The Serverless Trilemma

Syed Zarak Farid, 390194

Talal Zahid, 390182

Tuan Anh Bui, 338816

What is serverless computing ?

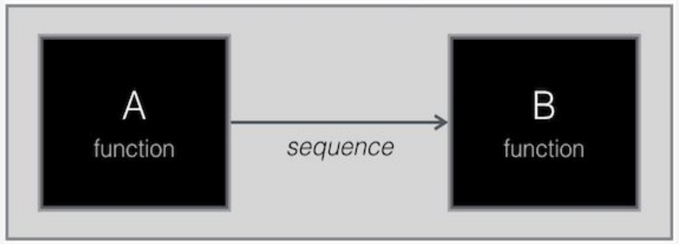
- Build and run applications without thinking about servers
- Functions as a Service
    - Micro-services are offered as separated “actions” or “functions”.
    - One function generates an output (example JSON) that acts as input to any other function.
- Event-driven invocations
    - The function should invoked based on events.
    - For example: When a function build completes, it “triggers” the other function(s).
- Function composition
    - Rather than create a single monolithic function, it is often desirable to separate the concerns of schema alignment and notification.

The problem

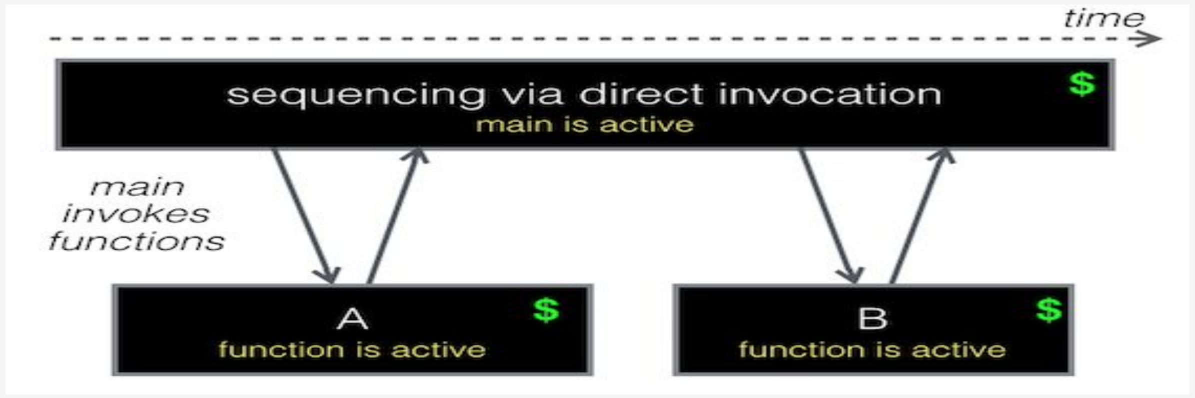
- Composition-as-functions must violate at least one of the 3 constraints:
    - Functions should be considered as black boxes.
    - Function composition should obey a substitution principle with respect to synchronous invocation.
    - Invocations should not be double-billed.

The Serverless Trilemma – I

- This desired sequential combinator as then i.e. a.then(b).then(c) => c (b (a()))



- Composition by Reflective Action Invocation (Double Billing Constraint).



