How is your project architecture related to the theory taught in the lecture?

Logical Architecture

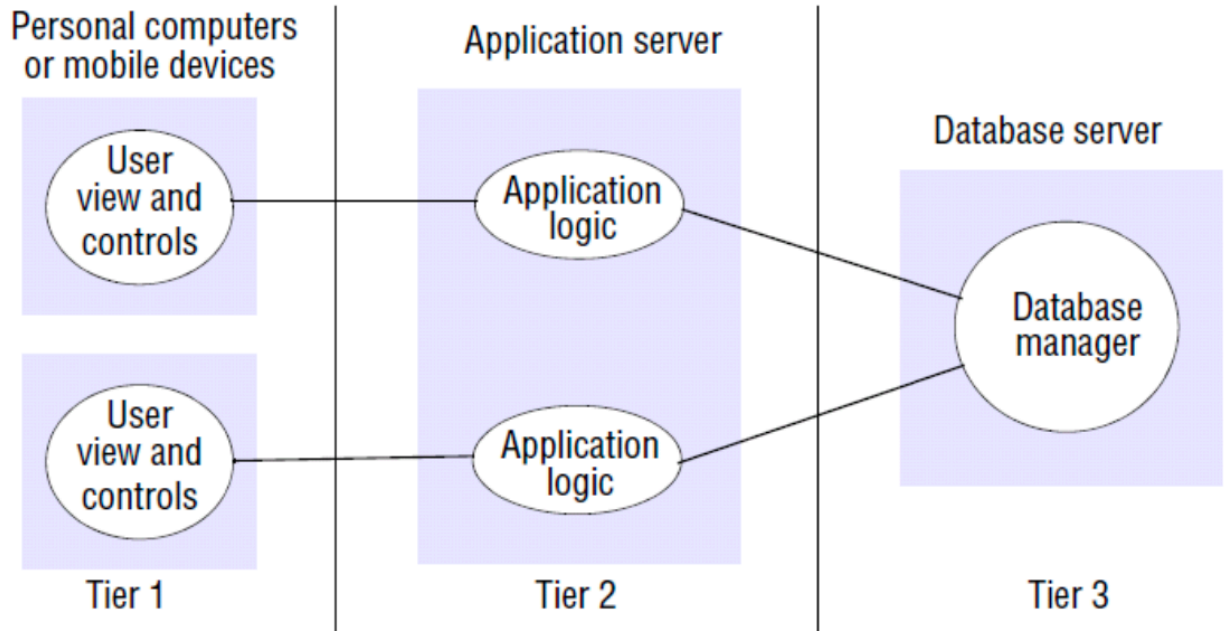
The chatbot uses Line's bot function to interact with users, uses github for code version control, and the database uses redis as high-performance key-value database storage service, and the chatbot is implemented on the cloud platform heroku.

Physical Architecture

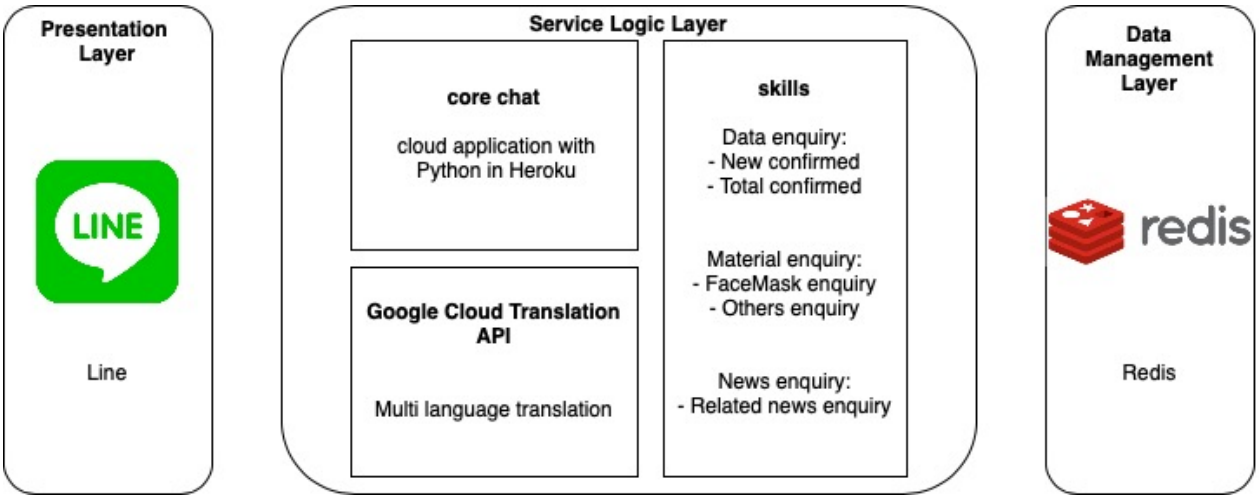
The physical architecture of the distributed system is mainly used, including mainframe, integration server, application server, proxy server, storage server, report server, Web server, network shunt, etc.

System Architecture

The architecture adopts Three-tier Architecture, which separates presentation logic, application logic and data logic, each tier has a well-defined role.

