Trends in Application Development

For Enterprise Web Applications

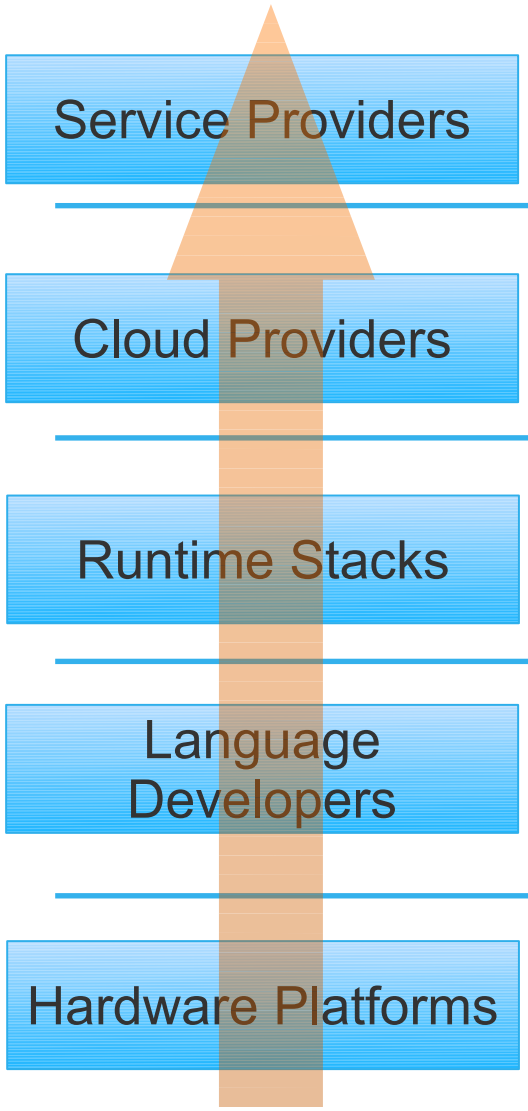
Chris Bailey
IBM Runtime Technologies

© 2015 IBM Corp

#1
Language Choices

© 2015 IBM Corporation

The Ecosystem is Increasingly Polyglot



Leverage the platform capabilities to accelerate computation and communication regardless of language runtime

Need a premium deployment platform where the language runtime showcases their density, performance, scaling and reliability

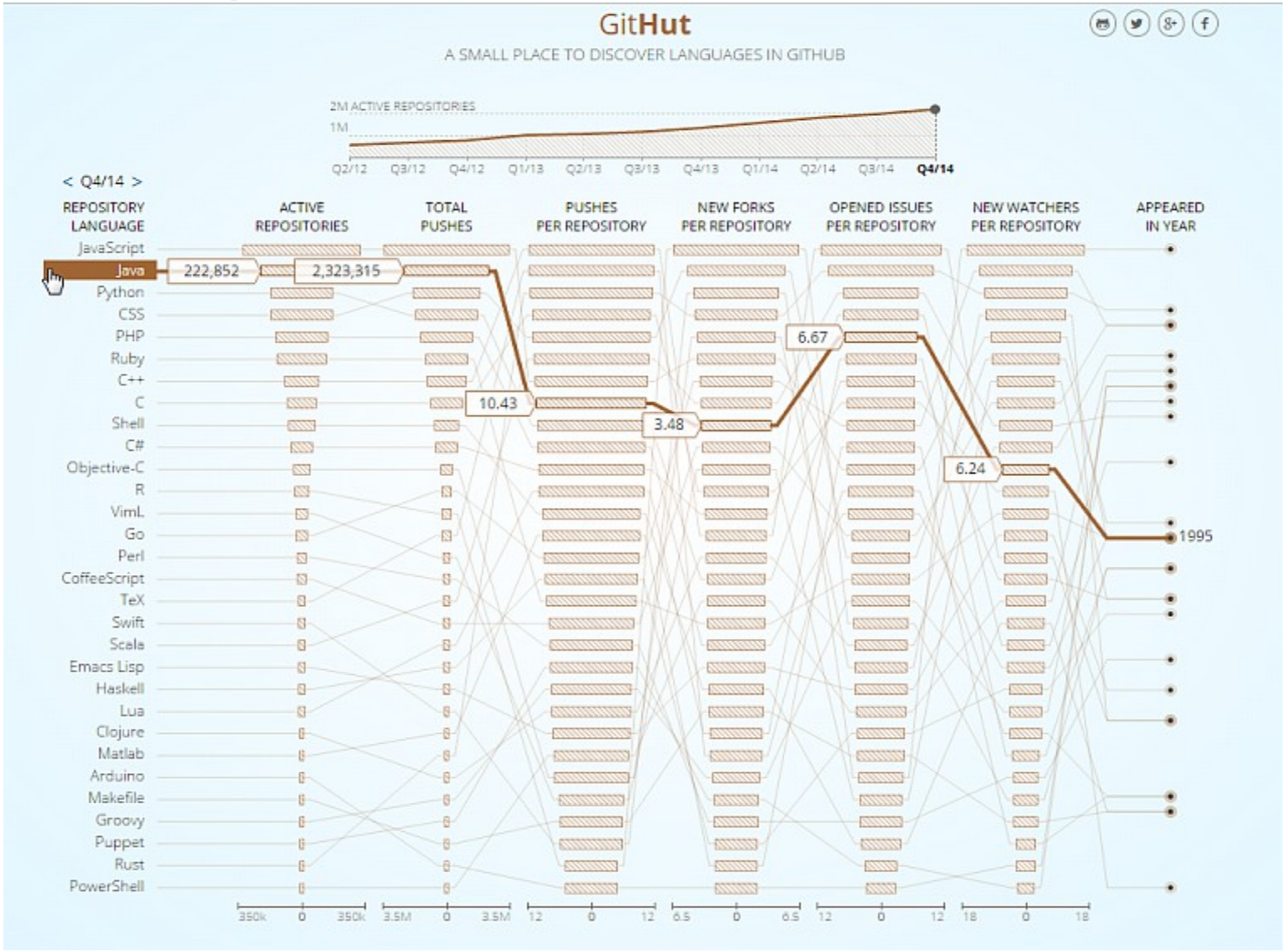
Want hardware platform benefits to showcase their stack regardless of the language runtime engine

Want high performance infrastructure but lack expertise. Lack a common point of investment for runtime acceleration on all platforms

Want to ensure quality implementations of language runtimes on their HW platforms and drive innovation to exploit hardware

