Workload Characterization of Server-Side JavaScript

Takeshi Ogasawara
IBM Research – Tokyo

Motivation

- Modern JavaScript compilers greatly improved the performance.
  - Dynamic type-feedback based optimization
- Node.js is one of the most successful server-side JavaScript frameworks.
- Is the JavaScript compiler optimizing Node.js applications?
- Are there new opportunities for performance improvement for Node.js programs?

Our Findings

The characteristics that are unique to Node.js:

- High CPU time on the V8 C++ library
  - Only 3.2% for the benchmarks on Web browsers
- Potential opportunities for performance improvement in Node.js
  - Two major reasons for high CPU time on the V8 C++ library:
    - C++ implementation of Node.js
    - Library call of the V8 runtime from JavaScript code

Node.js

- An event-driven, non-blocking I/O model based framework
  - Node.js API registers a callback to a specific event type.
  - The callback is executed when the event is detected.

Node.js Implementation

Experimental Environment

Perf performance analysis tool for Linux

SUT
Node.js (ver 0.11.9)

Benchmark Driver
WebFramework Acme Air

Database
mongo mysql

4-core 3.5-GHz Xeon E3-1270v2
32-GB mem
Fedora 20

var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200);
  res.end('Hello World
¥
');
}).listen(1337, '127.0.0.1');
Analyze how the V8 C++ library is called

- When the CPU time is spent in the V8 C++ library code, the native code was called from JavaScript code.
- We would like to understand why the JavaScript code calls the native code.

Native code group – We grouped the native code by functionality.

Node C++ Group

Plainext Workload
- C++ code that implements a JavaScript function:
  ```
  void StreamWrap::Writev(...) {
      writeSize = Number::New(isolate, size);
      req->Set("bytes", writeSize);
  }
  ```
- Corresponding JavaScript Code:
  ```
  req.bytes = writeSize;
  ```

V8 C++ Group (for handling objects)

Plainext Workload

- Corresponding JavaScript Code:
  ```
  Node JavaScript _writeGeneric()
  ```

CPU Time Distribution among Native Code Groups

- V8 C++ Lib to handle objects

Conclusions and Future Work

- Node.js applications spend 48% of the total CPU time in calling the native code.
- A Node.js unique reason is that a part of Node.js functions are implemented by C++.
- V8 compiler does not target Node C++ code.
- There are opportunities for performance improvement in the Node.js code.
- Also there are opportunities for performance improvement in the V8 compiler since the compiled code of the Node JavaScript code still calls generic V8 API functions.