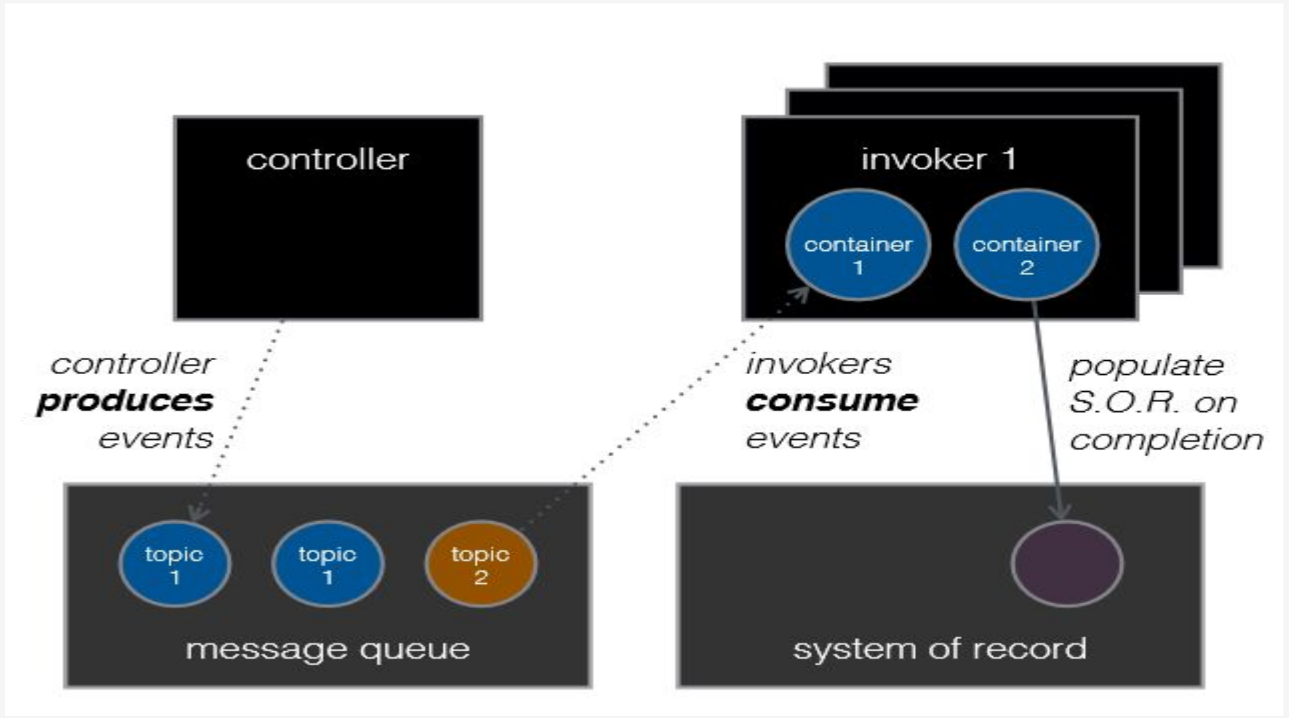
Reference: https://medium.com/openwhisk/composing-functions-into-applications-70d3200d0fac/

The Serverless Trilemma - II

- Composition by Fusion of Actions (Black-Box/Polyglot Constraint)
    - To avoid the double billing, we can infuse all functions in one source code.
    - Challenges: The source to every action is available, and in the same language
- Interlude: The Serverless Substitution Principle
    - Compositions-as-actions conform to the the JSON in, JSON out protocol of actions
    - Implies a single entry-single exit structure
    - Replace it with async/await pattern
- Client-Side Scheduling (Abnegation)
    - since it runs on the Client side scheduler not implemented as an action
    - There is no black box, no double billing
    - Satisfies substitution
    - The approach doesn’t work always

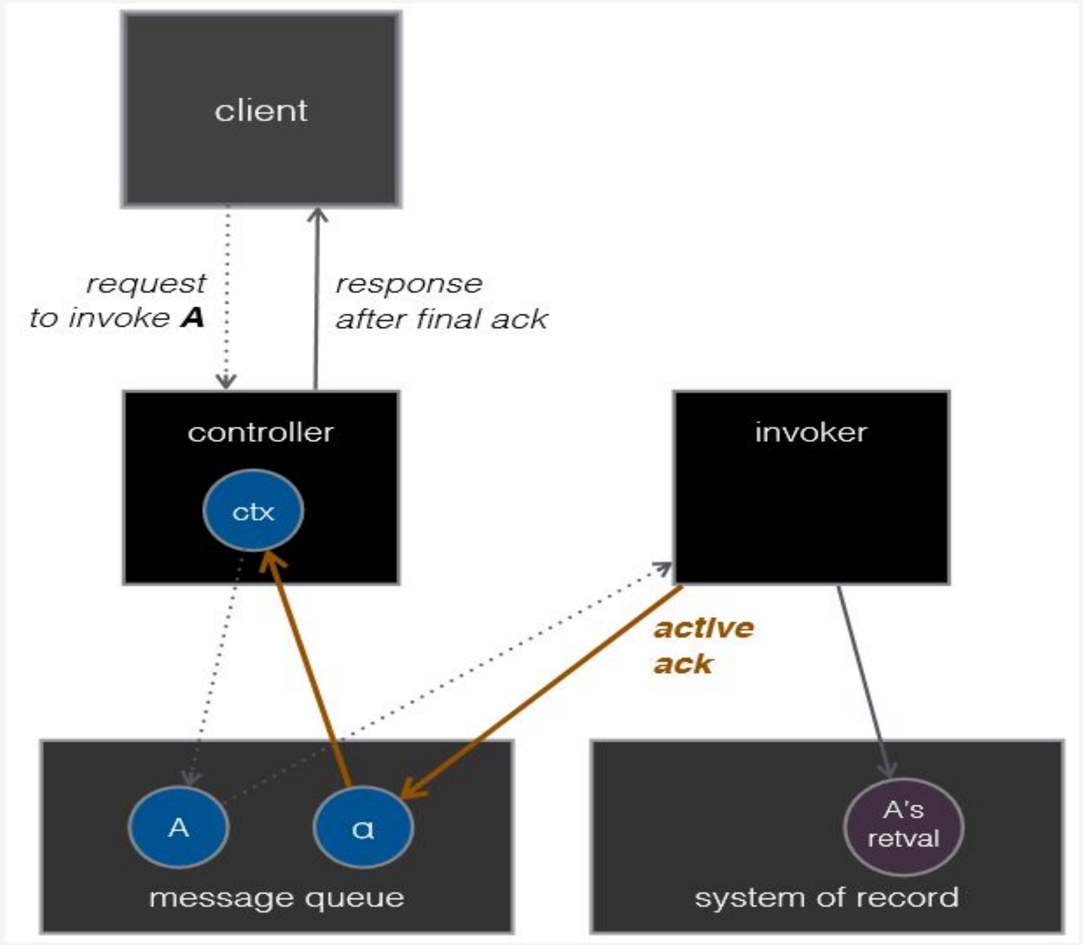
Trilemma-safe Sequential Composition - Overview

