Name - Abhay Singh Rathore
NetId - asr22015

Project Proposal - FaasFlow

Introduction

Serverless computing (also known as Function-as-a-Service, or FaaS) has become widely popular nowadays. Instead of renting virtual machines with varying specifications, users simply submit functions to the vendor, who then manages the scheduling and execution of those functions automatically and efficiently. Some serverless applications still rely on simple functions, but more advanced applications are utilizing fine-grained functions to deliver complex functionality by linking them into workflows. These workflows run based on the user-defined logic and control the flow of the functions. They often involve parallel branches and have dependencies on data. The major cloud providers implements the serverless workflow systems typically utilize a centralized workflow engine on the master node to control the execution state of the workflow and distribute function tasks to worker nodes. This scheduling pattern is known as the "master-side workflow schedule pattern" (MasterSP) because the central workflow engine in the master node makes the decision to trigger function tasks.

Problem with the Existing System

The MasterSP approach does not align well with the event-driven nature of serverless computing. MasterSP results in poor performance for serverless workflows due to high scheduling and data movement overhead. On one hand, the central workflow engine is responsible for dynamically managing and scheduling all functions, leading to frequent transfer of function execution states between the master node and worker nodes, causing significant scheduling overhead. As functions are typically short, this transfer occurs frequently. On the other hand, the engine distributes triggered functions to worker nodes for load balancing, and cloud vendors have limits on the input and output data size of functions to prevent excessive network bandwidth usage. In real-world serverless platforms, users often have to use additional database storage services for temporary data storage and delivery, leading to significant data movement overhead.

Proposed Solution

To make serverless workflow execution more efficient, we propose a "worker-side workflow schedule pattern" (WorkerSP) to reduce scheduling overhead. In WorkerSP, the master node only handles workflow graph partitioning, while task scheduling is handled by each worker node. Each worker node's per-worker workflow engine independently determines if it can trigger a local function based on the execution state of its predecessor functions received from other nodes. To alleviate the expensive data movement overhead of remote storage, we also use FaaStore, an adaptive storage library that allows functions on the same worker node to communicate directly through shared main memory instead of remote storage.