apigee

May 2014

Introduction to Trireme

Greg Brail

What is Node.js, Really?

Node.js

- Single-threaded event engine
  - Non-blocking TCP I/O
  - Non-blocking UDP datagrams
  - Timers
  - Non-Blocking File I/O
- “Buffer” object
- Module loading system
- Utility modules
- Third-party components
  - V8 JavaScript engine
  - OpenSSL
  - ZLib

What is Trireme?

Node.js

- Single-threaded event engine
  - Non-blocking TCP I/O
  - Non-blocking UDP datagrams
  - Timers
  - Non-Blocking File I/O
- “Buffer” object
- Module loading system
- Utility modules
- Third-party components
  - V8 JavaScript engine
  - OpenSSL
  - ZLib

Trireme

- Single-threaded event engine
  - Non-blocking TCP I/O
  - Non-blocking UDP datagrams
  - Timers
  - Non-Blocking File I/O
- “Buffer” object
- Module loading system
- Utility modules
- Third-party components
  - Rhino JavaScript engine
  - Java SE
  - Bouncy Castle (crypto, optional)

Trireme at a Glance

- Runs Node.js code (10.x) inside the Java VM
  - Supports Java 6 and up
- Supports *most* of the Node API and *most* third-party modules
- Designed for embeddability
    - Run many Node.js apps inside the same Java VM
    - “HTTP Adapter” lets it run inside existing containers
    - “Sandbox” restricts file and network I/O access
- Deliberately few dependencies
  - Java 6, Slf4J
  - Rhino for JavaScript
  - BouncyCastle is optional, required for some crypto and TLS

Why On Earth Did You Do This?

We wanted to add Node.js capabilities into our existing product

- Highly mission-critical, in production, and built in Java

We wanted to efficiently use existing Java code from Node.js

We didn't want to assemble a whole Node.js "PaaS" as well

Trireme requirements:

- Support most Node.js features and modules
- Run inside an existing Java VM container
- Support many (hundreds) of scripts in a single VM
- Isolate different scripts from each other and from the machine

Trireme Architecture

One thread per Node.js application

- Async I/O handled via NIO within that thread

Additional thread pool for blocking operations

- File I/O (especially on Java 6)
- DNS lookups

Replace native code from Node.js with Java alternatives

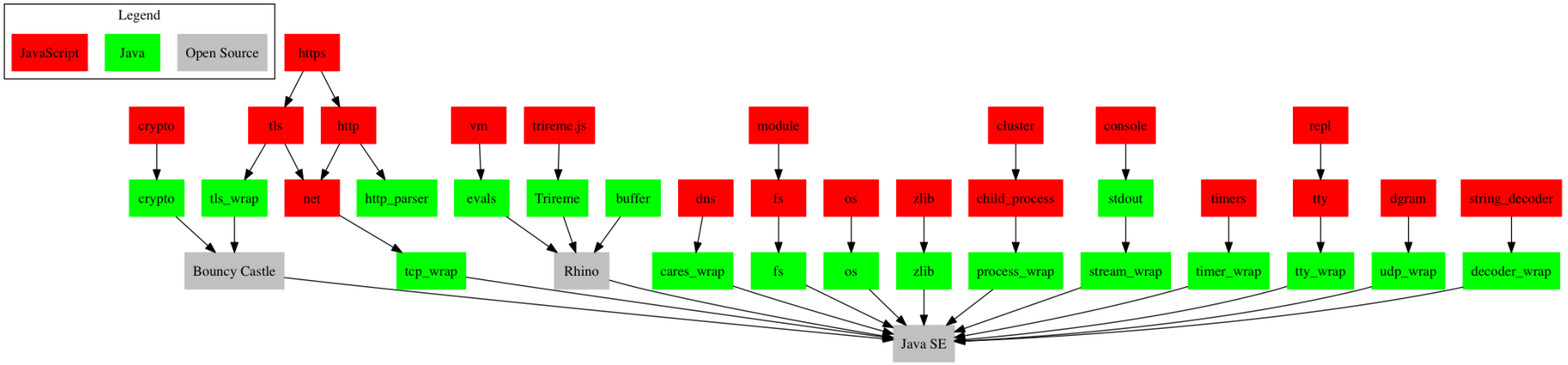
- Internal modules such as “tcp_wrap,” etc.

