Pressure Feedback for LRU Maps

Joe Stringer
Isovalent

Agenda

- Background
- eBPF LRU hashmap deep dive
- Discussion

It all starts with an incident...

- Packets are being dropped after a production upgrade
- Two curious clues upon closer inspection:
  - Policy drops for packets towards ephemeral port range
  - "CT Map insertion failure" metric count.
- At the time, no metrics for flow count directly
  - Churn on CT map? Eyeballed at 10Ks of entries changing in seconds in a ~250K size map

Cilium's connection tracker

- Implement CT via LRU hashmap for firewall & NAT
- Properties we like?
  - Hash table properties
  - Garbage collect as you go

Cilium's connection tracker

- Implement CT via LRU hashmap for firewall & NAT
- Properties we like?
  - Hash table properties
  - Garbage collect as you go
- Difficulties?
  - Understanding current contention + signalling impact
  - LRU doesn't respect Cilium timers
  - Tied fates for CT and NAT?

Contention

Cause

> We have identified the primary cause of the drops as a set of very connection-heavy ingress pods that ended up overflowing the conntrack tables on select nodes. By spreading these ingresses more evenly using anti-affinity rules, we have eliminated the most negative effects and stabilized the env.

Contention

How do we make this more obvious?

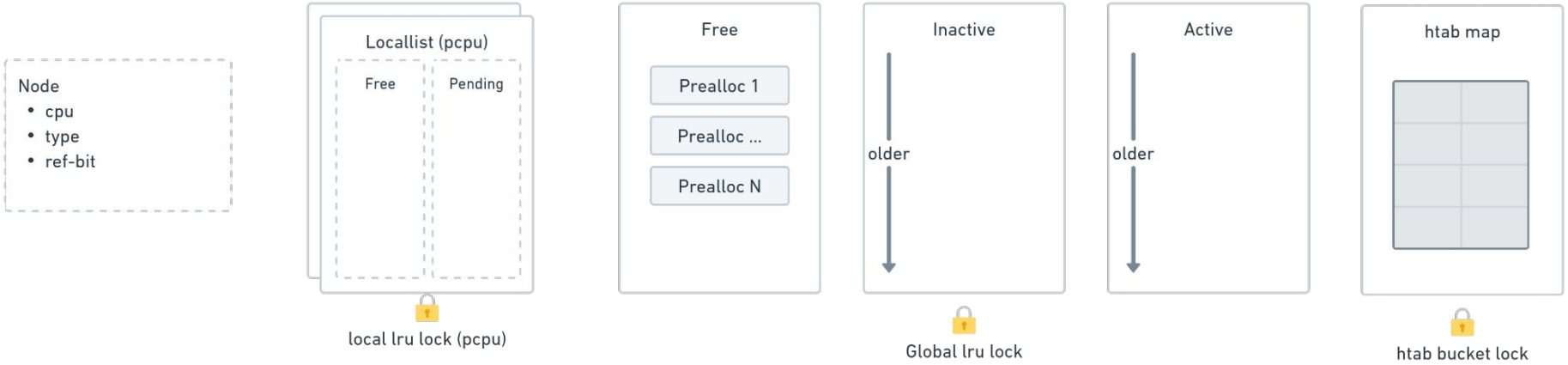
- Strong signal: CT Map insertion failure
  - Count: 14 instances over hours.
  - Not sensitive enough?
- How "full" is the map?
  - High rate of change. Can dump & count (expensive)
  - Inc counter on insert, dec counter on delete?
    - LRU doesn't allow us to count delete by LRU
    - As soon as table is full, cannot track how full.

Idea: Signal in return code


```
--- a/include/uapi/linux/bpf.h
+++ b/include/uapi/linux/bpf.h
@@ -1570,6 +1574,13 @@ union bpf_attr {
  *		**BPF_ANY**
  *			No condition on the existence of the entry for *key*.
+ *		**BPF_F_PRESSURE**
+ *			If the update would successfully replace an existing
+ *			entry per the map properties, this helper replaces the
+ *			entry and returns **-EINPROGRESS**. This flag is only
+ *			valid for the following map types:
+ *			* **BPF_MAP_TYPE_LRU_HASH**
+ *			* **BPF_MAP_TYPE_LRU_PERCPU_HASH**
  *
  *		Flag value **BPF_NOEXIST** cannot be used for maps of types
```

