UNIVERSITY of WASHINGTON

**Paper:** A Serverless Publish/Subscribe System.

**Authors:** N. Pezhman and HA Jacobsen.

**Journal/Institution:** arXiv/ Cornell University

**Reviewers:** Jasleen Kaur and Naman Bhaia

0

## Talk Outline

- Introduction
- Background/Related Work
- Summary of New Approach and Technology
- Experimental Evaluation
- Authors' Conclusion
- Critique: Strengths
- Critique: Weaknesses
- Critique: Paper Evaluation
- Identify Gaps

1

1

## Introduction: Paper Overview

- **Problem Statement:** Applicability of the serverless paradigm to scalable stateful distributed pub/sub systems.
- **Why is it a problem?** Serverless platforms are stateless and do not persist the state of the functions across multiple executions.

2

2

## Introduction: Paper Overview

- **Hypothesis:** The serverless platforms can use the storage services offered by cloud providers to persist the state of the serverless applications.
- **Goal:** To build a scalable, stateful publish/subscribe serverless architecture.

3

3

## Introduction: Publish/Subscribe Systems

- A pub/sub system consists of three primary parts:
  - **Subscribers:** Express their interest in the specific data type to the broker.
  - **Publishers:** Broadcast publications containing some data to the broker.
  - **Brokers:** Receive the publications, use a matching scheme to match the publication with the subscriptions and forward the publications to the appropriate subscribers.

4

4

## Background/ Related Work

- **Papers:**
  - **Serverless video processing system** (Fouladi et al.) for executing cloud functions in parallel to edit, transform and encode videos.
  - **Chatbot based on a serverless platform** (Yan et al.).
  - **Serverless approach for encapsulating and deploying web components** (Ast et al.).
- **Learning**: Serverless architecture for running tasks in parallel
  - Decreases the latency,
  - Decreases development effort,
  - Maintains scalability and extensibility.
- **Takeaway**: To link multiple publications with subscribers in parallel, cloud-based resources can be beneficial.

5

5

## Background/ Related Work

- **Paper: Modular system to maintain and execute serverless microservices** called Stafu (Spillne).
- **Learning:** Integrating stateful services such as object stores and file storage helps overcome the limitation of serverless functions being stateless.
- **Takeaway:** To preserve state across multiple sessions, necessary state data can be stored in cloud-based databases (DBaaS).

6

6

## Background/ Related Work

- **Paper: Serverless pub/sub broker** that performs content-based and topic-based matching (Nasirifard et al.)
- **Learning:**
  - Topic based matching
  - Content based matching
- **Takeaway:** The author has extended the paper above and proposed a novel Function based matching.

7

7

## Summary of New Approach

- A serverless pub/sub system that performs **topic-based**, **content-based**, and **function-based** matching.
- Function-based matching is an evolution of Content Based Matching.
  - Publisher provides data along with a function type.
  - Subscribers register function types along with the source code of the mentioned functions.

8

8

## Summary of New Approach: Matching Algorithms

- **Topic-Based:** Compares categories/subjects/tags.
  - Publication: ($publicationData$, [$topic_1$,$topic_2$, …, $topic_n$])
  - Subscription: ([$topic_1$, …, $topic_m$])
- **Content-Based:** Compares values shared by the publisher against constraints defined by the subscriber.
  - Publication: ([$key_1$ : $value_1$, …, $key_m$ : $value_m$])
  - Subscription: ([$key_1$ : constraint$_1$, …, $key_m$ : $constraint_m$])
- **Function-Based:** Compares function type and function output.
  - Publication: ($publicationData$,$matchingFunctionType$)
  - Subscription: ($matchingFunctionType$, $Function$)

9

9

## Summary of New Approach: Key Contributions

- Improve matching algorithm performance by caching.
  - Serverless functions maintain an ephemeral cache.
  - The proposal entails caching subscriber IDs and their subscriptions.
  - Cache Hit: Refer to FaaS cache.
  - Cache Miss: Refer to persistent DB .
- Propose a novel function-based matching scheme.
- Evaluate performance and latency of proposed system.

10

10

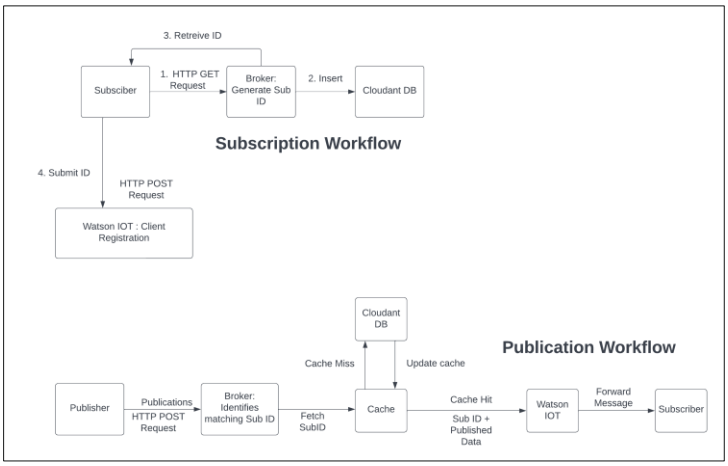
## Summary of New Technology: IBM Bluemix

