Distributed Systems
Project assignment - Building a distributed data-manager

Hadrien Croubois
hadrien.croubois@ens-lyon.fr

All documents are available on my website: `http://hadriencroubois.com/#Teaching`

This project is due for april 30, 2017, 23h59, Lyon's time (UTC+2).

Part 1

Introduction

Cloud platforms are becoming the go to solution for everyone wanting to deploy online platforms. The "pay as you go" paradigm allows anyone to deploy any platform size without having to care for the maintenance of the hardware or the cost of unused servers during off-peak hours.

However, those very dynamic platforms need special mechanisms to manage all the resources in this highly dynamic context.

Part 2

Assignment

Now about the assignment. In this project we aim at building a distributed data-management tool which could take care of all or data and share them among the nodes of our infrastructure. Note that fulfilling all the proposed features, however interesting it is, is not required to have a good mark. Don't be stressed if don't answer all the question, rather choose the one you feel interesting and capable of doing in order to build your own solution at an objectively very complex problem.

Beyond the code itself, you are expected to handle a report which should includes both details about your choices of implementation and discussion about the strength and weaknesses of your solution.

Agent deployment

The first job in deploying a distributed platform is to design agents that will run on the different machine and communicate with one another. Once one or multiple agents are deployed, you should be able to add new nodes or remove existing nodes while maintaining a coherent topology.

While all erlang instances can send message to any other, sending a message requires to know the target identifier. Therefore, for each agent, the list of other agents known by him represent a subset of the platform which is his neighborhood. Exchanging information with further nodes requires to hop from nodes to nodes.

Your platform should offer some features to handle the topology to dynamically adapt to the nodes deployment.

Question 1

a) **[Required]** Spawn an agent that waits for remote command;

b) **[Required]** Have this agent be part of a coherent topology that allows command to pass through;

c) **[Required]** Be able to have a new agent join an existing topology;

d) **[Required]** Have the possibility to remotely kill an agent while maintaining the coherent topology among the remaining agents;

e) **[Optional - Hard]** Have the possibility to detect and react to an agent dying without prior notice.

You should also implement some communication mechanisms on this topology

Question 2

a) [**Required**] Broadcast: send a message to all nodes. Nodes should consider the message only once.

b) [**Optional**] Scatter: distribute a list of message among the nodes. All message should be considered by only one node. Distribution should be as even as possible.

c) [**Optional**] Leader election.

Using the deployed agent for data-management

Agents are interresting in that they are the base for deploying services. Now that you have a set of agents, we will use it to build a data-manager.

Question 3

a) [**Required**] Your agents should be able to receive pieces of information, remember them, and send back a unique identifier (UUID) use to point at this specific piece of data;

b) [**Required**] Your agents should be able to send back the piece of data corresponding to a UUID. The request can be done to any node, not necessarily the one who received the piece of data on the first place;

c) [**Required**] Make sure that no data is lost when a nodes is asked to shut down;

d) [**Optional - Easy**] Have a mechanism to release a specific data (deleting it from all nodes where it is stored);

e) [**Optional - Medium**] Have a mechanism to distribute all the pieces of data evenly across the platform so that no node store them all;

f) [**Optional - Hard**] Have a mechanism that replicates the pieces of data across the nodes so that multiple nodes have it (be careful of not flooding the platform, we do not want every node to have everything !);

g) [**Extra**] Feel free to discuss and implement any other features that you might find interesting.

Agent monitoring

An interesting tool to have is a client that is able to interact with a topology of agents through a CLI. This agent could:

Question 4

a) [**Required**] Given an hostname, remotely spawn an agent on that node and have it join the topology;

b) [**Required**] Given an hostname, remove the corresponding node from the platform and return the hostname of a node still in the topology;

c) [**Required**] Send data to a platform identified by the hostname of a node in the platform;

d) [**Required**] Given an UUID, retrieve the corresponding piece of data from a platform identified by the hostname of a node in the platform;

e) [**Optional - Easy**] Given an UUID, release the corresponding piece of data from a platform identified by the hostname of a node in the platform;

f) [**Optional - Simple**] Query the topology for statistics;

g) [**Optional - Medium**] Build a script that, given a set of nodes, automatically deploy the platform.

Part 3

Handover details

Apart from the code itself, you are requested to provide a `.pdf` report detailing your choices. I'd recommend this report to be written using LaTeX, but you are free to use applications like Microsoft Word or Open Office (as long as you export it as a PDF). This report, and all your code must be provided put in a `.tar.gz` archive named `name.tar.gz` in which you must have a folder `name` (replace `name` by your name !). You are also encouraged to provided detailed information on how to deploy and test you code, or even better, have a script do it automatically !

Your code is your property, protect it with a license ! Also, when sending your archive don't forget to provide us with a `md5` checksum so we can verify its integrity.