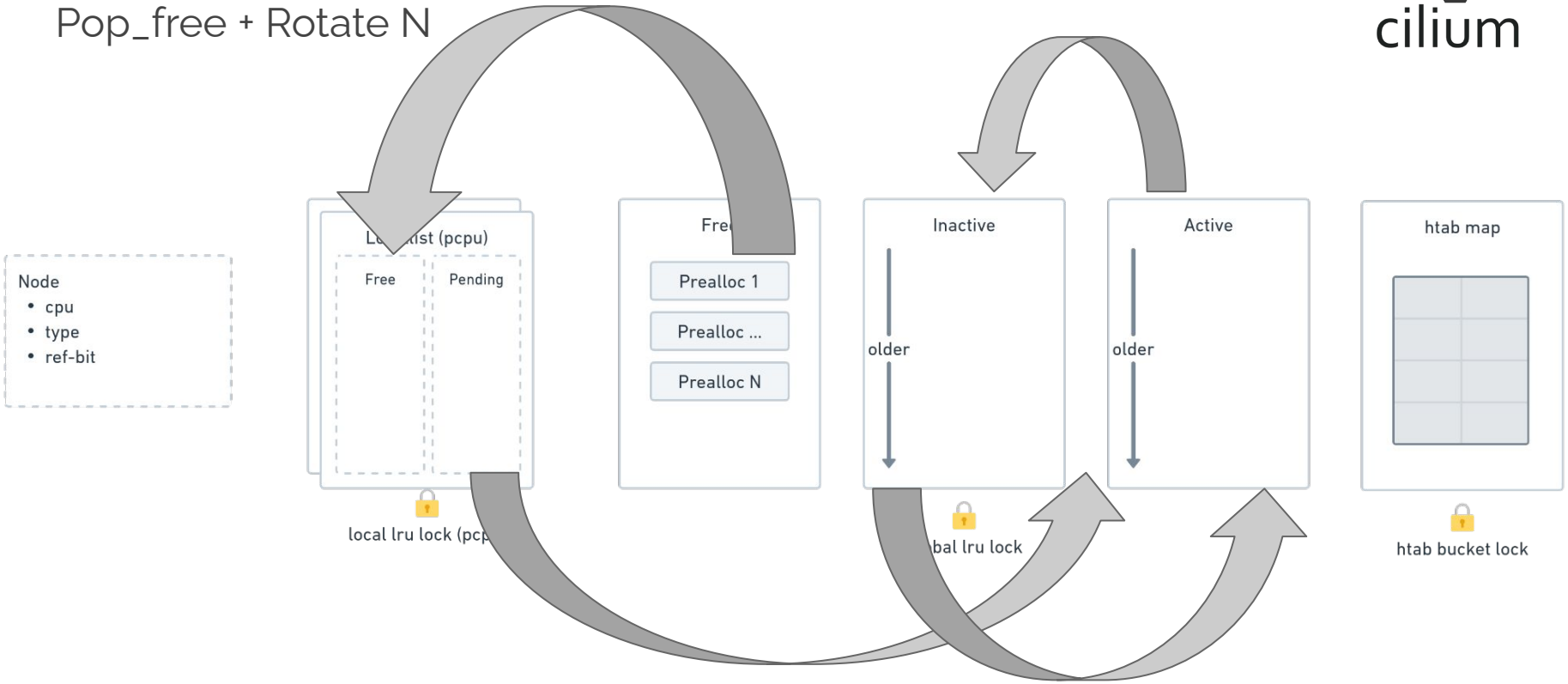
LRU deep dive

Structure


Node
• cpu
• type
• ref-bit
Locallist (pcpu)
Free
Pending
local lru lock (pcpu)
Free
Prealloc 1
Prealloc ...
Prealloc N
Inactive
older
Global lru lock
Active
older
htab map
htab bucket lock