- **IBM Cloud Functions** (Serverless FaaS platform)
  - Event based serverless programming model based on Apache OpenWhisk.
  - Rest API provides the event (Trigger) to execute stateless functions (Actions) based on pre-defined Rules.
  - Perform the matching methodologies between publications and subscribers.
- **IBM Cloudant** (NoSQL database)
  - DBaaS that stores and queries data as JSON objects
  - Used to persist the application's state
- **IBM Watson IoT** (communications platform)
  - Used to deliver the publications to the subscribers.

11

11

## Summary of New Technology: Workflow








## Experimental Evaluation

- Implement a serverless pub/sub system with a broker making use of different matching algorithms.
- At the Broker, for each matching algorithm:
  - Maintain 2 databases
    - 1. Records subscriber ID versus matching parameter*
    - 2. Records matching parameter* versus subscriber IDs.
  - Performs matching against cache first
  - Query data from Databases only in the case of cache miss.

* Here, matching parameter can be a topic, content or function type.

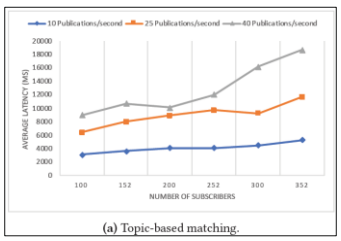
## Experimental Evaluation

- Conduct a distributed experiment to measure the latency of different matching algorithms
  - Measured at the subscriber.
  - Measured against increasing number of subscribers and publications.
- For each iteration,
  - Fixed number of publications per second from 1 publisher
  - Fixed number of subscribers evenly distributed among four virtual machines
  - Size of published data = 1KB
  - 1 Broker implementing all three matching algorithms

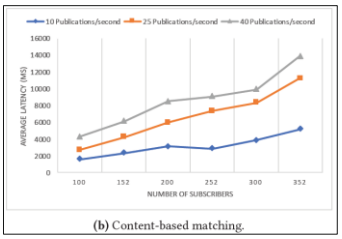
14



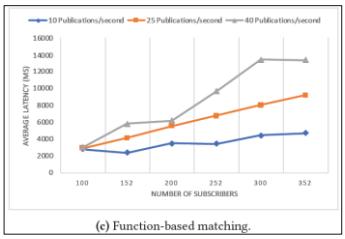
## Experimental Evaluation: Results




(a) Topic-based matching.




(b) Content-based matching.




(c) Function-based matching.

- The serverless broker scaled up resources to accommodate increased workload.
- Latency increased with workload.

15

## Authors' Conclusions

- **Advantages:**
  - Reduced overhead of operational activities related to the broker.
  - Systems are cost effective as you are billed only for up-time.
- **Limitations:**
  - Unavailability of local testing environments.
  - Limited logging and the inability to filter logs based on action or message type in case of failures.
  - Cloudant DB:
    - Limits the number of lookups per second and becomes a bottle neck.
    - Cache misses are frequent and unpredictable.

16

16

## Critique: Strengths

- Serverless technology is **easy to learn** and therefore transition to.
- The **cost of the model is low** as IBM Bluemix provides pay-as-you-go-plan
- The **performance of the system is good** as the latency in passing messages to subscribes increases with workload
- The author's experiments surmise that **system is scalable** since the Broker FaaS scales with workload.
- Since the subscription information is redundantly stored in the DB and FaaS cache, the **proposed system is Fault tolerant**.

17

17

## Critique: Weaknesses

- The proposed system is **affected by the limitation of serverless** applications:
  - **Stateless**
  - **Allowed usage**. Serverless service providers limit the number of hits/queries which then act as a bottleneck.
- Even though the authors have tried to cache data and make it redundant, **caching is not robust** and persistent enough to support a large ecosystem.

18

18

## Critique: Paper Evaluation

- **Strengths:**
  - The authors have provided the source code to allow reproduction of results.
  - Explained serverless computing concepts before diving into its applications.
  - Potential future work was explained well.
- **Weaknesses:**
  - Grammatical Mistake: Section 3 – First paragraph.
  - Images are too far from the text referring to them.
  - The structure of Section 4 can be improved. All subscribers are mentioned in one heading (4.3.2) but each publisher has a different heading(4.3.3-4.3.5).
  - The paper does not provide the results to substantiate that the FaaS scales up with the workload.

19

19

## Identify Gaps

- Author does not compare novel broker algorithm to existing ones in terms of performance or theoretical advantages.
- Without examples for the following databases, it is hard to imagine the implementation of the paper.
  - Content-based databases and how the constraints are evaluated.
  - Function-based databases and the relation between function type and source code.
- The logical next step for the paper would be to identify ways to make the cache more predictable/reliable.





UNIVERSITY of WASHINGTON

# Questions?