BSE 3208: Distributed Systems Development

Exercise 2 (Peer-to-Peer Systems)

Required: 23rd December 2021

1. Introduction

Crane Cloud is implementing a multi-cluster setup of computing nodes (servers) distributed over multiple sites. The clusters are used for hosting containerized applications (think Docker containers) through a front-end software abstraction dubbed 'omicron'. The users access omicron and deploy applications by providing details such as name, description and the container image repository. The clusters represent a logical grouping of nodes (as shown in figure 1) with resources (compute, storage and memory) to host applications and have them provide the required services. At the moment, the project has three clusters based on their locations ***mpologoma*** (Kampala), ***simba*** (Dar-es-Salaam) and ***mkango*** (Lusaka) as shown in Figure 2. Each cluster provides an API through a public IP address for access to omicron and the users. Given possible cluster downtime, the Crane Cloud Dev team is considering a Peer-to-Peer design (architecture) for the multi-cluster system with the following requirements:

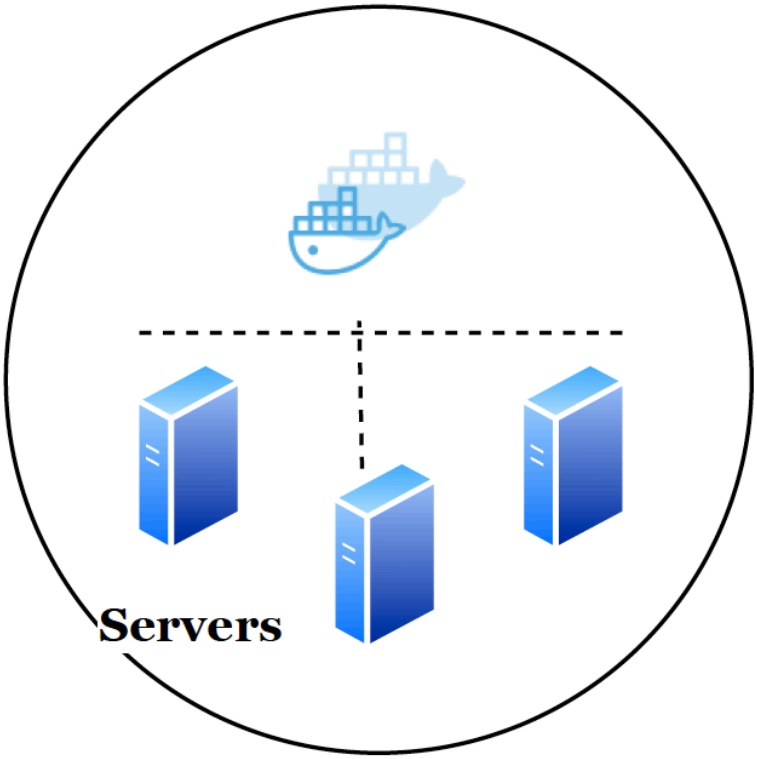

Figure 1: A typical cluster

- A cluster should join and leave the network
- Resources should be moved from one cluster to another
- The network should scale (New clusters should be easily added to the network)

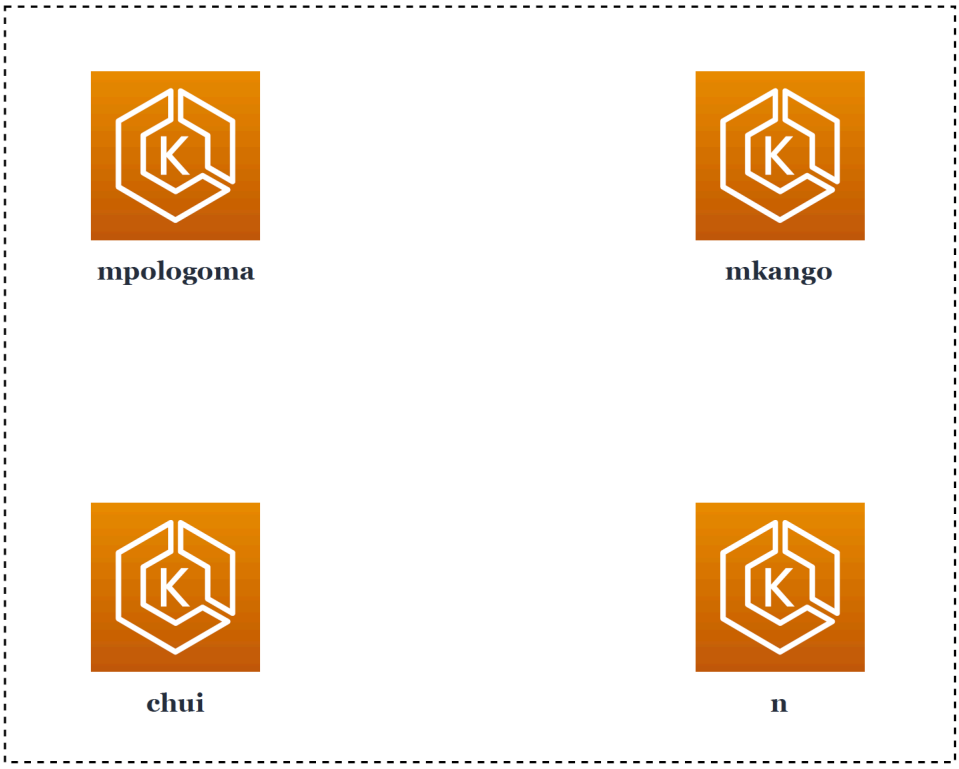

Figure 2: Multi-cluster

2. Exercise

In this exercise, you will help the Crane Cloud Dev team to design and develop a P2P system to meet the above requirements. More specifically:

- A node such as a desktop or laptop should represent a single cluster
- At the start, only two nodes (say laptops) are part of the network. You start node 1 and join node 2 to the network.
- A third node (and subsequent nodes) should be able to join and/or leave the network
    - If a node leaves the network, the resources that it holds should be relocated to another node.
    - If a node rejoins the network, its previous resources should be re-allocated to it
    - You may develop a simple algorithm to effect the above operations (for example, the resources could be relocated to the closest node. The distance can be measured by the RTTs between the nodes)
- A client application should be able to search (query) one of the live nodes for a specific item hosted. The query should be routed appropriately to the node that has the resource. The response should indicate which node provided the response.
- You may need to build a simple LAN to mimic an overlay network for the P2P system

3. Additional Information

- You can make extensive use of sockets and write out your code in your preferred programming language (If you can do Python, great but not a requirement)
- The resource to be shared are simple text files (It should be possible for a single resource to be available on two nodes)
- The fetching of resources is 1:1, a client can download a resource from a single peer
- You can do further reading on Hash Tables & Distributed Hash Tables (DHTs) and their implementations
- Your nodes may need to specify the maximum number of peers that they can connect to in form of a peers table

Required: An implementation report with well documented code and screenshots where necessary