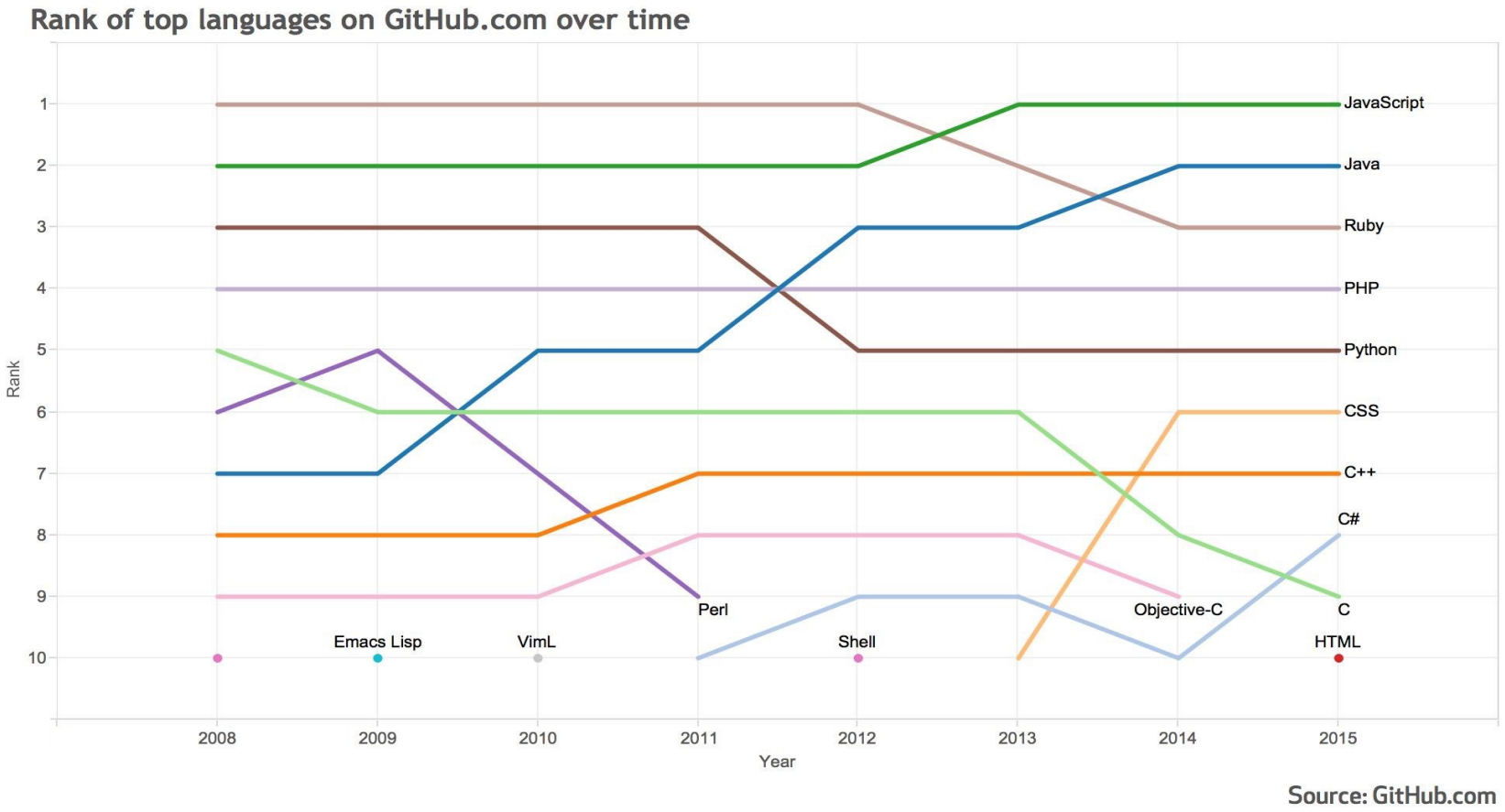
© 2014 IBM Corporation

GitHub Adoption



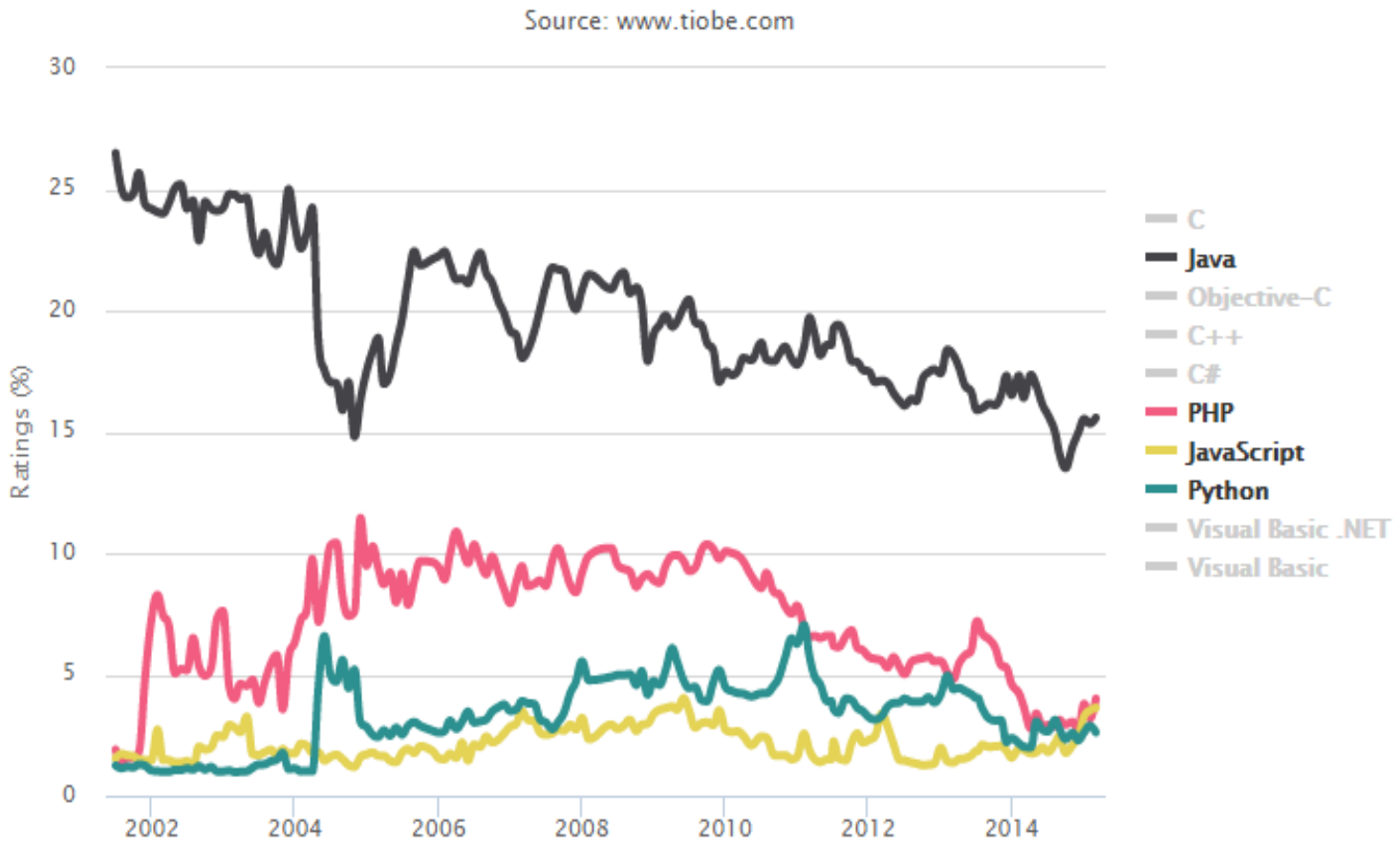
© 2014 IBM Corporation

GitHub Trends



© 2014 IBM Corporation

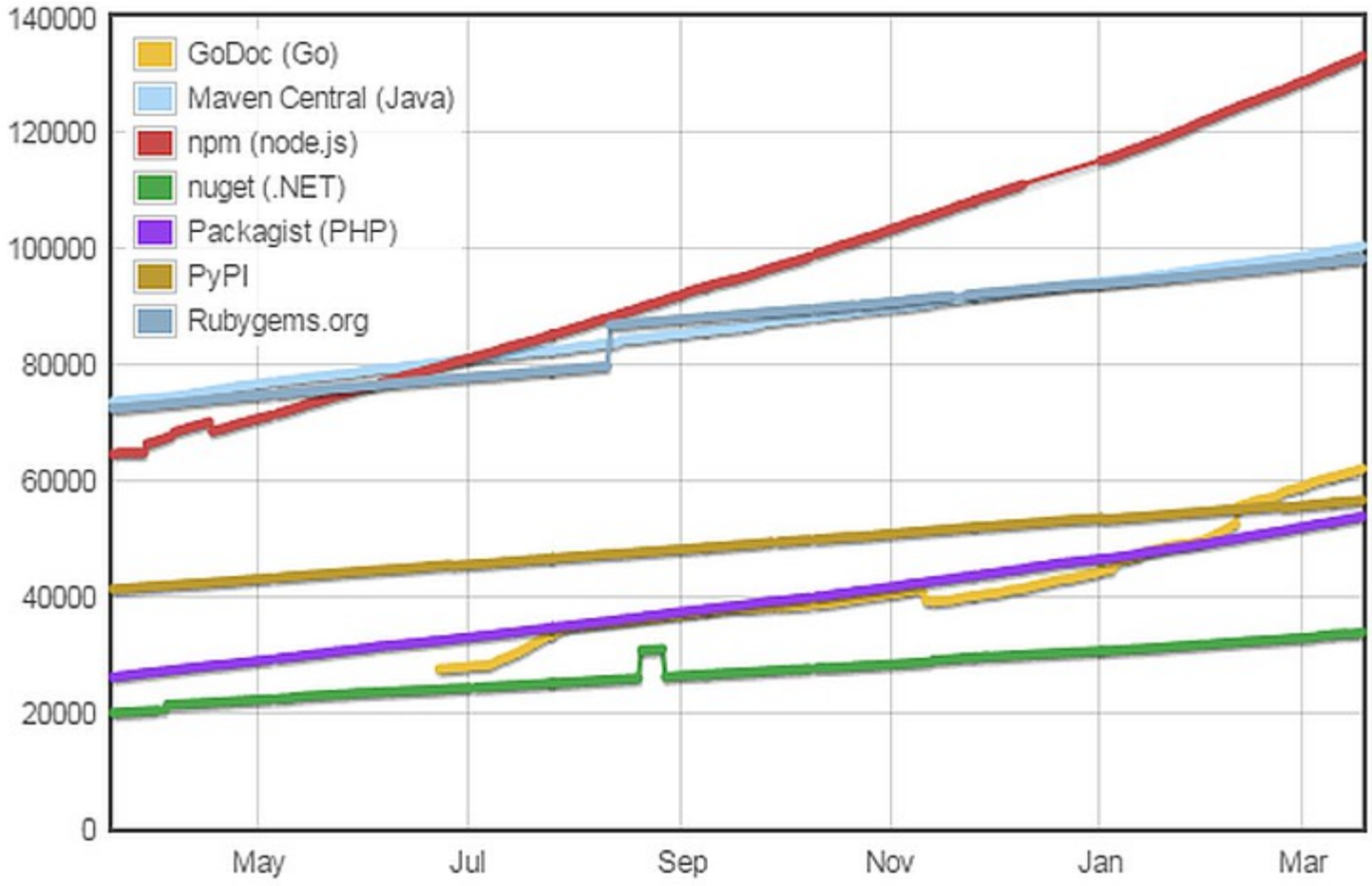
Tiobe Community Programming Index



Ratings based on the number of skilled engineers, courses and third party vendors.

