Node.js

ryan@joyent.com

April 9, 2011

These slides are online:

`http://nodejs.org/codeconf.pdf`

Joyent

Somewhat popular

- Mailing list: 4000 people (For comparison: Ruby on Rails: 22000, Tornado: 1300, EventMachine 300)
- IRC: 500 people daily
- Half-dozen conferences in planning.
- Documentation translations: Japanese, Russian, Spanish, Persian
- Three books in progress
- 1600 modules posted on the NPM registry (see `https://github.com/joyent/node/wiki/modules`)
- Spamming Hacker News daily.

Joyent

History:

- Initial release: May 2009
- v0.2: August 2010
- v0.4: February 2011

Development will continue in unstable v0.5 releases. Stable v0.6 release in 2-3 months.

Joyent

Roadmap for v0.6:

- Good Windows support (more about this)
- Remove internal DNS resolver (c-ares), use system resolver instead.
- Unified event handling: allows hooks for tracing the history of errors.

Windows support

Important to provide a good experience.

Non-ignorable amount of Windows servers (45% physical, 15% emulated at a large hosting company)

Non-ignorable amount of Windows developers.

Windows support

Currently Node runs only on the Cygwin libraries–a Linux emulation layer.

Cygwin is slow, old, buggy. Users hate it.

Windows support

Is it even possible to decent network programs in Windows?

Not by naive direct port from Unix-style.

Windows support

Decent network programing on Unix means:

- Set sockets non-blocking
- Wait for file descriptor state changes with an O(1) I/O multiplexer like kqueue or epoll.

Windows has `select()` and `O_NONBLOCK` for waiting for socket readiness but `select()` becomes slow when polling many file descriptors (like, more than 100).

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

Windows support

Joyent

Sad face.

Windows support

However, it *is* possible to write efficient high-concurrency servers on Windows with I/O Completion Ports (IOCP).

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

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

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

Windows support

Node's API is already quite similar to IOCP.

For example:

```
1   socket.write('hello world', function() {
2     console.log('finished writing');
3   });
```

Windows support

Since possible Node should provide the best possible backend on Windows.

We will take this opportunity to geek-out and write a proper abstraction library over IOCP and Unix-style I/O multiplexers:

