A Cloud-based Simulation of Disease Tracking System using AWS Amazon

Laleh Rostami Hosoori, Dr Stephen W. Clyde

Department of Computer Science, Utah State University

Introduction

The poster is about a cloud-based simulation of Disease Tracking System with many electronic medical record systems, health district systems, and disease analyzers. This simulation was completed as an assignment in CS7930, which involved

1. Implementing the processes in the system.
2. Implementing vector timestamps for tracking logical time.
3. Implementing and setting up a simple name server.
4. Setting up access to a cloud.
5. Provisioning the necessary resources to run the system.
6. Automating the deployment of the software to the cloud.
7. Automating the launching of the system's execution.

The Inter-process Communication Pattern

The distributed disease tracking system consists of three programs:

1. **EMR** (Electronic Medical Record) simulator
2. **HDS** (Health-District System)
3. **DOA** (Disease Outbreak Analyzer)

Figure 1a. Logical Design of Disease Tracking System.

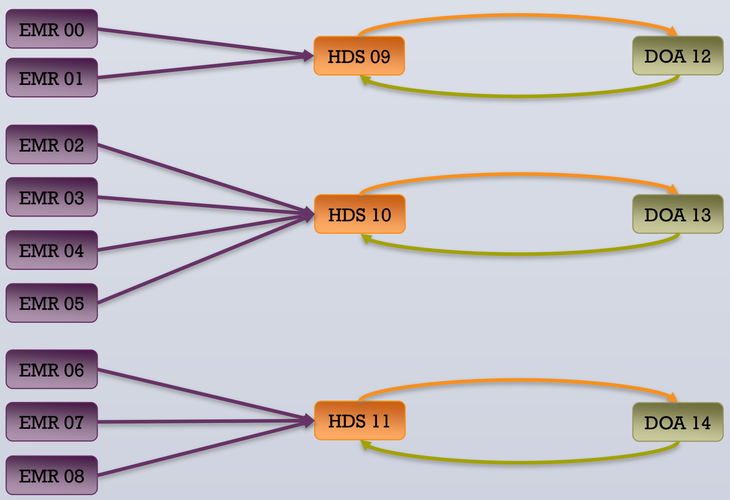

Figure 1b. Configuration of Disease Tracking System Processes.

Vector Timestamps

Vector timestamp is a way to partially order events in a distributed system. It is a logical clock in lack of a global clock in distributed systems.

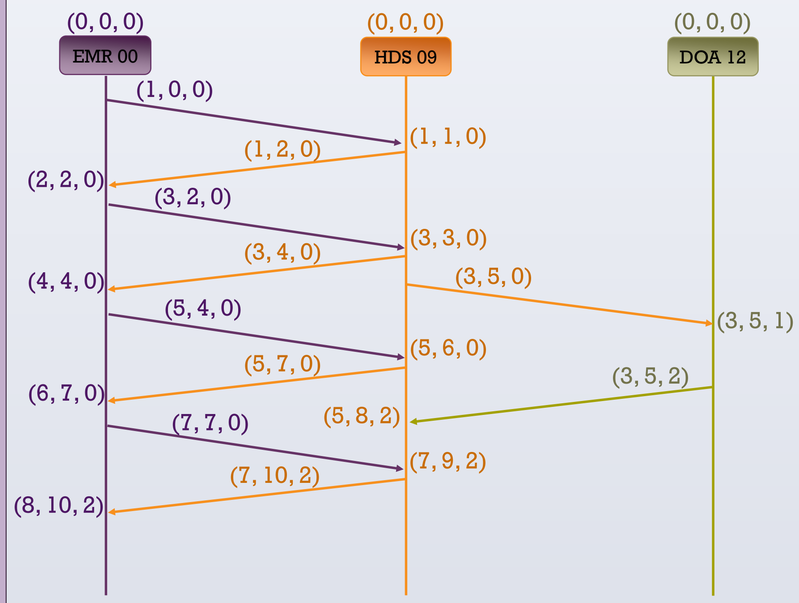

Figure 2. Vector Timestamps.

Name Server

The name server lets all processes to register themselves and query for other processes as well. Since, EC2 instances are assigned dynamic IPs whenever we start them, there is a need for a name server to provide the processes' endpoints .

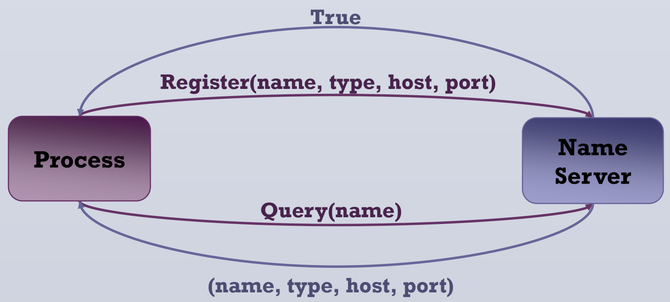

Figure 3. Name Server.

AWS Amazon

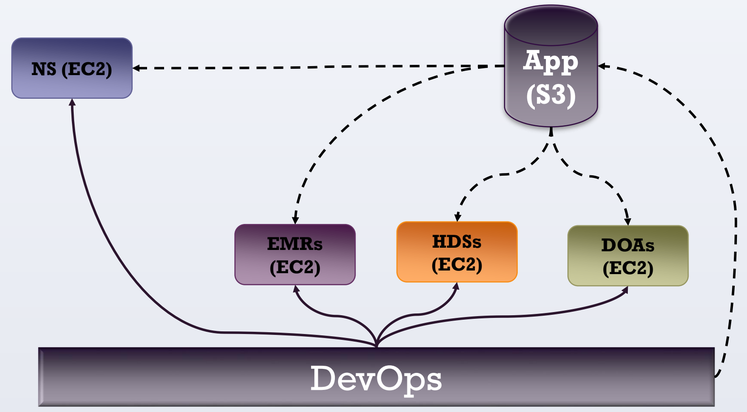

Figure 4. AWS Amazon Resources.

I leveraged the following resources of AWS Amazon:

1. **EC2**: Run the processes.
2. **S3**: Store the whole application and timestamp log files.
3. **DevOps**:
    1. Create Instances.
    2. Start instances.
    3. Deploy the application.
        1. Upload from a local machine to a S3 bucket.
        2. Deploy from the S3 bucket to the all instances.
    4. Stop instances.
    5. Terminate instances.
4. **AWS nodejs SDK**:
    1. Run an arbitrary number of the processes on each instance.
    2. Upload the timestamp log files to the S3 bucket.

Conclusions

In vector timestamps, the x-th element in the timestamp of the x-process is always larger than or equal to the x-th element from any other process's timestamp. Using a cloud, it is quite tricky to manage usage to get lower bills, although it is really cheap for low usage. The cloud users depend on the cloud environment pretty much.

Contact

Please contact me at *laleh.rostami@aggiemail.usu.edu*.