Tips on High Performance Server Programming

Joshua Zhu

June 9, 2009

Agenda

- **Fundamentals**
- Best practices and common tricks
- Design issues
- Tools and resources

Key Rules

- Do NOT block
- Avoid excessive system calls
- Cache/preprocess as much as possible
- Use efficient algorithms and data structures
- Threads are usually a bad idea
- Separate the I/O code from the business-logic code
- Keep the most heavily used data near the CPU
- Tune against the bottleneck

I/O Models

- Blocking
- Non-blocking
- I/O multiplexing
- Signal-driven I/O
- Asynchronous I/O

I/O Multiplexing

- Select
- Poll
- /dev/poll
- Epoll/kqueue
  - Level triggered
  - Edge triggered

I/O Strategies

- 1 thread, non-blocking, level-triggered
- 1 thread, non-blocking, edge-triggered
- 1 thread, AIO
- 1 thread per client
- Build the server code into the kernel

Let's Face the C10K Problem

- 10000 connections
  - CPU/memory usage per connection
- 10000 hits/sec
  - 10 us per hit
    - Instruction (ns)
    - System call (us)

Agenda

- Fundamentals
- **Best practices and common tricks**
- Design issues
- Tools and resources

The Event-driven Model

```
while (true) {
        for t in run_tasks:
                t.handler();

        update_time(&now);
        timeout = ETERNITY;

        for t in wait_tasks: /* sorted already */
                if (t.time <= now) {
                        t.timeout_handler();
                } else {
                        timeout = t.time - now;
                        break;
                }

        nevents = poll_function(events, timeout);
        for i in nevents:
                task t;

                if (events[i].type == READ) {
                        t.handler = read_handler;
                } else (events[i].type == WRITE) {
                        t.handler = write_handler;
                }

                run_tasks_add(t);
}
```

Scheduler

- Task scheduling
  - Run queue
  - Wait queue
- Timers
  - Time jumps
  - The next timeout
    - Pass to select()/poll()/epoll_wait()...
  - Data structures
    - Red-black tree
    - Min-heap
    - Timer wheel

Unify Multiple Event Sources

- I/O events
- Timer events
- Signals/threads
  - Use a pipe or socketpair
    - Register the read end in the event loop
    - Notify the event loop by writing to the write end

Buffer Management

- Fixed size or not
- R/W pointers operation
  - Producer/consumer
- Circular buffer
- Single buffering
- Buffer chain

Memory

- Memory fragment
  - Memory pool
    - Fixed size
      - MRU
    - Non-fixed size
  - Slab allocator
- Memory operations are expensive
  - Allocation/free
  - Data copies
- Pre-allocation/static-allocation
  - Array

Strings

- Gotchas
  - Strlen()
    - Sizeof() - 1
  - Strncpy()
    - Strlcpy()
  - Vsnprintf()
  - …
- Algorithms
  - KMP/BM
  - AC/WM
  - …

Caching

- Time
  - Reduce the number of gettimeofday() calls
  - Time resolution
    - ms or sec?
- Content
  - Files
    - In-memory buffering
  - Database
    - Memcached
- Preprocessing

Accept() Strategies

- Accept-limit
  - Multi-accept
  - Accept-mutex
- The thundering herd problem
  - Polling functions
  - Kernels
  - The amount of processes/threads

Concurrency

- Threads
  - Pros
    - Utilize all the CPUs
    - True CPU concurrency
  - Cons
    - Context switches
    - Locks hurt performance
      - Deadlocks
      - Starvation
      - Race conditions
    - Error prone and hard to debug
    - Hard limits of operating systems

Concurrency (cont'd)

- Categorize your service
  - CPU-bound
  - I/O-bound
- Models
  - Thread pool
  - Process pool
  - SEDA
  - Master/workers
- CPU affinity

Advanced I/O Functions

- Gather read, scatter write
  - Readv/writev
- Sendfile
- Mmap
- Splice and tee

Take Advantage of TCP/IP Options

- TCP/IP options
  - SO_REUSEADDR
  - SO_RCVBUF/SO_SNDBUF
  - TCP_CORK
  - TCP_NODELAY
  - TCP_DEFER_ACCEPT
  - …

Misc.

- Byte order
  - Little-endian
  - Big-endian
- Max open file number
  - Ulimit/setrlimit

Agenda

- Fundamentals
- Best practices and common tricks
- **Design issues**
- Tools and resources

Protocol

- Readability
  - Binary
  - Textual
- Compatibility
- Security
  - Heart beat
  - Invalid data
- The state machine (FSM)
  - Granularity
  - Traceability

Security

- Timeouts
- Bad data
- Buffer overflow
- DOS attack
- Encryption
- Privilege
- Working directory

Flexibility

- Modularize
  - Modules
  - Plugins
- Scriptable
  - Lua
- Hot code update
- HA

Introspection

- Statistics
  - SNMP
- Debugable
- Tunable
- Logs

Patterns

- Reactor
- Proactor
- Half-Sync/Half-Async
- Leader/Followers
- …

Choose a Suitable Language

- C
- C++
- Java
- Erlang
- …

The Devil Hides in the Details

- Find out the shortest plank in the bucket
  - Profile/benchmark
- Take care of every line of your code!
  - Return values
  - Error code
    - EINTR
    - SIGPIPE
    - …
- Tune your operating system if necessary
  - /proc
- Make sure hardware is not the bottleneck
  - Network cards
  - Disk
  - …

Agenda

- Fundamentals
- Best practices and common tricks
- Design issues
- **Tools and resources**

Sharpen Your Tools

- Ping
- Traceroute
- Netstat
- Lsof
- Strace
- Nc
- Wireshark
- Gprof
- Valgrind
- Vmstat/iostat/dstat
- Dot/gnuplot

Understand TCP/IP Well

- Three-way handshake
- Connection tear-down
- TCP state transition
  - TIME_WAIT
  - CLOSE_WAIT
  - …

Learn By Reading Code

- Frameworks
  - ACE
  - Libevent/libev
  - Boost::asio
  - …
- Servers
  - HAProxy
  - Nginx
  - Lighttpd
  - Memcached
  - MySQL Proxy
  - …

Recommended Books

- *Advanced Programming in the UNIX Environment*
- *UNIX Network Programming*
- *TCP/IP Illustrated*
- *Effective TCP/IP Programming*
- *Pattern-Oriented Software Architecture*
- *C++ Network Programming*
- *Programming With POSIX Threads*
- *Introduction to Algorithms*

Thank You!

Visit www.zhuzhaoyuan.com for more info