Roles: a viable alternative to Microservices and Monoliths

Septimal Mind Ltd

team@7mind.io

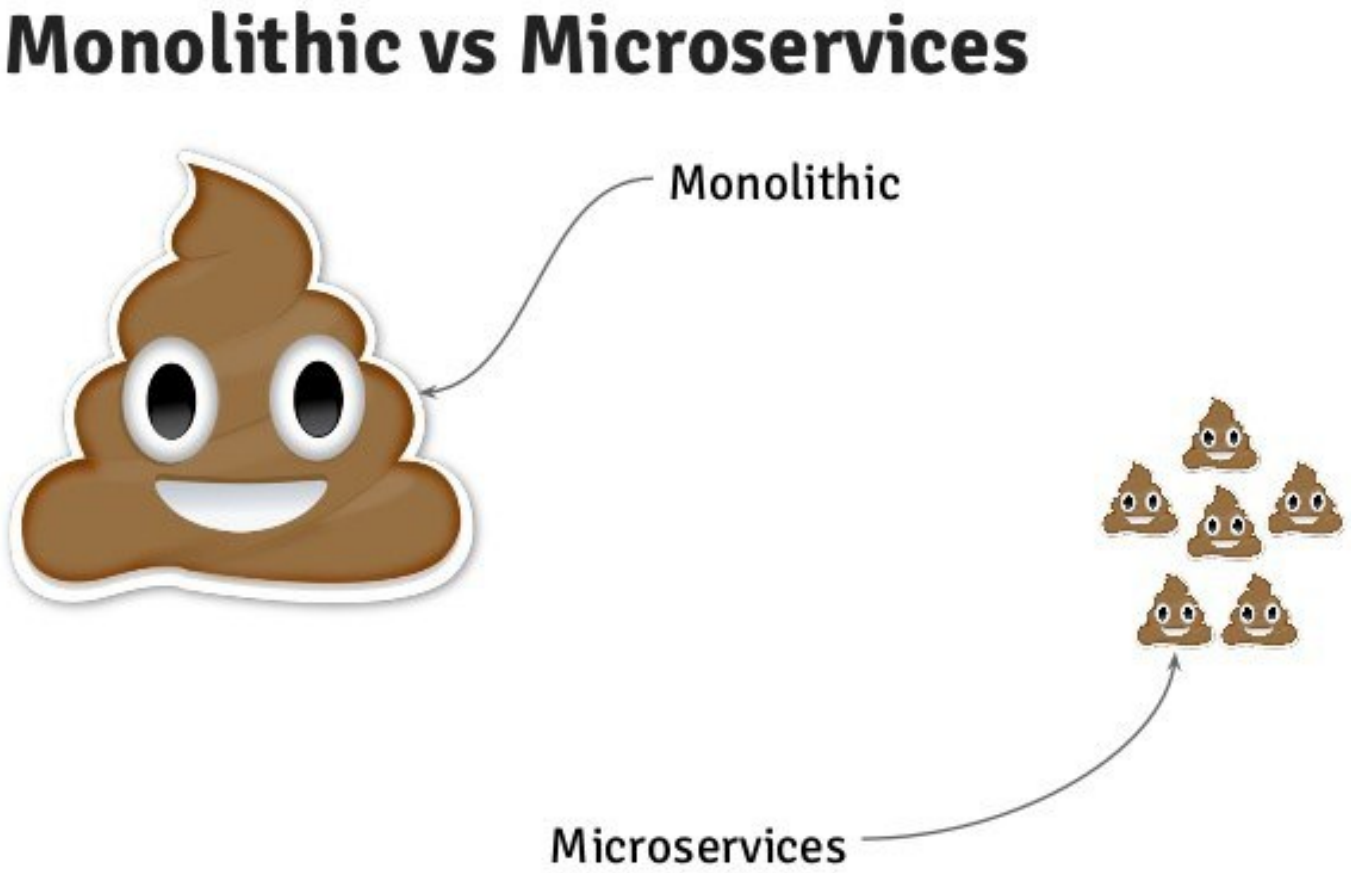
July 24, 2018

Monoliths are broken

We all know that Monoliths aren't nice:

- Inconvenient when your codebase is large: you may need to rebuild to much, etc,
- Provoking tight-coupling and non-SOLID code,
- Startup time is usually high,
- When you update you need to update everything (maybe not a bad things though),
- Issues with ,
- Some issues may be kinda mitigated by taking care of artifact structuring, pulling conservative parts out of the main codebase into independent libraries.