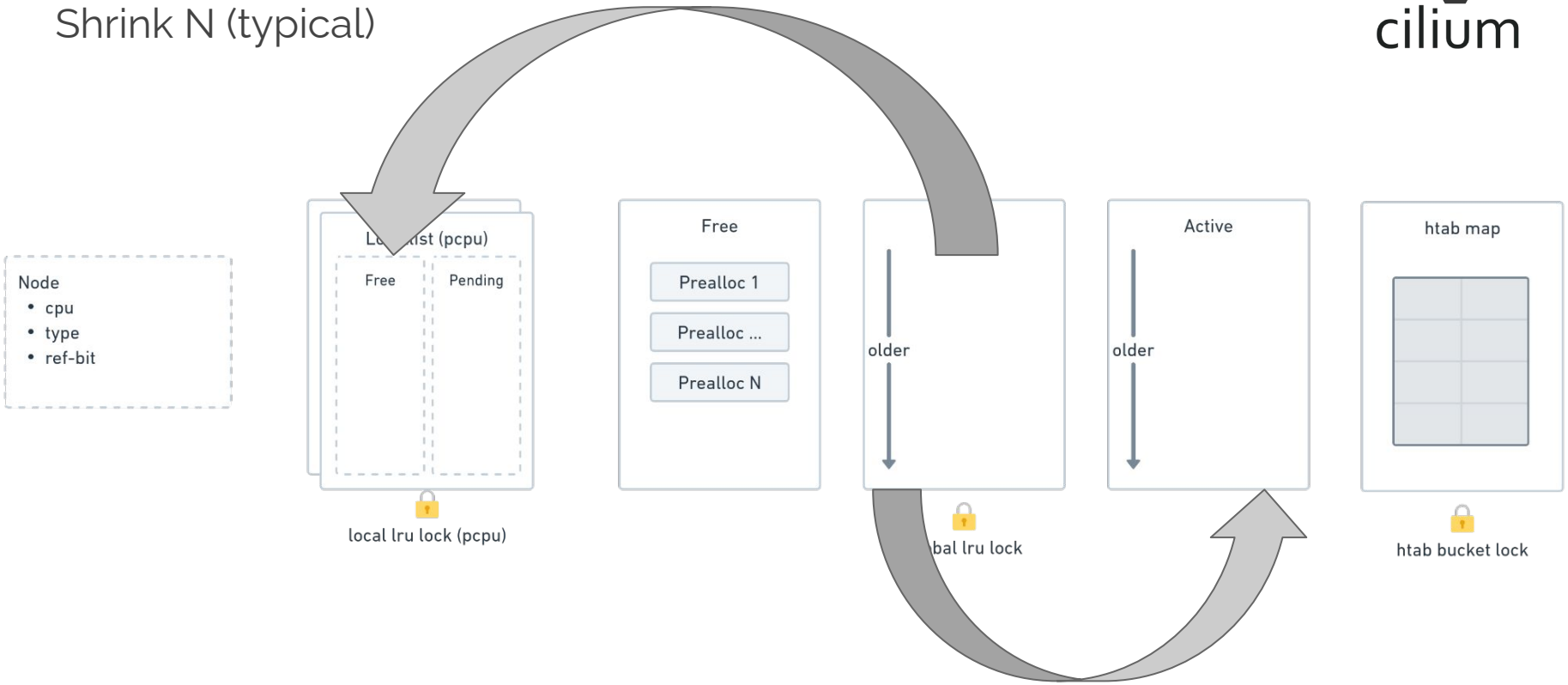
Structure

Pop_free + Rotate N



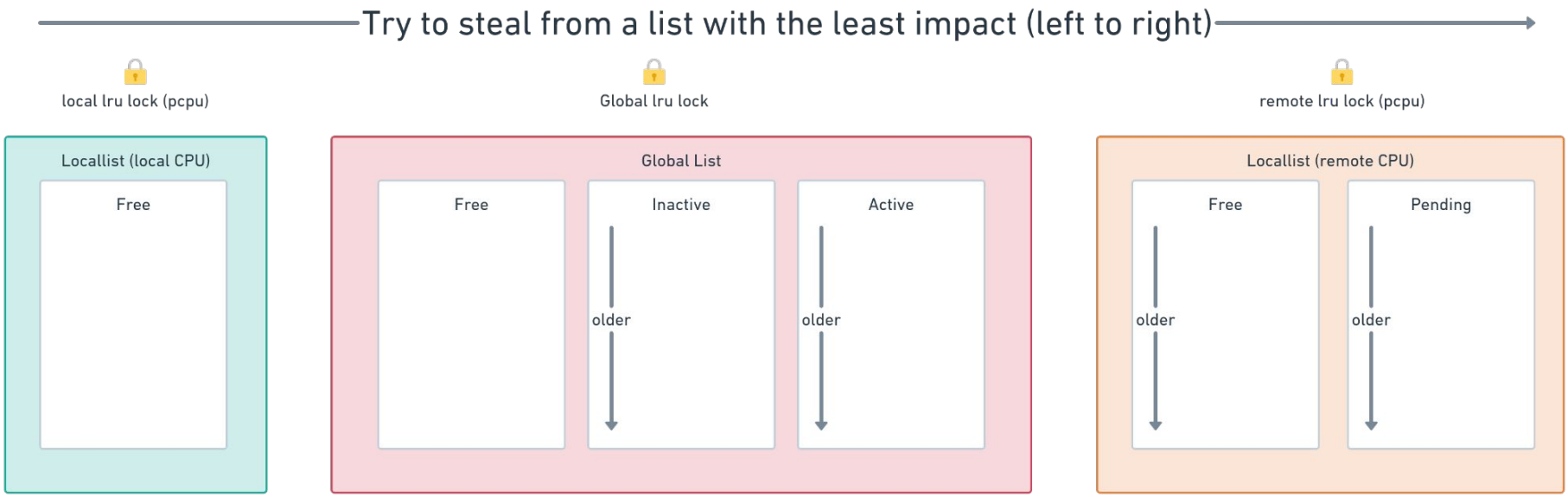
Structure

Shrink N (typical)



