Scheduling Serverless Workflows

Professor: Ana Klimovic (aklimovic@ethz.ch)
Postdocs: Dmitrii Ustiugov (dmitrii.ustiugov@inf.ethz.ch)
PhD student: Lazar Cvetković (lazar.cvetkovic@inf.ethz.ch)

1. About the Project

Serverless computing platforms, in particular FaaS offerings, enable users to launch thousands of fine-grained, short-lived tasks (functions) in parallel, to run interactive jobs with near real-time performance. In addition to individual functions, one can deploy and invoke functions organized into directed acyclic graphs (workflows) that are becoming increasingly popular and used in practice (e.g., AWS Step Functions). Once the entry function of the workflow has been triggered, the system automatically orchestrates the execution of the function chain following the rules of data flow machines, i.e. successors of graph's nodes get invoked as soon as their arguments are available. Related work has just scratched the surface of serverless workflows in the context of scheduling, orchestration, programming model, performance, and fault tolerance.

Academic research on the topic of serverless workflows is hindered by the lack of infrastructure. Currently, no open-source research serverless platform has built-in support for workflow orchestration. The related work has invested a lot of effort in porting applications to serverless various workflow orchestration systems (2.1.a). However, their representativeness is questionable as they were not initially designed for the serverless workload. Also, such ported systems have not managed to achieve efficiency in practice and still suffer from tens of milliseconds of scheduling latency, poor exploitation of data locality (key to solving serverless workflow challenges), unpredictable tail latency, and increased network bandwidth utilization. Additionally, there is a lack of a representative benchmark for serverless workflows. Related work has mostly ported or synthesized benchmark suites that were used only for a single evaluation, thereby making the comparison of evaluation sections between different publications not trivial.

The goal of this project is to add a workflow orchestration feature to Knative. In our current understanding, only the Activator component of the Knative framework, which serves as a buffer for incoming requests and as a proxy for all the bidirectional communication with function executors (instances), requires modification. Additionally, to bring Knative support for workflows, the Knative service specification format (YAML-based) has to be extended to support directed-acyclic graphs, as well as the database for storing the actual cluster state. To evaluate the implemented system, benchmark functions should be synthesized (possibly with some dummy work in the function body) according to the information contained across different real-world traces that have been published so far (e.g., DAG width, DAG depth, node input/output arity, number of nodes in the graph, size of function arguments).

2. Overview of Initial Steps

1. Theoretical introduction and overview
    a. FaaSFlow paper
    b. ORION paper
    c. Pheromone paper
    d. Kubernetes docs
    e. Knative docs
    f. Knative reference - tips
2. Create an account on Cloudlab and try setting up a vHive multinode cluster
    a. Cloudlab
    b. vHive Github Repository
    c. vHive Quick Start Guide
3. Try running some experiments on a vHive cluster
    a. You will find simple workloads with instructions on how to run them in the quick start guide (see 2.2.b)
4. Start implementing requested features in Knative

Final Deliverables:

- *Project report* describing the motivation, background, design, implementation, evaluation, related work, conclusions and future directions.
- *Source code* and documentation of the system implementation and experiments.
- *Presentation* summarizing the project outcome.

3. Project meetings

- We will have weekly meetings to discuss ideas and progress on the project.

4. Administrative Aspects

- You should create an Element/Matrix account with your ETH credentials. We use this system for easier communication within the Systems Group, without using email.

Remote Converged Snapshot Storage for Serverless Clouds

Project Proposal
Autumn 2022

Serverless computing, a.k.a. Function-as-a-Service (FaaS), has emerged as the next generation cloud computing paradigm. With FaaS, cloud application developers can structure the business logic of their applications as a set of functions connected in a workflow. At the same time, cloud providers, such as Amazon Web Services and Azure, take complete responsibility for managing cloud resources allocated to the functions, by adjusting the number of active function instances at any time based on the observed invocation traffic for each function. This labor separation improves programmability and reduces time-to-market for application developers, however raising significant system design challenges for cloud providers. In particular, functions often exhibit high response time in the presence of highly dynamic invocation traffic to cloud applications, due to the necessity to spawn additional function instances at real time.

Prior work [1, 2] has shown that high response time is related to the so-called **cold-start** delays, appearing when spawning new instances. The results obtained from benchmarking the leading cloud providers [2] demonstrate that these systems are bound by data movement. One of the biggest sources of these overheads is moving the initialization state (e.g., container images) to the datacenter node chosen to host a new instance. In the state-of-the-art FaaS platform called vHive, the initialization state of a function instance comprises a container disk image and a MicroVM snapshot, in which the complete state (i.e., the memory and architectural state) of an active function instance is captured. Such state of all deployed functions is stored in remote storage (further referred as **remote snapshot storage**), efficient access to which is essential to minimize the start-up time of function instances.

This project's goals are to obtain and analyze the breakdown of the function instance start-up latency in vHive and explore the design of an efficient remote snapshot storage subsystem. The first milestone is to evaluate the experimental support for remote MicroVM snapshots in vHive, understanding its latency and throughput characteristics. Measurements should be done for an unloaded system and in the presence of a high function load. The second step is to set up and evaluate the stargz snapshotter [4, 5] for container disk images, which stores images as sets of filesystem chunks retrieved on demand, and compare it to a conventional snapshotter that only supports full disk image retrievals. After that, we will explore the design of a Remote Converged Snapshot Storage (RCSS) that contains both the disk and MicroVM images, exploring its architecture, state management policies, and performance/cost trade-offs. The final step will be a detailed quantitative comparison with the baseline vHive system.

