CSE 520S Course Projects - Cloud Tutorial

TA for class CSE 520S, Fall, 09/07/2016

Chong Li, Chao Wang
IaaS (Infrastructure as a Service)
- Real-Time Performance Measurement in
  - Amazon: Xen Virtual Machine and Docker Container
  - Google: Compute Engine and Container Engine

PaaS (Platform as a Service)
- Real-Time Streaming
  - Amazon: Kinesis
  - Google: Cloud DataFlow
- Internet-of-Things
- Real-Time Messaging Service
  - RabbitMQ, NSQ, ZeroMQ, Google Cloud Pub/Sub
Amazon IoT

- IoT in public cloud provides bi-directional communication between sensors, actuators, embedded devices, smart applications and cloud services

- Amazon IoT (https://aws.amazon.com/iot/)
  - APIs for frontend apps
  - Flexible protocols for message delivery
  - Message processing & storage in cloud
Amazon IoT

Demo

- Create a “thing” (like a message aggregator)

- Write a frontend application, using AWS IoT SDK
  - Leverage OpenSSL for authentication
  - Collect (sensor) data; encapsulate data into (mqtt) messages; send messages to the “thing”

- Create rules at the “thing”
  - Define message filter and how to process interesting messages
    - In our case, trigger Amazon SNS (Simple Notification Service)
Real-Time Messaging Service

- Messaging service provides
  - Message routing, delivery, persistency for applications
  - A typical communication paradigm: publish/subscribe
    - Publishers publish messages of different topics; middleware forwards messages of interest to corresponding subscribers

- RabbitMQ, NSQ, ZeroMQ, Google cloud pub/sub
- Try different configurations
  - Number of publishers/subscribers, number of topics, message size, sending rate, enable/disable persistency, number of brokers
  - Measure latency and locate system bottleneck
Real-Time Messaging Service

- Demo – using NSQ as messaging broker
  - Setup Golang environment
    - Golang:
      - C-style with garbage collection
      - Flexible and powerful concurrency model
      - Docker, Kubernetes, NSQ … (https://github.com/golang/go/wiki/GoUsers)
  - Install and run NSQ broker
  - Write your own publisher and subscriber, using NSQ client library
Pub/Sub Architecture

- Apps communicate using **publish/subscribe** mode
  - Publishers: publish msgs on topics
  - Subscribers: receive msgs on topics of interest

- Broker receives, queues and forwards messages for every application
  - NSQ broker is a daemon process, which can be **co-located** with applications or be deployed **independently**
Using the Container Engine

- Container Engine on the Google Cloud Platform (https://cloud.google.com/container-engine/)

- Monitor the cloud services via on-line console (https://console.cloud.google.com)

- Run projects using the cloud shell (click the icon on the top-right of the above page)

- Quickstart (https://cloud.google.com/container-engine/docs/quicksart)
Using the Cloud Dataflow

- Cloud Dataflow on the Google Cloud Platform (https://cloud.google.com/dataflow/)

- Run projects using the cloud shell
  - Locally: VM + Dataflow SDK
  - Remotely: Cloud storage + monitor, via on-line console

- Quickstart (https://cloud.google.com/dataflow/docs/quickstarts)
Pointers

- Amazon IoT
  - [http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html](http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html)

- Amazon SNS
  - [http://docs.aws.amazon.com/sns/latest/dg/welcome.html](http://docs.aws.amazon.com/sns/latest/dg/welcome.html)

- Resource list for course projects