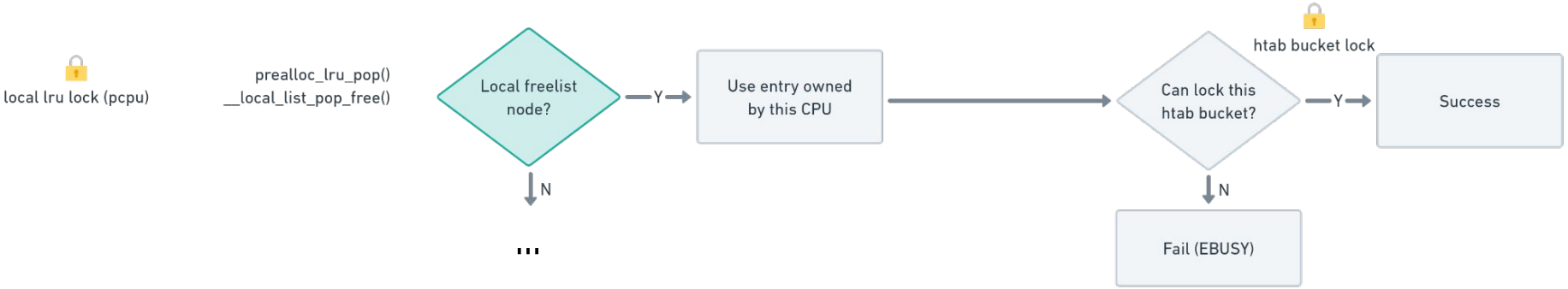
LRU Update

Priority order for finding a "least recent" entry



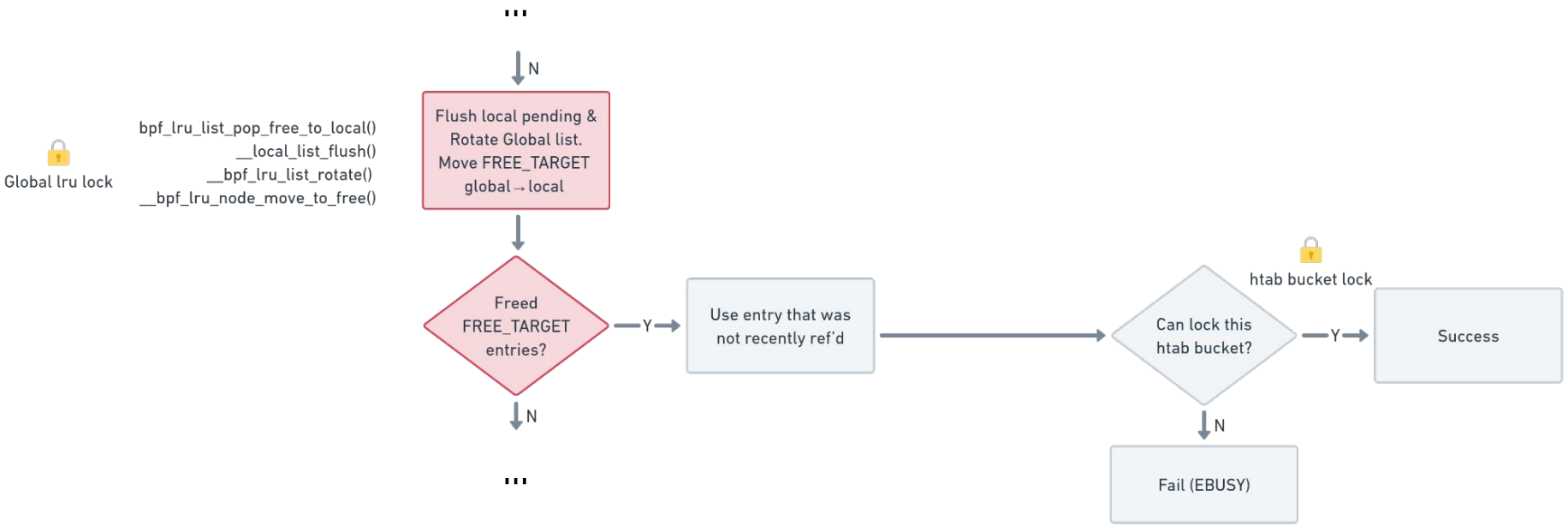
LRU Update

Initial updates in preallocated map



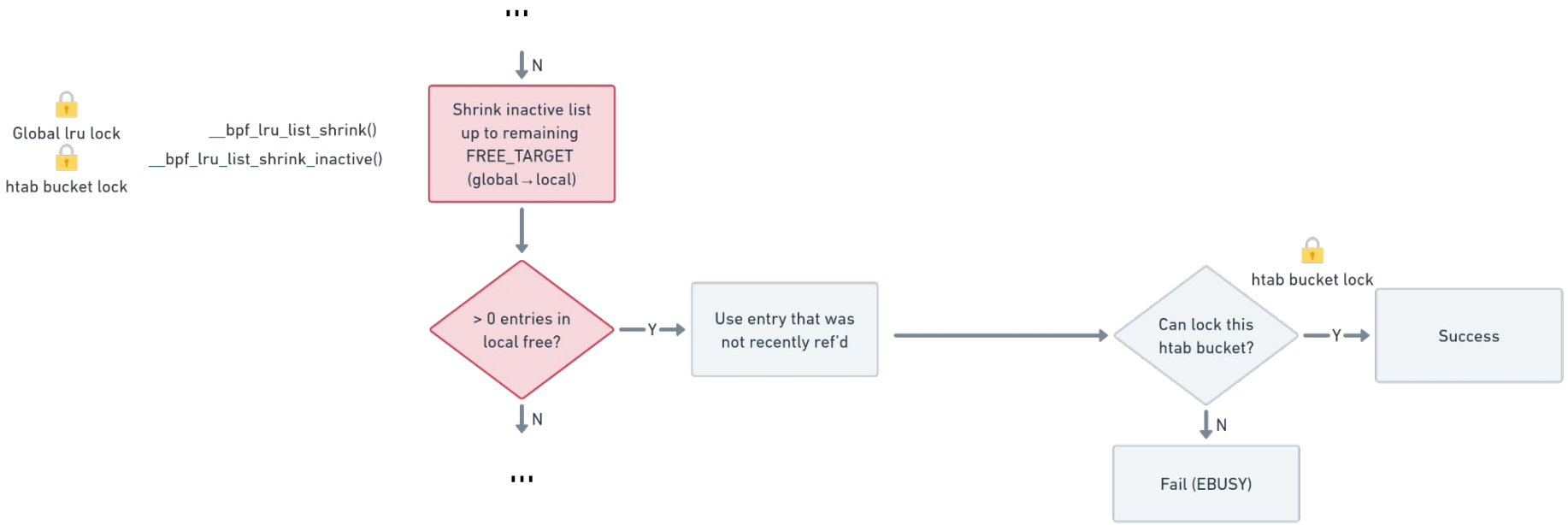
LRU Update

No entries available on local CPU. Rotate global list.



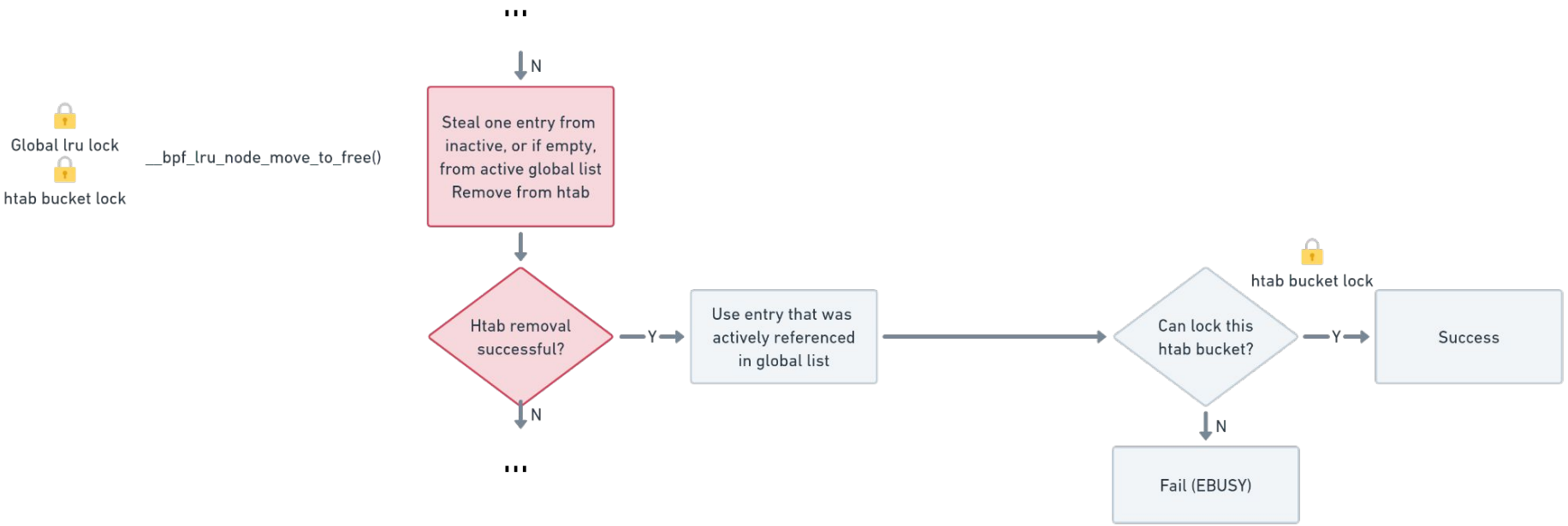
LRU Update

Global freelist did not have FREE_TARGET entries. Shrink.



