Chapter 2: Architectures

Nguyen Ngoc Lam - 20162316

Friday 27th March, 2020

Contents

1 Problem of Physical Distance

From what I gathered, the physical distance only has a significant affect on latency if you are to implement a system that capture a live data and than process it in almost real time for sciencetific purpose. If you built that kind of system (strongly depend on time), a simple solution would be to put the processing station near the source of information and then transmit a processed file.
Some of the common strategies to deal with latency are:

- caching data on the client - i.e. avoiding to make request again if possible
- combining several smaller requests into bigger ones
- compressing the request
- reusing the same connection for many requests
- using different protocols (i.e. a lot of games use UDP to avoid latency at the cost of making other things much more complex)
- caching data on the server
- redesigning the way to make requests (pre-loading data, pre-allocating resources, creating parallel requests)
- using content delivery networks (CDN) for static data
- running a code through a profiler and redesigning it to eliminate bottlenecks
- minimizing critical sections of code
- designing the service as stateless so that it can be deployed on many servers in parallel
- correctly sizing, designing and maintaining the system (OS, database, virtualization, load balancers, DNS caches, etc.)
- correctly sizing, designing and maintaining the network (having enough bandwidth, using fast routers, traffic prioritization, resource reservation)

2 Three-tiered Client-server Architecture

A three-tiered client-server architecture consists of three logical layers, where each layer is implemented at a separate machine, in principle. The highest layer consists of a client user interface, the middle layer contains the actual application, and the lowest layer implements the data that are being used.

3 Differences Between a Vertical Distribution and a Horizontal Distribution

- Vertical distribution refers to the distribution of the different layers in a multitiered architectures across multiple machines. In principle, each layer is implemented on a different machine.
- Horizontal distribution deals with the distribution of a single layer across multiple machines, such as distributing a single database.

4 Disadvantage of Topology of the Overlay

The main problem here is dealing with logical paths. In practice, two nodes A and B which are neighbors in the overlay network may be physically placed far apart. As a result, the logically short path between A and B in this instance required routing a message along a very long path in the underlying physical network.

5 Main Problems with Sequential Organization when Taking a Look at the Request-reply Performance at Process P_1 (The Last One)

- Performance can be expected to be bad for large n since the problem is that each communication between two successive layers is, in principle, between two different machines.
- The performance between P_1 and P_2 may also be determined by $n2$ request-reply interactions between the other layers.
- Another problem is that if one machine in the chain performs badly or is even temporarily unreachable, then this will immediately degrade the performance at the highest level.

6 Goodness of Routing Policy that Forward messages to the Closest Node Toward the Destination

According to the pictures of the question sheet, consider forwarding message from node $(0.2; 0.3)$ (called A) to node $(0.9, 0.6)$ (called B). There are 2 routes:

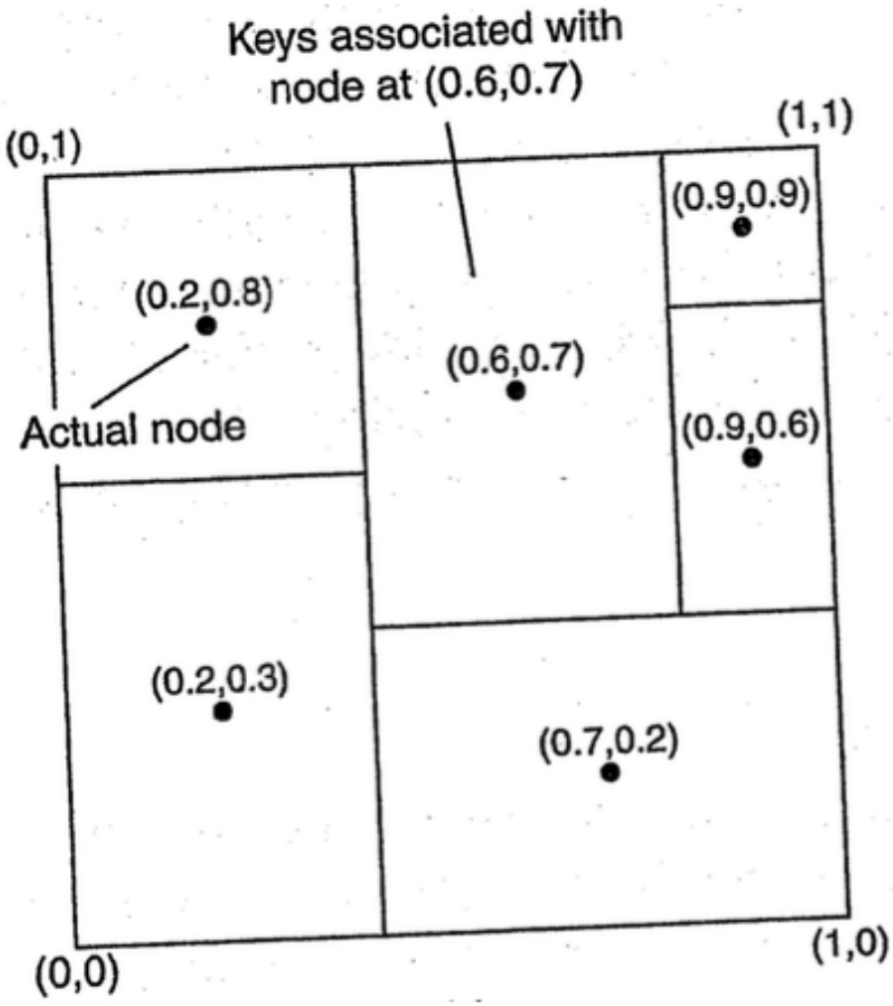

Figure 1: CAN network

1. A, through $(0.6; 0.7)$ (called C), to B:

$$AC = \frac{2\sqrt{2}}{5} \approx 0.566$$

$$CB = \frac{\sqrt{10}}{10} \approx 0.316$$

$$TotalDistance = \frac{\sqrt{10} + 4\sqrt{2}}{10} \approx 0.882$$

2. A, through $(0.7; 0.2)$ (called D), to B:

$$AD = \frac{\sqrt{26}}{10} \approx 0.510$$

$$DB = \frac{\sqrt{5}}{5} \approx 0.447$$

$$TotalDistance = \frac{\sqrt{26} + 2\sqrt{5}}{10} \approx 0.957$$

Because our policy is to forward a message to the closest node toward the destination, which in this case D, we will actually go a little longer than if we take route 1. In conclusion, this policy is a resonable one, but it is not always the best one.

7 Benefits of Microservices Architecture Compared To Monolithic Architecture

- Microservices architecture makes systems and applications much faster to develop, and much easier to understand and maintain since it tackles the problem of complexity by decomposing application into a set of manageable services.
- Microservices architecture enables several teams to works on a projects, each can focus on one service or serveral closely-related services thus allowed services to be developed independently.
- Microservices architecture reduces barrier of adopting new technologies since the developers are free to choose whatever technologies make sense for their service and not bounded to the choices made at the start of the project.
- Microservice architecture makes continuous deployment possible for complex applications because it enables each microservice to be deployed independently.
- Microservice architecture enables each service to be scaled independently.

8 Design an E-commerce System Using Microservices Architecture

Services in the system:

- HTTP apche: Get the HTTP request and forward it to the right service
- Customer: Handling customer data like: name, address, age, gender , preferences et cetera

- Order: Handling order information: who buying, from whom, what sort of item/items, how many item and for how much
- Catalog: Handling item information: of what catergory, how many variants (color, size, shape, version), contain what items (for bundle), how many items left, unit price, rating
- Vendor: Handling vendor information: name, address, vendor rating, what items are they selling, number of items sold in a time period (for customer to know if they have been receiving lots of requests)

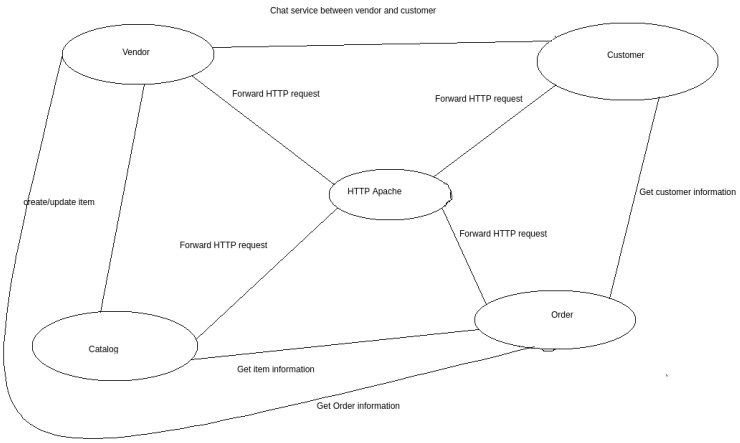

Figure 2: Figure to illustrate how the system interact