Quick Update on Network Stack Parallelism

Robert N. M. Watson

Security Research Group
Computer Laboratory
University of Cambridge

FreeBSD Developer Summit
15 May 2008

The Big Picture

TCP/IP MPSAFE since FreeBSD 5.3

6.x, 7.x, and 8.x non-trivially lower overhead, improve lock granularity, greater parallelism

Improving highly concurrent workloads

- Many cores, many threads
- BIND9/nsd/memcached (UDP)
- Apache (TCP)
- MySQL/PostgreSQL (Local Sockets, TCP)

UDP: Problems

- BIND9, nsd, memcached
  - One(ish) sockets per process, many threads
  - Tens/hundreds of thousands of simultaneous clients
    - TCP neither feasible nor desirable
- Very high contention on a number of locks
  - pcbinfo for input path
  - inpcb during input and output processing
  - receive socket buffer lock for thread receive
- Excessive overhead from socket buffer routines

UDP Solutions

- Motivates two locking improvements
  - Read-write locks for pcbinfo and inpcb
    - Read lock most common paths
  - Reduction of socket layer overhead
    - sosend_dgram, soreceive_dgram
- UDP itself no longer a source of contention in these workloads – significant performance win
- Still contention sources in receive socket buffer, routing locks, ifnet transmit queue locks

TCP Problems

- Apache, MySQL, PostgreSQL
  - Server: one listen socket, many data sockets spread over many threads or processes
  - Client: many simultaneous unrelated sockets
- Very high contention on several locks
  - pcbinfo for input path and user state transition paths
  - inpcb contention during send/receive
  - socket buffer contention

TCP Solutions

- Improvements in lock granularity, primitives
  - sblock now optimized sx lock
  - Read-write locking for pcbinfo and inpcb
    - Primarily pcbinfo, not inpcb, due to complex per-TCP state
    - Somewhat complicated because cannot tell a priori from header flags if global state transition will be triggered when looking up connection
  - Decompose pcbinfo lock into connection groups
    - Reduce necessarily exclusive access to globals
- inpcb lock, socket buffer send/receive, routing, ifnet transmit queue locks remain significant

Stack Parallelism

- Direct dispatch vs. netisr
  - Exploring multiple netisrs to improve loopback performance, but direct dispatch remains win
- Multiple input queues
  - 10gbps cards/drivers support 4+ input queues, represented as (n) ithreads balanced with RSS
- Multiple output queues
  - Kip has prototype in Perforce, some discussion of ordering issues
  - Need to align stack, driver, card flow logic

The Plan

Convert inpcb/pcbinfo mutex to rwlock in 8.x, 7.x

Move to read locking wherever possible in UDP for pcbinfo and inpcb in 8.x, 7.x

Move to input path read locking of pcbinfo in TCP input path where possible in 7.x, 7.x

Prototype connection group decomposition of pcbinfo in 8.x, MFC may be possible

Retain non-loopback direct dispatch default

Use parallel netisr for loopback, IPSEC, etc