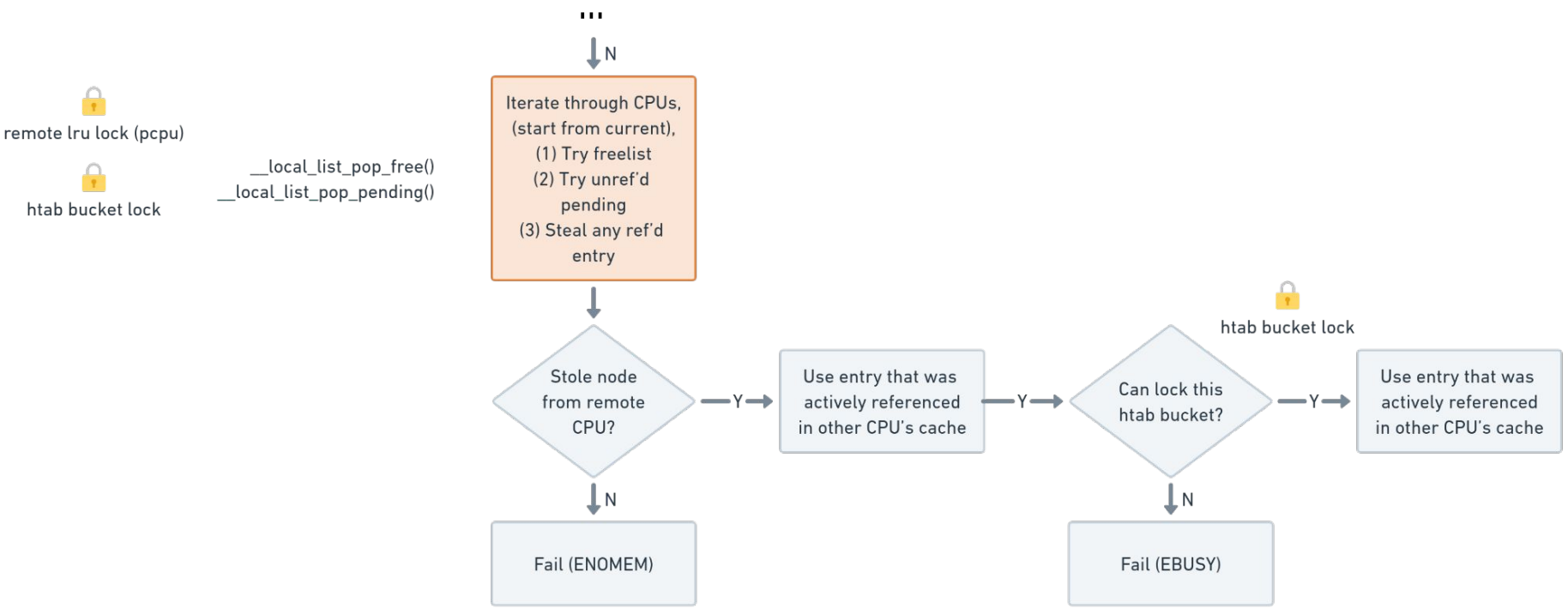
LRU Update

Despite shrink, no inactive entries identified. Steal from global map.



LRU Update

Stole an entry, foiled by htab contention. Steal from another CPU.



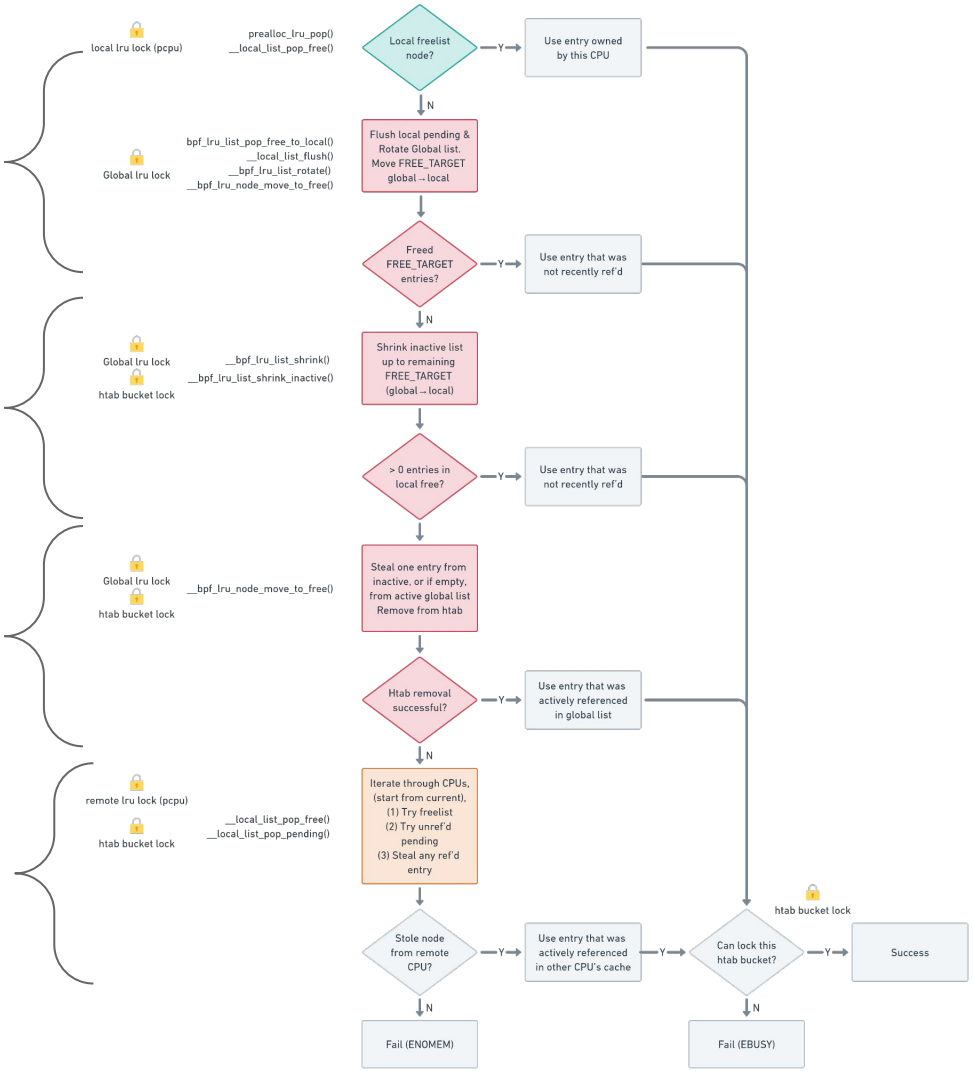
Discussion

Ideal range: sufficient free entries on local CPU, or easy to rotate entries out to local CPU.

GC kicks in. Start using htab lock. Additional contention?

Start to panic - choose any entry. Fails if cannot delete any entry from htab.

Unable to locate any inactive entries globally, so just start stealing new ones from free or pending lists regardless if active.



