CSE 520S Course Projects

TA for class CSE 520S, Fall, Sep/06/2017

Haoran Li, Chong Li, Chao Wang
A Quick Preview: Project Requirements

- **Build Up Your Own Cloud Service**
  - Run in public cloud, using open sourced component

- **Evaluation on Real-Time Feature**
  - Benchmark Application
  - Metrics: Latency + Other Features related to your service

- **Will grade based on**
  - project difficulty
  - quality and depth of work
  - workload distribution among team members (peer review)

- **Milestones:** proposal, demo1, demo2, final demo, report.

- **Start early! Discuss with us and Dr. Lu**
Cloud Computing

- Cloud computing provides shared pool of configurable computing resource to end users on demand

- Three service models
  - IaaS (Infrastructure as a Service): virtual machines, storage, network …
  - PaaS (Platform as a Service): execution runtime, middleware, web server, database, development tool …
  - SaaS (Software as a Service): email, virtual desktop, games …
Cloud Services: On-premise Software

- Traditional
  - installed and runs on personal computer
  
- You Manage and Deploy
  - Hardware
  - OS
  - Software

- Example
  - This presentation
Infrastructure as a Service (IaaS)

- **IaaS**
  - "physical server box"
  - Virtual Machine
    - Memory
    - Storage
    - CPU
    - Network

- **Example**
  - AWS EC2
  - AWS HPC

- **Usecase**
  - Build up your VM cluster
Platform as a Service (PaaS)

- **PaaS**
  - You get a framework
  - Host Application
  - Tools

- **Example**
  - AWS IoT

- **Use case**
  - Build up you’re smart A/C controller

---

<table>
<thead>
<tr>
<th>APP</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime</td>
<td>Middleware</td>
</tr>
<tr>
<td>OS</td>
<td>Virtualization</td>
</tr>
<tr>
<td>Server</td>
<td>Storage</td>
</tr>
<tr>
<td>Network</td>
<td></td>
</tr>
</tbody>
</table>
PaaS Example: Amazon IoT

Tools
- AWS IoT DEVICE SDK
  - Set of client libraries to connect, authenticate and exchange messages

Framework
- AWS IoT
  - AUTHENTICATION & AUTHORIZATION
    - Secure with mutual authentication and encryption
  - DEVICE GATEWAY
    - Communicate with devices via MQTT, WebSockets, and HTTP 1.1
  - DEVICE SHADOWS
    - Persistent device state during intermittent connections
  - REGISTRY
    - Assign a unique identity to each device
  - AWS IoT API

Applications
- AWS SERVICES
  - With these endpoints you can deliver messages to every AWS service.
Software as a Service (SaaS)

- SaaS
  - You get a whole solution

- Example
  - Gmail
  - Dropbox
  - Office365
XaaS: A Recap

On-Premise
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

IaaS
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

PaaS
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

SaaS
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

You Manage

Service Provider Manages
How you build your service?

What’s the open sourced counterpart of a device gateway?

Is Mosquitto (MQTT) Message middleware a good choice for messaging service?

How do you manage the shadows / maintain its states?
Project: Your Own Cloud Service

IaaS
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

Your Own PaaS
- APP
- Data
- Runtime
- Middleware
- OS

Your Own SaaS
- APP
- Data
- Runtime
- Middleware
- OS

You Build and Deploy

You Build and Deploy

You Build and Deploy
Too Overwhelming?

Your Own PaaS

- **APP**
- **Data**
- **Runtime**
- **Middleware**
- **OS**

You may focus on one important component in your proposed PaaS and implement it by using open source building blocks, for example:

- Message Middleware: NSQ
- Streaming Processing: Flink
- In Memory DB: Redis

Then tried to integrate your own implementation against existing AWS services

- E.g. AWS DynamoDB, S3, Kinesis

Ready to Kick it up a Notch?

- Try to use several and only open source components for building your PaaS.
- Feed it with “Real Data”
Evaluation: In terms of Real-Time

- Evaluation:
  - Latency Measurement (Real-Time)
  - Other Metrics relevant to your service/components

- Data and workload:
  - Synthetic Data and Workloads are fine
  - Get extra credit for using “Real Data”
    - E.g. Twitter Data (API) ➔ Streaming Platform
Where to start?

- Bunch of Services
- Find the one you interested:
  - If you are running a start-up:
    - How do you implement (part of) that service on your cluster?
  - Is there any open sourced component can be used in this project?
- Your own Idea / Insight
Example: Yahoo Streaming Benchmark
Real Time Advertising: Examples

- Native Advertising
  - Collect User Real-Time Event
  - Generate Relevant Ads

- Campaign Provisioning
  - Yahoo Gemini[^2]
  - Real-Time Ads Trend

[^2]: [https://gemini.yahoo.com/advertiser/home](https://gemini.yahoo.com/advertiser/home)
Yahoo Streaming Benchmark

- **Story**
  - Yahoo Adopt Apache Storm as stream processing platform in 2012
  - Want to know the limitation when compared to other platform
  - Existing benchmarks: **not testing real world usecase**
  - **Yahoo Streaming Benchmark**-- one specific real world usecase: Advertisement Campaign
  - Metrics: Comparing the **Latency** that a particular system can produce

Yahoo Streaming Benchmark

➢ Real-Time Advertisement: Counting Relevant Events

- Read an event from Kafka
- Filter out irrelevant events \((event\_type)\)
- Get relevant fields \((ad\_id \text{ and } event\_time)\)
- Join each event by \(ad\_id\) with its associated \(campaign\_id\)
- Take a windowed count of events per campaign
- Store each window along with a timestamp

