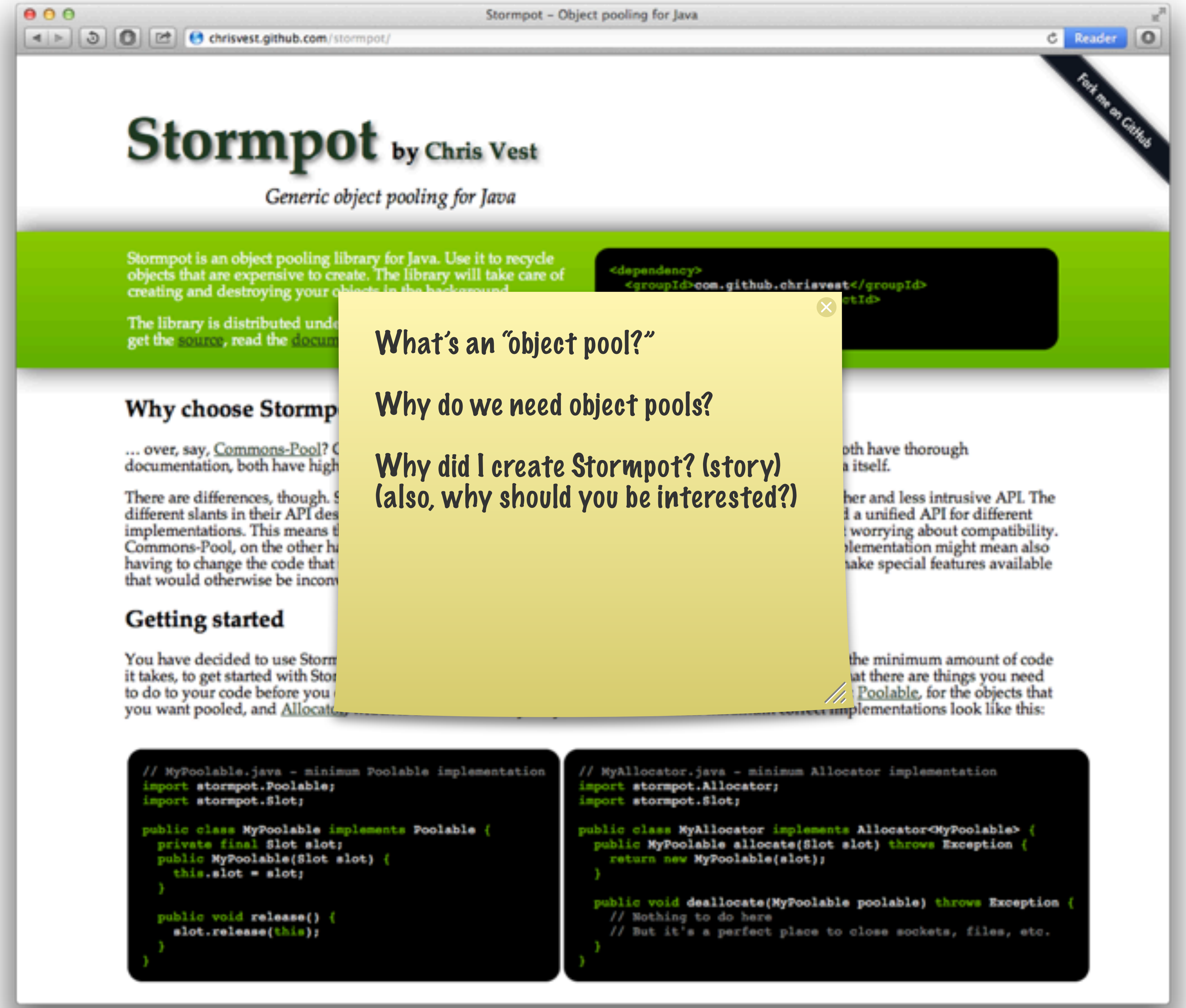
Stormpot

@chvest



Stormpot – Object pooling for Java
chrisvest.github.com/stormpot/
Reader
Fork me on GitHub

Stormpot by Chris Vest

Generic object pooling for Java

Stormpot is an object pooling library for Java. Use it to recycle objects that are expensive to create. The library will take care of creating and destroying your objects in the background.

The library is distributed unde
get the source, read the docum

```
<dependency>
  <groupId>com.github.chrisvest</groupId>
                            ctId>
```

What's an "object pool?"

Why do we need object pools?

Why did I create Stormpot? (story)
(also, why should you be interested?)

Why choose Stormp

... over, say, Commons-Pool? C   oth have thorough
documentation, both have high   a itself.

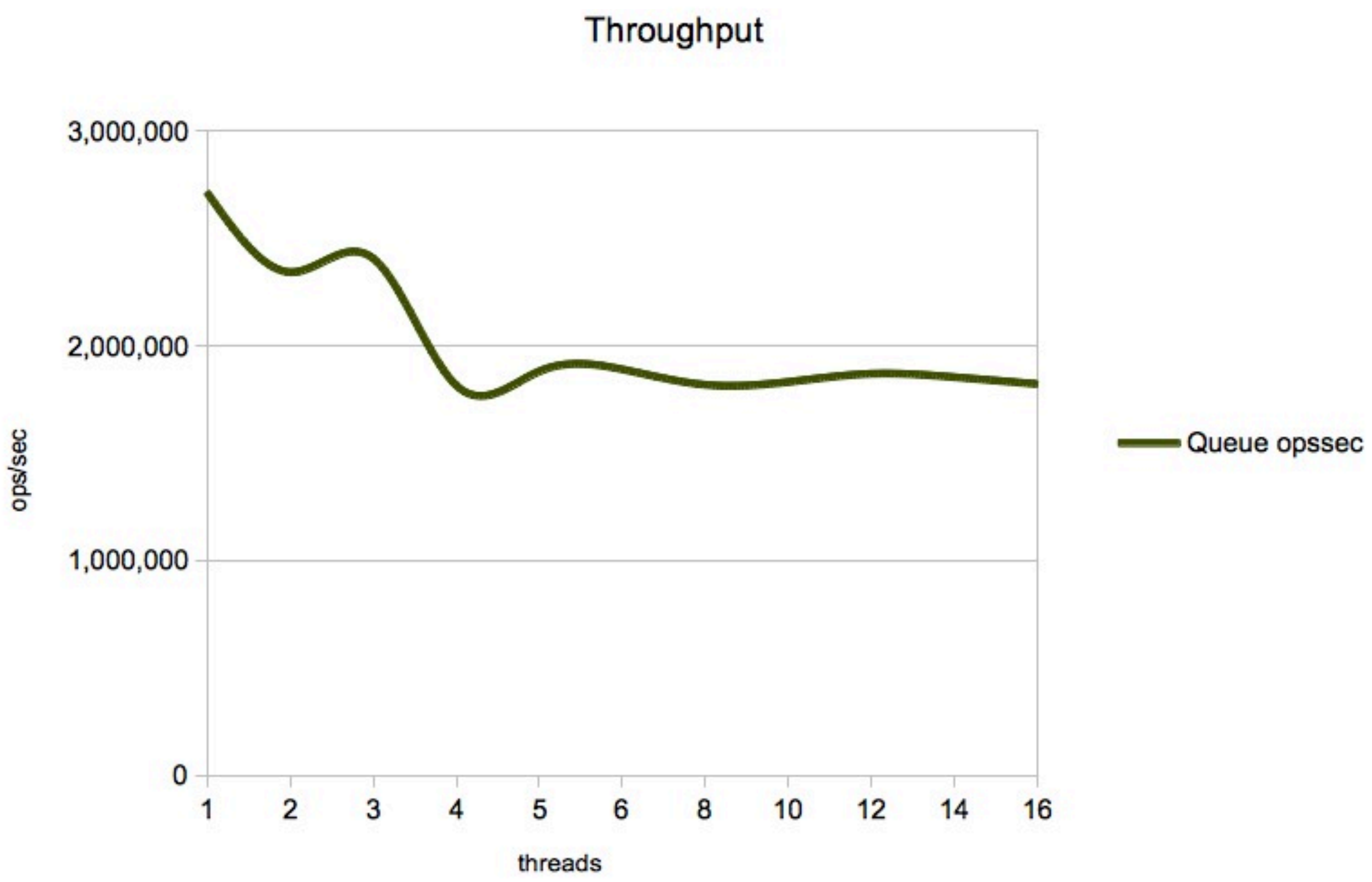
There are differences, though. S   her and less intrusive API. The
different slants in their API des   d a unified API for different
implementations. This means t   worrying about compatibility.
Commons-Pool, on the other ha   plementation might mean also
having to change the code that   nake special features available
that would otherwise be incon