© 2014 IBM Corporation

modulecounts.com



© 2014 IBM Corporation

Computer Science Zone Jobs Report

LANGUAGE	JOB OPENINGS*	SALARY
SQL	211,017	$55,000
Java	148,216	$84,000
Javascript	88,013	$80,000
C#	58,923	$79,000
Python	56,635	$85,000
Perl	46,217	$81,000
Visual Basic	27,804	$78,000
Ruby	24,045	$85,000
PHP	24,045	$77,000
MATLAB	16,289	$71,000

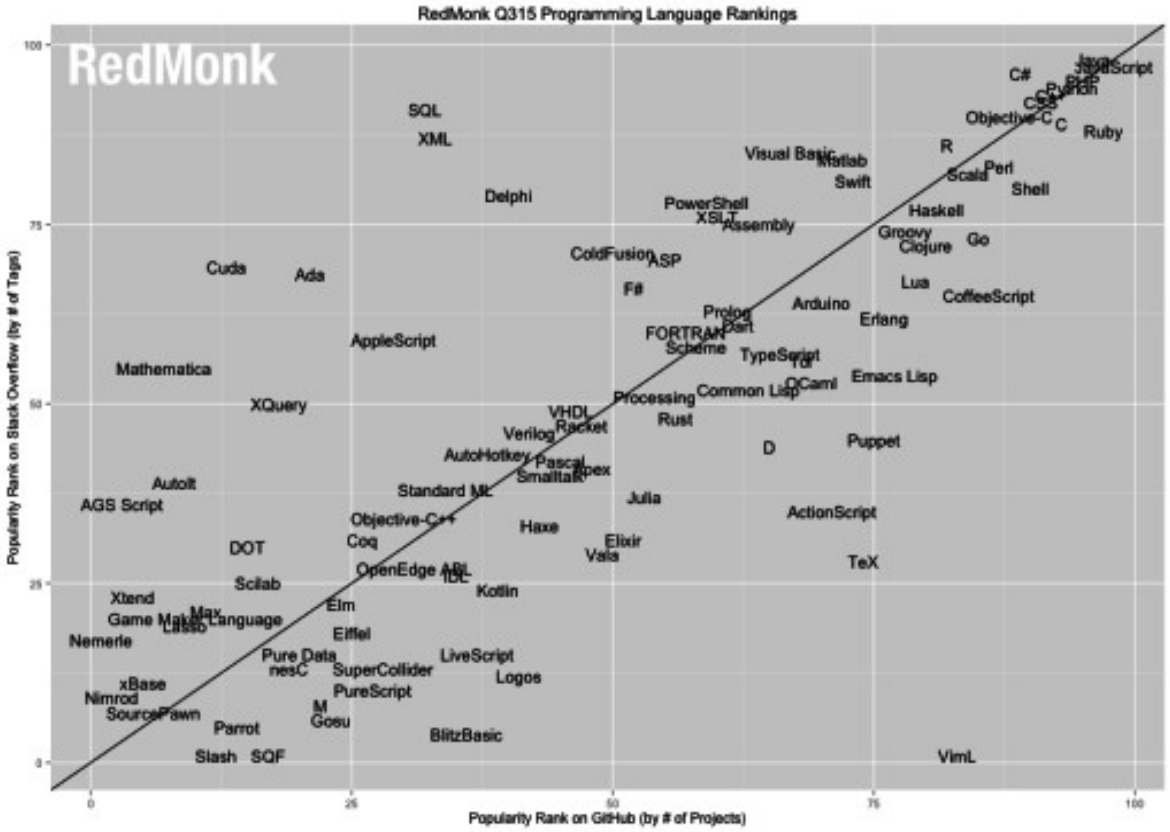
LANGUAGE	JOB OPENINGS*	SALARY
VB.Net	9,260	
Ruby on Rails	8,403	$84,000
Objective-C	7,336	$81,000
Scala	3,509	$112,000
Groovy	3,376	$86,000
ActionScript	1,532	$61,000
Delphi	672	$64,000
Arduino	188	$63,000
Rust	56	$81,000

*December 2014

Average salary and job vacancies (Computer Science Zone)

© 2014 IBM Corporation

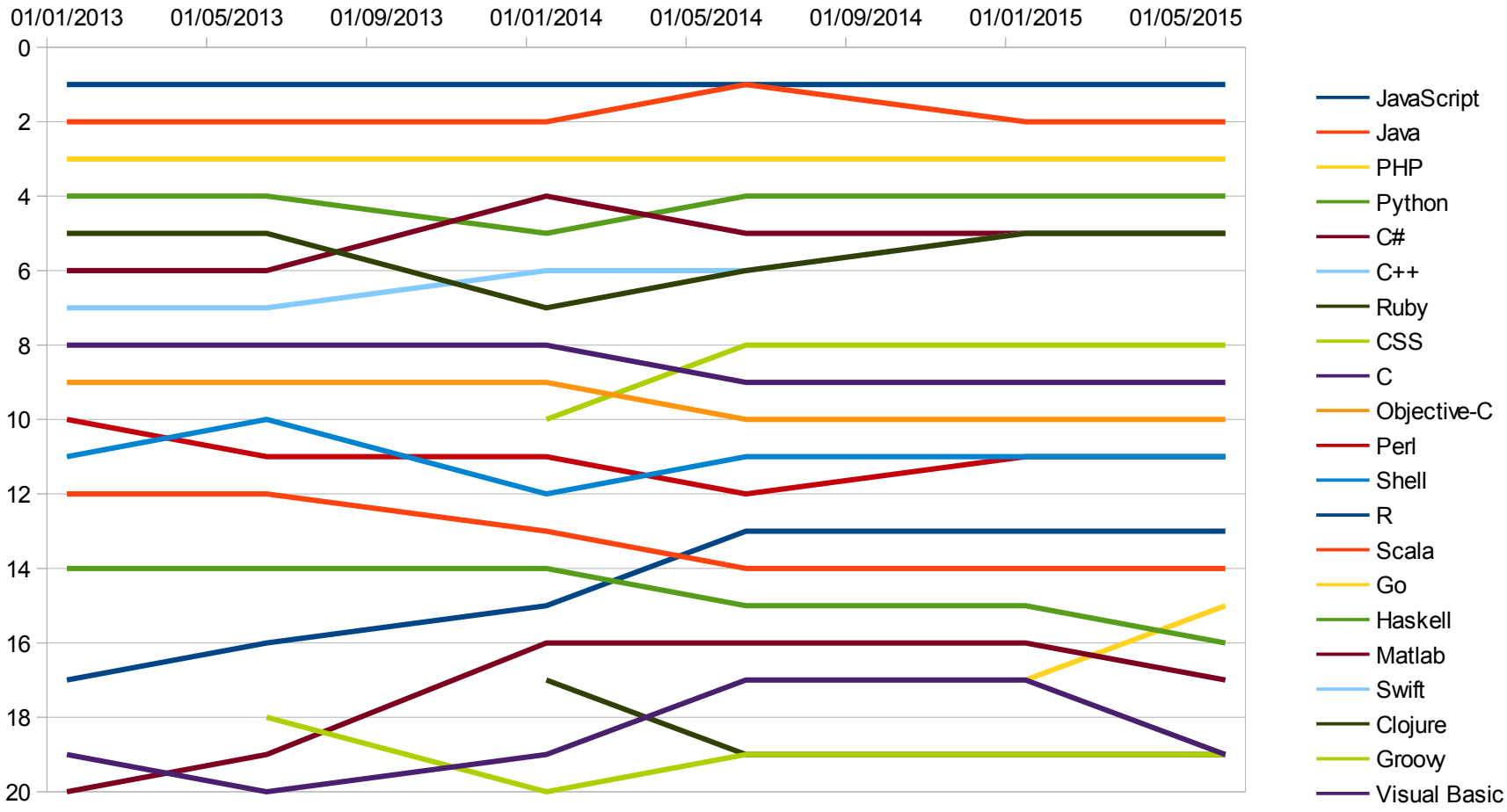
RedMonk Language Rankings



1 JavaScript
2 Java
3 PHP
4 Python
5 C#
5 C++
5 Ruby
8 CSS
9 C
10 Objective-C
11 Perl
11 Shell
13 R
14 Scala
15 Go
16 Haskell
17 Matlab
17 Swift
19 Clojure
19 Groovy
19 Visual Basic