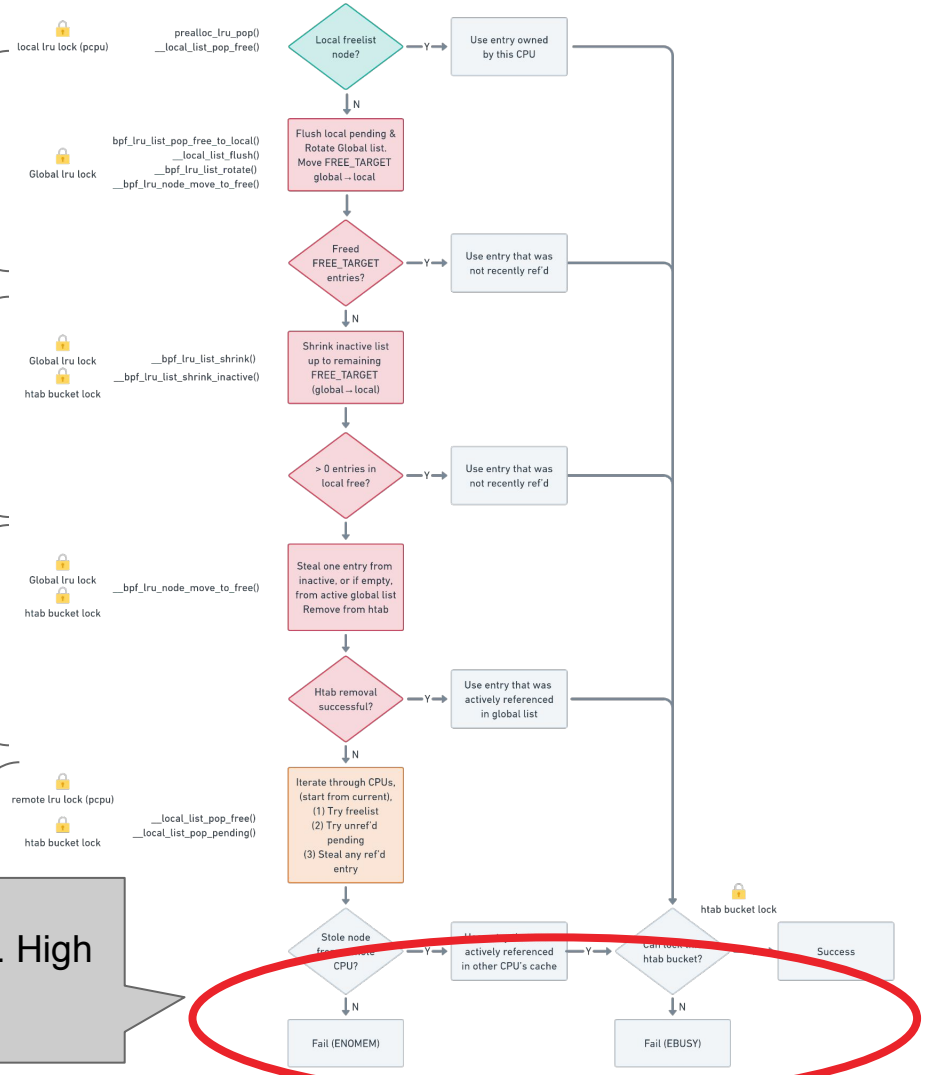
Ideal range: sufficient free entries on local CPU, or easy to rotate entries out to local CPU.

GC kicks in. Start using htab lock. Additional contention?

Start to panic - choose any entry. Fails if cannot delete any entry from htab.

Unable to locate any inactive entries globa
stealing new
pending lists

We monitor these today. High signal, low frequency.



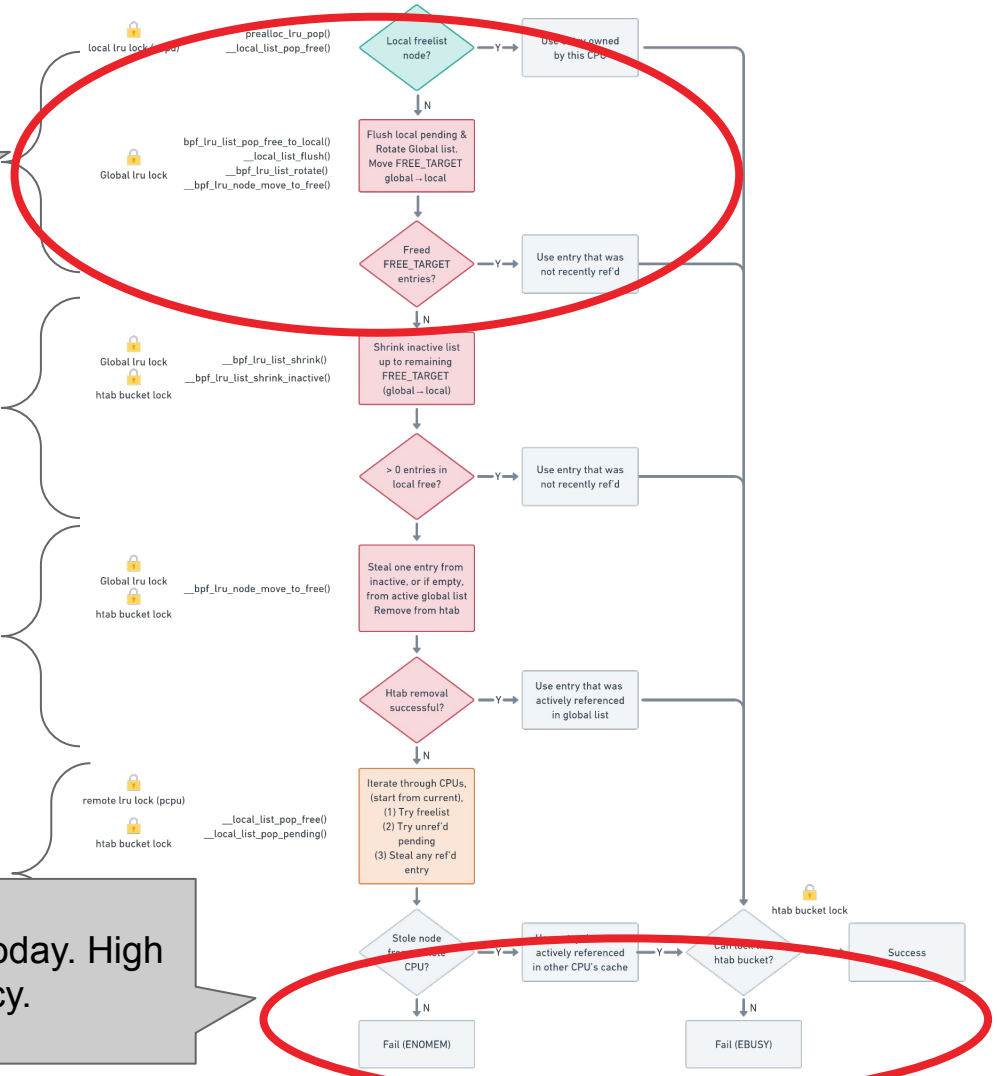
Count every implicit delete
Report usage: % utilization
Weak signal: map full

GC kicks in. Start using htab lock.
Additional contention?

