubicundus




HDR
a b c
a-ptr b-ptr c-ptr
val-ptr

Rax facts

- 2600 lines of code
- Less memory used
- Faster than dict.c for most use cases
- Ordered! Powerful iterators
- 100% fuzz tested

Rax usages

- Cluster key <-> slots tracking (done in 4.0)
- Streams data structure (4.2)
- Memory efficient Hashes and Sets (4.2)
- No longer double encoded! \o/
- Main dictionary? Expires?

Listpack

Main problem…

Ziplist encoding

Listpack encoding

Reverse length

Listpack facts

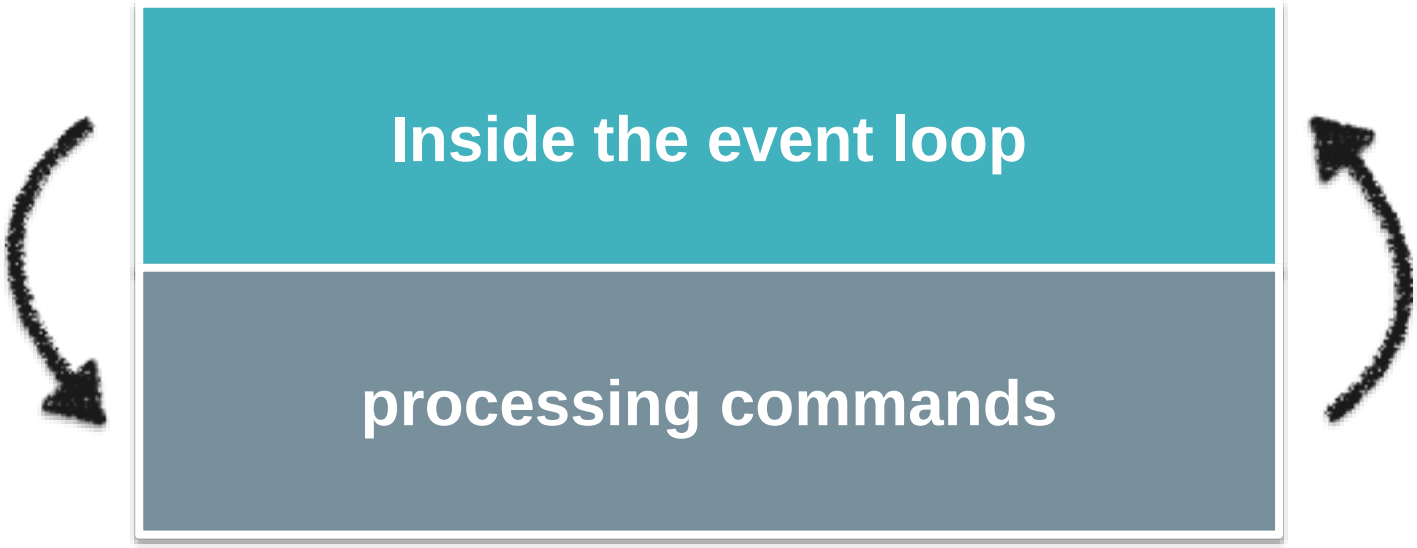
- We have a specification! By Yuval, Oran and me
- 1/3 the source code size of ziplist :-)
- More memory efficient for Redis data model
- work in progress (alpha published)
- TODO: 100% fuzz testing

Thread safe contexts

TSC: what we had

- blocking commands in modules
- RedisModule_UnblockClient() from thread
- No Redis Modules API from threads :-(

Redis event loop



Redis event loop

Non blocking KEYS *

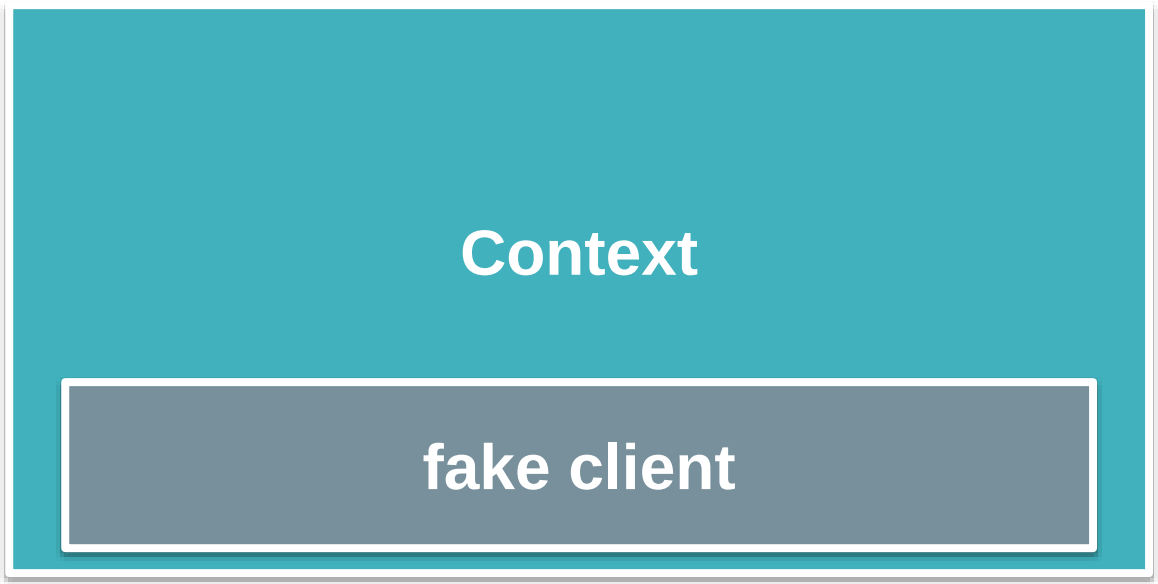
```
do {
  RedisModule_ThreadSafeContextLock(ctx);
  RedisModuleCallReply *reply =
       RedisModule_Call(ctx,
            "SCAN","l",(long long)cursor);
  RedisModule_ThreadSafeContextUnlock(ctx);

  …
}
```

