Squid: Networking Stuff

Adrian Chadd <adrian@squid-cache.org>

Disclaimer!

- This is very Squid-2 centric
- Most should be applicable to Squid-3 if the network API direction is finally chosen!
- As with the storage talk, the “science” is missing - I’ll fill that in when I’ve got access to the servers w/ data!

Overview

- Squid's network core isn't all that bad
- How it's used is pretty bad
- Its use of C STDIO routines is shocking
- It's not suitable for efficient threading in its current state
- It's not suitable for high-performance network IO (windows completion IO, for example) in its current state

Overview

- Core - just an event loop
- Users register for FD read/write interest
- Comm layer on top of that implements basic socket operations
- Squid-2: notably, comm_write() is implemented, but all reading is done via registration and subsequent read()
- SSL is still a bit of a hack

Performance

- Single-thread performance isn't all that bad
  - Cacheboy - separated out the core routines from Squid itself
  - Allowed for benchmarking of the network comm code
  - Results are in line with other network applications
  - - ie, can reach 1Gbps throughput for large transactions; or 10,000 small transactions/sec

Performance (ctd)

- Where we fall short:
  - per-FD description - lots of little memcpy()'s that are -only- used for the per-FD statistics
  - per-FD IP address string - inet_ntoa() uses STDIO functions.. :/
  - Lots of memory allocation going on which just isn't needed

Network IO Sizes

- For normal WAN and small object sizes, never really reach over a couple kbyte per read
- For large/streaming objects - may be a different story
- Am lacking data to make an adequate judgement on this

Network IO calls

- Are we doing network IO in an efficient manner?
- No!
  - Not using scatter/gather IO
  - Not doing zero-copy IO where possible
  - .. and Squid itself does a lot of data copying which isn't needed!

Network IO calls

- Supporting a writev() style API would help with writing HTTP requests/replies
- Supporting a readv() style API - not so obviously useful at the moment?
- This could be done today with minimal changes to the codebase (ie, an addition, not a “change”.)

Network IO calls

- A lot of time is spent in kernel-space copying network data to/from user-space
- Its less of an issue on current hardware, but still there
- Can Squid support it?

Network IO calls

- In short - ‘No’.
- cbdata and the lack of explicit IO cancellation in the comm API make it difficult
- Comm layer and comm users would need to be redesigned to handle cancellations and failed cancellations!
- “close handlers” - similar issues!

Network - SSL

- SSL is implemented by some fudging
- FD_READ_METHOD / FD_WRITE_METHOD
- Comm layer hides the "SSL" IO event notifications somewhat
  - Some reads need writes, and vice versa
- Solution: a comm_read() / comm_write() only API, hiding the event notifications!

Network - threading

- Current comm code is “heavy”
  - lots of per-FD state, lots of state in the fd_table[]
- Cleanly threading this will be ugly
- Solutions?

Network - threading

- Windows IOCP style - worker thread pool, request callbacks can occur in any thread
- UNIX thread style - multiple IO queues, request callbacks occur in callee thread
- Varnish style - blocking IO everywhere; spawn threads to handle concurrent IO load
- Multiple processes style (ie, ignore threading entirely!)

Network - threading

- I'm leaning to the UNIX thread style used in things like libevent, memcached, etc
- Each "app" thread has its own local IO thread, like a "squid process" today
- One thread handles incoming requests and punts them to other threads
- Other threads handle non-IO work queues

Network - threading

- Figure out what needs to handle concurrency and what doesn't
- Treat the rest of Squid at the moment as "one thread"
- Implement a generic inter-thread work queue
  - submit, retrieve and cancel work
  - inter-thread communication

Network - threading

- The "core" is mostly easy to thread
  - (ie, everything in Cacheboy that isn't in src/)
- .. except MemPools, which need to be turfed
- cbdata semantics make inter-thread communication difficult for existing callbacks
  - ie, the "can immediately invalidate at any time" makes threading impossible

Network - Windows

- Efficient windows support is going to be difficult!

1. Get threading support working
2. Get overlapped IO support working
3. Turn “fd” into an opaque type rather than an integer
4. Write some glue to translate between the IOCP threading model and the Squid threading model!

Questions?