Optimizing and Integrating Node.js for the Cloud

Panagiotis Patros¹, Maria Patrou¹, Maxim Uzun¹, Jiapeng Zhu¹, Kenneth B. Kent¹, Michael Dawson²
University of New Brunswick, Faculty of Computer Science¹
IBM Canada²
{Patros.Panos, Maria.Patrou, muzun, jiapeng.zhu, ken}@unb.ca, Michael_Dawson@ca.ibm.com

Why Node.js?
Developers love Node.js: 315k modules on npmjs.com 3x growth rate vs other runtimes/languages

Platform as-a-Service (PaaS) Clouds
- Abstract large parts of the software and hardware stack
- Run user code and provide external services (e.g. DBs)
- Usually on isolated Linux containers
- Potentially multiple containers on the same host
- Install environment and code
- Language runtime, Application server, User code
- Scaling
  - Vertical (More Resources) and Horizontal (More Instances)

What is missing in Node.js?
- No multithreading
- Startup optimizations to speed scale out
- Hard to develop and debug live on the cloud
- Incomplete support for various new hardware features
- Needs to be maintained for new/emerging language features

Proposed Research Areas
- Develop Node.js scalability-oriented benchmarks
  - Identify areas for improving the scalability of Node.js
- Develop Node.js best-practices, modules and core updates towards better scalability
- Investigate deployment and external-service communication features missing from Node.js
  - Propose and evaluate improvements
- Investigate live-development tools and features missing from Node.js
  - Develop solutions
- Investigate hardware features Node.js does not utilize
  - Develop Node.js improvements for hardware support
- Investigate Node.js extended language features
  - Incorporate extended language features into Node.js

Why the growth?
- It is JavaScript (Server Side)
  - Availability of JavaScript talent
- Same language on client + server side
- End-to-End JavaScript stack
  - Greater productivity and integration with JSON APIs
- Event-driven, single-threaded model
  - Eliminates concurrency and thread-safety worries
- Thousands of concurrent connections with minimal overhead

Where Node.js?
Good fit for highly scalable web applications

Inherently event based: perfect fit for asynchronous non-blocking I/O

Blocking I/O Example (e.g. Java)
Parcel collection depot
'Service Deck'
Move customers to the order line

Asynchronous I/O Example (Node.js)
Fast food restaurant
'Server'...

The Project
Research Program at UNB with IBM Canada support
- Funding through CAS and NSERC
- Duration 3 years (2016-2019)
- Researchers
  - 2 PhD and 2 MCS students
    - Talk to us and apply for grad school at UNB CS, if interested!
      - http://www.unb.ca/admissions/applying-to-unb/
- 3 Undergrad Summer students
- UNB Faculty and IBM personnel

Aug 22, 2016
http://www.modulecounts.com