Advanced Matching and Word Counting (involve in-memory database)

Relevance to your project

- Example: Build a Streaming service on a private cluster

- Components: Open Source
  - Kafka as Message Middleware
  - Flink as Streaming Processing
  - Redis as In-Memory Database

- Evaluation: Latency & Throughput
  - Data: Synthetic
    - (Clojure Project ➔ Kafka)

- Deployment:
  - Distributed

---

The GitHub Version: Single Node

- Verify if the system performs correctly in VM:

![Diagram of AWS cluster with EC2 Node, Kafka, Zookeeper, Flink Workers, and Redis]

Attention: You need 8GB JVM Memory for running this testcase.
Deploy it on Multi-Node

- Try Different Strategies and Evaluate Performance

![Diagram of a multi-node cluster with VMs for Kafka, Zookeeper, Flink, and Redis.]
Message Middleware:
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
Real-Time Messaging Service

- Messaging middleware includes:
  - Broker(cluster)
  - Client API

- Open source choices:
  - RabbitMQ, NSQ, Mosquitto, Kafka…
Real-Time Messaging Service

- Deploy messaging broker in Amazon EC2 VMs
  - Optional: broker cluster with load balancer
- Write your own publisher/subscriber with client APIs
  - Synthetic workloads
  - Or simulate realistic (IoT/micro-service) use cases
  - Example: [https://github.com/Chong-Li/Messaging-Service-Test](https://github.com/Chong-Li/Messaging-Service-Test)

- Try different configurations
  - Number of publishers/subscribers, number of topics, message size, sending rate, enable/disable persistency, number of brokers, etc
- Measure end-to-end latency and locate system bottleneck
Streaming Platform: Flink
Apache Flink

- An open-source stream processing platform

- Streaming-first
  - continuous processing

- Fault-tolerant
  - stateful computations

- Scalable
  - to 1000s of nodes and beyond

- Performance
  - high throughput, low latency

https://flink.apache.org
Example: Ad analytics
Example: Ad analytics

Service provider: “Good spot for Ads?”

Vendor: “effective?”
Example: Ad analytics

Service provider: “Good spot for Ads?”

How many # of clicks on the Ad?

Vendor: “effective?”
The click count example

- **Data sources:** Ad instances
  - producing tuples \{ad1, 1\}, \{ad2, 1\}, \{ad1, 1\}, …
- **Stream transformation:** aggregate the click counts for each Ad
  - \{ad1, 1\}, \{ad2, 1\}, \{ad1, 1\}, … \rightarrow \{ad1, 2\}, \{ad2, 1\}, …
- **Data sinks:** vendors / service provider
  - sinks subscribing to ad1 receive \{ad1, 2\}, …
Apache Flink features

- Stream transformation (built-in and user-defined)
- Windowing (e.g., density of clicks on certain Ad)
- Scalability (stream processing on a machine cluster)
- Fault tolerance (e.g., recovering from service interruption)
- …

https://flink.apache.org
Project ideas if you pick Flink

- Streaming usability and applicability
  - Use cases?
  - Achievable using Flink? Working along with other IoT IaaS?
  - Implementing those use cases
  - Evaluation (whether it meet the goal and how good it is)

- Streaming performance measurement and analysis
  - End-to-end latency?
  - Performance bottleneck?

- Streaming systems design and configuration
  - Good / bad recipes in each scenario? And why?
    - Example: processing transformations of different level of priority
  - Providing with empirical evidences
Steps

1. Choose your favorite topic
2. Form a team
3. Propose a plan
   - Proposal Presentation: Sep/18
   - Proposal Due on Sep/18, 11:59PM
4. Demo: 1, 2, final
   - Demo 1: Oct/9, Oct/11
   - Demo 2: Nov/13, Nov/15
   - Final: Dec/4 (1-4pm)
5. Write a technical report
   - Due on Dec/6, 11:59PM
Project Requirements

- **Build Up Your Own Cloud Service**
  - Run in public cloud, using open sourced component

- **Evaluation on Real-Time Feature**
  - Benchmark Application
  - Metrics: Latency + Other Features related to your service

- **Will grade based on**
  - project difficulty
  - quality and depth of work
  - workload distribution among team members (peer review)

- **Milestones:** proposal, demo1, demo2, final demo, report.

- **Start early! Discuss with us and Dr. Lu**
One More Thing: Security

➤ DON’T UPLOAD YOUR KEYS!!!
What if... $50,000 AWS Bill!

My AWS account was hacked and I have a $50,000 bill, how can I reduce the amount I need to pay?

For years, my bill was never above $350/month on my single AWS instance. Then over the weekend someone got hold of my private key and launched hundreds of instances and racked up a $50,000 bill before I found out about it on Tuesday. Amazon had sent a warning by email at $15,000 saying they had found our key posted publicly, but I didn't see it. Naturally, this is a devastating amount of money to pay. I'm not saying I shouldn't pay anything, but this just a crazy amount in context. Amazon knew the account was compromised, that is why they sent an email, they knew the account history and I had only spent $213 the previous month. I almost feel they deliberately let it ride to try to earn more money. Does anyone have any experience with this sort of problem?

Source: https://www.quora.com/My-AWS-account-was-hacked-and-I-have-a-50-000-bill-how-can-I-reduce-the-amount-I-need-to-pay
Thank You!

TA for class CSE 520S, Fall, Sep/06/2017

Haoran Li, Chong Li, Chao Wang