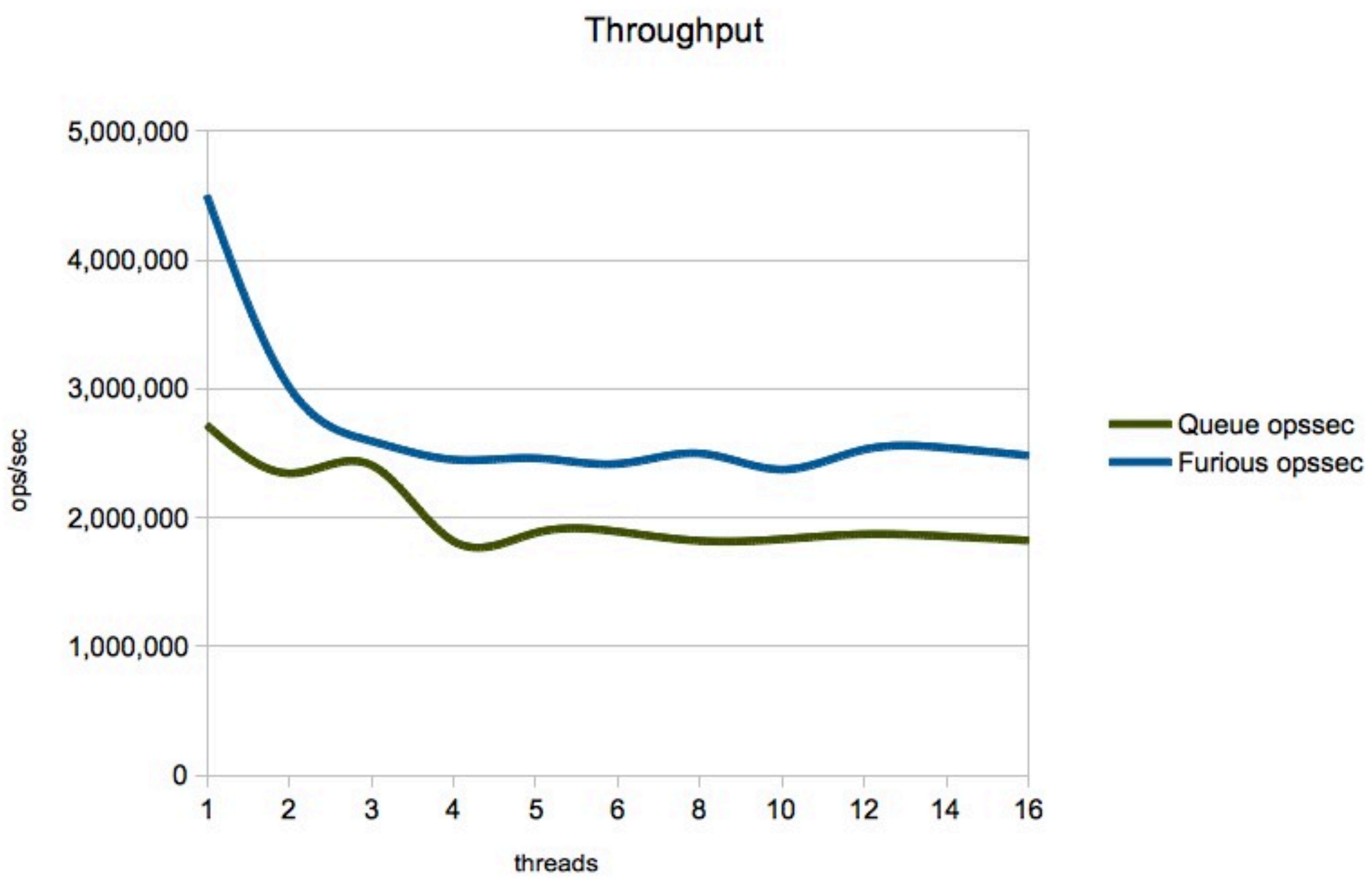
Getting started

You have decided to use Storm   the minimum amount of code
it takes, to get started with Stor   at there are things you need
to do to your code before you   Poolable, for the objects that
you want pooled, and Allocato   implementations look like this:

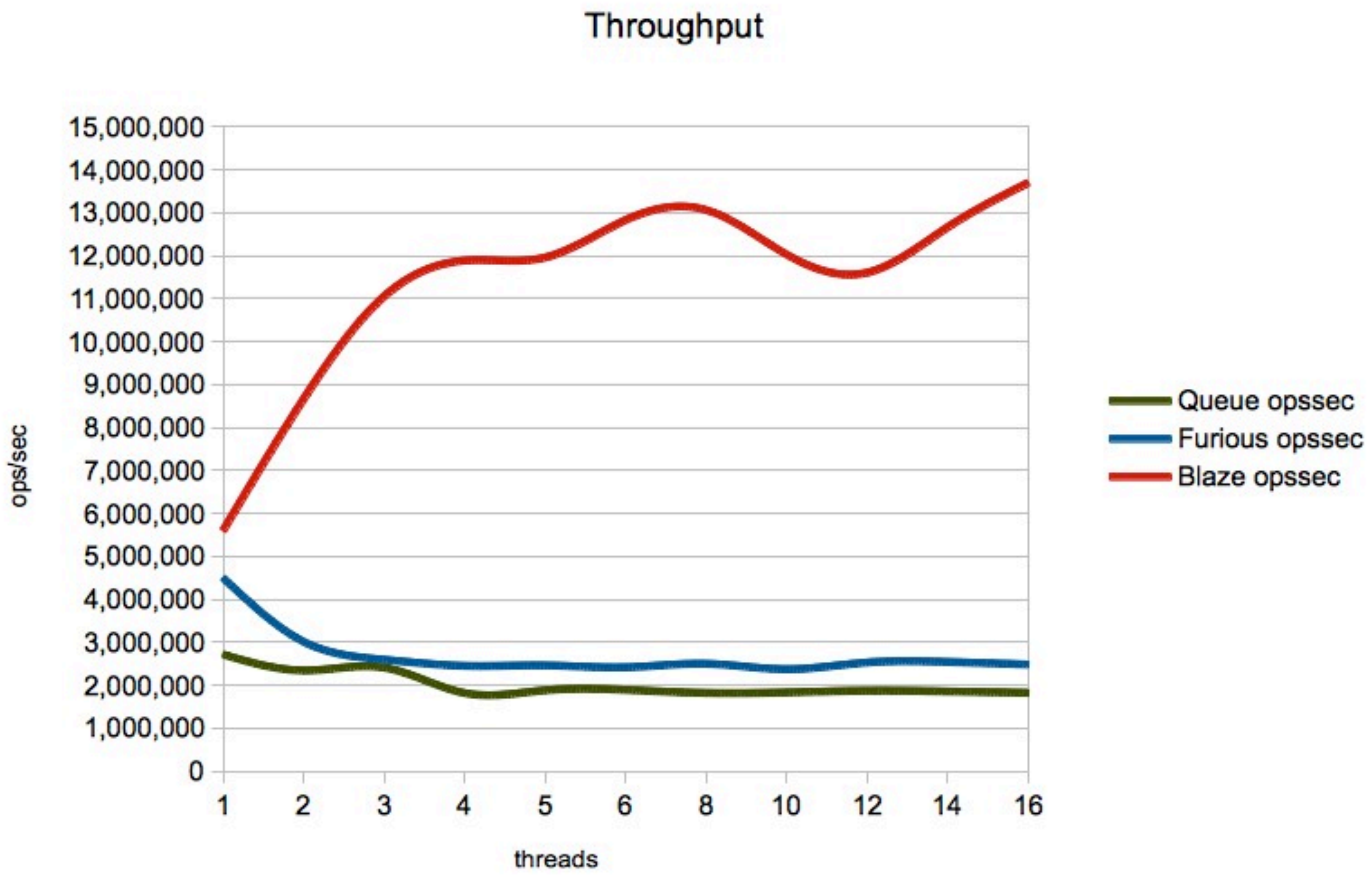
```
// MyPoolable.java - minimum Poolable implementation
import stormpot.Poolable;
import stormpot.Slot;

public class MyPoolable implements Poolable {
  private final Slot slot;
  public MyPoolable(Slot slot) {
    this.slot = slot;
  }

  public void release() {
    slot.release(this);
  }
}
```

```
// MyAllocator.java - minimum Allocator implementation
import stormpot.Allocator;
import stormpot.Slot;

public class MyAllocator implements Allocator<MyPoolable> {
  public MyPoolable allocate(Slot slot) throws Exception {
    return new MyPoolable(slot);
  }

  public void deallocate(MyPoolable poolable) throws Exception {
    // Nothing to do here
    // But it's a perfect place to close sockets, files, etc.
  }
}
```