- Serverless core must offer more than actions, rules, and triggers to satisfy all the three constraints



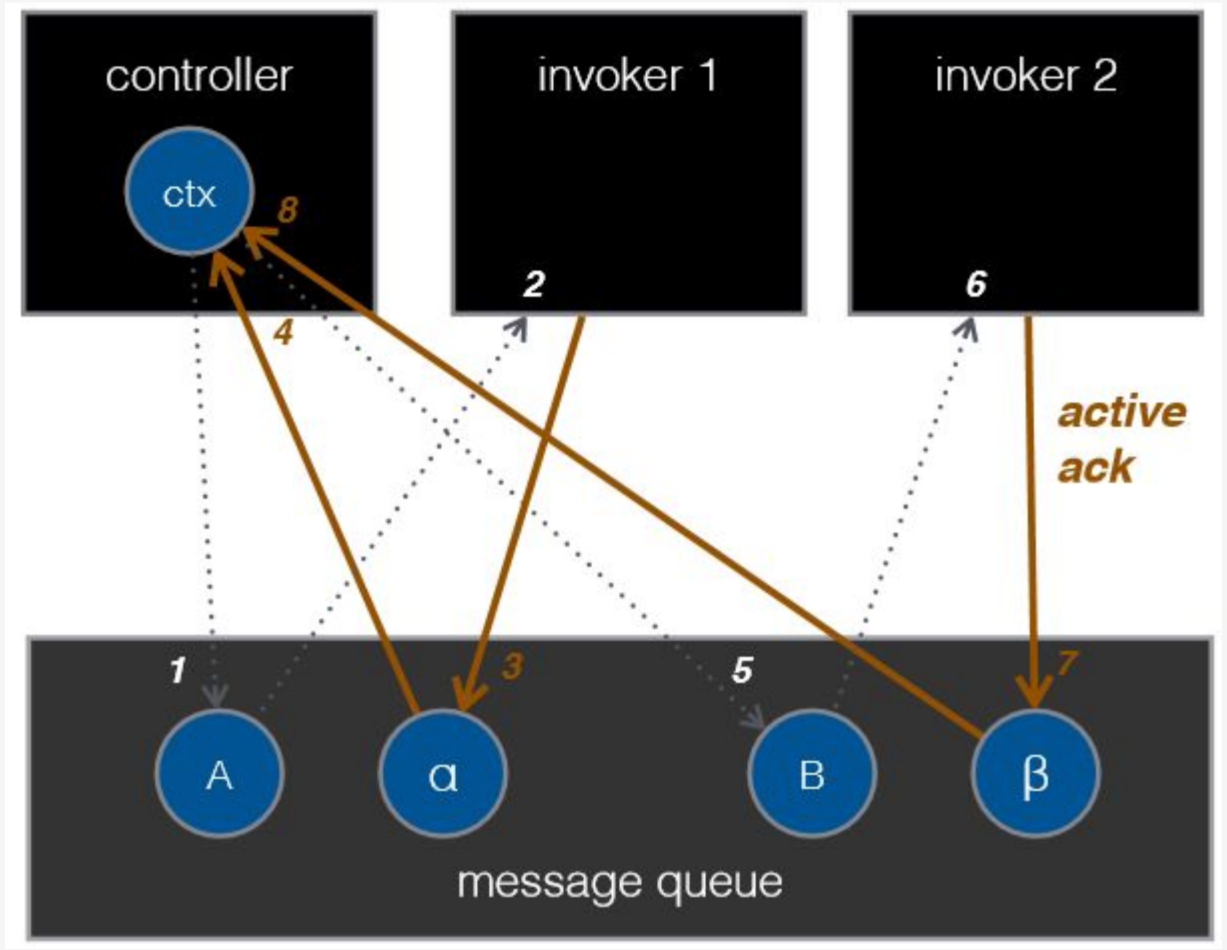
[1]