Implement a few popular native modules with Java code

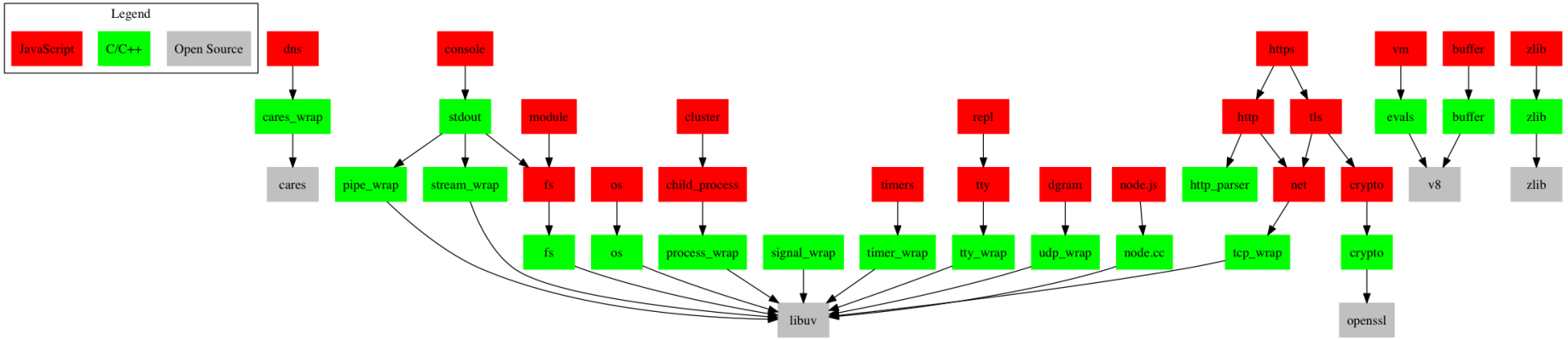
- “iconv,” “node_xslt”, eventually others

Architecture

Trireme



Node.js



Why Use Trireme?