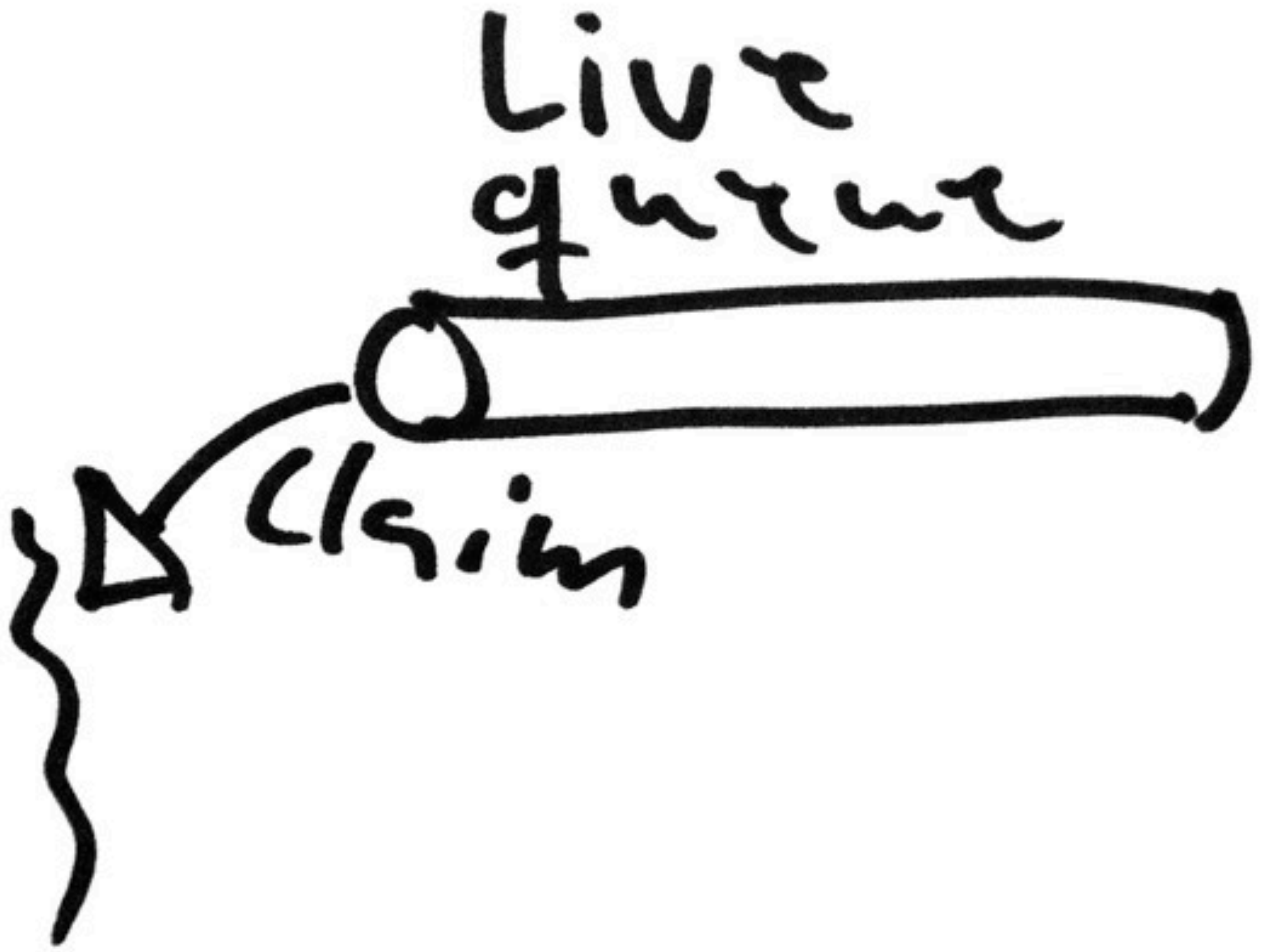

Throughput
3,000,000
2,000,000
1,000,000
0
ops/sec
1
2
3
4
5
6
8
10
12
14
16
threads
Queue opssec

