OpenWhisk Experiment Documentation

Release 0.1

Hyungro Lee, Geoffrey Fox

Oct 08, 2018

Contents

Lightweight dynamic applications on cloud computing have been migrated to serverless computing platforms, such as Apache OpenWhisk, recently due to its elasticity and simplicity of compute resource provisioning. Mobile applications, image data processing and system log analysis are implemented as a stateless function to process given tasks on serverless computing environments. For example, less than 300 lines of a Python function with TensorFlow library identifies a thousand of training image data from ImageNet Large Visual Recognition Challenge in 2012 within a few seconds using serverless concurrent function invocations. Functions for heavy workloads with external libraries, however, would not be loaded on the serverless environments due to its size and its long execution time. Applications for big data, deep learning, computer vision and genomics typically require multiple and complicated libraries along with cpu and data intensive tasks.

We have three objectives to achieve during this experiment:

- Sucessful deployment of Apache OpenWhisk on public clouds to comare its behavior with a series of functions for big data, deep learning and genomics applications
- Benchmark regarding to concurrent function invocations on IaaS-powered serverless computing environments
- Practical guide to building services on public clouds with the experience of deploying Apache OpenWhisk.

Target cloud providers are:

- Oracle Cloud Infrastructure
- Amazon EC2

Target serverless platform is:

- Apache OpenWhisk

CHAPTER 1

Development Schedule

This page shows a list of items to achieve during this experiment with the tentative date. Changes may occur based on discussions.

- Schedule: 04/02/2018 - 05/04/2018 (1 month)

1.1 Installation & Configuration

Multi-node installation is desired with DevOps tools to demonstrate a repeatable cluster setup at any platform.

Two versions of OpenWhisk installation:

- **standalone deployment, Apr 9th, 2018**
  - Automation with Terraform (https://github.com/oracle/terraform-provider-oci)
  - Configuration by Ansible
- **Cluster setup with 3+ nodes, Apr 13th**
  - by kubernetes pods (https://github.com/apache/incubator-openwhisk-deploy-kube)

1.2 Applications

Three applications from different domains would be integrated to depict use cases.

- Big Data Benchmark from UC Berkely AMPLab, Apr 13th
- Tensorflow ImageNet, MNIST, Apr 20th
- Genomics tool, Rail-RNA sequencing alignment, Apr 20th

1.3 Platform

Various server types will be tested between the selected platforms. The current setups are:

- AWS EC2 i3 vs OCI with object storage, Apr 20th

1.4 Practical Guide

The purpose of this section is to share lessons that we learned during this experiment regarding to software development on public clouds. Comparisons to development environments would be provided as well. For example, we demonstrate the differences of between using AWS botocore API and Oracle oci SDK regarding to virtual machine management e.g. start, stop, or terminate.

- API Usage Guide
- CLI Usage Guide
- Community Support
- Documentation, Apr 27th (above all)

1.5 Final Report

This experiment will generate a final report which includes information that we have collected and tested such as system configuration, devops scripts, experiment results and summary of what we found during this activity, focusing on software development and experiences.

- due on May 4th

CHAPTER 2

OpenWhisk Deployment

We anticipate to build a serverless computing environment using open source projects e.g. Apache OpenWhisk and Fn projects on public clouds. During this activity, we will learn how to build platform using public cloud services and understanding differences regarding to hardware performance, and level of supported development tools i.e. APIs and CLIs.

Target cloud providers to test are:

- Amazon EC2
- Oracle Cloud Infrastructure

2.1 Deployment Option

- single node
- cluster mode (3+ nodes)
- high-end performance server types regarding to CPU, Network and Memory

2.2 Integration

- block storage
- object storage
- 10G network
- SSD vs HDD local disk

2.3 Fn Project (Optional)

TBD

2.4 Kubeless (Optional)

TBD

CHAPTER 3

Benchmark

This section shows benchmark results among different cloud platforms.

The current platforms included are:

- Amazon EC2
- Oracle Cloud Infrastructure

3.1 Concurrent invocations

A thousand of small tasks are given on serverless computing environments to complete in parallel. CPU-intensive, a file I/O-intensive and heavy network traffic functions are designed to measure concurrent throughput on these cloud platforms.

3.2 CPU Intensive Function

Javascript function for Matrix multiplication and binary tree creation is implemented to stress CPU time.

3.3 Network Intensive Function

Downloading a large file from its object storage creates network traffic and network bandwidth for downloading will be displayed for concurrent executions.

3.4 File I/O Intensive Function

Intermittent files are frequently generated while a set of functions runs and writing and reading file speed is a measure to avoid overhead of processing logics in functions.

CHAPTER 4

Practical Guide

Poor management of cloud resources may cause unnecessary resource provisioning, misconfigured infrastructure and delays in troubleshooting of cloud components. Nowdays, however, cloud providers distribute several software development tools and APIs to enable proper handling of cloud resources by managing them automatic and programmable.

This section explains technical details of managing cloud resources using Python and REST APIs by walk-through of the serverless deployment experience. General comparisons will be provided at the end to give some idea of deploying applications on these cloud providers as a beginner. The differences and similarities are also demonstrated.

4.1 Account Authentication

Oracle User Credentials:

- IAM
- console password
- api signing key
- Swift password
- Amazon S3 Compatibility API Keys
- SMTP Credentials

AWS EC2 User Credentials:

- IAM
- Multi-Factor Authentication (MFA) for AWS websites
- Access Keys (id and secret key pairs)
- CloudFront key Pairs
- X.509 certificate

4.2 System Choice

Operating Systems

- Windows
- Linux distributions

Server Types

- general
- cpu intensive
- io intensive
- bare metal

4.3 Account Limit

AWS API Limits

- 10k requests/sec

AWS EC2 Limits

- 5 public ips/classic ec2 instance
- 5k key pairs

4.4 Supported Development Tools

Oracle:

- Python SDK: https://github.com/oracle/oci-python-sdk
- CLIs: http://API_URL/tools/Linux/install-IaaS-CLI.sh

AWS:

- Python SDK: https://github.com/boto/boto3
- CLIs: https://github.com/aws/aws-cli

4.5 Comparisons

TBD

4.6 Summary

TBD

CHAPTER 5

Summary

TBD