The Gap between Serverless Research and Real-world Systems

Qingyuan Liu[1,2], Dong Du[1,2], Yubin Xia[1,2,3], Ping Zhang[4], Habio Chen[1,2]

[1] Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University
[2] Engineering Research Center for Domain-specific Operating Systems (MoE)
[3] Shanghai AI Laboratory
[4] Huawei Cloud

11/28/2023 TCSS 562 Presentation
Review by Lucas Lu, Yexuan Gao, Christopher Henderson

UNIVERSITY of WASHINGTON

Talk Outline

- Paper overview and background
- Summary, key contributions and conclusions
- Critique
- Q & A

Introduction: Paper overview #1

- Discrepancy between research works and real-world systems
  - Research works are often based on oversimplified assumptions that hide real-world issues
  - Why would this happen?
  - Example: how to reduce cold start latency
- FunctionGraph (Lambda), Huawei Cloud (AWS)
- Knative, K8s

Introduction: Paper overview #2

- Challenges of Serverless Computing
  - Asynchronous Start
  - Declarative Tax
  - Scheduling Cost
  - Balancing Scheduling Policies
  - Costs of Sidecar

Introduction: Paper overview #3

- Why is it important ?
  - The paper identified challenges that represent real-world obstacles and limitations
- Why it's of interest to solve?
  - Solving these challenges aligns with the broader goals of advancing cloud-native computing

Background / Related Work #1

- Zhipeng Jia and Emmett Witchel. 2021. Nightcore: Efficient and Scalable Serverless Computing for Latency-Sensitive, Interactive Microservices.
  - Nightcore: a FaaS runtime
  - Aim to optimize instance initialization
  - Fail to address other overheads including scheduling costs, and communication costs

Background / Related Work #2

- Tian Zhang, Dong Xie, Feifei Li, and Ryan Stutsman. 2019. Narrowing the Gap Between Serverless and its State with Storage Functions. (Outside the paper)

"Data shipping problem" : Overheads associated with data movement

⇒

Shredder: allow to compute on durable data at its location of record

Limitations:
Relies on JavaScript

Challenge I: Asynchronous Start

Background

- Synchronous starts
  - Most existing works
- Asynchronous starts
  - Focus of this paper

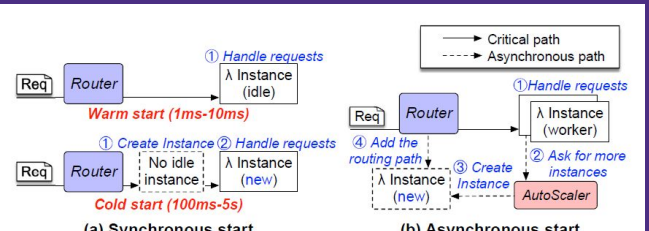


Figure 1: Synchronous start v.s. asynchronous start.

Challenge I Asynchronous Start
Gaps & Challenges

- Initialization latency is magnified by factors like
  - Queue size
  - Execution time
  - Arrival rate of incoming requests
- Challenges balancing initiation with queuing latency
- Essential to consider
  - Proactive auto-scaling policies
  - Queue design

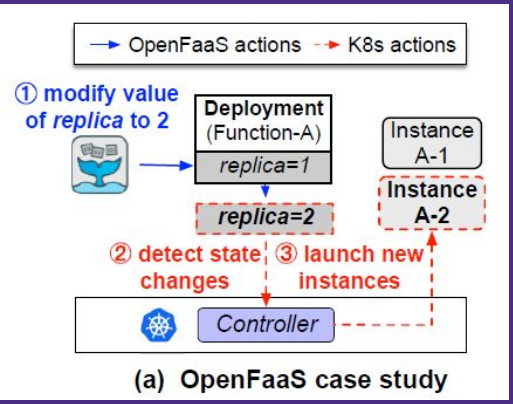
Challenge I Asynchronous Start
Opportunities & Suggestions

- Important to rethink design choices in
  - Scheduling
  - Routing
  - Queuing
- Minor adjustments in design can lead to significant performance improvements
- Late binding with centralized queue can halve tail-latency