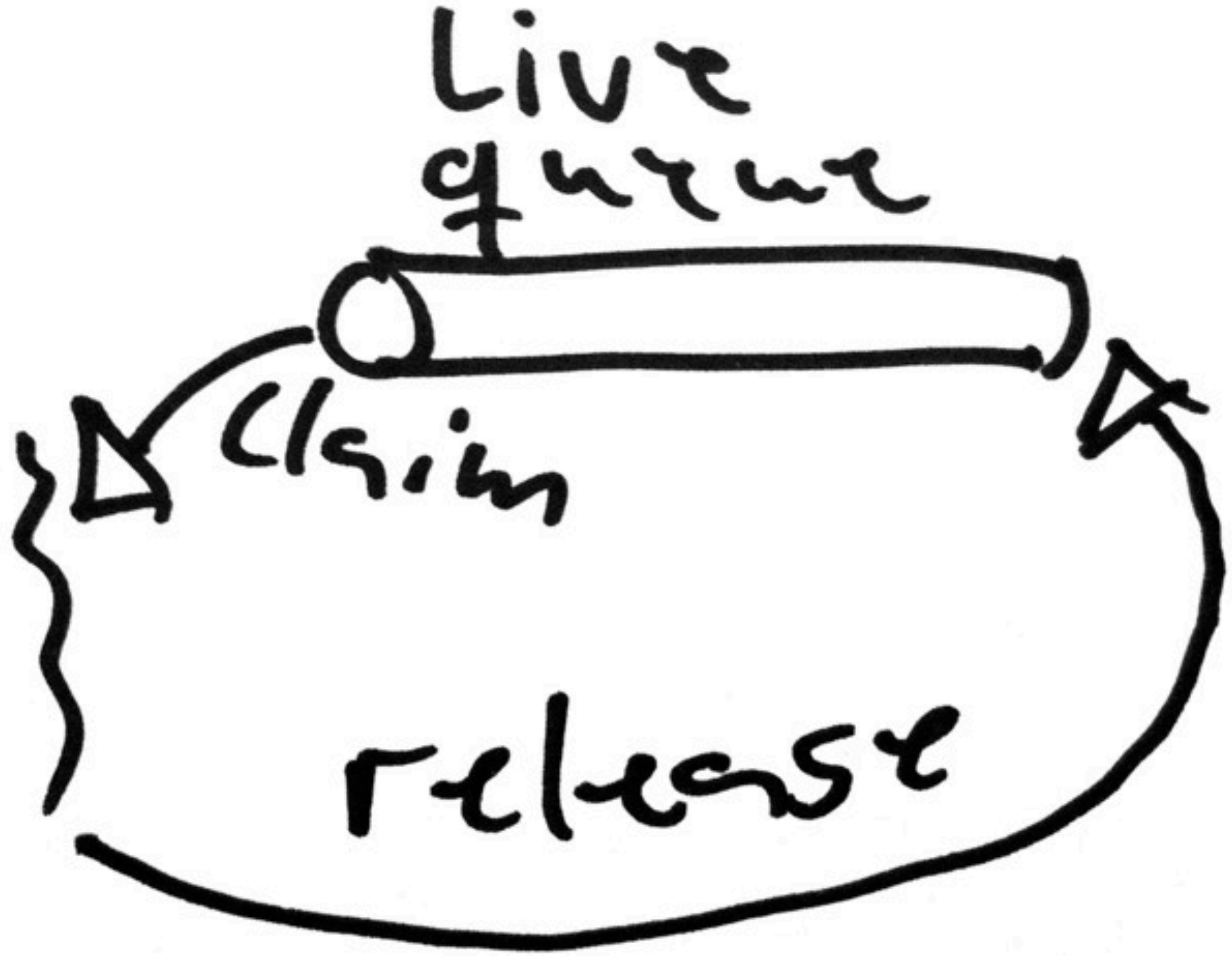


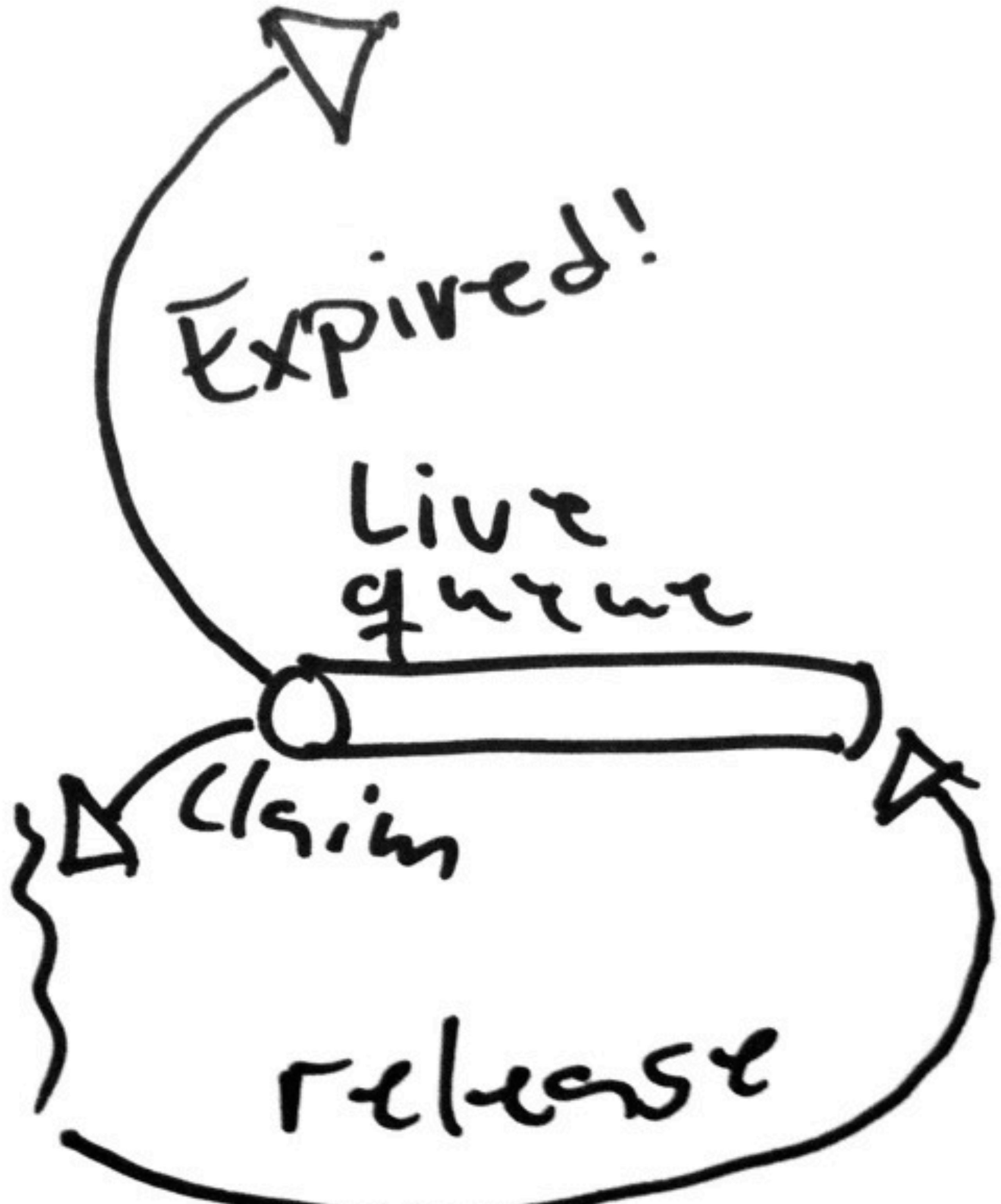
Throughput
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
0
ops/sec
1
2
3
4
5
6
8
10
12
14
16
threads
Queue opssec
Furious opssec

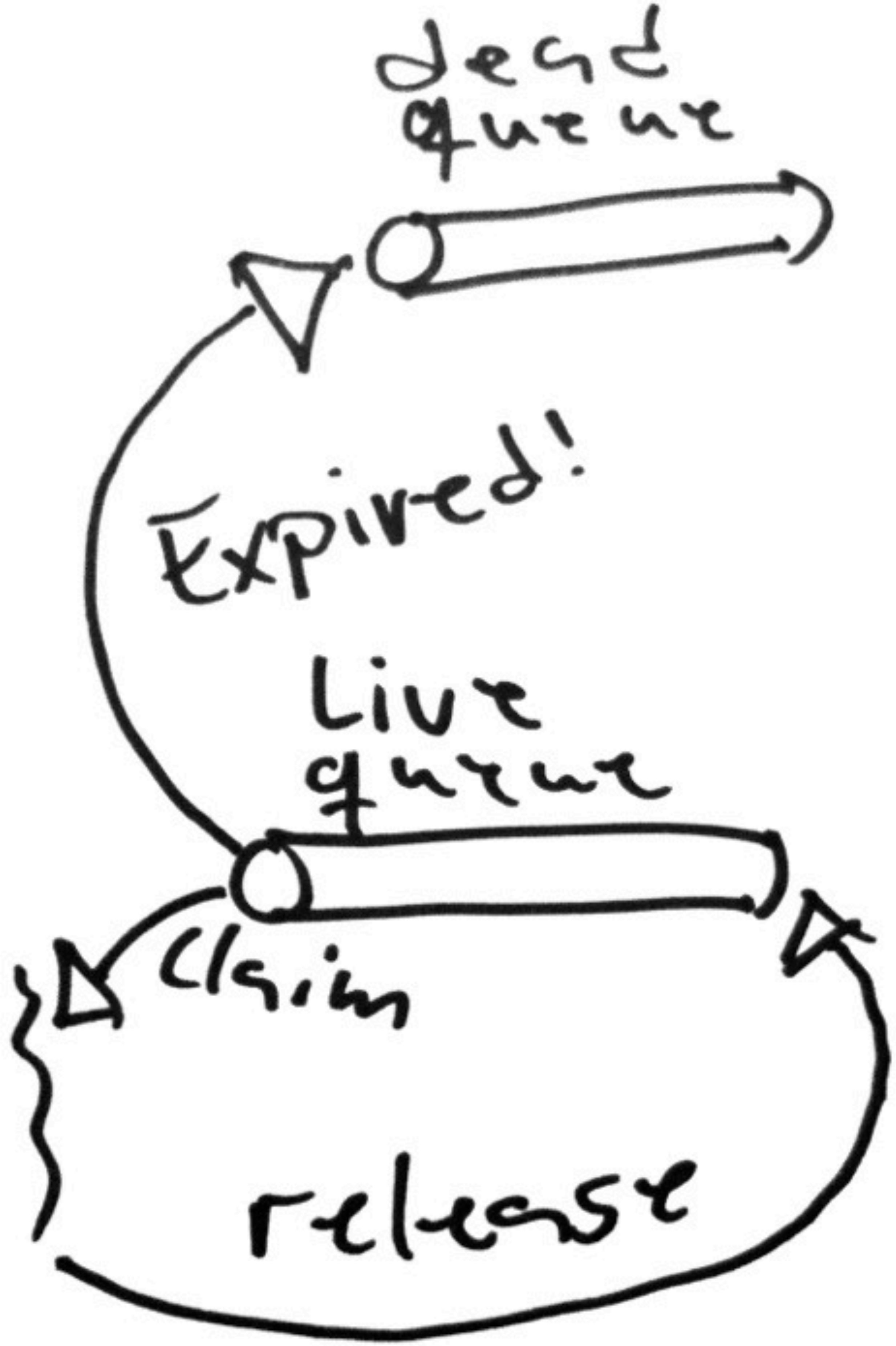

Throughput
15,000,000
14,000,000
13,000,000
12,000,000
11,000,000
10,000,000
9,000,000
8,000,000
7,000,000
6,000,000
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
0
ops/sec
1
2
3
4
5
6
8
10
12
14
16
threads
Queue opssec
Furious opssec
Blaze opssec

