Why you should use Node.JS for CPU-bound tasks

Neil Kandalgaonkar (“Neil K”)
Node Brigade, Dec 17 2013
Vancouver
http://neilk.net/
Programming JS & web servers since the 90s

• ActiveState here in Vancouver

• The California adventure: Google, Flickr, Wikipedia

• now, solo projects, often with Node
  (seeking cofounders?)
WHY?!
CPU-bound JS

Data-Driven Documents

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3's emphasis on web standards gives you the full capabilities of
CPU-bound JS

emscripten
But not Node.js?

- “Node.js is cancer” - fibonacci
- The event loop (libuv)
- By default, switches tasks only on I/O events
LetterPwn

● A Letterpress solver I wrote in Node.js
● Deliberately “doing it wrong”
● Do I even like Node?
LetterPwn demo

![LetterPwn Demo Image](image-url)
Why it stresses Node

• Very large in-memory database
• just a giant JS data structure
• Free Heroku hosting... 512 MB limit
Why it stresses Node

- Very CPU intensive - lots of math and bitwise operations.

...and no I/O events while processing
The problem

Lots of data to search through

Lots of stuff to sort

But we want to be responsive!
How bad?

most common letters in English

Vocabulary: sesquipedalian

eleutheromanias

eleutheromania

Server is monopolized for nine seconds

telomerisation

271,377 words searched for this vocabulary.
57,025 words found for this board. 1,024,295 possible moves ranked in 8.836 seconds, at 115,922 moves/second. hide stats

Friday, December 20, 13
Standard single-tasking Node.js server

Internets

difficult task

easy task

time

difficult task done on time
easy task delayed :(

Server

Node.js

BLOCKED! :(
Co-operation
Cooperative multi-tasking

- A 1980s model of concurrency
- manually pass control to next task
Cooperative multi-tasking

- process.nextTick
- setImmediate
Cooperative multi-tasking

- `process.nextTick` (do it before anything else)
- `setImmediate` (do it after next I/O)
Single-tasking Node.js server, multitasking via `setImmediate()`

Internets

- difficult task
- easy task

Server

Node.js

- `setImmediate`
- `setImmediate`
- `setImmediate`
- `setImmediate`
- `setImmediate`

time

- easy task done, still delayed :(
- difficult task done

Friday, December 20, 13
Not everything can cooperate

• `Array.sort()` a million items?
Getting more help
Multiple processes?

- `child_process.fork` - misnomer
- fork and exec?
- Well, okay... let's see
// co-ordinator

var backgrounder = require('backgrounder'),
    ...

var worker = backgrounder.spawn(
    path.join(__dirname, "../bin/letterpressMoves.js"),
    { 'children-count': 5 },
    function() {
        console.log("worker children started");
    }
);
// co-ordinator (cont’d)

var message = {
  board: board,
  ...
};

worker.send( message, function (m) {
  res.send([sequence, m.topMoves, stats]);
});

// worker

process.on('message', function(message, callback) {
  var movesObj = lp.getMovesForBoardInGameState(...);
  ...
  callback({
    dictionaryLength: dictionaryLength,
    wordsLength: wordsLength,
    movesLength: movesLength,
    topMoves: topMoves
  });
});
Node.js server with pool of 2 workers (200MB each)

difficult task

easy task 1
easy task 2
easy task 1 done
easy task 2 done
difficult task done

Internets

Server
Multiple processes?

• cluster - now we’re talking
• forks with shared memory
// all in app start code!

var cluster = require('cluster'),

// heroku config compatible
var MAX_PROCESSES = process.env.MAX_PROCESSES || 5;

...

if (cluster.isMaster) {
    // fork!
    for (var i = 0; i < MAX_PROCESSES; i++) {
        cluster.fork();
    }

    cluster.on('exit', function(worker, code, signal) {
        console.log('worker ' + worker.process.pid + ' died');
    });
}

else {
    http.createServer(app).listen(app.get('port'), function(){
        console.log("Express server listening on port " + app.get('port'));
    });
}
Node.js server with 5 forked processes (~150MB shared)

Internets

- difficult task

Child processes

- easy task 1
- easy task 2

- easy task 1 done
- easy task 2 done

- difficult task done

Server

Parent

Child processes

sharing CPU and memory

Friday, December 20, 13
The end?