Challenge II: Declarative Tax

Background

- The declarative approach of K8s
  - Pros and cons
- OpenFaaS case study



Challenge II Declarative Tax Gaps & Challenges

- Declarative methods used by K8s requires extensive communication and synchronization
  - Causes non-trivial overheads
- Challenging to provide deterministic performance guarantees
- Difficult to program the controller
- Bottleneck in low or real-time latency applications

Challenge II Declarative Tax Opportunities & Suggestions

- Optimize low-level infrastructure system design, balancing performance with an easy-to-use interface
- Explore:
  - Speed up synchronization via API server/etcd
  - Adjust queuing mechanisms within controllers to reduce latency variation
- Ensure modularity in optimizing multiple components

Challenge III: Scheduling Cost

Background

- The scheduling cost is part of cold start latency

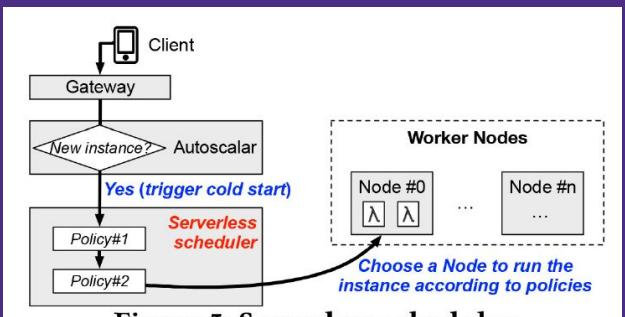


Figure 5: Serverless scheduler.

Challenge III Scheduling Cost Gaps & Challenges

- Scheduling costs are critical in large-scale clusters
  - Can be ~100x higher than start-up overhead
- Existing scheduling policies
  - Have complex calculations that are infeasible for real-world platforms
  - Fail to address large-scale implications

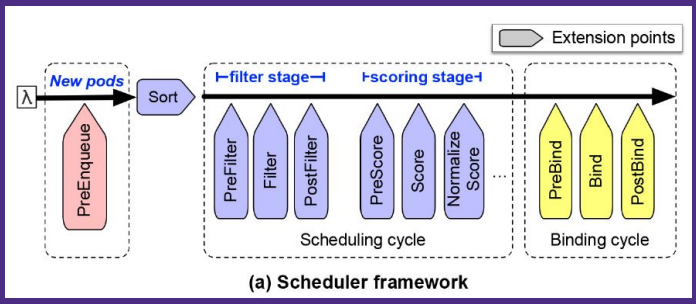
Challenge III Scheduling Cost Opportunities & Suggestions

- Focus on
  - Designing scalable scheduling policies
  - Optimizing the scalability of scheduling systems
- Try parallel binding and other mechanisms to eliminate unscalable designs in K8s
- Utilize simulation tools like Kwok for large-scale cluster testing to model virtual nodes and Pods

Challenge IV: Balancing Scheduling Policies

Background

- Supports for multiple policies by serverless schedulers



(a) Scheduler framework

Challenge IV Scheduling Policies Gaps & Challenges

- Platforms use multiple scheduling plugins & policies
  - Are challenging to balance effectively
  - Can conflict and/or interfere
- Current methods (e.g. plugin weights) can lead to suboptimal scheduling
- Lack of systematic approaches to analyze and balance multiple policies

Challenge IV Scheduling Policies Opportunities & Suggestions

- Collaborative efforts from industry and academia to
  - Design effective balancing mechanisms
  - Share data traces
- Comprehensively define "optimal scheduling"
- Suggests using reinforcement learning to dynamically adjust weights of plugins

Challenge V: Costs of Sidecar

Background

- Benefits of Sidecar
  - Additional features
  - Modify functionality
  - Facilitates dynamic CPU resource allocation
  - Modular design

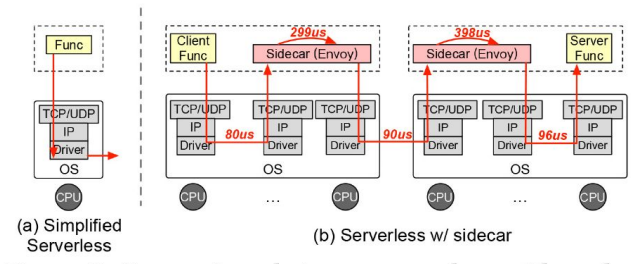


Figure 8: Comparison between serverless with and without sidecars. *We label the latency for an emoji-vote application using Istio/Envoy as our sidecar. Client and server functions are running in the same machine in this case.*