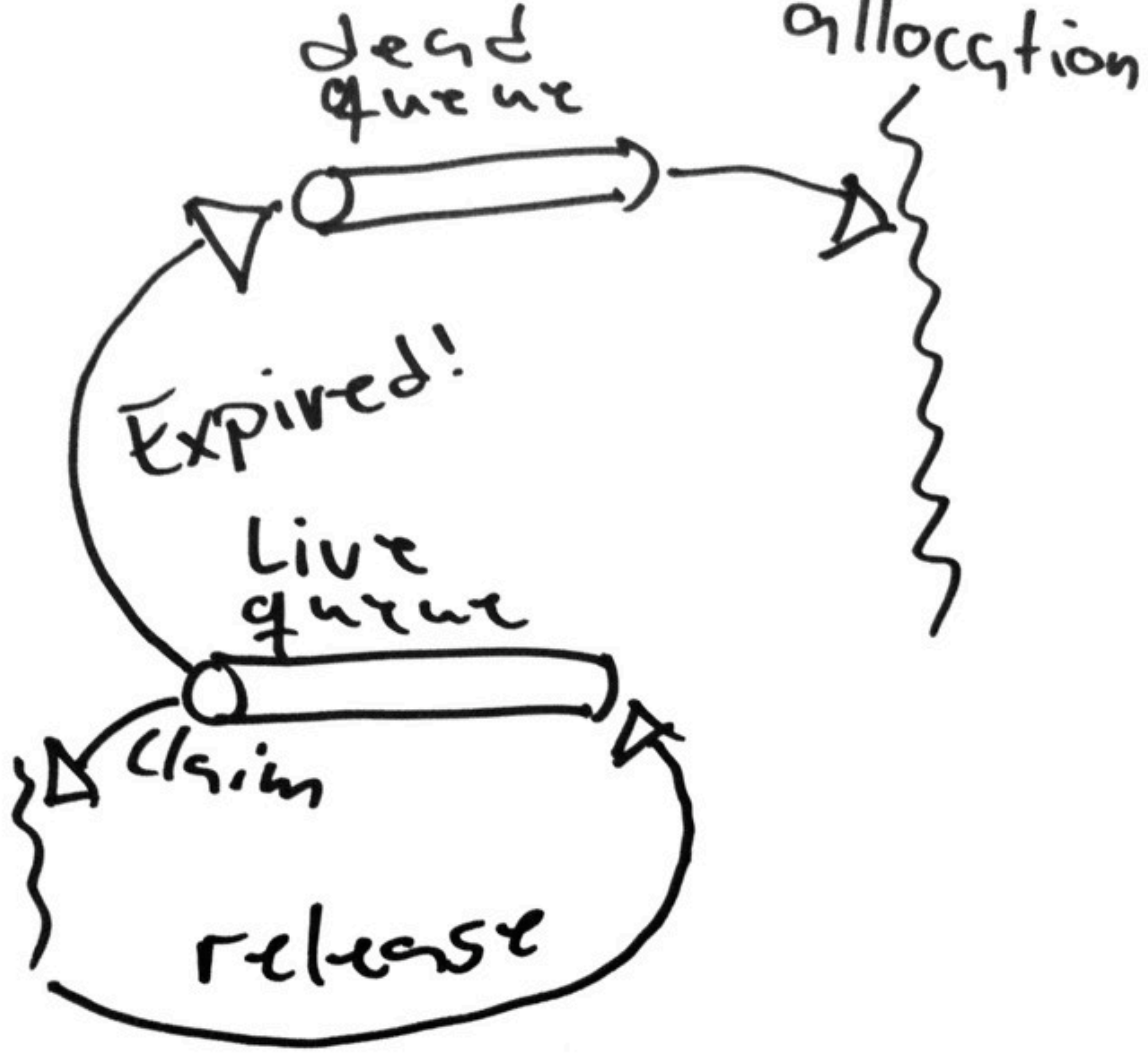
QueuePool

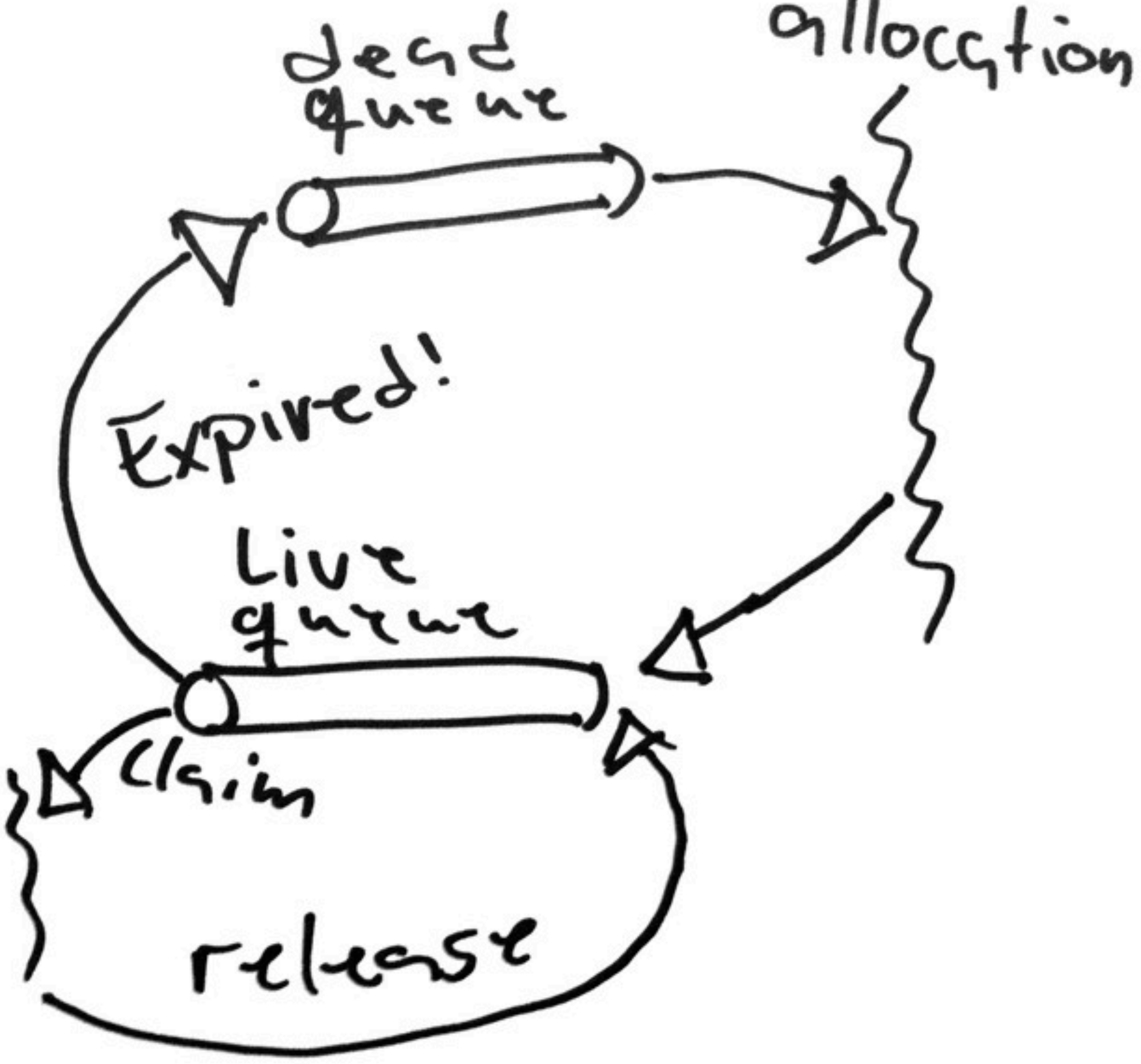

Live queue
Claim


Live queue
claim
release

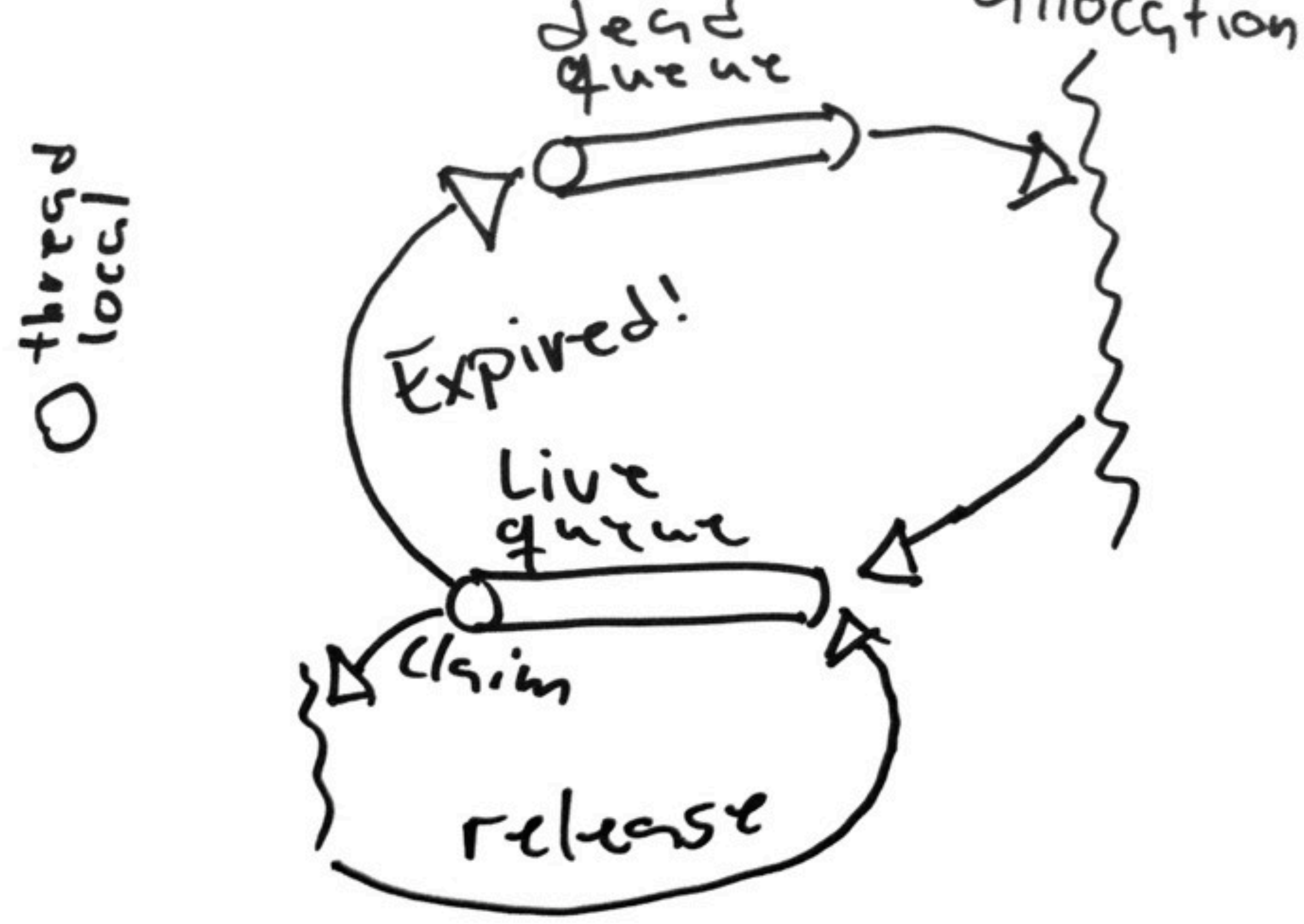

Expired!
Live queue
claim
release

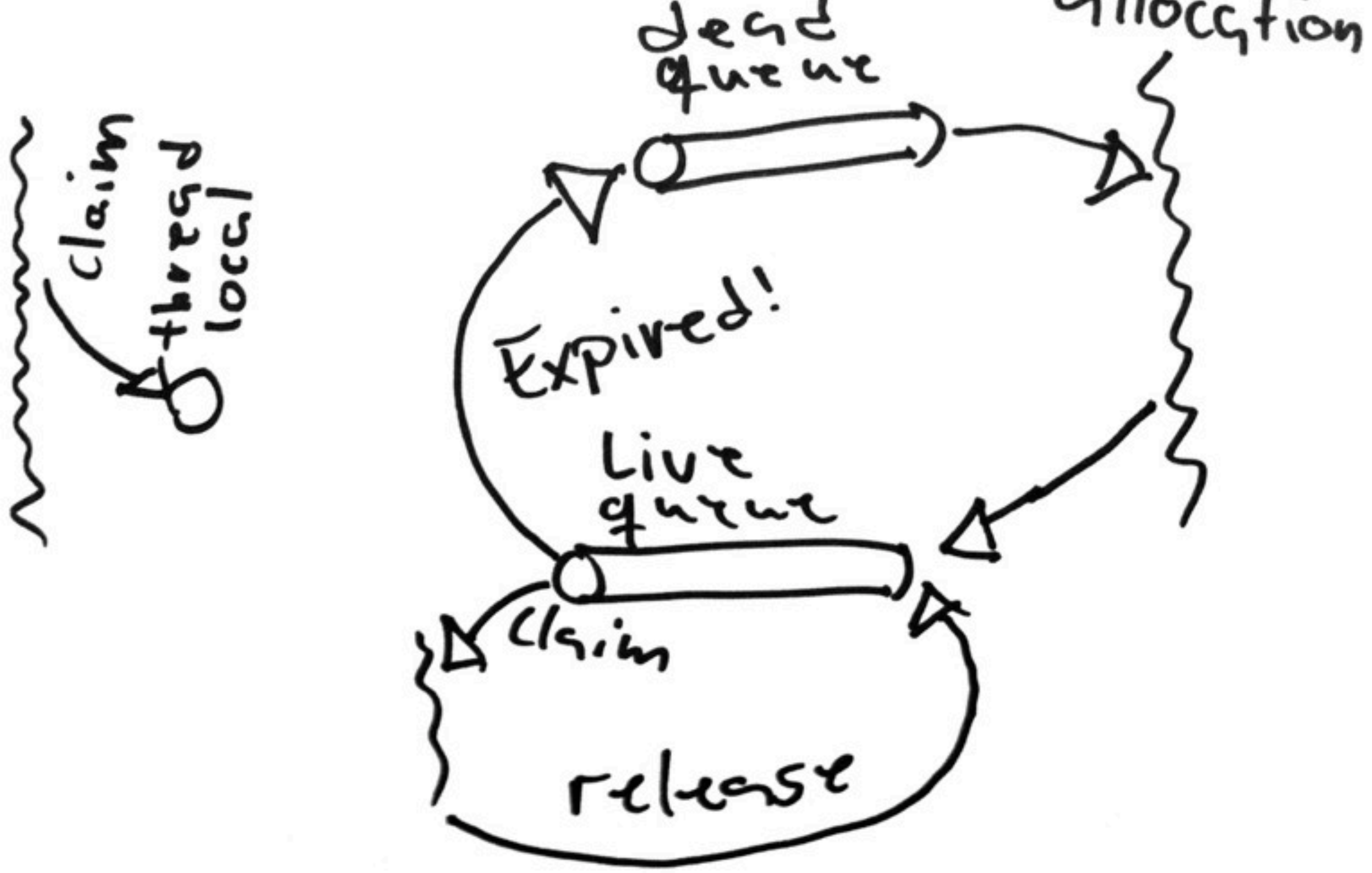

dead queue
Expired!
Live queue
Claim
release

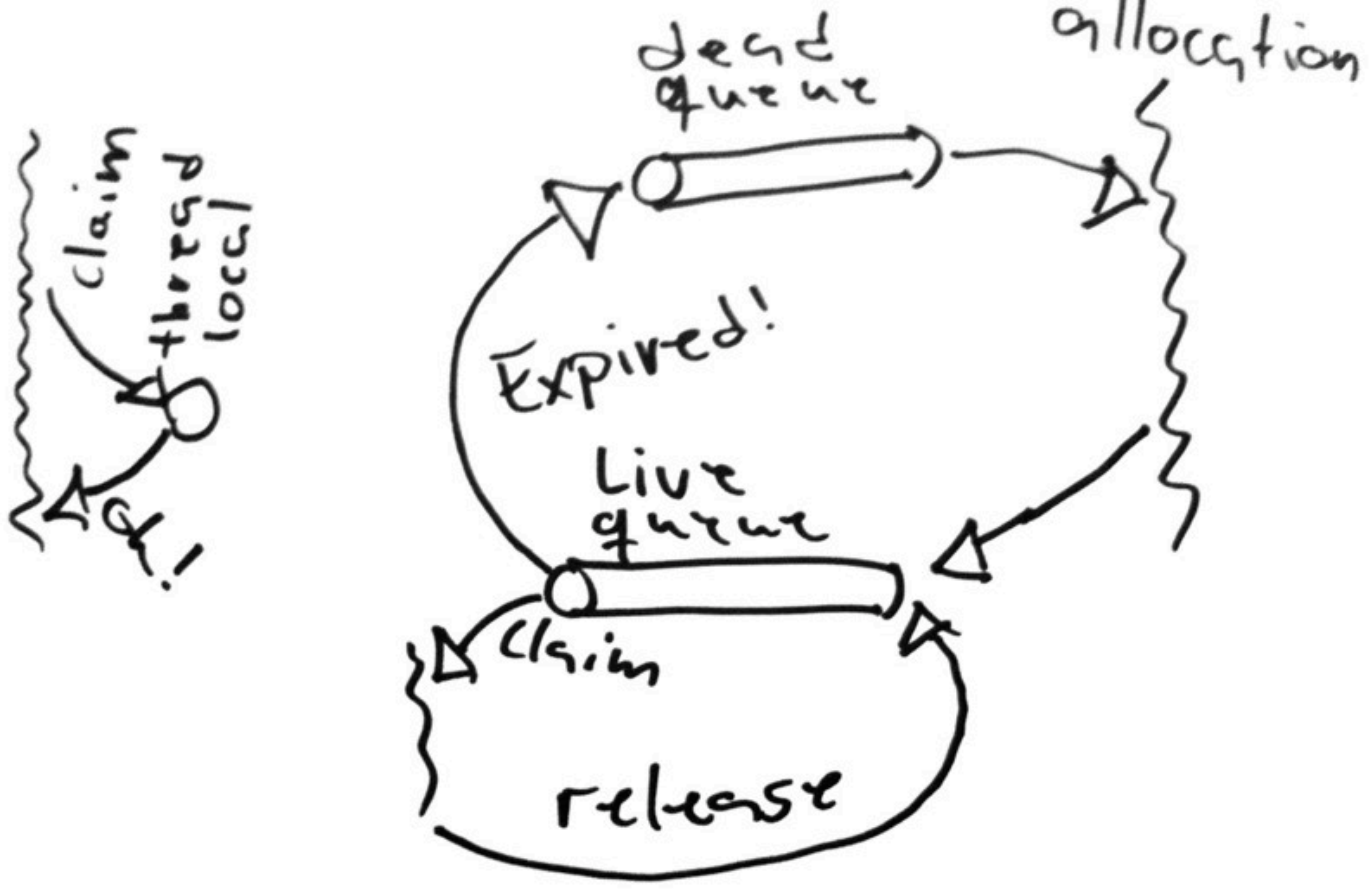

