Microservices with Ballerina
Open Source Summit 2019

Sameera Jayasoma
@sameerajayasoma
WSO2
Agenda

- The problem
- How Ballerina attempts to solves it
- Demo
  - Write a microservice in Ballerina
  - Deploy to Docker
  - Deploy to Kubernetes
  - Sequence diagrams
- What makes Ballerina unique for writing microservices
- Ballerina Platform
The problem
Increasing demand is causing disaggregation

DEMAND

1970s
MAINFRAME
IBM

1980s
IT AWAKENING
SAP

1990s
INTERNET
Microsoft

2000s
MOBILE

2010s
IOT / AI
NETFLIX
slack
UBER

2020+
DIGITAL NATIVE

APP
DISAGGREGATION
TO MEET DEMAND

1
MONOLITHIC
BUSINESS APP

10
ENTERPRISE
APPS

$10^2$
DEPARTMENTAL
APPS

$10^3$
SAAS
APPS

$10^5$
PUBLIC / PRIVATE
APPS

$10^9$
SERVERLESS AND MICROSERVICES
Disaggregation leads to more endpoints

Everything is An Endpoint

- Functions
- APIs
- Data
- SaaS apps
- Legacy apps
- Devices
Networked interactions are everywhere

Bazillion SaaS services to use
  - The new shared library

Everything you make is useful to someone else
  - Serve, or be forgotten

Need for agility
  - Recompose, recompose, recompose
Networked interactions are everywhere...
Resilient communication between endpoints

- It isn’t easy

- The challenges include
  - Compensation
  - Transactions
  - Events
  - Circuit breakers
  - Discovery
  - Protocol handling
  - Various message formats

- These are all hard problems
All about data

- Many messages are exchanged between endpoints
- Typically schemas describe these messages
- Data binding converts these messages into types/values in programming languages
- Creates a lot of friction for application that exchange messages
Network interactions are slow

- Requires a lightweight thread implementation to maximize the usage of computing resources.
- Software unit of concurrency (thread) vs domain’s unit of concurrency (user, transaction, interaction).
- Inefficient to match the scale of domain’s unit of concurrency with kernel threads.
Ballerina

Microservices with Ballerina
Ballerina was born out of frustration

- With existing frameworks and languages
  - For the work we do with customer at WSO2
- They don’t provide right abstractions for the job
“A programming language for network-distributed applications”
“programming language”

- Not a DSL
- Full programming language with rich set of general purpose features
- General purpose features are optimized for “network distributed applications”
Almost all existing programming languages designed for a world where the normal case is to integrate components running on one machine.

In the cloud world, the normal way for a program to interact with its environment is over the network.
Not a systems-level programming language

Easy to write and modify is more important than squeezing the ultimate compute performance

Suitable for application programmers
Design Principles

- Works for Mort, Elvis and Einstein
- Familiar
- Pragmatic
  - not a research language
- Readability
- Design language together with platform
Demo
What makes Ballerina unique as a programming language
Two features that works together

- **Providing and consuming services**
  - first-class language concepts with their own syntax and semantics, not a library
  - inherently concurrent

- **Sequence diagrams**
  - graphical view of most fundamental aspect of the semantics of a network distributed application
  - sequence diagram view only possible because syntax and semantics of the language were designed to enable it
  - higher-level concurrency abstraction
  - message sends/receives are matched up at compile-time
Type system

● Ballerina has chosen pragmatic point on the typing spectrum
  ○ Too static leads to inflexibility and/or complexity
  ○ Too dynamic means unreliable and poor IDE experience
Designed to work well with data

- **Structural types**
  - Describes shape/structure of values
- **Semantic subtyping**
  - Types are just sets of values
- **Untagged unions**
  - Allows for choice of A or B
- **No type hierarchies**
- **Most similar to TypeScript**
Error handling

- Errors are normal part of network programming (one reason why "network transparent" does not work)
- Errors handled as part of normal control flow
- Error is a separate data type
- Possibility of errors described by type system in same way as rest of the language
- Leverages untagged unions
Concurrency

- Provides a lightweight thread implementation called “strands”

```ballerina
public function main(string... args) {
    future<Coordinates | error> portlandF = start getCoordinates("Portland,US");
    future<Coordinates | error> seattleF = start getCoordinates("Seattle,US");
    future<Coordinates | error> bostonF = start getCoordinates("Boston,US");

    Coordinates | error coords = wait bostonF | portlandF | seattleF;
}
```
Ballerina Platform
Ballerina platform

- Collection of tools that helps you to develop, build, test, run and deploy programs written in Ballerina.
  - A language specification
  - Compiler and runtime
  - A set of standard libraries
  - Ballerina tool — the build tool++
  - A centralized module management architecture
  - LSP implementation, IDE plugins
  - and many more...
Two compilers

- One that produces Java bytecode
  - Java interoperability is provided

- One that produces platform-specific binaries
  - Not production ready yet
Summary

- Ballerina is an attempt to build a modern application development programming language
  - For a future with lots of network endpoints
- Abstractions for providing and consuming services
- Type system is designed to make network data processing easier
- Framework to encourage more secure code
- Fully open source and developed openly
How to get involved

Learn more → http://ballerina.io

Open source → http://github.com/ballerina-platform/

Get support → Stack Overflow #ballerina tag

Contact me ;-) → sameera@wso2.com @sameerajayasoma