FreeBSD Developer Summit Network Locking Status

June 30, 2004

Robert Watson
rwatson@FreeBSD.org

Introduction

- Background on network stack locking
- Current status
- Performance testing
- Where to go next

Network Stack Locking
Some Credits

- A long list of contributors over many years
  - BSDi, Jonathan Lemon, Jennifer Yang, Jeffrey Hsu, Sam Leffler, Brooks Davis, Max Laier, Julian Elischer, George Neville-Neil, Pawel Dawidek, Maurycy Pawloski-Wieronski, Ruslan Ermilov, Roman Kurakin
  - Kris Kennaway!
- More contributors welcome :-)

Network Stack Locking Background

- SMPng goals:
  - Kernel will execute in parallel on multiple CPUs
  - Adopt a kernel architecture that facilitates explicit synchronization
- Method
  - Introduce additional synchronization primitives
  - Move towards a more threaded architecture (ithreads, etc)
  - Take a first cut at locking; then refine
  - Explore implications on scheduler, threading, etc

Network Stack Locking Components

- Underlying infrastructure
  - Memory allocators, device drivers
- Interface layer
- Socket layer
- Protocol layer
  - IPv4, IPv6, netatalk, ..., UNIX domain sockets
- Socket consumers (NFS, smbfs, AIO, ...)
- Misc: NetGraph, IPFW, et al

Network Stack Locking Activity through 2003Q3

- Ifnet queue
- Mbuf allocator
- Partial routing
- Arp
- Ifaddr references
- Partial inpcb Ipv4
- Netgraph edges
- ithreads
- Some drivers, somewhat

Network Stack Locking Activity through 2003Q4

- Raw IP
- Divert sockets
- IPFW2
- DUMMYNET
- Ethernet bridge
- IP fragment queues
- Routing entries
- FAST_IPSEC
- Parallel netisr
- IP forwarding path complete
- Syncache
- Partial TCP, UDP

Network Stack Locking Activity through 2004Q1

- if_disc, if_faith, if_gif
- ip_ecn
- PF

Network Stack Locking Activity through 2004Q2 (1)

- if_tap, if_tun, if_loop
- Netatalk AARP
- MAC inpcb labels
- ip_encap
- if_clone metadata
- UNIX domain sockets
- Socket buffers
- Sockets
- Fifofs
- NFS server
- Netatalk PCBs
- debug.mpsafenet
- Rtsock uses netisr
- MAC ifnet labels
- Rawcb list

Network Stack Locking Activity through 2004Q2 (2)

- Accept filters
- IGMP, mrouter
- ALTQ
- A number of netgraph nodes
- Portalfs
- More TCP

Network Stack Locking Some Current WIPs

- IPv6
- More netatalk
- More netgraph nodes
- SLIP
- NFSv4 server
- Additional ifnet state
- Additional network device drivers
- General cleanup to socket locking
- Some remaining issues in soreceive
- NFS client
- RPC code
- Netipx
- Netisr/ithread/pcpu exploration

Areas Requiring Owners

- PPP
- SLIP testing
- net*atm
- More netgraph nodes
- More netipx
- KAME IPSEC

Network Stack Locking Performance

- Performance measurement and optimization is now the focus
- Don't have a very good picture of current performance
  - Adhoc benchmarks reveal continuing performance issues on UP relative to 4.x
  - Adhoc benchmarks reveal dramatic performance enhancement on SMP relative to 4.x
- Solution: more hands, netperf cluseter

Network Stack Locking
Network Performance Testbed

- Creating a network performance testbed
  - Sentex donated rack space, connectivity, management system
  - FreeBSD Systems, FreeBSD Foundation sponsored hardware
  - Some of my own also :-)
- Support netperf research and development activities
  - “Check out” model
  - Pretty reserved for the next few months

Network Stack Locking
More Testbed Thoughts

- Permit numerous variables to be explored
  - Network topology variations possible due to full connectivity between some nodes
  - Pxeboot, remote power, remote serial console
  - Operating systems (FreeBSD, DFBSD, Linux, NetBSD – Windows would be nice but unlikely)
  - Operating system versions (5.*, 4.x, ...)

Network Server Locking Benchmarks

- “Raw” network benchmarks
  - Host-Host, Host-Bridge-Host, Host-Router-Host
- Application benchmarks
  - Local MySQL (Host)
  - Distributed MySQL (Client-Server)
  - HTTP (Client-Server)
  - ...
- Generate, publish historic performance information to track changes

Network Stack Locking Variables

- UP/SMP/HTT
- Mpsafe (or not)
- Scheduler Choices
  - 4BSD, ULE
- Synchronization Optimization
  - ADAPTIVE_MUTEXES, wakeup, hashes, ...
- NETISR model
  - Direct ithread, one netisr, multiple netisr
  - Coallescing models to amortize per-packet cost

Network Stack Locking
Where to go next?

- Continue the locking work
  - Many of the biggest hurdles are overcome
  - Will be refinement and bug fixes for a while
  - As of next week, sufficient locking in CVS to run UNIX domain sockets, IPv4 without Giant
- Performance optimization
  - Measurement
  - Synchronization, scheduling improvement
  - Locking improvement