dead queue
allocation
Expired!
Live queue
claim
release

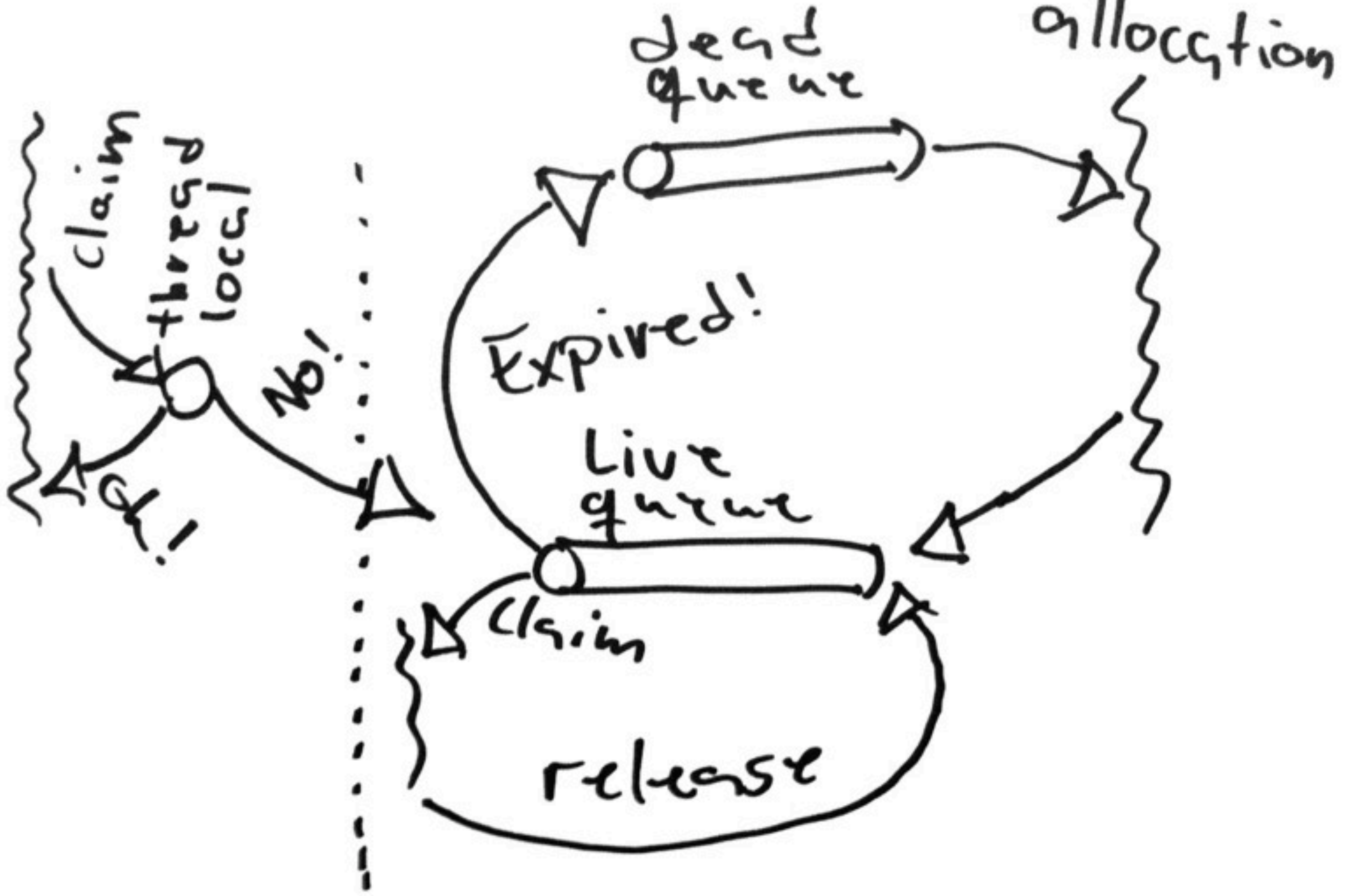

dead queue
allocation
Expired!
Live queue
claim
release

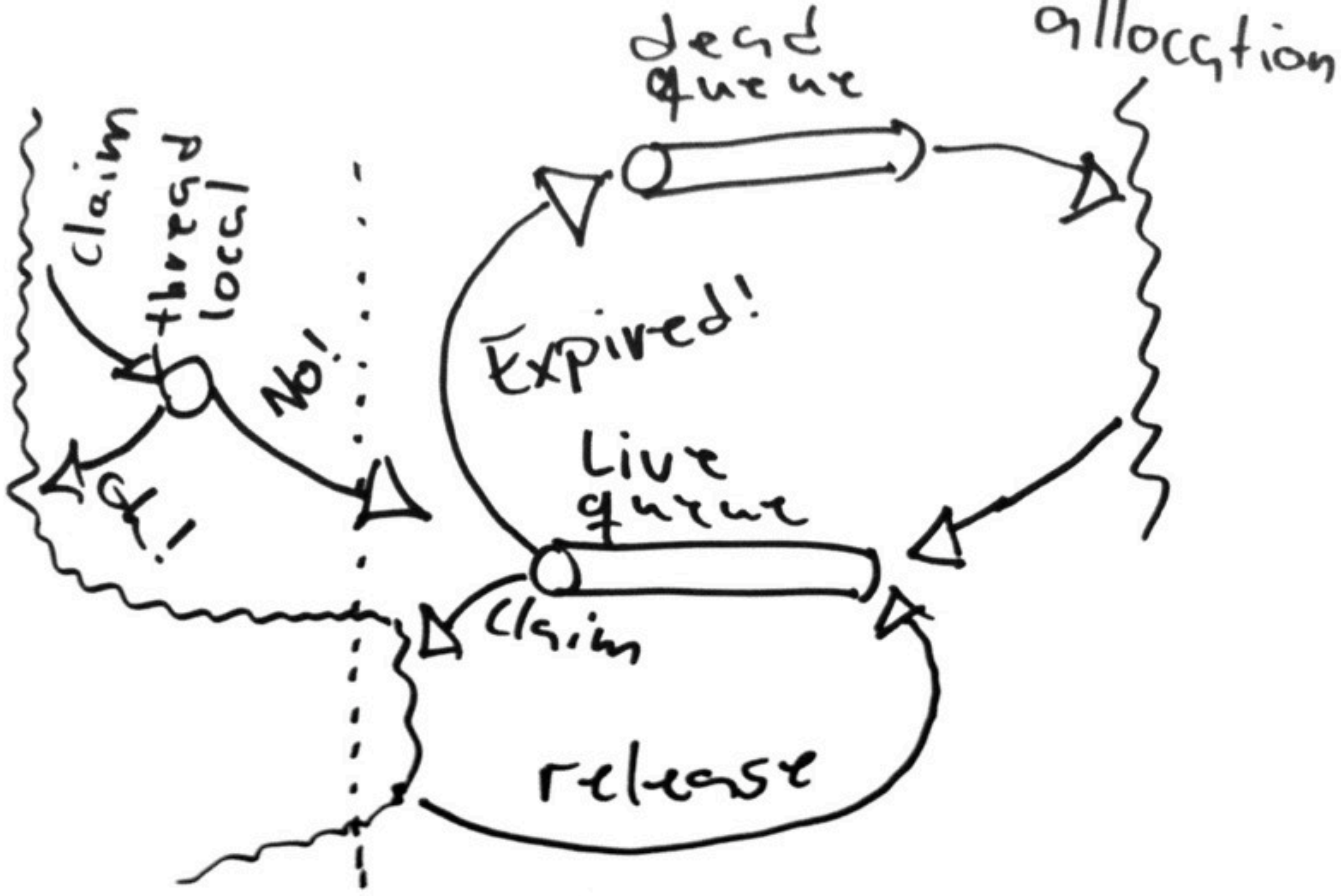
BlazePool


dead queue
allocation
thread local
Expired!
Live queue
claim
release


dead queue
allocation
claim