You've seen this before



The alternative – Microservices – is just sick

- Evil Dependencies and neccessary *Shared State*,
- Deployment and maintenance becomes more complicated with each new dependency,
- Inevitably Distributed: all *fundamental* problems are in place. CAP theorem?,
- Cascade failures,
- Accidental Incompatibilities: you may notice them too late. Amplified by cascade failures,
- Computing Density: 50 JVM microservices in Docker on a single machine? Really?,
- Overcomplicated Development Flows: simple (integration[1]) test requires many things done in right order,
- Time, interference and reproducible builds: isolated environments are expensive to setup.

[1] Not a constructive definition. Subject of another presentation

Observations on Microservices

- *Software components* are still in place – in form of Services. All the potential pitfails of component design apply to microservices,
- Many checks which may be done by your compiler in case of monolithic design have to be performed by integration test suites (in case you have them, it's hard to write),
- Many tasks are delegated to operations: in case of a monolith your dependency graph is being processed by your DI framework, in case of microservices it is. . . in worst case processed by people, in best case requires an orchestration tool. But we don't have any single great tool yet,
- An orchestration tool does *exactly the same job* your DI framework may do. And lot more things.

Roles: the idea to rescue

What we need?

- Something with all the positive traits of Microservices and Monoliths but without negative.

So:

- Let's run multiple services. . . in one service. Well, one process,
- Let's choose which *roles* we wish to assign to a before we start it,
- It resembles Containers and OSGi,
- Containers have some disadvantages. They provide you a lot so we get used to think that they are sluggish on startup. Isolated classloaders are inconvenient. Dynamic DI is a nightmare,
- Let's get rid of Dynamic DI. And let's have isolation optional.

Benefits

Missing things and potential pitfails

- Startup time. Do you remember Tomcat? You have to pay attention to your design – and there are some recepies[1],
- Heterogenous systems: you can't fit C#, Scala and Go into a single process. Though you still can greatly improve heterogenous systems by using roles,
- Dependency convergence. In case you want roles to be completely independent you need isolated classloaders. Solved in OSGi. Expensive, inconvenient.

[1] We have something to add to well-known answers. Subject of another presentation

DISTAGE

Next-gen Dependency Injection Framework for Scala

Generative, Modular, R✪les and Garbage Collection included

Quick overview

Status and things to do

Distage is:

- ready to use,
- in real production for 4 months.

Our plans:

- Make Roles opensource. ASAP,
- Support optional isolated classloaders (in foreseeable future),
- Check our GitHub: https://github.com/pshirshov/izumi-r2.

DIStage is just a part of our stack

We have a vision backed by our tools:

- Idealingua: transport and codec agnostic GRPC alternative with rich modeling language,
- LogStage: zero-cost logging framework,
- *Fusional Programming and Design* guidelines. We love both FP and OOP,
- *Continous Delivery* guidelines for Role-based process,
- *Percept-Plan-Execute* Generative Programming approach, abstract machine and computational model. Addresses Project Planning (see Operations Research). Examples: orchestration, build systems.

Altogether these things already allowed us to significantly reduce development costs and delivery time for our client.

More slides to follow.

Thank you for attention

https://izumi.7mind.io/distage/

We're looking for clients, contributors, adopters and colleagues ;)

About the author:

- coding for 18 years, 10 years of hands-on commercial engineering experience,
- has been leading a cluster orchestration team in Yandex, "the Russian Google",
- implemented "*Interstellar Spaceship*" – an orchestration solution to manage 50K+ physical machines across 6 datacenters,
- Owns an Irish R&D company, https://7mind.io,
- Contacts: team@7mind.io,
- Github: https://github.com/pshirshov