Start to panic - choose any entry.
Fails if cannot delete any entry
from htab.

Unable to locate any inactive
entries globa
stealing new
pending lists

We monitor these today. High signal, low frequency.



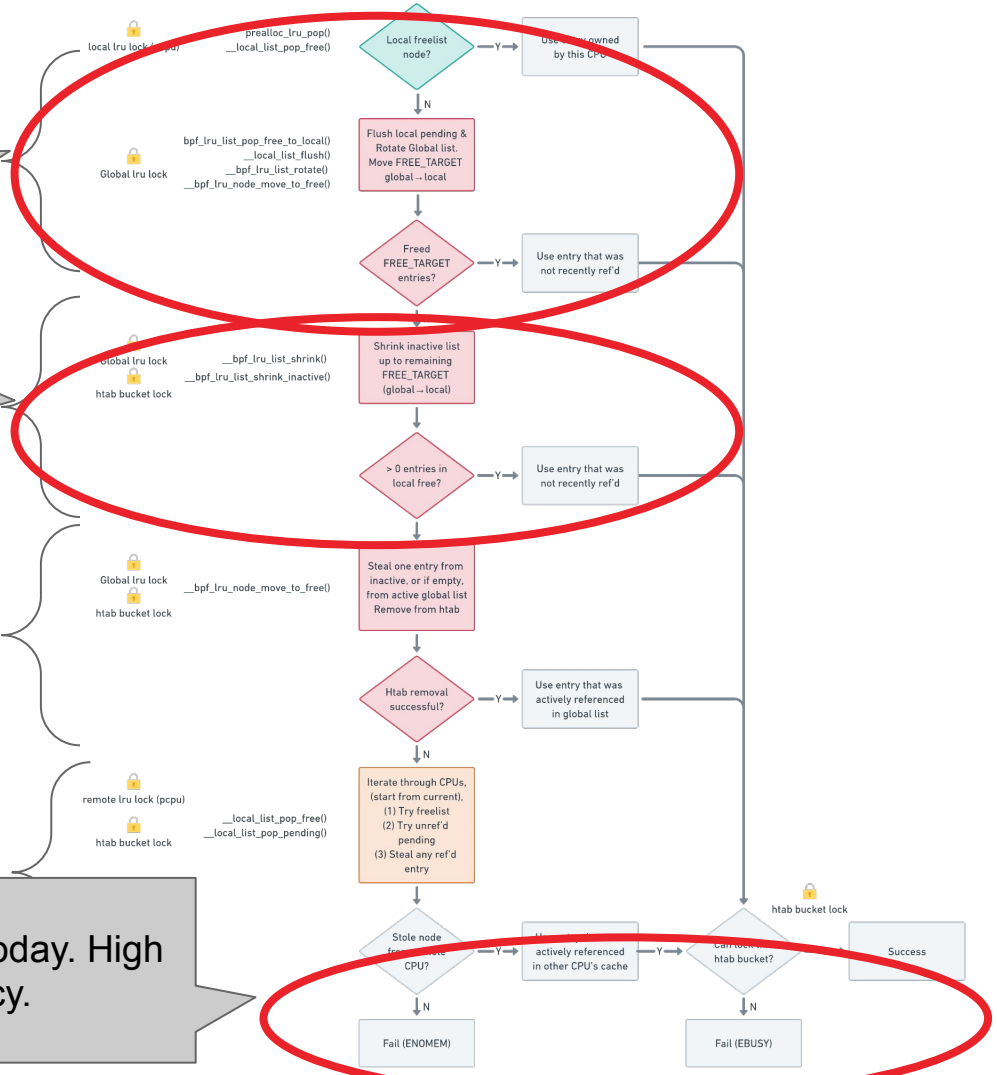
Count every implicit delete
Report usage: % utilization
Weak signal: map full

Count shrink runs or shrinked count?
~GC rate?

Start to panic - choose any entry.
Fails if cannot delete any entry from htab.

Unable to locate any inactive entries globa
stealing new
pending lists

We monitor these today. High signal, low frequency.