thread local
Expired!
Live queue
claim
release


claim
thread local
ok!
dead queue
allocation
Expired!
Live queue
claim
release


dead queue
allocation
claim
thread local
No!
ok!
Expired!
Live queue
claim
release


dead queue
allocation
claim
thread local
No!
Ok!
Expired!
Live queue
claim
release

Contention

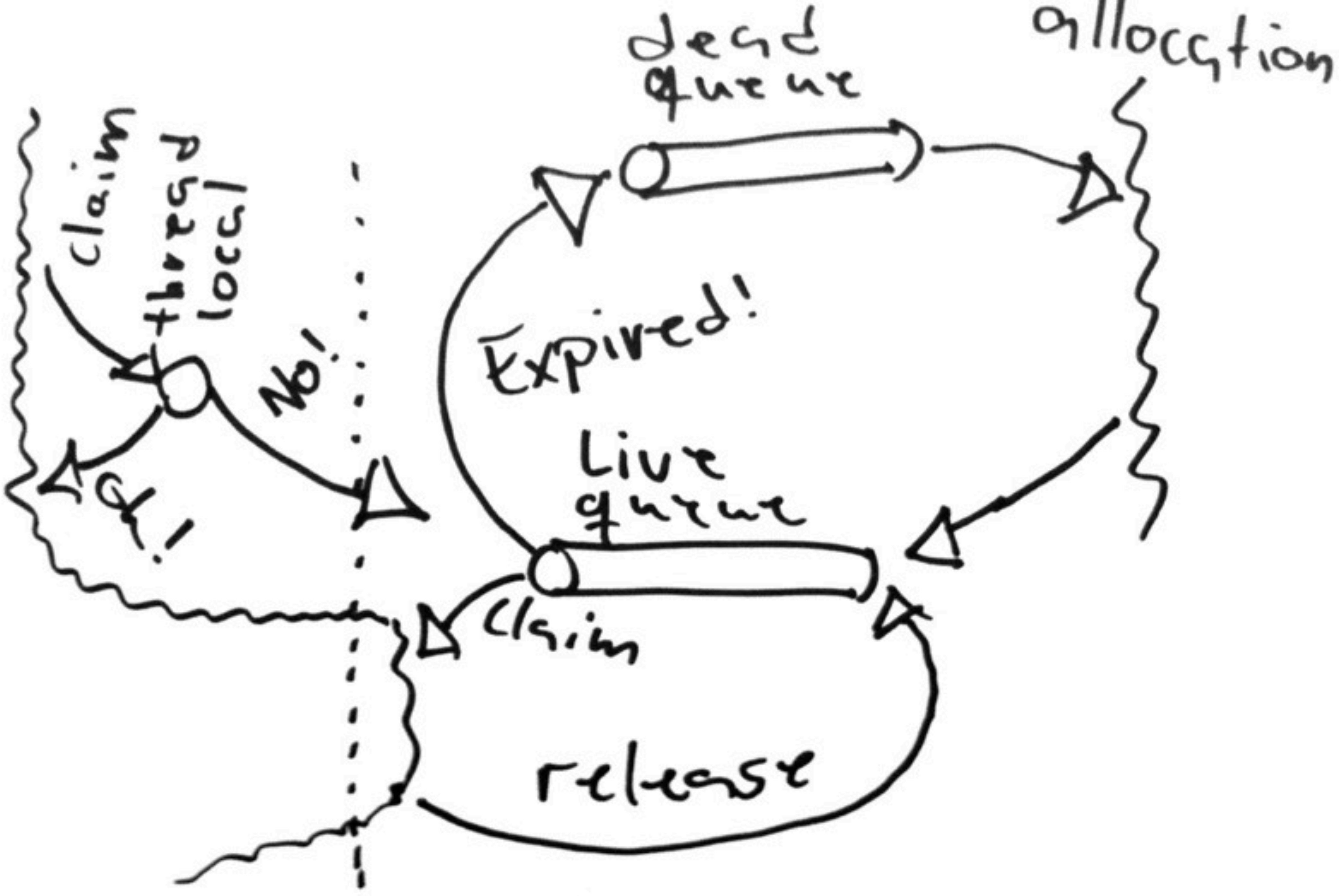
$$=$$

Slow

Single Writer Principle!

or stripe memory writes
... if you need the performance
... and you can afford the complexity

You can never compromise correctness!


dead queue
allocation
claim
thread local
No!
ok!
Expired!
Live queue
claim
release

?

@chvest

http://chrisvest.github.com/stormpot/