Trilemma-safe Sequential Composition – I



[1]

Trilemma-safe Sequential Composition – II

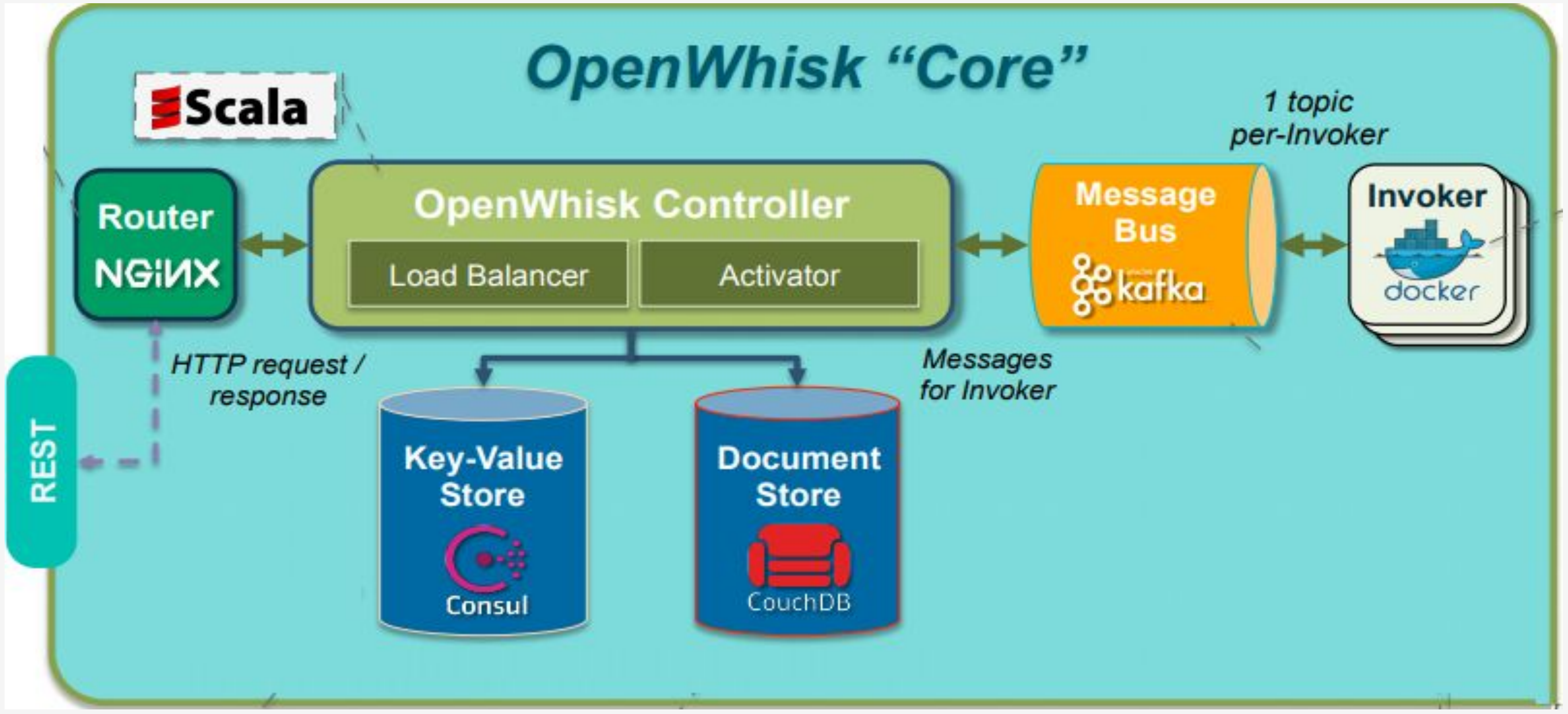


[1]