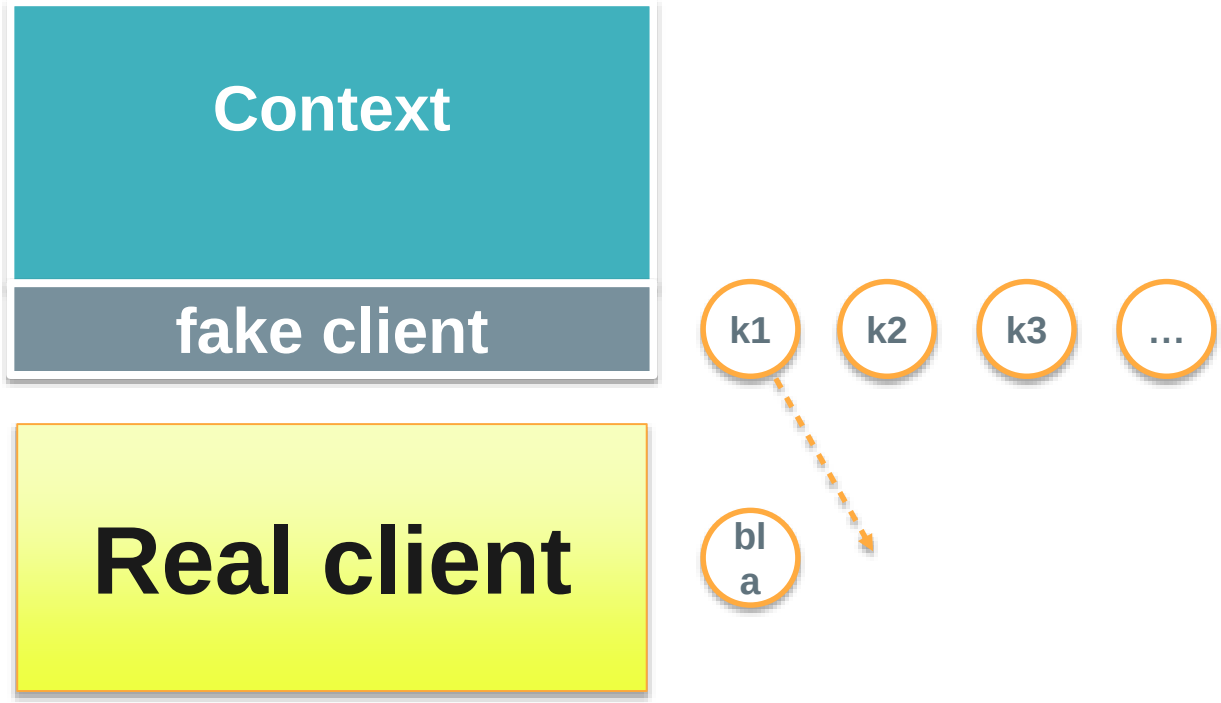
```
/* Note: no context lock to reply. */

do {
   …
   for (size_t j = 0; j < items; j++) {
     RedisModuleCallReply *ele = …;
     RedisModule_ReplyWithCallReply(ctx,ele);
   }
}
```

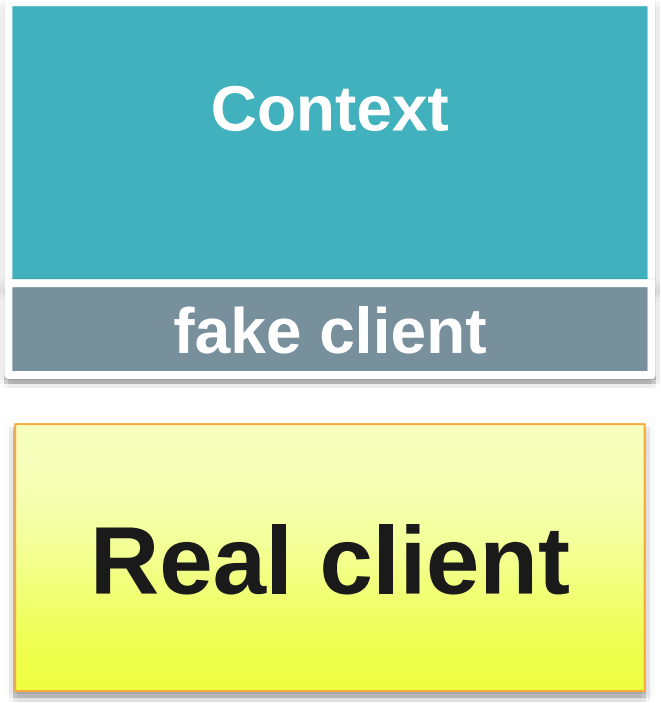
Replies #1



Replies #2



Replies #3



O(1) operation

Future

Modules threading future

- Key-locking
- read-write locks
- Automatic blocking of clients accessing busy keys
- No need for data structures-level locks

Thank You

Salvatore Sanfilippo
Redis Labs

Tweet me at @antirez