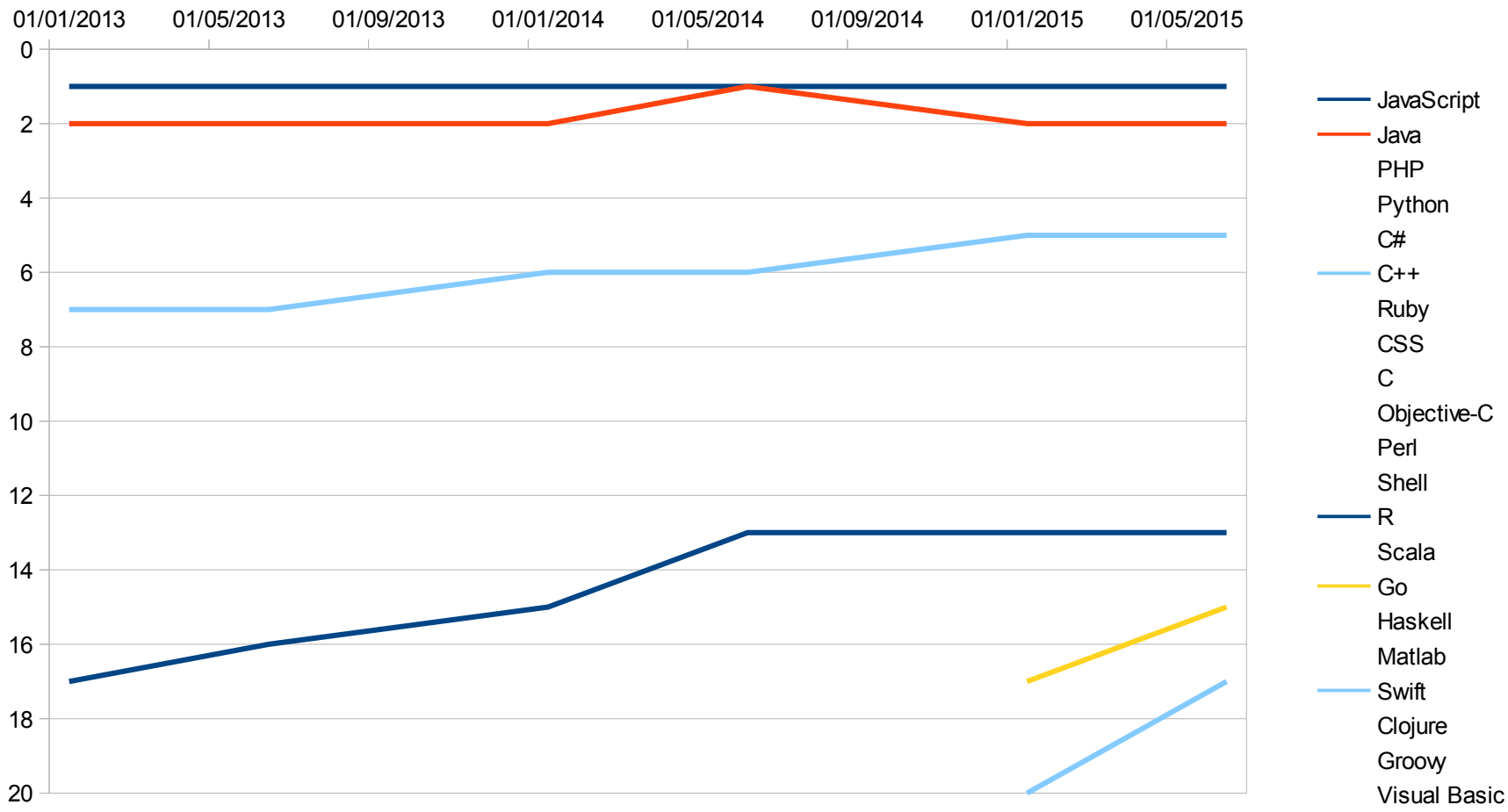
© 2014 IBM Corporation

RedMonk Language Rankings Trends



© 2014 IBM Corporation

RedMonk Language Rankings Trends



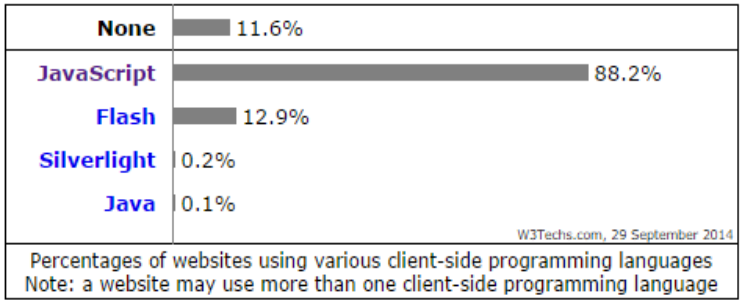
© 2014 IBM Corporation

#2
Engaging Applications

© 2015 IBM Corporation

Browser Applications

- JavaScript is ubiquitous in the browser
  - Supported in every browser
  - Integration with HTML and CSS



- JavaScript is not affected by negative publicity....

CERT | Software Engineering Institute | Carnegie Mellon University

Homeland Security
Sponsored by the DHS Office of Cybersecurity and Communications

***Unless** it is **absolutely necessary** to run Java **in web browsers, disable it** as described below, even after updating to 7u11. This will help mitigate other Java vulnerabilities that may be discovered in the future.*

US-CERT
UNITED STATES COMPUTER EMERGENCY READINESS TEAM

*This and previous Java vulnerabilities have been widely targeted by attackers, and new Java vulnerabilities are likely to be discovered. **To defend against** this and **future** Java **vulnerabilities**, **consider disabling Java in web browsers**…*

© 2015 IBM Corporation

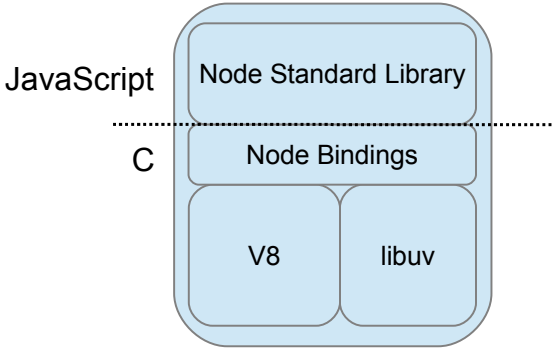
Browser Applications to Server Applications

- Java has originally targeted at for creating user applications
- Eventually started to migrate to the server:
  - JPE launched in 1998
- Today Java has rich platform support:
  - Linux x86, Linux POWER, zLinux
  - Windows, Mac OS, Solaris, AIX, z/OS
- JavaScript usage is starting to grow on the server

© 2015 IBM Corporation

Server Side JavaScript: Node.js

- Single Threaded Event based JavaScript framework
  - Uses non-blocking asynchronous I/O
- Wraps the Chrome V8 JavaScript engine with I/O interfaces
  - Libuv provides interaction with OS/system



- Designed to build scalable network applications
  - Suited for real time delivery of data to distributed client
- Available on a wide set of platforms:
  - Linux on x86, ARM, Power and Z
  - Windows, Mac OS, Solaris, SmartOS and AIX

© 2015 IBM Corporation

#3
Reactive Programming

© 2015 IBM Corporation

Reactive Programming

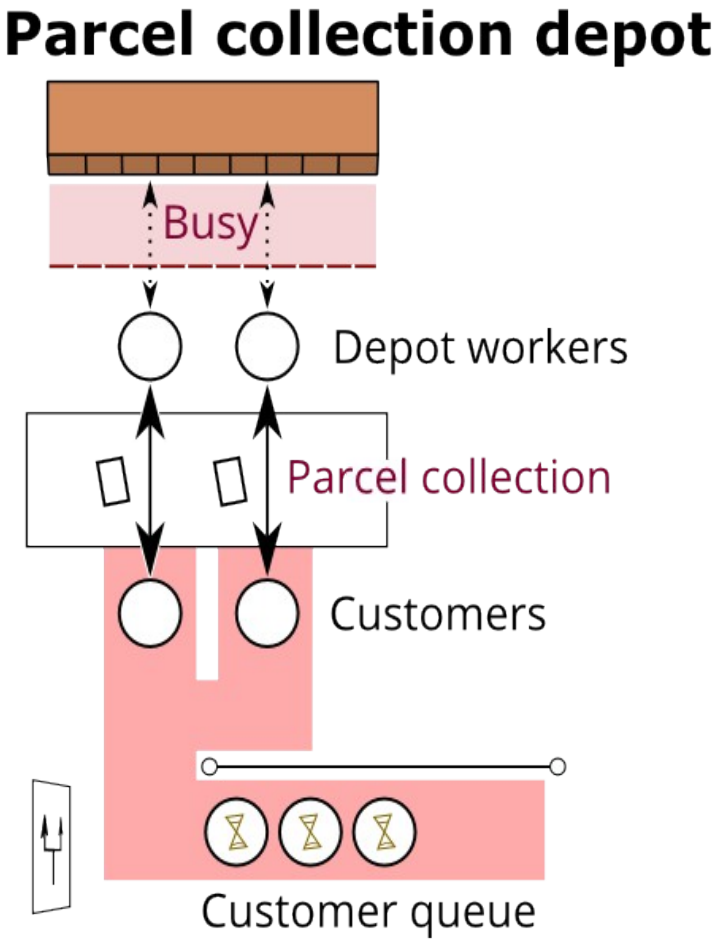
“a programming paradigm oriented around data flows and the propagation of change.”

- Can easily express dynamic data flows
- Execution model propagates changes through the model
- Typically makes use of asynchronous models to propagate events

© 2015 IBM Corporation

Typical approach to I/O

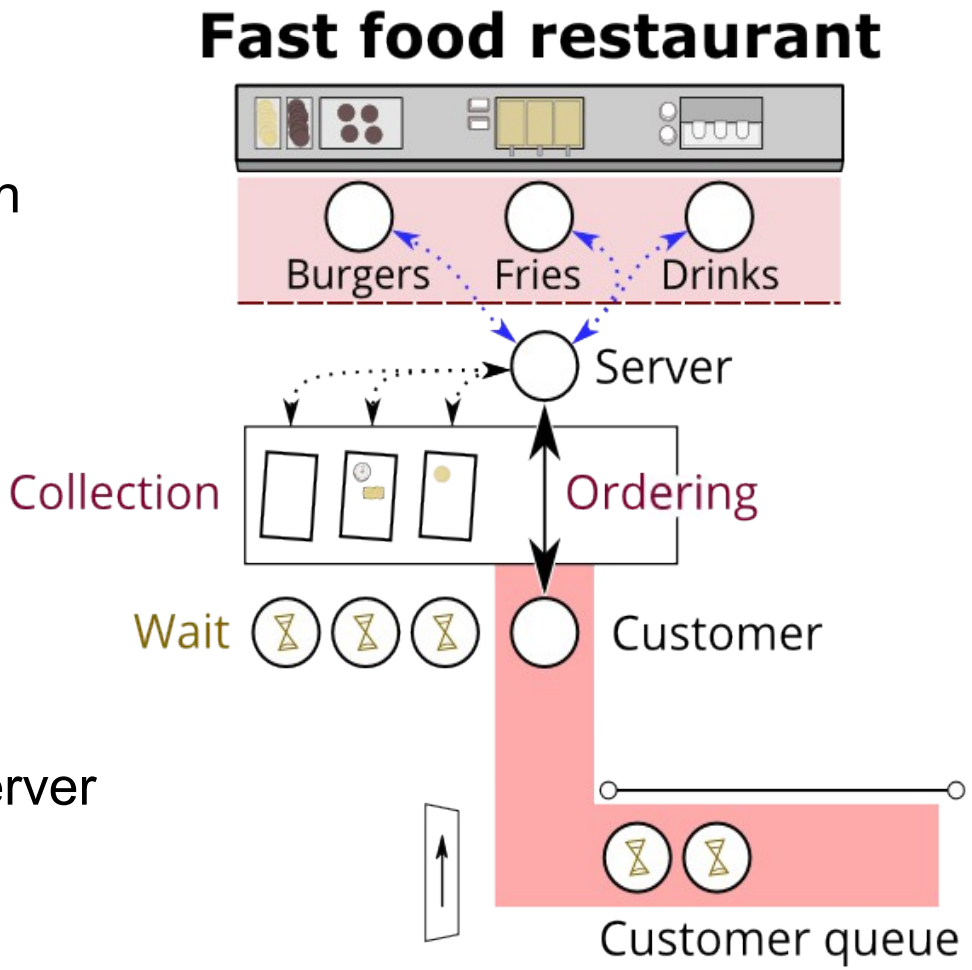
- One thread (or process) per connection
  - Each thread waits on a response
  - Scalability determined by the number of threads
- Each thread:
  - consumes memory
  - is relatively idle
- Number of concurrent customers determined by number of depot workers
- Additional customers wait in a queue with no response



