Cold Start Aware Agentic Video Search on Serverless Platforms

[Samarth Bhatia, Shreesh Kumar Jha]
MSc Computer Science [Politecnico Di Milano]

November 20, 2025

1 Introduction

Function as a Service (FaaS) platforms provide elastic scaling and simplified deployment for cloud applications, but they suffer from cold starts that can significantly increase latency, especially for interactive workloads. In parallel, agentic AI applications, in which several specialised components collaborate to complete a task, are increasingly used for media analysis and retrieval.

The existing *VideoSearcher* toy application used in the course already illustrates a basic video analysis pipeline, yet it is not structured as an agentic workflow and has not been evaluated in a serverless setting with respect to cold starts and configuration choices. Implementing a fully custom autoscaler inside OpenFaaS or deploying a dedicated VLLM service would be unrealistic within a 2.5 week time frame. Instead, this project focuses on a more feasible but still meaningful goal: refactoring VideoSearcher into an agentic multi stage pipeline, deploying it on OpenFaaS, and systematically studying how different cold start related configuration settings affect end to end latency.

The central problem addressed in this project is: how can we use existing OpenFaaS configuration mechanisms to obtain "cold start aware" deployments for an agentic video search application, and what are the performance and cost trade offs of different configuration regimes for such an application?

2 Related work

Previous work on serverless computing has analysed cold start overheads, resource allocation and autoscaling strategies for FaaS platforms, often using microbenchmarks or simple functions to characterise latency and scalability. These studies show that cold starts can dominate tail latency when functions frequently scale to zero, and that parameters such as minimum replicas and scale down timeouts have a strong impact on responsiveness.

At the same time, there is growing interest in workflow style and agentic applications that decompose complex tasks into multiple coordinated components, for example for image and video analysis. However, there is relatively little experimental work that combines these perspectives in a concrete application and focuses on what can be achieved purely by tuning existing FaaS configuration knobs rather than designing new autoscaling mechanisms. This project aims to contribute a focused case study in this direction using the VideoSearcher application and the OpenFaaS platform introduced in the course. A small, simplified experiment on AWS Lambda may be included as an optional comparison if time permits.

3 Research questions

The project is guided by the following research questions:

- RQ1: How can the VideoSearcher toy application be refactored into an agentic pipeline composed of independent stages suitable for deployment on OpenFaaS?
- RQ2: How do different OpenFaaS configuration settings related to scaling (e.g. minimum replicas, maximum replicas, and scale-to-zero timeouts) affect end to end latency and cold start frequency for the agentic VideoSearcher workload?
- RQ3: What trade offs between latency and approximate cost can be observed across these configuration regimes, and what practical guidelines emerge for designing similar agentic applications on serverless platforms?

The primary scope of the project is a detailed performance and configuration study on OpenFaaS. A full custom autoscaler integrated with Kubernetes APIs, a dedicated VLLM deployment, and a complete two platform replication of the entire pipeline are explicitly out of scope. If time allows, a small single function or reduced pipeline experiment on AWS Lambda may be added as a qualitative comparison.

4 Methodology

First, the existing VideoSearcher application will be analysed and decomposed into several logical stages, for example audio extraction, video clipping or frame extraction, feature processing, and frame explanation. An orchestrator component will be implemented to coordinate these stages as an agentic workflow, either sequentially or with limited parallelism. For the "explain frame" stage, the project will either use a lightweight hosted language model API or a synthetic CPU intensive task that mimics the latency profile of an AI component, thereby avoiding the complexity of deploying VLLM.

Each stage will then be packaged as an independent function and containerised using Docker. These functions will be deployed on OpenFaaS, using the course infrastructure when available. The orchestrator will trigger the functions and record request identifiers and timing information to allow end to end latency measurements.

Cold start aware behaviour will be explored by configuring multiple OpenFaaS deployment regimes. At least two to three configurations will be defined, varying parameters such as the minimum number of replicas per function, the maximum replicas, and the scale-to-zero timeout. Rather than implementing a new autoscaler, the project will rely on OpenFaaS' existing scaling mechanisms and configuration interfaces, possibly assisted by simple scripts that modify configuration files and redeploy functions.

Synthetic workloads of video search requests will be generated using Python scripts, covering low steady load, bursty arrivals, and moderate continuous load. For each configuration, the experiment will record end to end response latency and observable indicators of cold starts (e.g. increased latency for first requests after idle periods). Basic resource usage metrics and invocation counts will be collected where possible, and approximate cost estimates will be derived using published pricing models or reasonable assumptions. If time permits, a simplified version of one pipeline stage will be deployed on AWS Lambda to obtain a small, illustrative comparison.

5 Time planning

A detailed Gantt chart and week by week planning will be prepared in consultation with the supervisor. At a high level, the first part of the 2.5 week period will focus on refactoring VideoSearcher and deploying the agentic pipeline on OpenFaaS; the middle part will be devoted to defining configuration regimes, generating workloads and running experiments; and the final part will be reserved for data analysis and writing the report.

6 Tools and datasets used

The project will build directly on the course infrastructure and the provided VideoSearcher toy application. The main tools and platforms are:

- **OpenFaaS**: for container based serverless deployment and configuration of scaling related parameters such as minimum and maximum replicas and scale-to-zero behaviour.
- **Docker**: for building and managing container images for each function stage in the agentic pipeline.
- **Programming languages and libraries**: primarily Python for the orchestrator, workload generation and data analysis, together with the languages and libraries already used in VideoSearcher.
- **Language model or synthetic agent**: either a small hosted language model API to provide textual explanations, or a synthetic CPU intensive computation that emulates the latency characteristics of an AI based explanation component without requiring GPU resources.
- **Optional AWS Lambda**: if time allows, a reduced version of one or two pipeline stages may be deployed on AWS Lambda to gather a small amount of comparative data.

As input data, the project will use small video clips suitable for experimentation and already available in the course environment, or open datasets of short videos that can be processed within the practical time and resource constraints. No sensitive or personal data will be used.