

Milestone 4

Q1. How is your project architecture related to the theory taught in the lecture?

The whole Line Bot system framework is made up of three parts:

- 1) Line Client, the terminal interface used by the users. Usually it can be a LINE Channel;
- 2) Line Server, it is used to receive the receive an interactive request from the Line Client;
- 3) Webhook Server, it is where the business logic, pass back component plan and the Line Bot SDK works.

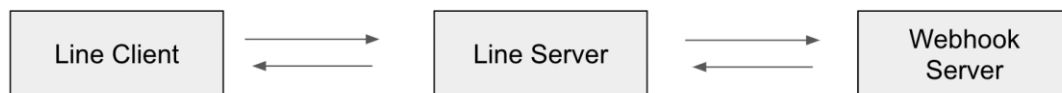


Fig.1

When Line Client receives a message from the user, it will pass it to Line Server who act like a translator. Then the server will send this request to Webhook Server, where we have deployed our code solutions already. After getting the results, the results will be passed back to client through the server again. Each passing represents a HTTPS protocol transmission (a HTTPS Request + a HTTPS Response).

Inside the Webhook Server, has deployed our code already. In order to enrich the query analysis ability and the decision making capability, some outside service are necessary. Such as Google Dialogflow, Microsoft Azure Bot Service...

The hardware architecture of Heroku

HTTP reverse proxy: With Nginx, this layer only handles HTTP- level processing.

HTTP Cache: For static content, use Varnish for caching.

Routing Mesh: Architecture components implemented in Erlang, routing and addressing, to improve availability and scalability.

Dyno Grid: The code deployed by the user runs here, which can be simply regarded as an application server cluster environment, but with a smaller granularity.

Database: PostgreSQL can also use a remote database.

Memory Cache: Memcached is a necessity for home travel architecture.

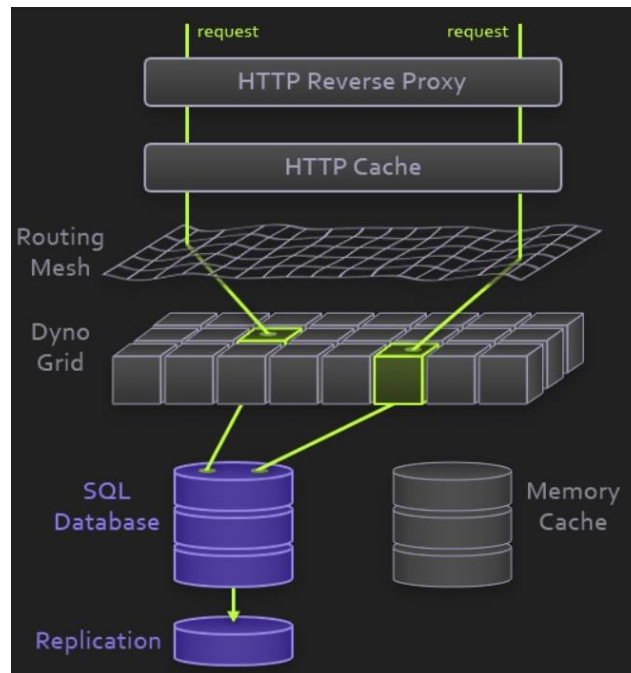


Fig.2

Python code framework Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

The simple use demo of Flask:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Fig.3

Q2. Can you demonstrate, with some screen cap, how to increase capacity of your chat bot service?

Method 1. Extend the access frequency in Messaging API.

To prevent massive access, each LINE Messaging API account has different API access frequency limits depending on the scheme, as shown in the following table:

方案	呼叫頻率限制	收訊者數量頻率限制
Developer Trial	1,000/min	20,000/min
其他方案	10,000/min	200,000/min

Fig.4

When the access frequency of the Messenger API exceed the threshold value, the API will response a 「429 Too Many Requests」 fault. If that happen, we can change the scheme in the API.

Here is the statistic for the access frequency:

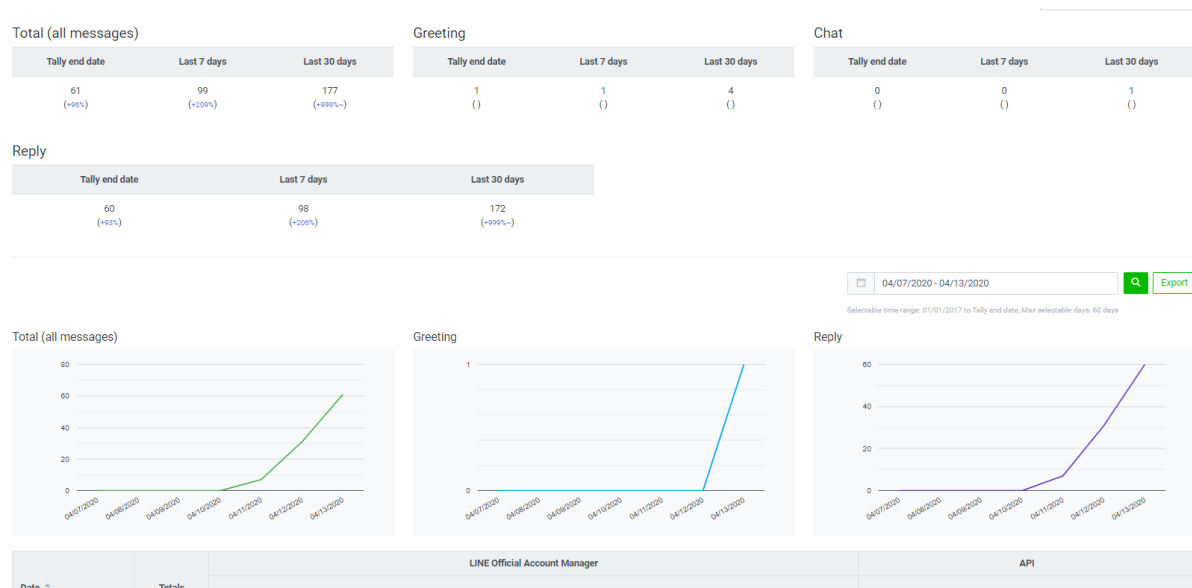


Fig.5

Method 2. Scale out/up in Heroku.

We can also increase our need by changing the scheme in Heroku to increase the capability.

- Scale up on each server. (Vertical scaling)







	 Free \$0	 Hobby \$7/dyno per month	 Standard 1x \$25/dyno per month	 Standard 2x \$50/dyno per month	 Performance M \$250/dyno per month	 Performance L \$500/dyno per month
What is it good for?	Ideal for experimenting with cloud applications in a limited sandbox.	Perfect for small scale personal projects and hobby apps.	Enhanced visibility, performance, and availability for powering your production applications.		Superior performance when it's most critical for your super scale, high traffic apps.	
RAM	512MB	512MB	512MB	1GB	2.5GB	14GB

Fig.6

b. Scale out by adding more servers (Horizontal scaling)

Simple horizontal scalability

With **Professional** dynos scaling your app out horizontally is as simple as dragging a slider in the Heroku dashboard or running one command from the Heroku CLI.

Mix Standard 1X, 2X, and Performance dynos to right-size your app and for greater performance.

STANDARD 1X
\$25/dyno per month

0

\$0

STANDARD 2X
\$50/dyno per month

0

\$0

PERFORMANCE M
\$250/dyno per month

0

\$0

PERFORMANCE L
\$500/dyno per month

0

\$0

Fig.7

Method 3. Rewrite the python code to get better operating performance.

a. Write the python code with smaller time complexity and space complexity

Q3. Can you identify if your bot is one of the examples of PaaS, IaaS, SaaS? Explain your answer.

What are IaaS, PaaS and SaaS?

IaaS: cloud-based services, pay-as-you-go for services such as storage, networking and virtualization.

PaaS: hardware and software tools available over the internet

SaaS: software that's available via a third-party over the internet.

My bot is based on a PaaS. PaaS is often used when a developer wants to create a unique application in a most cost-effective and time-effective way. I used Heroku to deploy my app. Heroku itself is a cloud platform as a service (PaaS) supporting several programming languages including python, which can be considered as a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages.

Several Advantages by using a PaaS platform:

- Accessible by multiple users. – Distributed system and router mechanism.
- Scalable – I can choose from various tiers of resources to suit the size of my business.
- Built on virtualization technology. – Better understanding and control of the physical resources.
- Easy to run without extensive system administration knowledge. – I can focus my time in creating, testing, and deploying the app instead of infrastructure management

I also use some SaaSs to enrich my bot function. The Messenger API, Amap API I used can be considered as the SaaS, since they provide a complete function with API which I can call. And it is no need for me to figure out the implementation details.

The Redis of the Heroku add-on is a Database-as-a-Service. It provides in-memory data structure store, which enables data persistence and high availability through replication and backups. Supporting various data structures such as strings, hashes, lists ...

Q4. Present your work on stage for not more than 5 minutes

See in the video.

References

[1] Document of Heroku Dev Centre; Available from <https://devcenter.heroku.com/articles/how-heroku-works>

[2] Blog of Fenng, Learning Heroku Structure; Available from https://dbanotes.net/arch/heroku_architecture.html

[3] Document of LINE Developer; Available from <https://developers.line.biz/en/docs/messaging-api/overview/#how-it-works>

[4] Tony Hou, IaaS vs PaaS vs SaaS Enter the Ecommerce Vernacular: What You Need to Know, Examples & More; Available from <https://www.bigcommerce.com/blog/saas-vs-paas-vs-iaas/#the-key-differences-between-on-premise-saas-paas-iaas>