© 2015 IBM Corporation

Asycnhronous Non-Blocking I/O

- One thread multiplexes for multiple requests
  - No waiting for a response
  - Handles return from I/O when notified
- Scalability determined by:
  - CPU usage
  - “Back end” responsiveness
- Number of concurrent customers determined by how fast the food Server can work
- Or until the kitchen gets slammed



© 2015 IBM Corporation

JavaScript and Asynchronous I/O

- JavaScript is inherently designed to be asynchronous
  - eg. onClick and onMouseOver events

- This applies easily to server applications as well

```
var http = require('http');

var server = http.createServer();
server.listen(8080);

server.on('request', function(request, response) {
        response.writeHead(200, {"Content-Type": "text/plain"});
        response.write("Hello World!\n");
        response.end();
});

server.on('connection', function(socket) {});
server.on('close', function() {});
server.on('connect', function(socket) {});
server.on('upgrade', function(request, socket, head) {});
server.on('clientError', function(exception, socket) {});
```


© 2015 IBM Corporation

#4
Cloud Deployments

© 2015 IBM Corporation

Cloud

*"a virtual, **dynamic environment** which maximizes use, is **infinitely scalable**, **always available and needs minimal upfront investment** or commitment"*

- Removes infrastructure as a bottleneck to rapid application delivery and expansion
- Provides "compute on tap"
- But taps come with meters and usage charge models

© 2015 IBM Corporation

Compute Costs

Offering	RAM Cost	CPUs
IBM Bluemix (CF)	$24.15 GB/Month	4vCPUs per instance
IBM Bluemix (Containers)	$ 9.94 GB/Month	4vCPUs per GB
run.pivotal.io	$21.60 GB/Month	4vCPUs per instance
Heroku (Hobby)	$14.00 GB/Month	1 "CPU share" per 512MB in an instance
Heroku (Professional)	$50.00 GB/Month	1 "CPU share" per 512MB in an instance
Amazon EC2 (SLES)	$16.56 GB/Month	1 vCPU per 4GB in an instance.

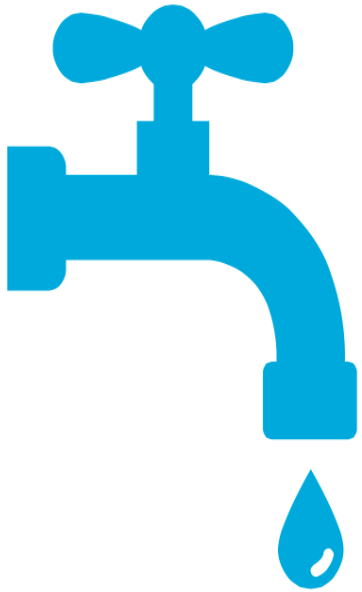
© 2015 IBM Corporation

Cloud Economics

-Xmx: $$$

© 2015 IBM Corporation

Cloud Economics

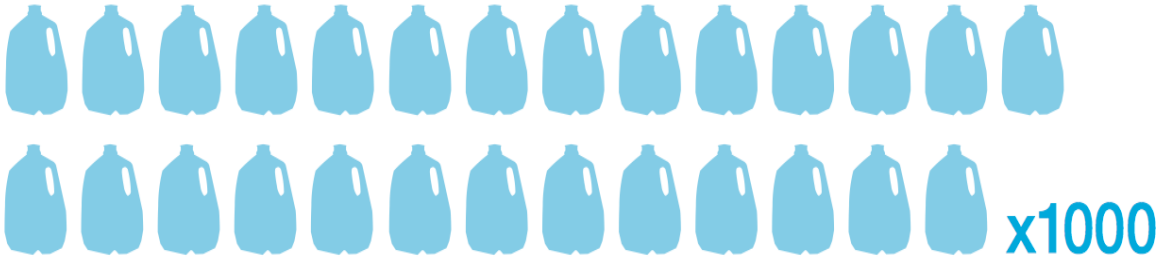


A faucet that drips just **once per second** wastes

2,700

gallons of water annually.[4]

