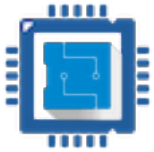
Distributed Platform Development with Groovy

Rob Fletcher – @rfletcherEW

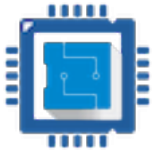
Dan Woods – @danveloper

What is a “*platform*”?



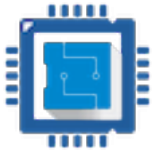
What is a "*platform*"?

"A broad set of functionality structured in a common way that exposes one or many business functions"



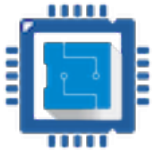
What is a "*platform*"?

- A pattern of architecting a "system"
- System is designed to expand and contract as business needs evolve



Distributed computing & data encapsulation

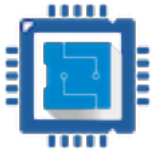
- Platform service components run in their own process and memory space
- Services are responsible for their own business logic and data processing rules



Platform as a service

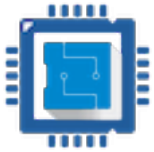
- Vertically sliced by business function
- Platform-oriented architecture means services are composable
- Pick-and-choose services as they relate to some business functionality

Why is platform architecture important?



Why is platform architecture important?

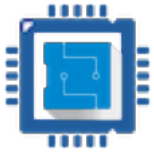
- Enforces modularity of concepts, not just code
- Easy to get a grasp on the “system” and its capabilities
- Allows functionality to scale as complexity grows
- “True” vertical slices



Microservice architecture

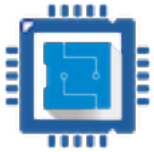
- Microservices are great, but don’t speak directly to Platform Architecture
- Platform microservices are designed as atomic subsets of capabilities in the platform
- A collection of microservices comprise a vertical slice of a system’s business function (encapsulating their own data domains)

Why choose Groovy for platform development?



Groovy has come a long way...

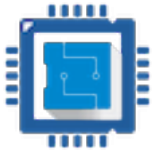
- Since Groovy 2.0.0
  - Static Compilation!
  - Type Checking!
- "*Indy*" support in Java 7+ gives near-static-compilation performance while maintaining dynamic nature
- Pick-and-choose your compilation strategy



Important note on indy support...

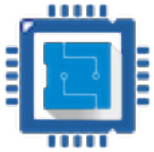
```
tasks.withType(GroovyCompile) {
  groovyOptions.optimizationOptions.indy = true
}
```

http://blog.freeside.co/2014/06/24/enabling-groovys-invokedynamic-support-in-gradle/



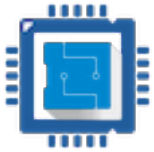
Groovy beans are superior!

- In microservice architecture, components encapsulate their data, and inherently the model
- Groovy beans cut the verbosity, and get right at the model of the data
- Rich models are easily built into Groovy beans, especially when considering the MetaClass



Great out-of-the-box support for microservices

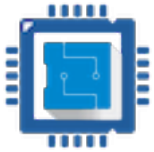
- Groovy SQL is a great foundation for data encapsulation
- Groovy String/URL extensions make REST calls a breeze
- The world's fastest JSON parser!



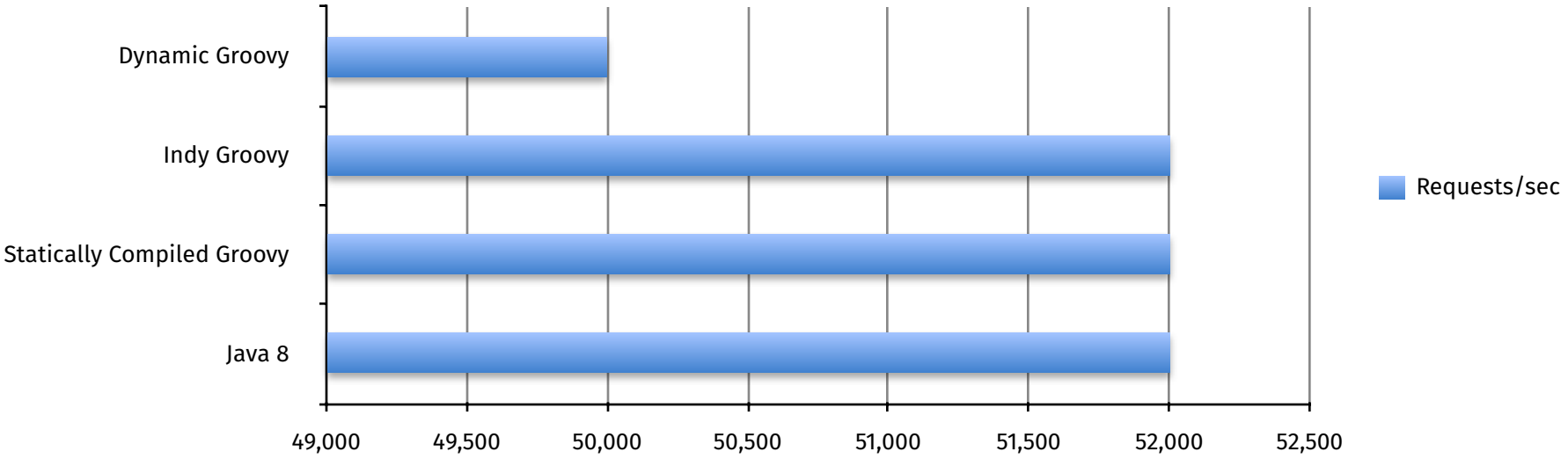
Groovy Extensions

- Groovy's extension framework allows compile-time analysis driven by business/ technical rules
- Annotation collectors can roll-up static compilation, as well as common service-layer annotations (JAX-RS, for example)