Count every implicit delete
Report usage: % utilization
Weak signal: map full

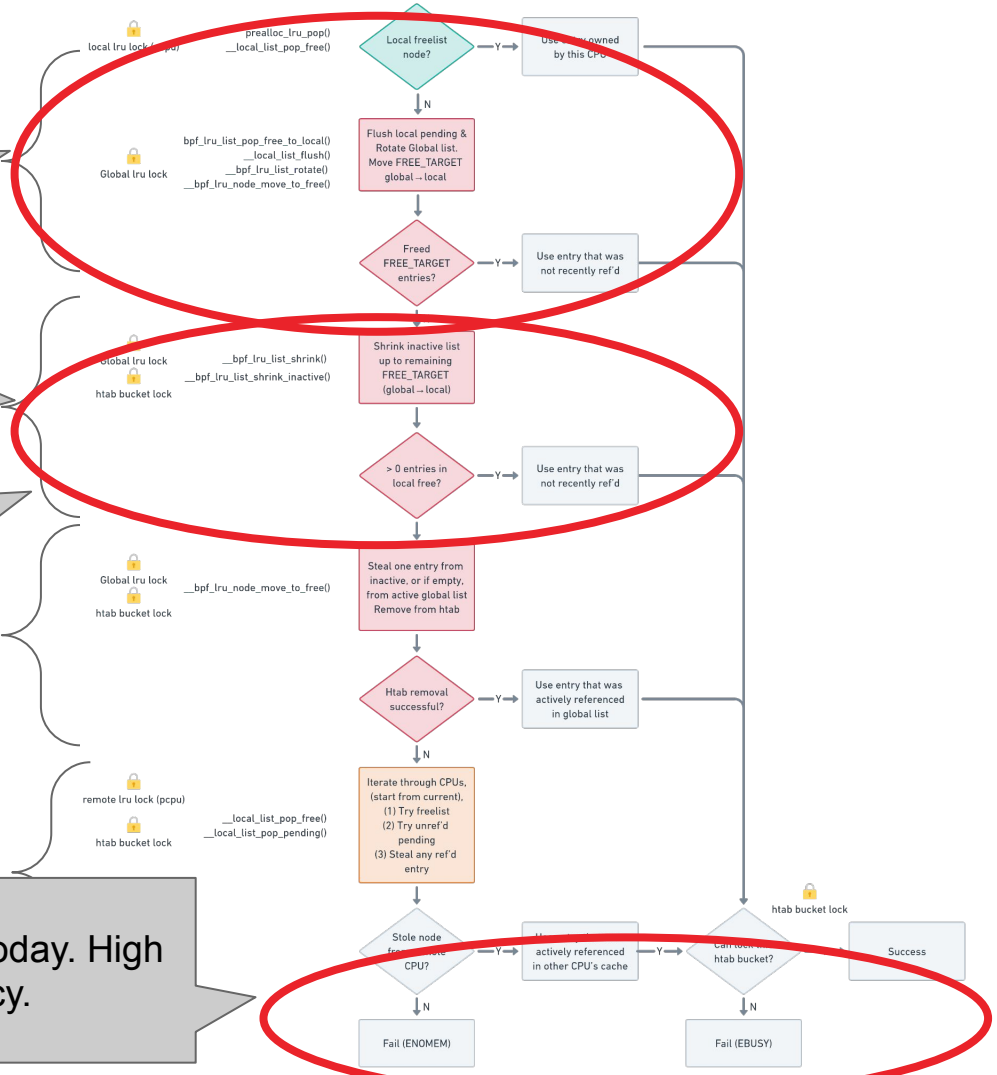
Count shrink runs or shrinked count?
~GC rate?

Use N / FREE_TARGET proportion as a signal of contention?

from htab.

Unable to locate any inactive entries globa
stealing new
pending lists

We monitor these today. High signal, low frequency.



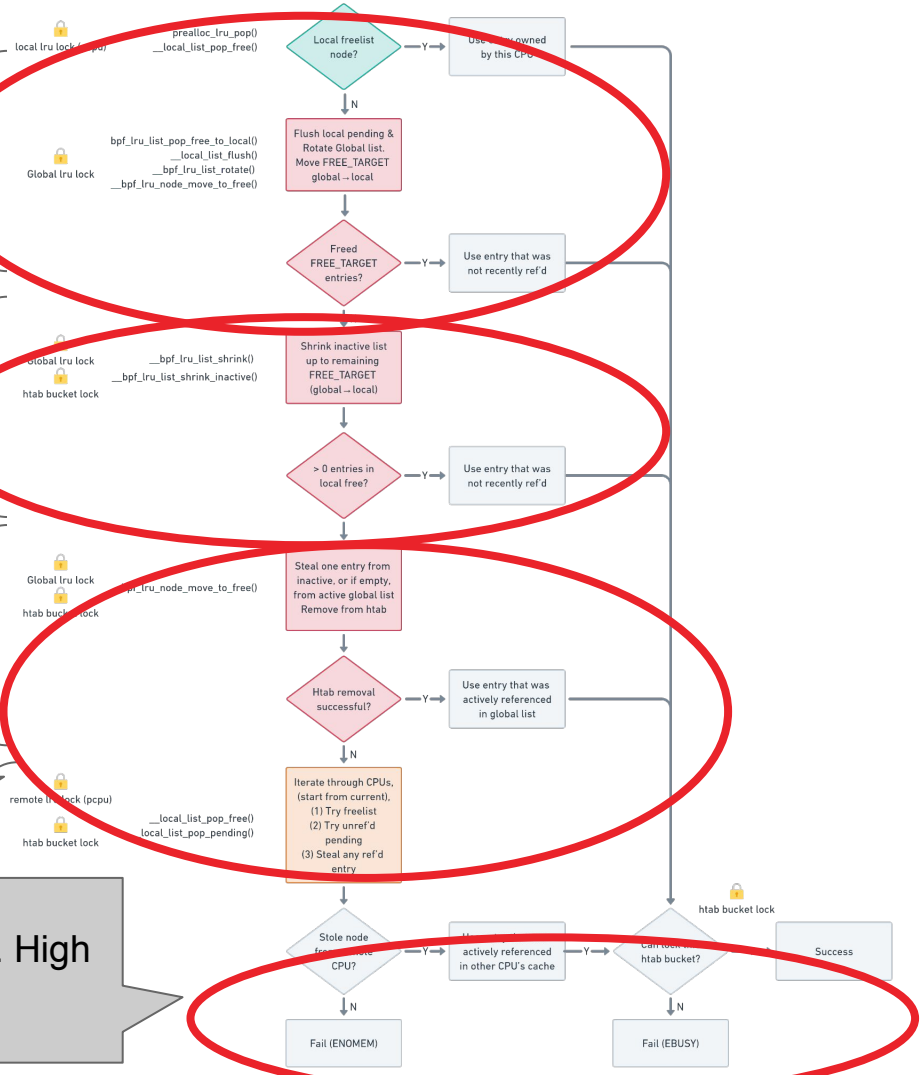
Count every implicit delete
Report usage: % utilization
Weak signal: map full

Count shrink runs or shrinked count?
~GC rate?

Use N / FREE_TARGET proportion as a signal of contention?

Report whether active entry was stolen from inactive / active list? Details…

We monitor these today. High signal, low frequency.



Or something more drastic?


```
From 3a08c2fd763450a927d1130de078d6f9e74944fb Mon Sep 17 00:00:00 2001
From: Martin KaFai Lau <kafai@fb.com>
Date: Fri, 11 Nov 2016 10:55:06 -0800
Subject: [PATCH] bpf: LRU List

Introduce bpf_lru_list which will provide LRU capability to
the bpf_htab in the later patch.

* General Thoughts:
1. Target use case.  Read is more often than update.
   (i.e. bpf_lookup_elem() is more often than bpf_update_elem()).
   If bpf_prog does a bpf_lookup_elem() first and then an in-place
   update, it still counts as a read operation to the LRU list concern.
```

@ciliumproject
@joestringernz