Someone will be looking at your leaky app

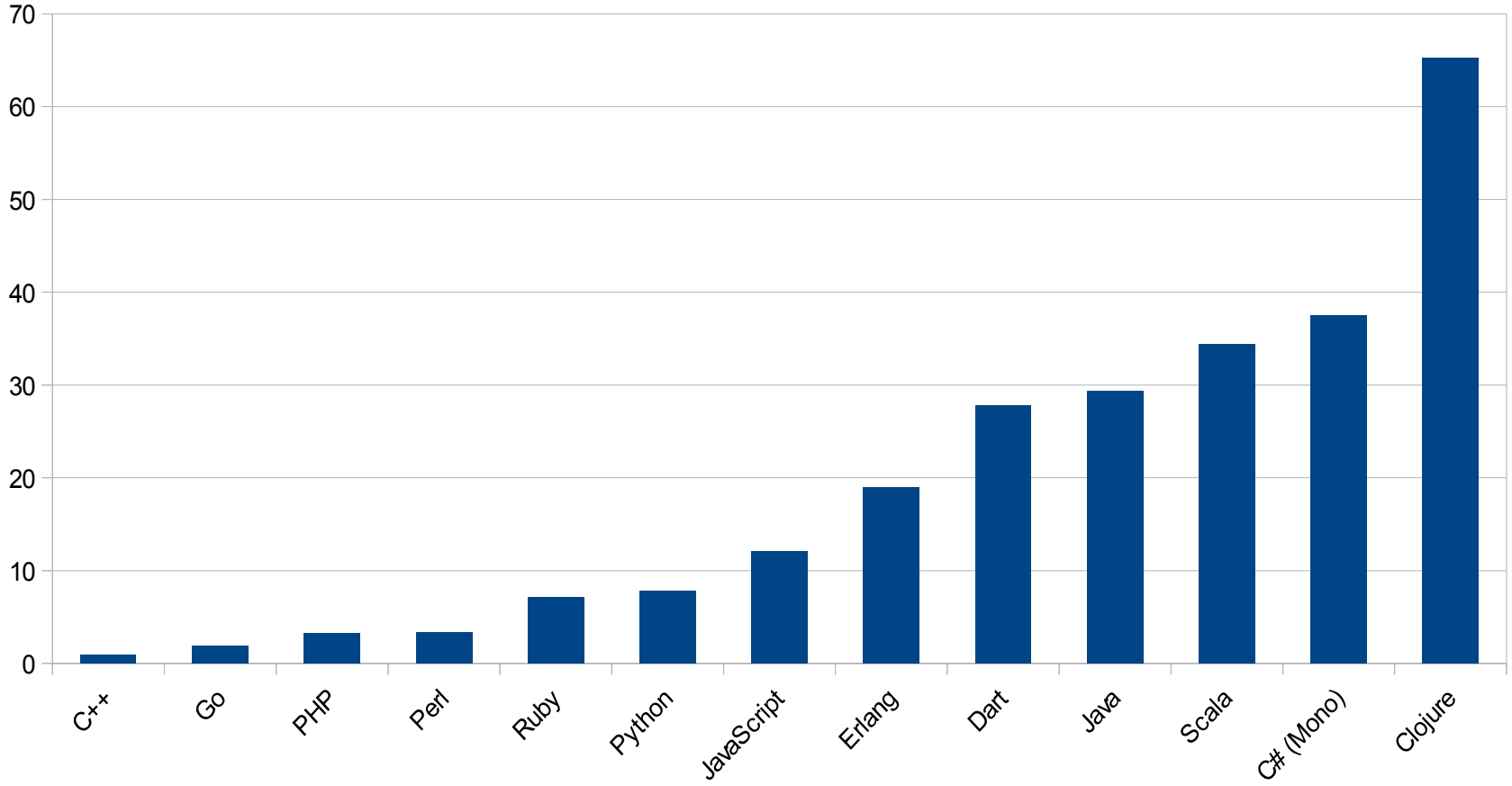


© 2015 IBM Corporation

Clouds are Polyglot

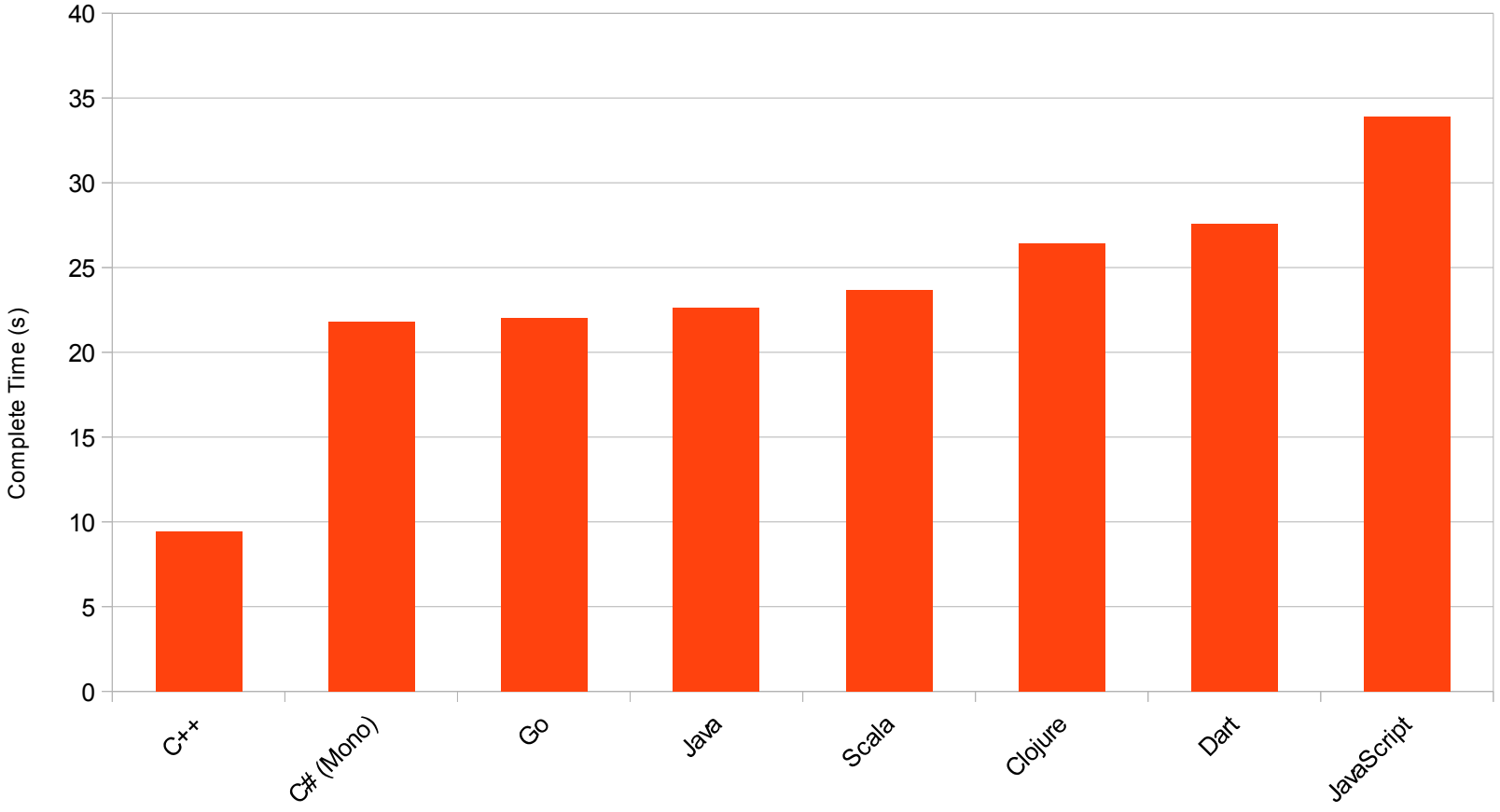
© 2015 IBM Corporation

N-Body Benchmark: Memory Footprints



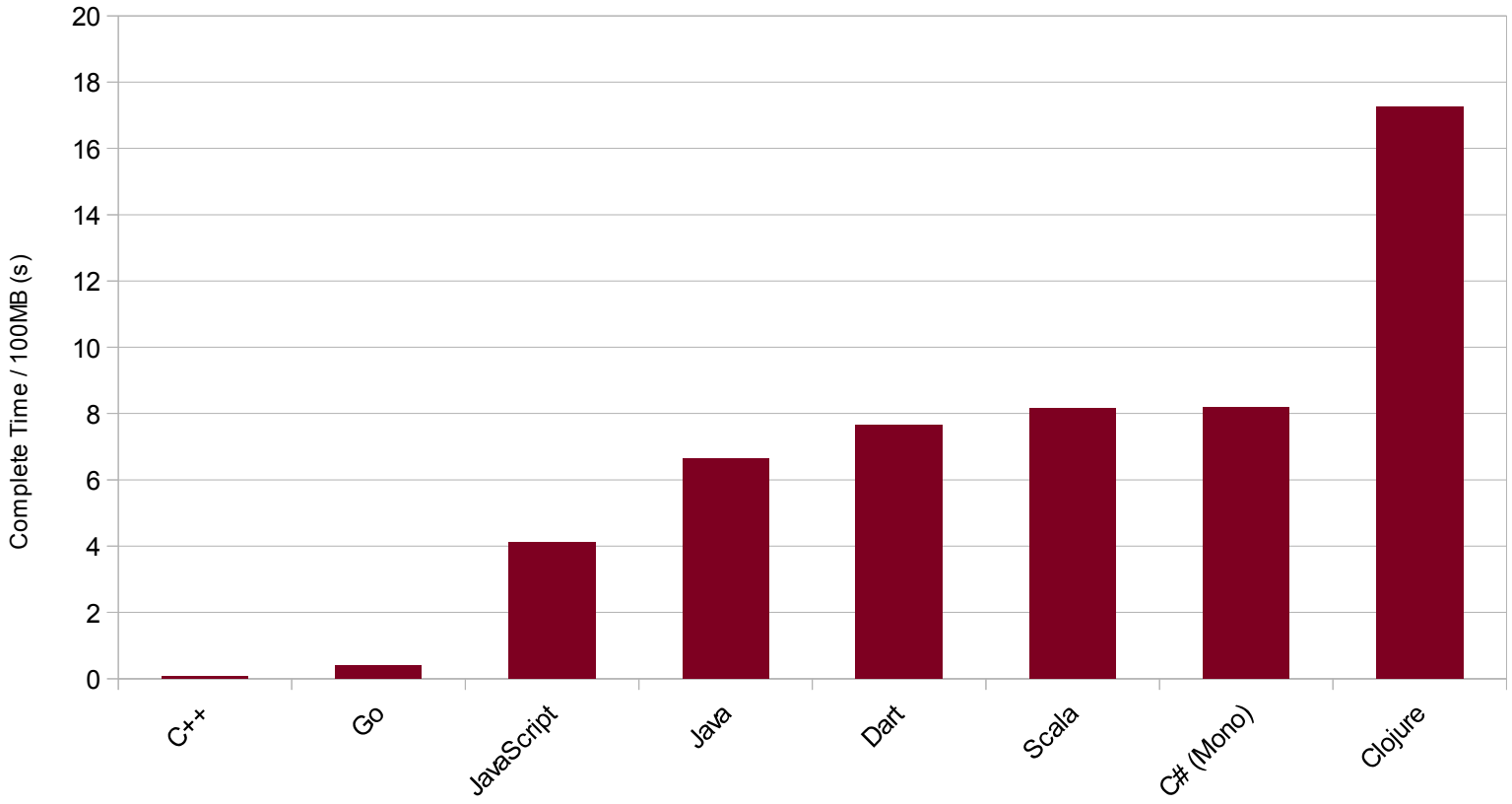
© 2015 IBM Corporation

N-Body Benchmark: Time to Complete



© 2015 IBM Corporation

N-Body Benchmark: Time to Complete



© 2015 IBM Corporation

#5
MicroServices

© 2015 IBM Corporation

MicroServices

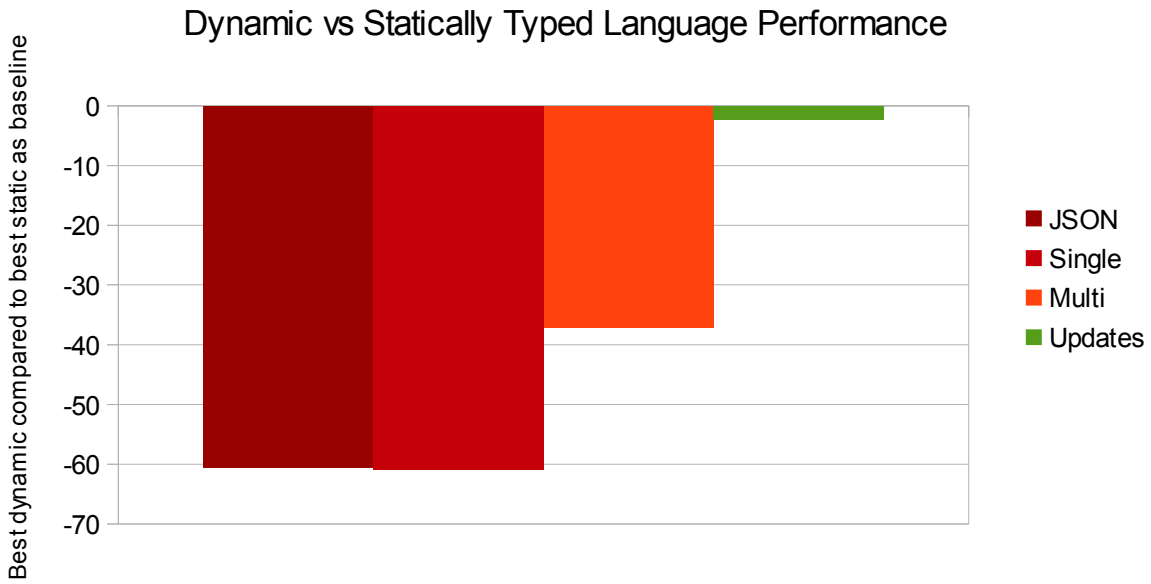
“Do one thing, and do it well”

