Update on the SMP Network Stack

Robert N. M. Watson
FreeBSD Foundation - University of Cambridge

17 November 2008
FreeBSD Developer Summit - MeetBSD

Agenda

- Reminder: the SMP network stack project?
- Some recent accomplishments
- Some works-in-progress
- Some things that need to happen
- Modifying the socket API

SMP Network Stack

- Apply SMPng principles to network stack
- Allow and encourage parallelism
- Millions of LoC, hundreds of subsystems
- Able to disable Giant in 5.3; default in 5.4
- Non-driver use of Giant eliminated by 7.0

The last few years...

- Performance optimization and cleanup
- From mutexes to rwlocks/rmlocks on inpcbs and inpcbinfos: fully parallel UDP/ UNIX domain sockets
- Move to direct dispatch
- Datagram socket optimizations
- Multi-queue input

Works-in-progress

- Further read-locking of global structures
- Multi-queue output
- More formal notions of affinity
- Hashing and replication of global structures
- New socket semantics to support load balancing for UDP receive and accept

Works-in-progress (2)

- Mbuf + cluster rethink
- Significant scheduler improvements
- Route flow cache

Things that want to be done

- Improve routing table scalability
- Rethink ifnet dispatch abstractions
- NUMA awareness in the VM system
- Revisit cache miss analysis of stack
- Revisit path-centric lock analysis of stack

The socket API and parallelism

- Some OS services imply synchronization
- Socket queues represent an ordering of packets/connections as required by APIs
- API guarantees can be stronger than application requirements
- So: cut corners on APIs, or change APIs?

Case study: UDP recv()

- UDP socket buffer maintains wire ordering of datagrams matched by binding
- Services often require flow ordering (IP/ port tuple) maintained by modern routers
- DNS/memcached support weaker orders
- Can we improve parallelism by weakening guarantees of socket buffer?

Proposal: subset binding socket option

- UDP sockets can have same binding today
- New socket option will allow colliding sockets to request different traffic subsets
- Application declares total number and instance using socket option (Borg 3 of 8)
- Kernel will maintain at least 4-tuple ordering, but no specific mapping guarantee

Related concepts

- Mapping to specific socket could simply be a hash on the tuple mod socket count
- Mapping could also be based on effective flow affinity to a specific queue or CPU
- Similar concerns exist with TCP accept: avoid contention on specific listen socket, and return sockets with locality to worker

Conclusion

- Focus remains on:
  - Locking and scheduling infrastructure
  - Locking granularity and contention
  - Improving opportunities for parallelism
- Going forward: increasingly optimal behavior for current semantics, how can we change semantics to improve performance?