Challenge V Costs of Sidecar: Gaps & Challenges

- Sidecars incur resource and performance overheads
- Challenges include
  - Designing system software and hardware to effectively support sidecar systems
  - Sidecars introduce complexities not considered in many existing research works

Challenge V Costs of Sidecar Opportunities & Suggestions

- Explore new software & hardware for
  - Offloading sidecar logic
  - Supporting serverless systems with sidecars
- Decouple processing logic of sidecars and share across multiple Pods
- Avoid using utilizing the wrapper of the function
  - compiled with the function code at deployment
  - worse modularity, no proxying and queuing

Key Contributions

- **Insight I (Asynchronous Start)**
  - Novel designs for systems: scheduling, queuing systems, and etc.
- **Insight II (Declarative Tax)**
  - New mechanism to optimize the costs of the declarative approach
- **Insight III (Scheduling Cost)**
  - Consider the costs associated with scheduling decisions
- **Insight IV (Balancing Scheduling Policies)**
  - Balance multiple scheduler policies
- **Insight V (Costs of Sidecar)**
  - Design efficient and lightweight sidecar containers

Author's Conclusions

- **The authors believe their**
  - **Observations**
  - **Identification of challenges**
  - **Proposed opportunities**
  - **Will address gap between research and industry**
  - **Create momentum for improving serverless platforms**

Critique: Strengths

Case studies and concrete examples for identified issues

Asynchronous vs. Synchronous

- End-to-end latency in *Knative* with per-instance queueing design modelled as: $Ld + T(1-\sigma)/\sigma$

Non-trivial scheduling costs in large clusters

- ***2000+*** concurrent pods cluster introduces ***~14.5s*** scheduling cost
- ***~100x*** than start-up overheads in large cluster

Sidecar container costs

- ***0.3~1*** **vCPU** and ***300~800 MB*** memory usage are observed
- Latency increased by ***9.5x*** to ***49.8x*** under different requests per second (RPS)

Critique: Strengths

- **Effective Use of Figures**
  - **Demonstration of architectures and mechanisms**
  - **Improve problem understanding**

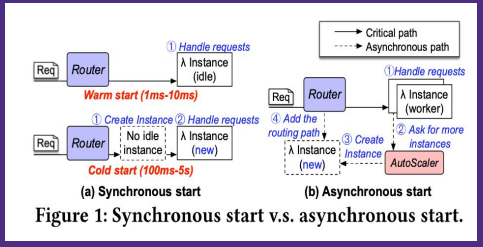


Figure 1: Synchronous start v.s. asynchronous start.

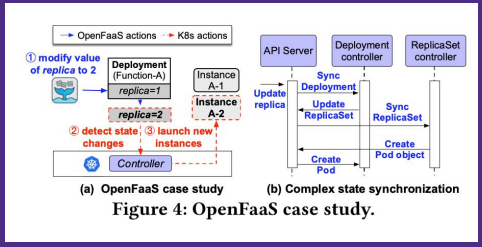


Figure 4: OpenFaaS case study.

Critique: Weaknesses

- ***Complexity in Reading:***
  - Require prior domain-specific knowledge (technologies & research)
- ***Few grammar mistakes***
  - Misuse of articles
  - Misuse of common phrases
- ***Limited Coverage in Solutions:***
  - Example: optimizing sidecar creation and management; control the ratio between sidecar containers and function containers

Critique: Evaluation

- ***Lack of Empirical Data***

"The challenges and insights presented in this paper are drawn mainly from our experience of applying research optimizations to real-world serverless systems" (pp.476)

Declarative Approach of K8s costs identified → No statistical evidence supporting the claim → How significant is its impact ?

Sidecar Container Cost identified → More memory cost in low RPS → Measured under what number of request per second (RPS)?

Remaining Gaps

- Discrepancies between research and industry in other areas such as "*Security*" in serverless systems are not discussed
- Evaluation on the feasibility of the authors' suggestions leads to future research

Questions

A break for questions.