- Services are small and targeted to their task
- Services are organized around capabilties
- Services are self contained, storing their own data

© 2015 IBM Corporation

Dynamically vs Statically Types Languages

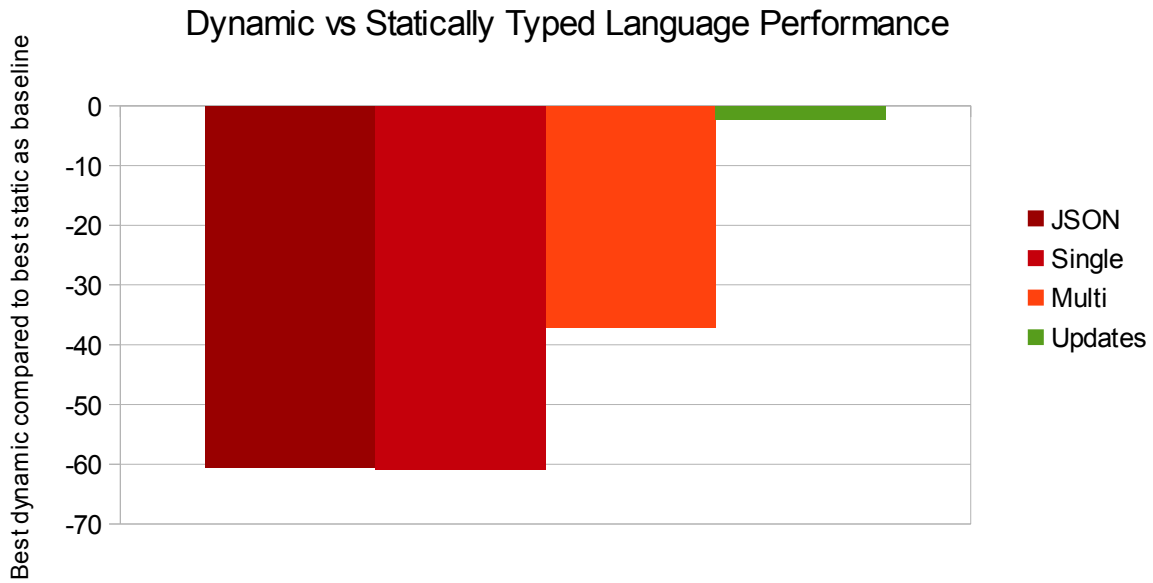
- Dynamically typed languages are harder to manage under the hood
- They have lower runtime performance for computational tasks



© 2015 IBM Corporation

Dynamically vs Statically Types Languages

- Dynamically typed languages are harder to manage under the hood
- They have lower runtime performance for computational tasks



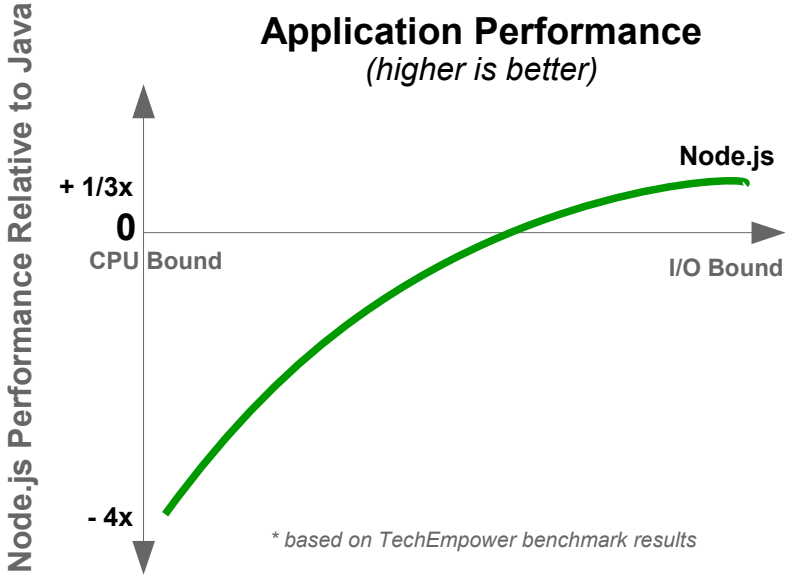
- They have higher scope for data integrity issues:

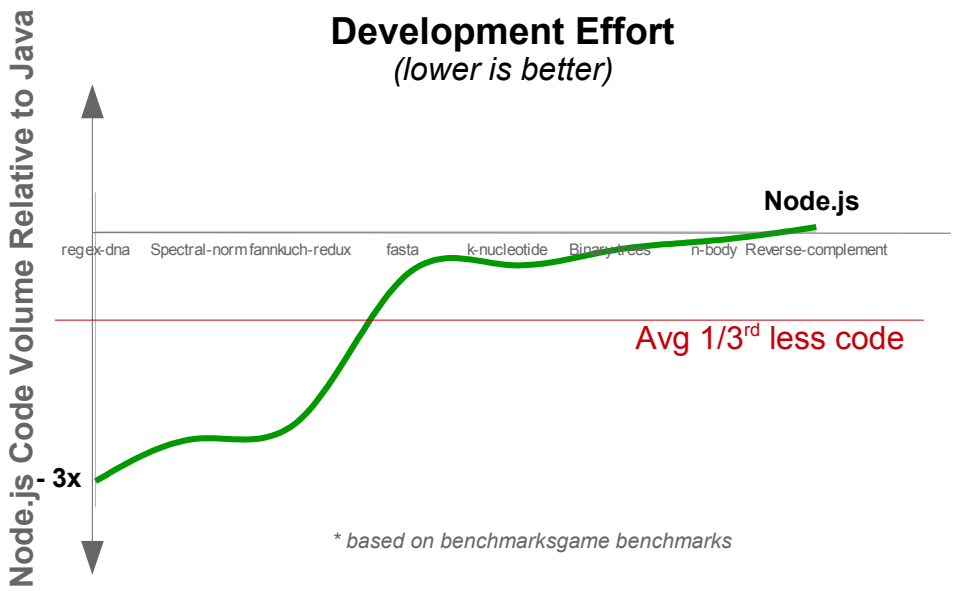
```
> 12 + 3
123                              // 12 or 3 previously used as text
```

- Statically typed languages throw error at compile time or runtime

© 2015 IBM Corporation

Choosing the Right Language for the Service





- Node.js has higher performance for I/O
  Fast async non-blocking framework for scalability
- Node.js allows “fullstack” webapp development
  End to end JavaScript for server and browser

However....

- Java is much faster at computational logic
  Node.js performance is non-ideal for transactions

- Node.js has higher developer productivity
  Many applications developed with significantly less code
- Rich module system simplifies development
  Reduces need to develop custom code

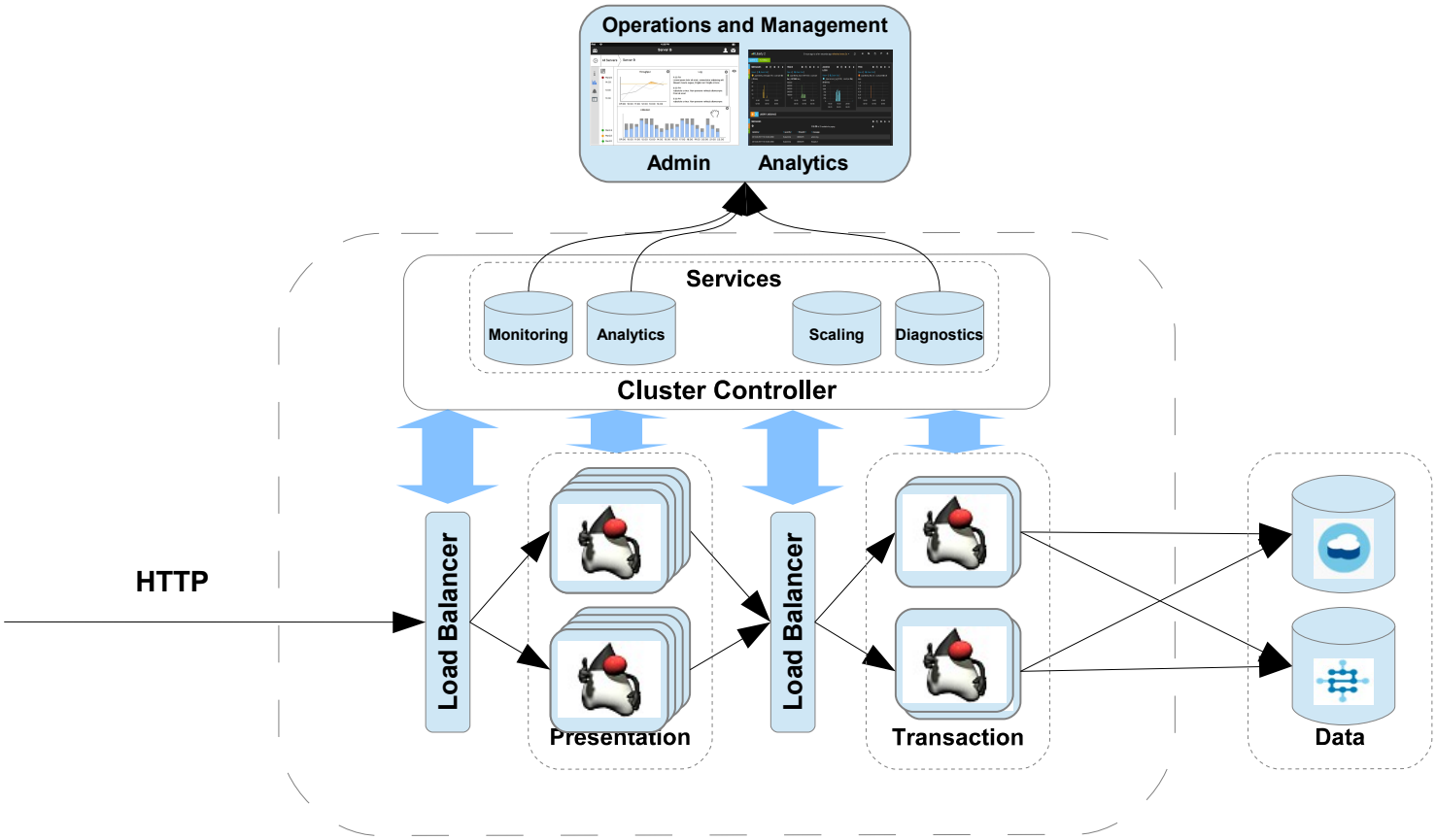
However...