- For almost all purposes, `cluster` meets our needs
- Simple model, easy to implement, performant
The end?

- But, I tried other stuff anyway...
“Threads” are weird in Node.js

- No shared variables, no locking!
- More like communicating w/processes
- load a single script file into the thread
- communicate by eval("code") !!!
- Standard module: threads_a_gogo
  (Use Audrey Tang’s fork on OSX)
var Thread = require('threads_a_gogo');

var numThreads = 5;
var threadPool = Thread.createPool(numThreads);

// file created with browserify, of all things!
// all necessary libraries smashed together - no require()!
threadPool.load('bin/serverMovesThreadRequireless.js');

... 

// and later...

var evalCall = 'getMoves(' + JSON.stringify(message) + ')';
threadPool.any.eval(evalCall, function(err, movesObjJson) {
  var movesObj = JSON.parse(movesObjJson);
  if (err) {
    handleErrors(err)
  } else {
    sendToClient('moves', movesObj.topMoves, { movesLength: movesObj.movesLength });
  }
});
Node.js server with thread pool for sorting

Internets

- difficult task
- easy task 1
- easy task 2

- easy task 1 done
- easy task 2 done

Server

- Node data process
- sorter thread
- sorter thread
- sorter thread
- sorter thread

Friday, December 20, 13
It’s life but not as we know it
Well, that mostly sucked

• But it gives me an idea
• If we’re communicating with a “thread” like a process...
Well, that mostly sucked

- But it gives me an idea
- If we’re communicating with a “thread” like a process...
- Why not do the same thing on the client?
Node.js server with client-side code for sorting

- **Browser**
  - Browser script
  - call server API
  - sort!
  - render results

- **Internets**

- **Server**
  - Node data process
  - sorter thread
GO HOME, CLOCK

YOU'RE DRUNK
(client-side thread)
Node.js server with client-side code for sorting

Browser

Browser script

call server API

sort!

render results

Internets

Server

Node data process

sorter thread
Node.js server with client-side Web Worker for sorting

Browser

Web Worker sorter

Browser script with access to DOM

- call server API
- receive response, call Web Worker for sorting
- render

Internets

Server

Node data process

sorter thread

time

Friday, December 20, 13
/** if client can calculate moves, just send the words. 
   Otherwise get a worker to 
   do the calculation here on the server */
if (isClientWorkerCapable) {
  sendToClient('words', wfb.wordStructs);
} else {
  /* call to thread pool, running virtually the same code! */
  var message = {
  ... 
  
  var evalCall = 'getMoves(' + JSON.stringify(message) + ')';
  threadPool.any.eval(evalCall, function(err, movesObjJson) {
    var movesObj = JSON.parse(movesObjJson);
    if (err) {
      ...
    } else {
      sendToClient('moves', movesObj.topMoves, { movesLength:
  movesObj.movesLength });
    }
    
  });
}
/* client-side worker initialization */
var workerSend;
/* can this browser even do Web Workers? */
if (typeof Worker !== 'undefined') {
    (function() {
        var worker = new Worker('./javascripts/browserify/clientMoveWorker.js');
        var callbacks = [];
        worker.onmessage = function(oEvent) {
            ...
        };
        workerSend = function(message, callback) {
            ...
            worker.postMessage(message);
        };
    })();
}
Friday, December 20, 13
Node.js server with client-side Web Worker for sorting

- Nearly identical code, running on client and server
Node.js server with client-side Web Worker for sorting

Nearly identical code, running on client and server

Try *that* in any other language
And finally!

• That’s why (*maybe*) you should consider doing CPU-bound tasks in Node.js

• Distributed processing across servers, processes, threads, and even between client and server, same code base

• 2013–14 trend! Rendering client or server – hood.ie, superfluous.io, etc.
Summary of tech

- cluster works great for a single machine for most purposes
- threads_a_gogo is a pain but gets you halfway to distributed processing
- Web Workers enable last minute rendering, sorting on client, without blocking DOM
Thanks

Neil Kandalaonkar

@flipzagging

neilk everywhere else

http://neilk.net/
http://github.com/neilk/letterpwn

Consulting in 2014? neilk@neilk.net