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

`http://tinyclouds.org/iocp-links.html`

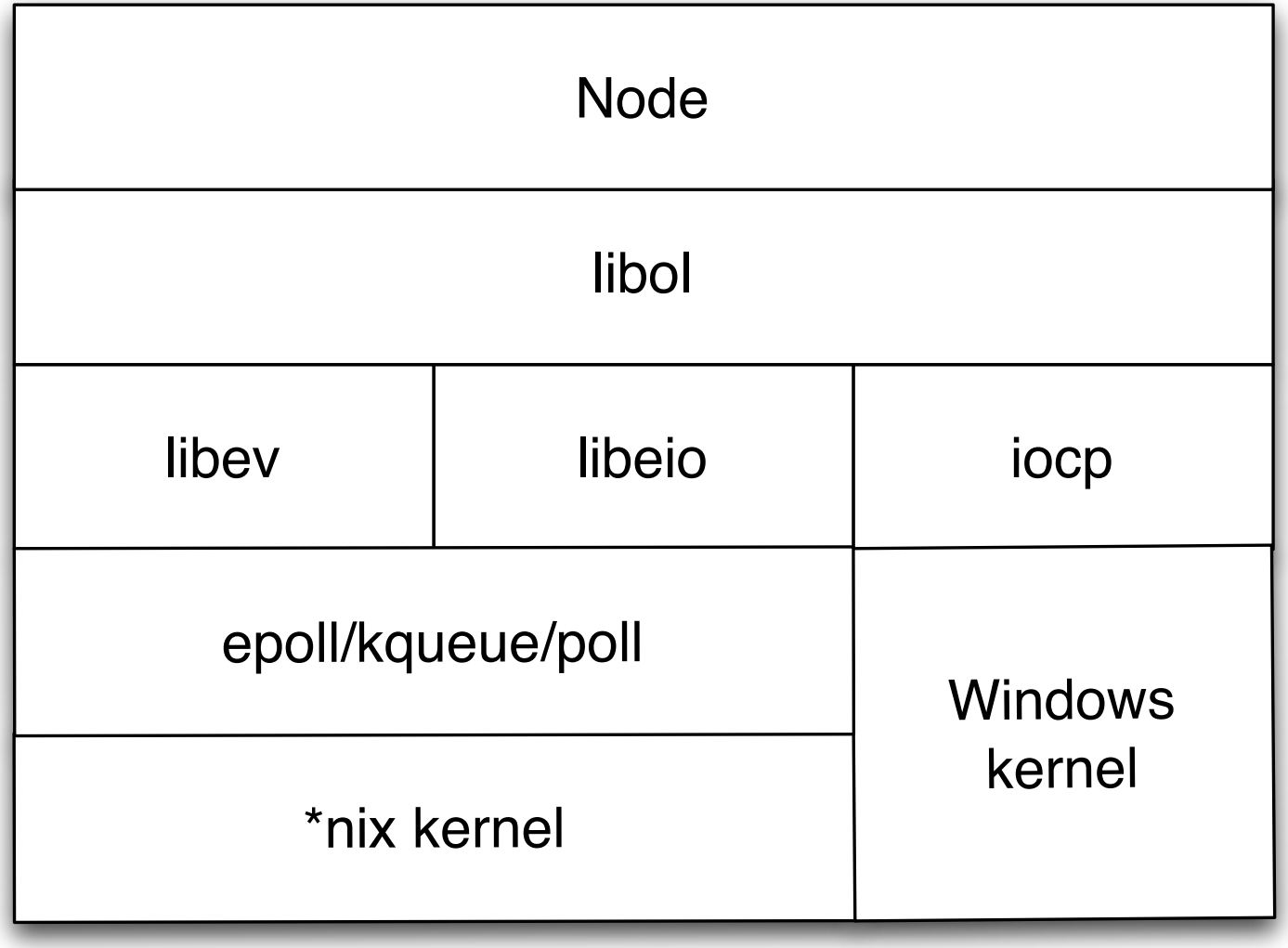
Windows support

Some other libraries address this need:

- **libev**–beautiful library but does not support IOCP.
- **libevent**–recently supports IOCP in version 2 but reportedly slow, also comes with a lot of baggage.
- **Boost ASIO**–works correctly but IMO unacceptably massive (300 files, 12000 semicolons). Problem is not that hard. Doesn't support regular files.

Windows support

Joyent



Windows support


```
1  typedef struct iovec ol_buf;
2
3  int ol_tcp_handle_init(ol_handle *handle,
4          ol_close_cb close_cb, void* data);
5
6  int ol_read(ol_handle* h, ol_req* req,
7          ol_buf* bufs, int bufcnt);
8
9  int ol_write(ol_handle* h, ol_req* req,
10         ol_buf* bufs, int bufcnt);
11
12 int ol_close(ol_handle* handle);
```

Windows support

IOCP works with **sockets** as well as **regular files**.

That is, the holy grail of **true asynchronous kernel file I/O is possible on Windows** unlike Linux.

Windows support

Node compiles its internal JavaScript code into the executable.

Node statically links the libraries it uses (modulo OpenSSL)

That is, a Node installation is literally 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.

Windows support

NPM, the Node package manager, 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 thus being retrofitted with the ability for authors to upload compiled versions their addons.

Windows support

The refactor to support Windows will allow for unified ”event emit” entry points.

This new hook will be used to provide ”long stack traces” that span events.

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

Windows support

Bert Belder is the man behind enemy lines doing the Windows work. We need more help.

If you're a Windows programmer: please help!

If you're a Unix programmer: you too can be useful building out the backend for libol.

Windows support

Unsolved problem: how to do a pre-fork IOCP server?

Joyent

Other features possibly in v0.6:

- Crypto module to match stream APIs (that is, use **`write()`** instead of **`update()`**)
- Consider reintroducing a promise object for async req/res operations. Probably follow jQuery's deferred API.
- Built in GZIP module
- Attempt to remove **`node-waf`** with a Node-based build system. (Used for building addons.)

Future

Node is a single thread, single process.

It's a good thread: can handle a lot of I/O–it's nearly optimal for a single core.

Ultimately limited to a single core.

A Node process serves many use-cases alone but is not suitable to massive distributed Internet applications.

Future

Scaling out beyond a single process requires some form

- load balancing
- shared state
- process management
- If dealing with multiple machines: some form of deployment
- Complex apps: some form of IPC

There are many existing solutions for all of these problems which can be applied to Node instances today.

Future

For example:

- some form of load balancing: **ZXTM**
- some form of shared state: **Redis**
- some form of process management: **Upstart**
- some method of deployment: **Capistrano**
- some form of IPC between Node instances: **AMQP with RabbitMQ**

Future

Some users are unhappy having to choose these options themselves.

They want the runtime to provide solutions.

Questions?

ryan@joyent.com