- Java is strongly typed, ensuring data correctness
  Node.js type mis-matches can result in incorrect results

Node.js fits the presentation tier, offloading to Java* for business transactional logic

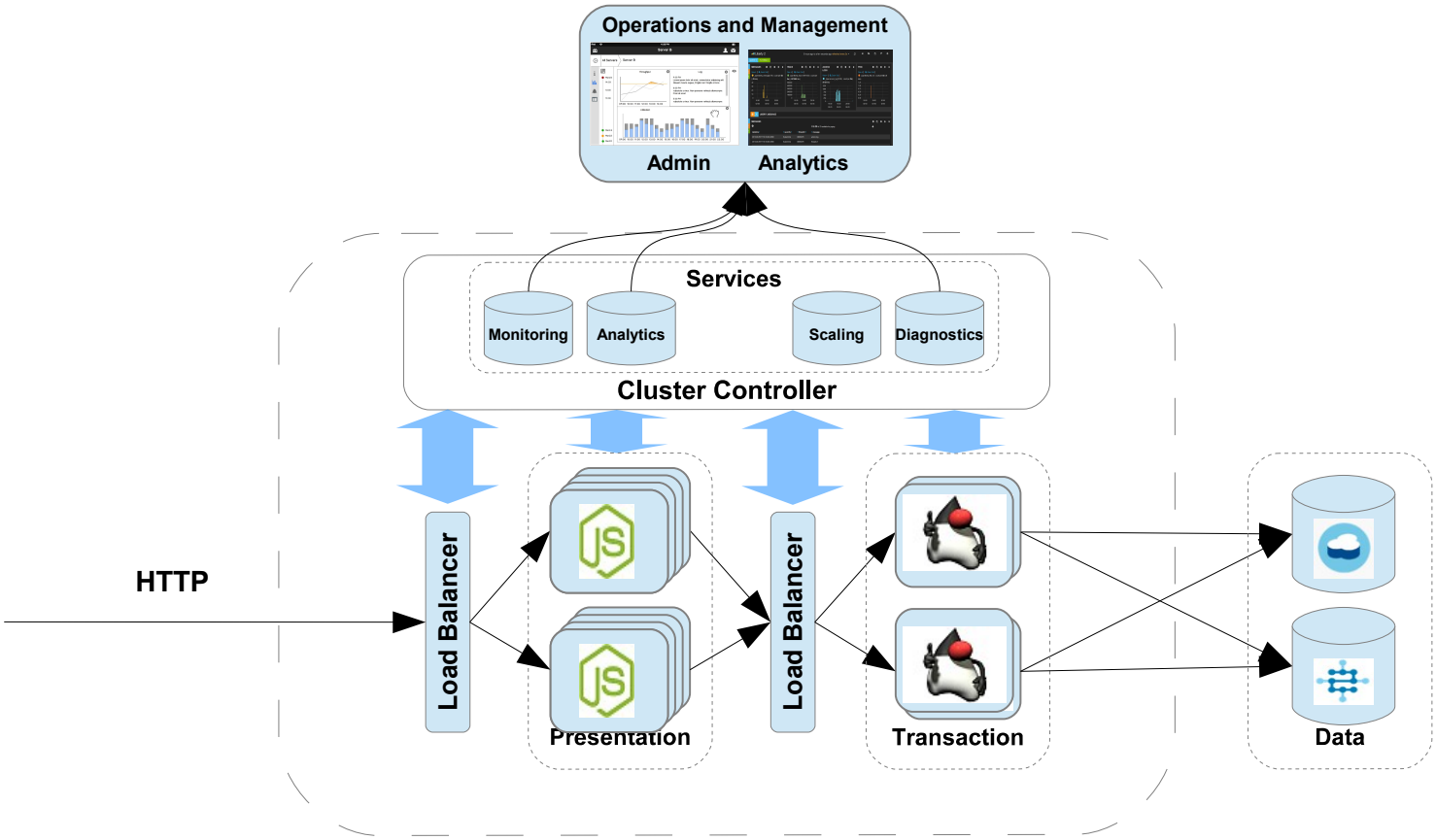
© 2015 IBM Corporation

Service topology for Web Applications



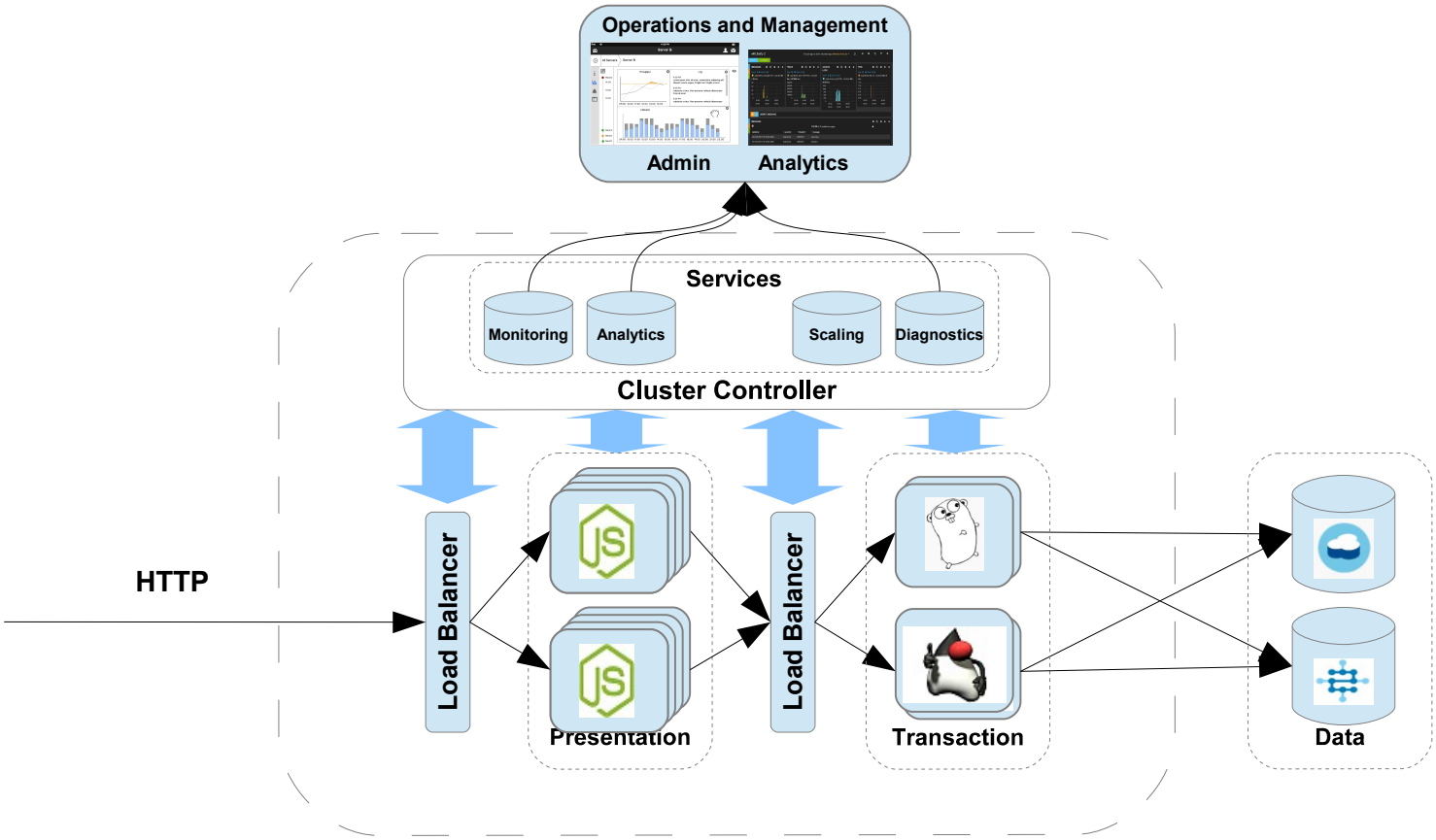
© 2015 IBM Corporation

Service topology for Web Applications



© 2015 IBM Corporation

Service topology for Web Applications



© 2015 IBM Corporation

© 2015 IBM Corporation