Need to embed inside an existing Java container

- Web app servers, Hadoop

Need to access existing Java technology in the same VM

- Hadoop, JMS, XML processing, Cassandra

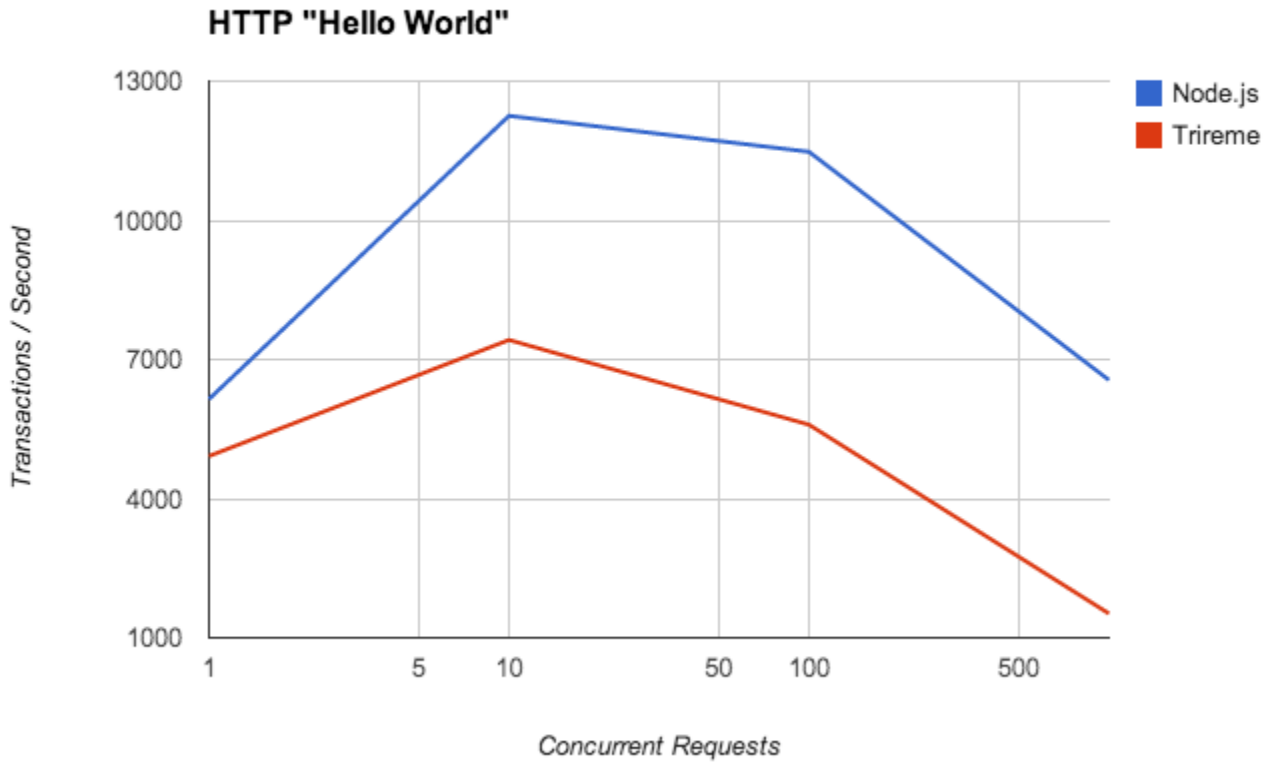
Want to distribute Node.js capabilities without binary code

- The Java VM runs pretty much everywhere

Performance

- Trireme's JavaScript Engine is much slower than V8
  - Something like 50x slower in the "V8" benchmarks
- So how does that affect Trireme?
  - It is slower than Node.js but less so for real apps:
  - About 50-60 percent the performance as an HTTP server
  - Better for use cases that are network or filesystem intensive
  - Worse if there is very complex JavaScript logic
- Can we do better with Rhino?
  - Need to look at optimizing the core event loop more
  - Didn't do any JVM tuning at all

Hello, World



Standard “Hello, World!” HTTP server

- Client and server on Amazon EC2 “c3.xlarge” instances (4-core 2.8 Ghz processors)
- Benchmark client is “apib” (https://github.com/apigee/apib)
- Java 7, default options, no tuning

HTTP Orchestration

For each HTTP request:

- Request five small JSON files from an nginx server in parallel
- Combine them into an array
- Sign it using RSA-SHA256
- Return the lot as JSON

So, 200 tps =

1000 server requests / sec

Myths

"Node was designed for V8"

- I would love to know the truth of that

"Rhino is too slow"

- Trireme is fast enough for many tasks

"Node is fast because it's single-threaded," or,

"Java is slow because it's multi-threaded"

- Java is capable of asynchronous network I/O just like Node.js
- Sometimes it's nice to use multiple cores

Limitations and Challenges

- Compatibility
  - Works with most things
  - Problems with native code (can't load C code)
  - Provide Java equivalents for a few (iconv, node_xslt)
- Performance
  - Yes, Rhino is much slower than V8
- What risks could the future hold?
  - Future changes to the JavaScript language
  - Further use of V8-specific features in the Node.js codebase
  - More big rewrites of the Node.js code base (hope not!)

