Carbon Aware Serverless System - Proposal and Timeline

Jaykumar Patel
The University of Texas at Austin
patel.jay4802@utexas.edu
JNP2369

Afnan Mir
The University of Texas at Austin
afnanmir@utexas.edu

Nidhi Dubagunta
The University of Texas at Austin
nidhi.dubagunta@utexas.edu

Abstract

Serverless computing enables developers to build and deploy applications faster without having to worry about provisioning resources and managing infrastructure. Existing serverless computing platforms allocate resources such as vCPU and memory without considering carbon footprint. Other policies, such as the keep-alive policy, are also carbon-agnostic. We propose a system that will manage resource allocation and the keep-alive policy to meet SLO while minimizing carbon footprint. Firstly, we define how to measure carbon emission and perform a measurement study. Then, we develop a serverless framework employing machine learning algorithms to drive carbon-aware decisions for the resource allocation and keep-alive policy. Further work includes creating a framework for scheduling and dynamic voltage and frequency scaling (DVFS) to minimize carbon footprint.

1 Timeline

- **Week 1:** Set up OpenWhisk, which is a serverless and open source cloud platform. This will allow us to deploy and invoke serverless functions.
- **Weeks 2 and 3:** Research and implement a measurement tool to measure the carbon footprint of cloud computing. This will be used to measure the carbon footprint of serverless function invocations.
- **Week 4:** Develop and run a measurement study. This will invole running a variety of workloads on serverless functions and measuring the carbon footprint of each invocation along with resource usage such as CPU and memory.
- **Week 6 and onwards:** The following weeks will be spent on developing a carbon aware serverless system. The system will involve the following components, in order of priority:
    1. **Resource Allocation:** Develop a resource allocation algorithm that minimizes the carbon footprint of serverless function invocations.
    2. **Keep-Alive Policy:** Develop a keep-alive policy that keeps serverless functions alive for a certain amount of time to reduce the carbon footprint of cold starts.
    3. **Scheduling:** Develop a algorithm that schedules serverless function invocations to minimize the carbon footprint.
    4. **Dynamic Voltage and Frequency Scaling:** Develop a DVFS algorithm that minimizes the carbon footprint of serverless function invocations.