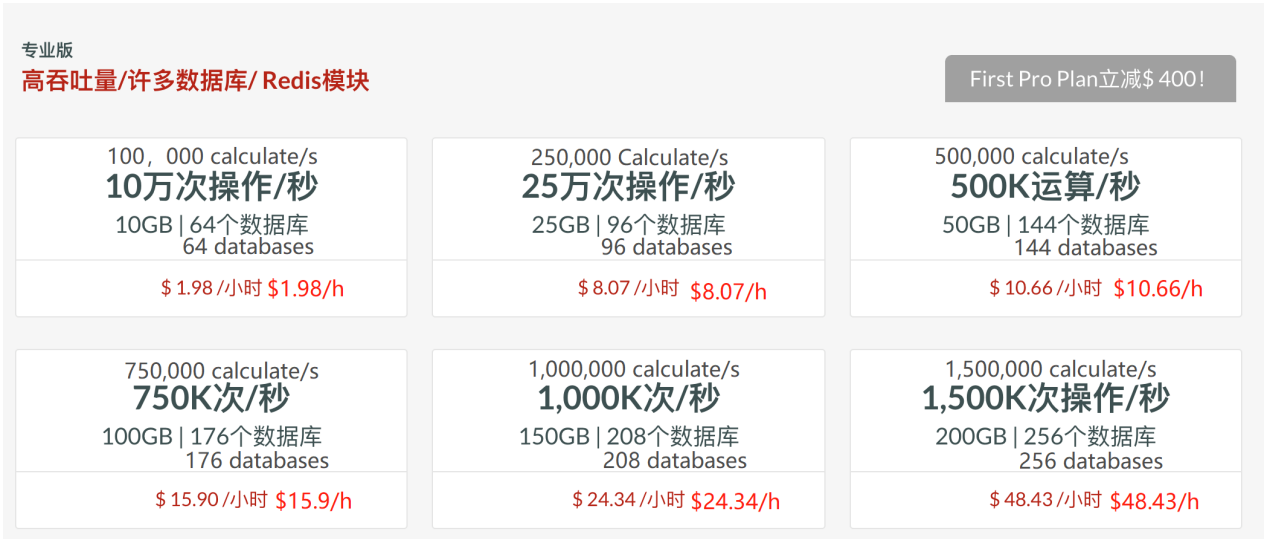


For our chatbot, the presentation layer is based on Line, and users can access Line from a mobile phone, PC, or other machines. Service Logic Layer includes core chat, google cloud translation API and skills. Core chat is written in Python on the Heroku platform; Google cloud translation API provides language translation for our chatbot; Skills includes virus data query, material query and news query. For the Data Management Layer, our data is stored in a redis database that can read data at high speed.



Can you demonstrate, with some screen cap, how to increase capacity of your chat bot service?

First of all, for a server with a larger capacity, we can first change from the choice of the API server itself. We can choose API servers that have more services but may be more expensive. As described in the picture, from left to right is the arrangement of prices from low to high. The two servers on the far right are equipped with an auto-scaling function to relieve the pressure on the server.



The second point is to use scale-up and scale-out for system expansion. And our common scale-up way is to increase the cache properly, use the existing storage system, and continuously increase the storage capacity to meet the needs of data growth. When the amount of data is large enough, the scale-up method may cost a lot of time and material costs to expand the original system, and at this moment will also affect the user's use. And we can use scale-out to expand the system. For example, a cloud application platform that supports automatic expansion is used for elastic scaling operations while using a microservice framework for optimization. Or an event-driven architecture, with nodes as the unit, each node will often include capacity, processing power, and I / O bandwidth. A node is added to the storage system, and the three resources in the system will be upgraded simultaneously.

For the optimization of database query in the actual project of chat robot, we can use the method of package query

```
•def packageQuery():
    # Set query num and time counter
    query_num = 0
    timeCount.start()
    # Increase the query queue when the user and time have not reached the
threshold
    while (userQuery() && timeCount < 1000 && query_num < 1000) :
        query_num += 1
        query_queue.add(userQuery.content)
    # Do package query and initialize timer
    timeCount.stop()
    query_num = 0
    result = redis.query()
    return result
```

Scaling

https://devcenter.heroku.com/articles/scaling

Scaling from the CLI

Scaling the number of dynos

You scale your dyno formation from the Heroku CLI with the `ps:scale` command:

```
$ heroku ps:scale web=2
Scaling dynos... done, now running web at 2:Standard-1X
```

The command above scales an app's `web` process type to 2 dynos.

You can scale multiple process types with a single command, like so:

```
$ heroku ps:scale web=2 worker=1
Scaling dynos... done, now running web at 2:Standard-1X, worker at 1:Standard-1X
```

You can specify a dyno quantity as an absolute number (like the examples above), or as an amount to add or subtract from the current number of dynos, like so:

```
$ heroku ps:scale web+2
Scaling dynos... done, now running web at 4:Standard-1X.
```

If you want to stop running a particular process type entirely, simply scale it to `0`:

```
$ heroku ps:scale worker=0
Scaling dynos... done, now running web at 0:Standard-1X.
```

Can you identify if you bot is one of the example of PaaS, IaaS, SaaS? Explain your answer.

Our chatbot belongs to SaaS. Our chatbot is deployed on the Heroku, which is a PaaS platform. The chatbot provides users with functions related to virus querying information and multi language translation. Users do not need to consider the system architecture and the composition of physical servers. They only need to use the function of chatbot, which is the characteristic of SaaS.