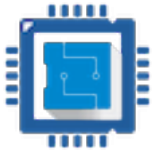
Debunking Groovy Performance Myths



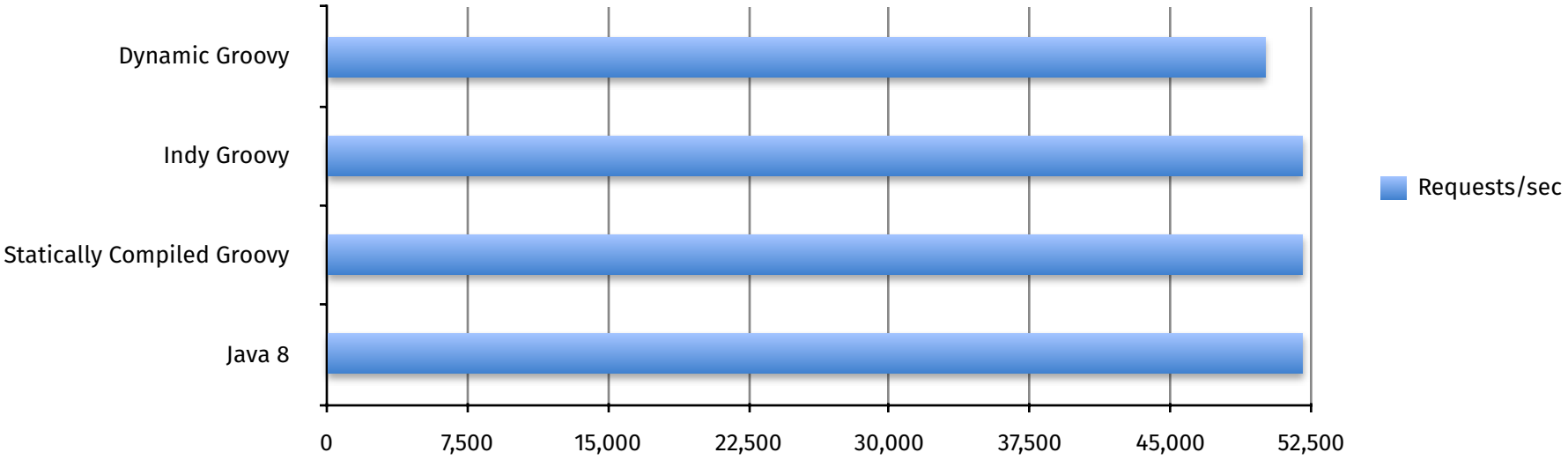
Ratpack Benchmarks



https://github.com/Netflix-Skunkworks/WSPerfLab

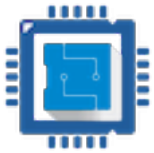


Ratpack Benchmarks



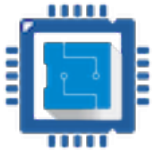
https://github.com/Netflix-Skunkworks/WSPerfLab

Platform Tools in the Groovy Ecosystem



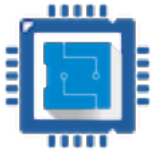
Grails

- Groovy-first web framework; full-stack defined
- Run in either container or in a stand-alone fashion
- Abstractions on data encapsulation (GORM)
- Extremely advanced REST support



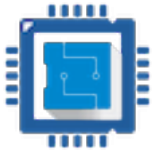
Spring Boot

- Opinionated framework for Spring
- Builds on all of the new Spring hotness, including conditional wirings, and advanced data bindings
- Advanced REST support, enables edge slicing
- Cloud-native, preferable to run in an embedded container



Ratpack

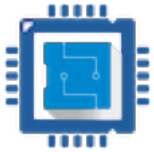
- JVM web framework, designed for high-throughput, non-blocking request system
- First class support for Groovy, seamless static compilation
- Seamless Groovy Sql integration (through the SqlModule)
- Excellent choice for developing a microservice



Vert.x

- Vertical slicing of applications, fits very well in the Platform Architecture concept
- Extremely high throughput
- Asynchronous, event-driven architecture model
- Packaged and run as a fat jar

To Summarize...



Summary

- Platform architecture is important for scaling systems as business complexity grows
- Platform verticals are defined by business functions
- Microservices can be built to compose the business functions
- Groovy provides a rich ecosystem for developing a distributed platform architecture