Node v0.5 Roadmap

ryan@joyent.com

May 5, 2011

The Community is Growing

- Mailing list: 4000 people (For comparison: Ruby on Rails 22000, Tornado: 1300, EventMachine 300)
- IRC: 600 people during peek hours
- Many meetups
- Documentation translations: Japanese, Russian, Spanish, Persian
- Three books in progress
- 1600 modules posted on the NPM registry
- Spamming Hacker News daily.

May 2009, initial release

August 2010, v0.2

February 2011, v0.4

Development will continue in unstable v0.5 release. Stable v0.6 release in about 2 to 3 months.

The main goal of v0.5 is reworking the plumbing for

Windows compatibility

Why is porting to Windows a priority?

The goal of Node, as with any programming platform, is total world domination.

- Windows accounts for 33% of servers
- Windows accounts for 87% of web browsers

There is no question. Node **must** run on Windows.

http://en.wikipedia.org/wiki/Usage_share_of_operating_systems

Today Node runs on Windows via Cygwin.

This is an unacceptable port. Cygwin is slow, old, buggy. Users hate it.

Is it even possible to do decent servers in Windows?

Yes. But **not** by naive direct port.

Very few web servers has achieved the goal of running properly in both Windows and Unix systems. (Apache - what else?)

Good server software in Unix means:

- Set sockets **non-blocking**.
- Wait for file descriptor state changes with a constant-time I/O multiplexer like **kqueue** or **epoll**.

Windows has `select()` and non-blocking sockets but `select()` is not constant-time. It becomes **noticeably slow** with more than 100 sockets.

Windows has `WSAEventSelect()` but it must be used with `WSAWaitForMultipleEvents()` which is **limited to 64 sockets per thread**.

AWESOME

It is possible to write efficient high-concurrency low-latency servers on Windows with

I/O Completion Ports

Unfortunately IOCP is not about file descriptor readiness like on Unix.

Unix: wait for file descriptor changes, do non-blocking I/O

Windows: call asynchronous functions on sockets, be notified of completion.

IOCP supports Sockets, Pipes, and **Regular Files**.

That is, Windows has true async kernel file I/O.

(In Unix we have to fake it with a userspace thread pool.)

Node’s API is already quite similar to IOCP:

```
socket.write('hello', function() {
  console.log('finished writing hello');
});
```

Asynchronous function calls.

We may be able to refactor the plumbing with minimal API disturbance.

Previous Work

What's required is an abstraction layer over the Unix and Windows styles of I/O. There have been several attempts at this:

- **libev** – beautiful library but does not support IOCP.
- **libevent** – recently supports IOCP but reportedly slow. The library caries a lot of baggage. Doesn't support regular files.

  `http://thread.gmane.org/gmane.comp.lib.libevent.user/1557`
- **Boost ASIO** – works correctly but is unacceptably massive (300 files and 12000 semicolons). Doesn't support regular files.

None are acceptable.

The problem isn't *that* hard–especially with the help of libev.

Solution: build our own layer.

`http://github.com/joyent/liboio`

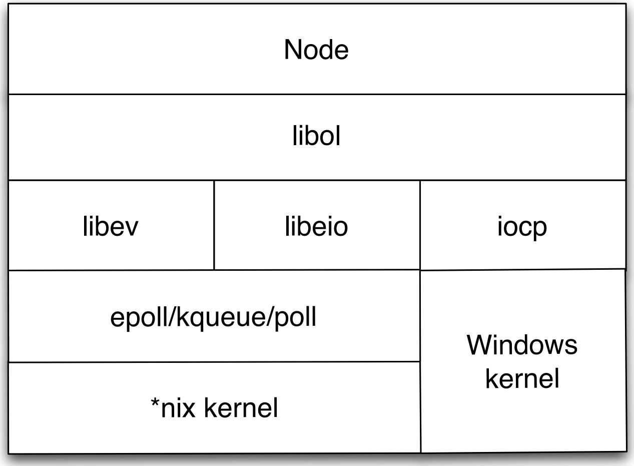
Currently work on liboio is being done in a separate repository so that we are forced to write all of the tests in C.

TCP socket/server is working cross platform now.

Will land liboio in the master branch very soon (probably next week).

Bert Belder of Cloudkick/RackSpace is our resident Windows expert – building the Windows side of liboio.

I am building the Unix side using libev and libeio.


Node
libol
libev
libeio
iocp
epoll/kqueue/poll
Windows kernel
*nix kernel

Avoiding `#ifdef` Hell

- Push all platform incompatibilies into the liboio layer (pipe, regular file I/O, timers, etc.) and preform testing.
- The V8 binding layer and pure JavaScript parts of Node will be platform independent.

Easy Install

Node compiles its internal JavaScript code into the executable.

Node statically links the libraries it uses (modulo OpenSSL)

A Node installation can be **a single executable**.

This plays well on Windows: we can provide a nice user experience on that by simply distributing **`node.exe`**. One file to get going.

NPM and Addons

NPM currently requires users to have a compiler toolchain available to install addons (AKA extension or native modules).

On Windows it is uncommon for users (even programmers) to have a complete compiler toolchain.

NPM is being retrofitted with the ability for authors to upload compiled versions their addons.

Long Stack Traces

Binding to liboio will allow for a single entry hook into all callbacks into Node.

Getting this hook is the main obstical to providing the “long stack trace” feature.

See `http://nodejs.org/illuminati0.pdf`

If we can't get at least gigabit throughput from a TCP pair over loopback **on every OS**, we've failed.

If we can't have 10,000 idle sockets connected to a server and use less than 100mb RSS **on every OS**, we've failed.

Questions...?

`http://nodejs.org/`

`ryan@joyent.com`