Predictive Scheduling for Functions-as-a-Service

Saurav Chhatrapati, Vincent Lyau, Ray Pan

Background

Serverless computing has gained significant attention recently with the growing popularity of auto-scaling Function-as-a-Service (FaaS) systems. Hydro is a stateful FaaS platform that focuses on function composition, direct communication, and low-latency access to shared mutable state. Function scheduling in Hydro currently focuses only on data locality while avoiding overutilization of individual executors. We are interested in improving scheduling to reduce latency for Hydro workloads. Workloads are anticipated to be composed of functions that will be frequently executed with varying input parameters, and we aim to predict runtime characteristics through regression models. These predictions allow us to implement the HEFT algorithm, an algorithm for scheduling DAGs of functions given expected runtimes. Combined, this results in our Predictive FaaS Scheduler (PFS).

Implementation

Scheduler

- Each scheduler uses a policy that contains the regression models and executes the HEFT algorithm to obtain an improved scheduling and function-to-executor assignment.
- Upon receiving execution metadata from the executor, the scheduler informs the policy, which periodically updates the models.
- If the policy lacks information about a function, a default heuristic-based policy is run.

Executor

- Each executor executes functions when triggered. Upon completion, execution metadata is sent to the scheduler.

Regression Models

- These encode the recorded execution metadata and make predictions upon future function executions.

Architecture

After functions are executed by the executors, function execution metadata including runtime, input sizes, and outputs sizes are sent to the policy models.

HEFT

HEFT (Heterogeneous-Earliest-Finish-Time) is a scheduling algorithm for a bounded number of heterogeneous executors. Like many scheduling algorithms for DAGs of functions, HEFT relies upon a knowledge of the runtime of individual nodes within the DAG prior to scheduling across the executors. The algorithm computes the upward rank for all nodes, a measure of the critical path between each node and the exit node. HEFT then aims to insert each function into the earliest idle time slot on each executor with regard to ready time, defined as the time when an executor has the input data available for the function. Functions are in this way iteratively assigned into time slots between other tasks.

Benchmarks

PFS is well-suited for DAGs with many parallel function calls, and outperforms the default Hydro scheduler. It is most effective when there is significant variation in the runtime of functions at the same DAG level. There is room for improvement when dealing with sequential graphs composed of functions that access the same dataset.

Conclusion

- Inter-scheduler communication for sharing and updating model
- Improvements to metadata handling
- Model persistence
- Fetching key sizes efficiently
- Metadata transmission between executor and policy
- Experiment with variations of HEFT such as look-ahead HEFT
- Compute costs with awareness of data location

Future Work

An overview of the Hydro architecture and how Predictive Scheduling for FaaS fits into the design.