Apache OpenWhisk – Introduction

- FaaS platform to execute code in response to events
- Available as Open Source via Apache Incubation
- Supports a variety of languages, such as JavaScript, Python, Java, and Swift
- Supports function compositions

OpenWhisk Architecture – Detailed View

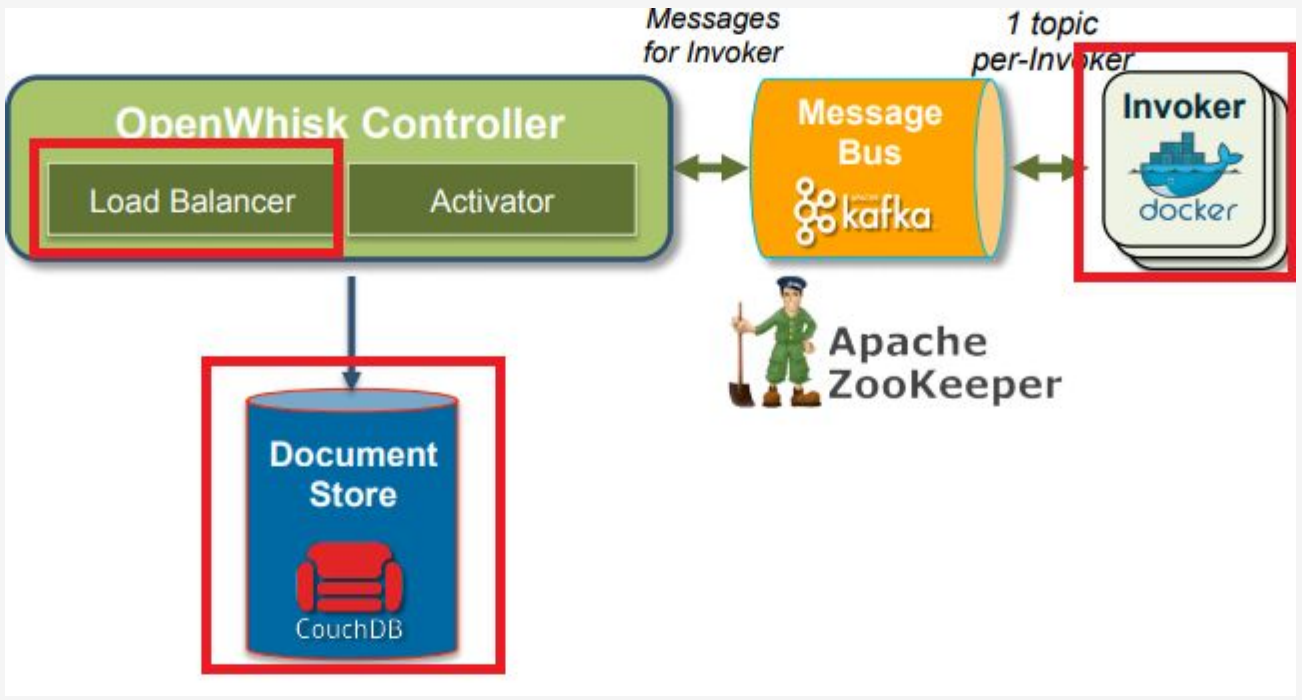


Reference: https://akrabat.com/wp-content/uploads/2019-01-20-ServerlessDaysCardiff-IntroducingOpenWhisk.pdf/

Which changes are necessary to coordinate function compositions that span multiple clusters (e.g., one near the Edge and one in the Cloud)?

OpenWhisk Edge Node Function Composition Solution

- Invokers or functions will be present on the edge node and cloud node depending on the functionality required.
- Load balancer decides which invoker to invoke.
- Finer grain invocation rules for the load balancer based on the requirements.



Reference: https://akrabat.com/wp-content/uploads/2019-01-20-ServerlessDaysCardiff-IntroducingOpenWhisk.pdf/

References

[1] Ioana Baldini, Perry Cheng, Stephen J. Fink, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Philippe Suter, and Olivier Tardieu. 2017. The serverless trilemma: function composition for serverless computing. In Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2017). ACM, New York, NY, USA, 89-103. DOI: https://doi.org/10.1145/3133850.3133855
[2] Vincent Hou, Software Engineer, IBM Cloud OpenTechnologies
[3] https://www.oreilly.com/library/view/learning-apache-openwhisk/9781492046158/ch01.html
[4] http://faculty.washington.edu/wlloyd/courses/tcss562_s2018/talks/G3-The_Serverless_Trilemma.pdf

Questions?