The Routing Cache is Dead, Now What?

David S. Miller

Red Hat Inc.

Netfilter Workshop 2013, Copenhagen, Denmark, 2013

Old Path

- Hash table lookup
- Keyed on all flow key members
- Slow path FIB lookup on hash lookup miss
- Hash table layout and contents unpredictable
- Therefore, performance unpredictable
- Table contents controllable by remote entities
- Design fundamentally lends itself to DoS attacks
- Garbage collection

Old Path - Outline

- Hash table demux, found? If yes, we're done.
- fib_lookup(), get FIB nexthop entry.
- Source validation.
- Build route cache entry, potentially run GC.
- Lookup neighbour, attach to route.
- Lookup inetpeer, attach to route.
- Insert route cache entry to hash table.
- Ok, finally, we're done.

New Path

- FIB lookup performed on every route lookup
- FIB nexthop entry contains prebuilt cached route
- Lots of tricks to make such route sharing legal
- ... and lots of tricks to make this not so costly
- Advantage: Lookup cost is consistent and predictable
- Remote entities have zero influence over table
- Therefore DoS like that of GC'd hash table is simply impossible

New Path - Outline

- fib_lookup(), get FIB nexthop.
- Source validation.
- Use any nexthop exceptions found....
- else use FIB nexthop cached dst, if exists...
- else build a new cached dst if possible.
- Done.

Uncached Neighbour Entries

- Our ipv4 routes no longer cache nexthop ARP entries
- This is necessary for sharing local subnet routes
- Instead, neighbour looked up at packet output time
- Compensated by new, cheaper hash, and hash demux inlining
- Bonus side effect, no more “neighbour cache overflow”
- Yoshifuji HIDEAKI recently removed route neigh caching from IPV6 too

TCP Metrics Cache

- All dynamically changing route metrics moved here
- Needed to increase route shareability
- All route metrics are now read-only, kernel wide
- Cache is maintained in an RCU quick-demux hash table
- Demux happens at TCP connection setup and teardown
- Per-hashchain LRU is employed, max chain depth is 5
- Metrics timeout increased to 60 minutes, was 5 minutes
- Old timeout an unintented side effect of ip_rt_gc_timeout
- Netlink based dump/get/del added by Julian Anastasov
- Many bugs fixed by Eric Dumazet and Julian

FIB Nexthop Exceptions

- PMTU and redirects are an impediment to route sharing
- Cache of exception entries hangs off of fib_nh
- Managed as an LRU hash table similar to TCP Metrics Cache
- When PMTU or Redirect hits shared route, we make an entry
- Route lookup consults exception table before using shared route
- Side note: Trying to do metrics in inetpeer was a huge mistake.

Reverse Path Filtering

- For input routes, we have reverse path filtering
- Validation that packet came from where it should
- This is expensive, up to two extra FIB lookups
- Old code actually did validation unconditionally
- Besides RP filtering, used also for traffic classification
- New code eliminates lookups completely when possible

Per-cpu FIB cached routes

- Initial implementation creates one shared route in FIB nexthop
- Causes cache thrashing on output, especially for loopback
- Eric Dumazet to the rescue
- FIB nexthop output route cache becomes per-cpu
- Input route cache is still just one entry

Socket Pre-Demux

- Route lookups without routing cache now faster!
- At ip_rcv() we call per-protocol pre-demux handler
- Only TCP supported at the moment
- Pre-demux done before input route lookup
- Pre-demux protocols cache input routes in sockets
- Upon pre-demux, cached input route attached to SKB
- ip_rcv() sees this and can skip route lookup
- Complex trie based route lookup eliminated
- Decrease number of demuxes on input by one

Remove Double-Demux

- Every FIB lookup does two trie lookups for non-local destinations
- Once in local table, then once in global table
- Doing two lookups is pointless in almost all cases
- Ordered lookups in 2 tables only necessary for overlap
- But nobody has overlapping entries between these tables
- A combined global+local trie would work just as well
- Problem is making it look sane to userspace
- Still need to report routes in seperate tables
- If overlapping condition created, revert to current behavior
- If using more tables via FIB rules, also revert
- Improvement is largest when FIB validation occurs

Tunnels

- Some ipv4/ipv6 tunnels cache routes
- But not all of them do
- Caching is hard in certain circumstances
- Problem is when parts of tunnel key is variable
- Example: TOS handling in IPIP, fixed or inherited

IPV6 alignment with IPV4

- IPV6 is actually more similar than one might expect
- Route entries are cached in FIB trie itself
- Problem is ipv6 does not try to share aggressively
- Base FIB routes are always cloned into cached ones
- Should be simple to share routes in ipv6:
    - We have TCP Metrics cache, even for ipv6, already
    - IPV6 routes no longer cache neighbours
- Nexthop exceptions-like mechanism needed
- Then sharing can be added

Cache Routes Aggressively

- When you have an object that represents one destination
- And you frequently do route lookups for it
- Cache routes in that object
- Just like sockets, which means validation before use
- IPVS already does this
- Tunneling is another area ripe for route caching

Thanks

- Eric Dumazet
- Julian Anastasov
- Steffen Klassert
- Herbert Xu
- Tom Herbert
- Jesper Dangaard Brouer