Demo Abstract: A Serverless Topic-Based and Content-Based Pub/Sub Broker

Pezhman Nasirifard
Technical University of Munich
pezhman.nasirifard@in.tum.de

Aleksander Slominski
IBM Research
aslom@us.ibm.com

Vinod Muthusamy
IBM Research
vmuthus@us.ibm.com

Vatche Ishakian
Bentley University
vishakian@bentley.edu

Hans-Arno Jacobsen
Middleware Systems Research Group
jacobsen@in.tum.de

Abstract

Building scalable, highly available publish/subscribe (pub/sub) systems can require sophisticated algorithms and a tremendous amount of engineering effort. This paper demonstrates a way to build a pub/sub broker on top of the OpenWhisk serverless platform that performs topic-based and content-based matching. This approach radically simplifies the design and significantly reduces the amount of code while still achieving scalability targets. Furthermore, we present a publisher/subscriber client application to interact with the broker as well as an evaluator application that enforces heavy workload on the broker to measure the scalability and latency of the pub/sub system and discover the potential bottlenecks.

CCS Concepts • **Software and its engineering → Publish-subscribe / event-based architectures**; **Cloud computing**;

Keywords Serverless, Function as a service(FaaS), OpenWhisk, topic-based pub/sub, content-based pub/sub

1 Introduction

Implementation of scalable production-grade server applications, such as pub/sub brokers, demands significant development effort and resources to guarantee stability, performance and fault tolerance. Recently, all primary cloud vendors start offering serverless computing resources to host and maintain scalable server applications. The serverless paradigm describes a programming model where the developer deploys a microservice, also known as a function, to the cloud, where the cloud provider takes responsibility for managing the resource for executing the functions. While a serverless platform provides powerful scalability and availability properties, building a pub/sub system on such a platform is not a simple matter of packaging existing pub/sub components as serverless functions. There are platform constraints, such as the stateless and short-lived nature of serverless functions, that necessitate additional components and careful design. For example, in this paper serverless functions are used to compute the matching logic, but these are augmented by a managed database service to store subscription state, and an IoT service to deliver notifications. Although integrating additional services can compensate for the serverless limitations, the restrictions on the scalability of the services can be a potential bottleneck to the scalability of the whole serverless architecture. In this demo, we present a scalable serverless topic-based and content-based pub/sub broker based on IBM OpenWhisk [1]. We also provide publisher/subscriber and evaluator applications to interact with the broker and evaluate its capabilities and realize its limitations.

Middleware Posters and Demos '17, December 11–15, 2017, Las Vegas, NV, USA

ACM ISBN 978-1-4503-5201-7/17/12.
https://doi.org/10.1145/3155016.3155024

2 Related Work

The serverless paradigm is rather new, and studies on building pub/sub systems based on serverless platforms are rare [4]. However, various serverless use cases have been explored to inspect different aspects of serverless environments. Fouladi et al. [5] present a low latency serverless high-definition video processing system and Yan et al. [6] demonstrate a chatbot with a serverless backend. They both argue that utilizing serverless recourses significantly reduces the development efforts while maintaining the scalability and reliability of the system.

3 The Serverless Broker's Overview

We break down and encapsulate the functional and nonfunctional requirements of the broker into subsystems which we can realize and deploy as OpenWhisk functions. A pub/sub system consists of three primary components, including brokers, publishers, and subscribers. The pub/sub broker requires persisting the state of subscriptions to perform the matching. However, since the OpenWhisk functions are inherently stateless, we integrate the IBM Cloudant NoSQL

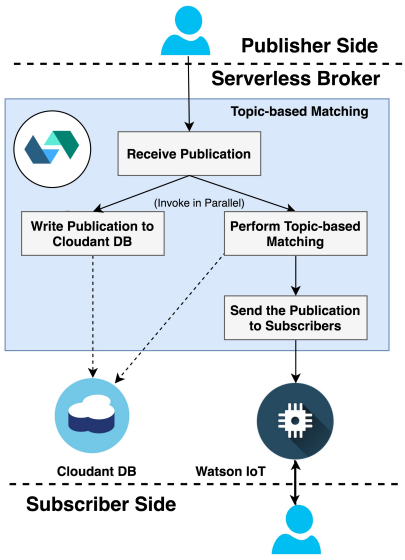

Figure 1. The topic-based matching workflow.