Work Plan

- Get familiar with the vHive architecture and the experimental support for remote MicroVM storage.
- Set up and evaluate the stargz snapshotter in vHive system.
- Devise the architecture of RCSS, containing the whole initialization state of functions deployed at a scale of a small FaaS cluster.
- Prototype RCSS in vHive and quantitatively study several data management policies.
- Contribute the RCSS code to the vHive serverless experimentation ecosystem.

Deliverables:

- *Project report* describing the motivation, background, design, implementation, evaluation, related work, conclusions and future directions.
- *Source code* and documentation of the system implementation and experiments.
- *Presentation* summarizing the project outcome.

Mentor Contacts:
Prof. Ana Klimovic <aklimovic@ethz.ch>
Dmitrii Ustiugov <dmitrii.ustiugov@inf.ethz.ch >

References

[1] Benchmarking, Analysis, and Optimization of Serverless Function Snapshots.
[2] Analyzing Tail Latency in Serverless Clouds with STeLLAR.
[3] Gigabytes in milliseconds: Bringing container support to AWS Lambda without adding latency
[4] https://github.com/containerd/stargz-snapshotter
[5] Startup Containers in Lightning Speed with Lazy Image Distribution on Containerd

Machine Learning Training on Dynamic Datasets

Project Description
Autumn 2022

While ML researchers commonly use static datasets (e.g., ImageNet) to develop and train new models, training datasets in real production environments often evolve dynamically. For example, training recommendation systems on click stream data requires adapting to fresh data becoming available over time and deleting data after a certain period of time to comply with data retention constraints. To adapt to dynamic data, ML models need to be continually trained online and/or periodically retrained offline. However, continuous training of ML models has high cost and consumes significant energy. As state-of-the-art neural networks grow in size every year, the datasets and compute power they require for training is also increasing.

To enable ML models to adapt to dynamic data environments at low cost and energy, several questions arise: how often should we re-train models? Which data should we retrain on, to capture both recent and seasonal trends? While ML researchers have proposed techniques to identify important training examples (e.g., approaches such as coresets [1, 2], active learning [3], and data perishability [4]) and analyzed the importance of weighting examples for ML [5, 6, 7], the ML ecosystem lacks system infrastructure to efficiently store dynamic datasets and automate complex training workflows for users [8].

This project takes a first step in a longer-term vision of building *data importance aware* ML systems infrastructure. In particular, we focus on deriving the requirements for a dynamic training data storage system and continuous ML training platform. Our long-term goal is to jointly optimize model accuracy and the energy required for continual training to achieve the target accuracy, by building systems that can identify which training examples are most useful to train on. We also aim to optimize storage costs by storing training data elements in appropriate storage tiers according to their importance.

The first step in this vision — and the focus of this project — is to build an open-source framework for running experiments with a set of open-source, representative workloads (models+datasets) for ML training on dynamic datasets. The goal is to find 2-3 motivating use-cases in which training with subsets of an evolving, dynamic dataset significantly impacts accuracy and using basic metrics of data importance can help us either attain higher accuracy and/or optimize the energy required to attain a target accuracy. Application domains to explore include ML-based weather forecasting (e.g., MetNet, FourCastNet, GraphWeather), autonomous driving on Waymo public datasets, fake news detection (e.g. on Twitter or Reddit datasets), financial modeling on stock market data, system autotuning (e.g., learned indexes).

Work plan:

- Literature review on data importance metrics.
- Find and test open source models and datasets with dynamic nature.
- Find and explore open source platforms for continual ML and dynamic dataset storage.
- Prototype system with integration of basic data importance metrics.
- Show motivating use-cases for training with subsets of data with 2-3 different examples.

Deliverables:

- *Project report* describing the motivation, background, design, implementation, evaluation, related work, conclusions and future directions.
- *Source code* and documentation of the system implementation and experiments.
- *Presentation* summarizing the project outcome.

Mentor Contacts:
Prof. Ana Klimovic <aklimovic@ethz.ch>
Foteini Strati <foteini.strati@inf.ethz.ch>

References

[1] Coresets for Data-efficient Training of Machine Learning Models. Mirzasoleiman, Bilmes, and Leskovec. ICML 2020. Also see talk: https://www.youtube.com/watch?v=Bmn8SNojTIA
[2] Coresets via Bilevel Optimization for Continual Learning and Streaming. Borsos Z., Mutný M., Krause A., NeurIPS, 2020.
[3] From Theories to Queries: Active Learning in Practice, Burr Settles, 2010.
[4] Time and the Value of Data. Valavi E., Hestness J., Ardalani N., Iansiti M., Harvard Business School, 2020.
[5] Not All Samples Are Created Equal: Deep Learning with Importance Sampling. Katharopoulos and Fleuret. ICML 2018.
[6] What is the Effect of Importance Weighting in Deep Learning? Byrd and Lipton. ICML 2019.
[7] Understanding the role of Importance Weighting for Deep Learning. Xu et al. ICLR 2021.
[8] Continual Learning in Practice. Diethe T., Borchert T., Thereska E., Balle B., Lawrence N. NeurIPS, 2018.