Next Steps

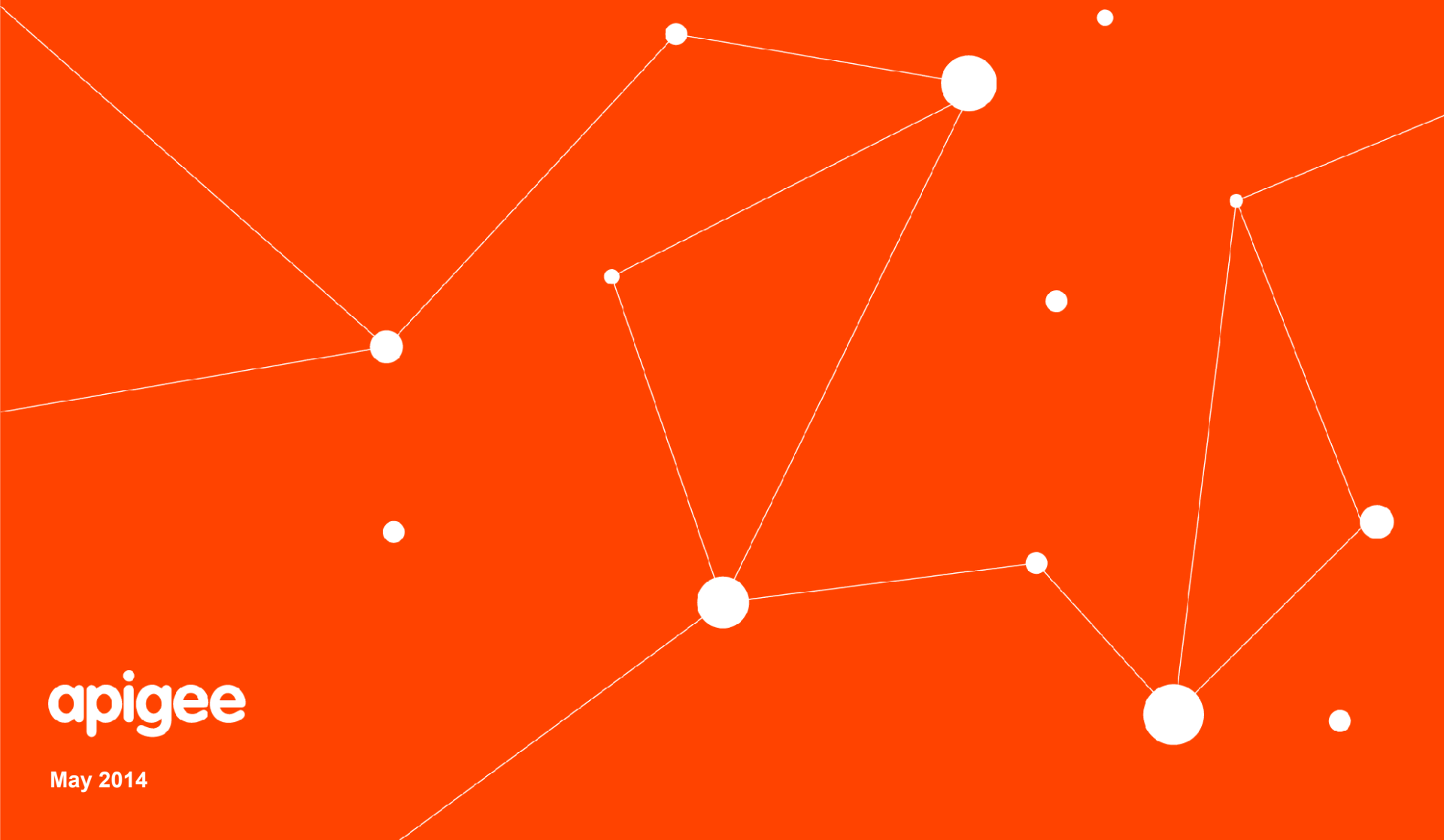
- Continue to refine and complete Trireme
  - Work to do in:
    - cluster, crypto, dns, repl
  - Performance work
  - Multiple Node.js versions in the same VM
    - Architecture in place
    - Support 11.x or 12.x?

Next Next Steps

Starting the successor to Trireme:

- Tentatively called “rowboat”
- Java 8 only
- Based on the Nashorn JavaScript Engine
  - Much faster in JavaScript-only benchmarks
  - Active development team
  - Spec compliance
- Make it simpler:
  - Write more JavaScript that calls Java directly
  - Will probably depend on the Java Security Manager

Thank you



apigee

May 2014