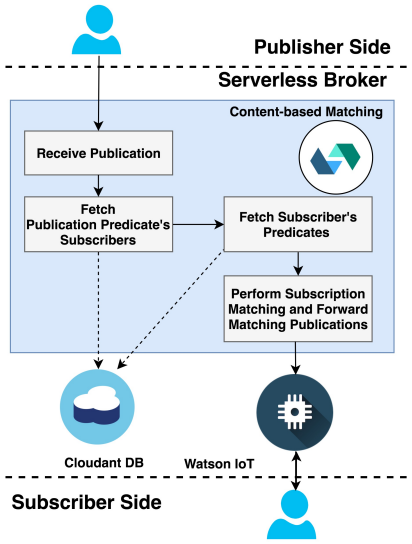

Figure 2. The content-based matching workflow.

database [2], an auto scalable Database-as-a-Service to maintain the state of the system.

To perform the matching, we chain several functions which are activated upon receiving publications, as illustrated in Figures 1 and 2 for topic-based and content-based matching respectively. In both scenarios, the publisher client dispatches a publication to the broker. Next, the function retrieves the subscriptions from Cloudant DB and passes the publications and subscriptions to an additional function as an input to perform the matching. Finally, the broker sends the matching subscribers to the Watson IoT Platform [3], which forwards the publication to the subscriber client using a real-time bi-directional communication channel.

4 Software Demonstration

To demonstrate the capabilities of the serverless broker, we provide a publisher/subscriber desktop application to interact with the broker. The application enables the users to unsubscribe/subscribe to subscriptions. Users can also make use of the publisher client to issue publications and observe the delivery of matching publications on the subscriber client in real-time.

Moreover, to evaluate the scalability and extensibility of the broker, we offer an evaluator application that creates heavy workloads for the broker by running up to 1000 subscriber instances simultaneously and issuing up to 1000 publications with 100 ms delay between each publication. Meanwhile, the application measures the latency of the broker and provides the user with evaluation result which helps with detecting the potential bottlenecks of the system.

The evaluator application shows that the serverless broker scales in response to the increasing workload of subscribers and publications. However, the vendor-specific constraints on OpenWhisk and Cloudant DB, which limits the number of parallel function invocations and database interactions per second, are bottlenecks to the scalability and enforce a threshold to the number of subscribers running simultaneously as well as the number of issued publications.

5 Conclusions

In this demo, we present an automatically scalable serverless approach to a pub/sub broker based on IBM OpenWhisk which is capable of performing complex topic-based and content-based matching. Furthermore, we provide a publisher/subscriber applications to interact with the broker as well as an evaluator application for measuring the performance and discovering the bottlenecks of the design.

References

[1] 2017. IBM Bluemix OpenWhisk. https://azure.microsoft.com. (August 2017). Accessed: 2017-08-22.

[2] 2017. IBM Cloudant DB. https://www.ibm.com/analytics/us/en/technology/cloud-data-services/cloudant/. (August 2017). Accessed: 2017-08-10.

[3] 2017. IBM Watson Internet of Things (IoT). https://www.ibm.com/internet-of-things/. (August 2017). Accessed: 2017-08-11.

[4] Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. 2017. Serverless Programming (Function as a Service). In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*.

[5] Sadjad Fouladi, Riad S. Wahby, Brennan Shacklett, Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. 2017. Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*.

[6] Mengting Yan, Paul Castro, Perry Cheng, and Vatche Ishakian. 2016. Building a Chatbot with Serverless Computing. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs (MOTA '16)*.