

รายงานความก้าวหน้าวิชา CE Project

ครั้งที่ 2

ระหว่างวันที่ 29 ส.ค. 65 ถึงวันที่ 10 ก.ย. 64

1. ชื่อโครงการ (อังกฤษ) Image Processing Application using Task Scheduling on Network System

2. การดำเนินงานมีความก้าวหน้า 23% (ใช้ค่า % **Complete** จาก MS Project)

มีความก้าวหน้าเพิ่มขึ้นจากรายงานความก้าวหน้า ครั้งก่อน 21%

☐ เร็วกว่าแผน 0 วัน ☐ ช้ากว่าแผน 0 วัน

3. รายละเอียดความก้าวหน้า

สำหรับการพัฒนาในส่วนของ Server ผู้เรียนได้ศึกษาการทำงานของ Kubernetes (MicroK8S) และได้ติดตั้งใช้งานโดยการทำ VM มา 3 เครื่องโดยมีสเปคดังนี้

- Server#1 OS Ubuntu 20.02 CPU 8 Core, Ram 16 GB, GPU RTX 2060
- Server#2 OS Ubuntu 20.02 CPU 4 Core, Ram 8 GB
- Server#3 OS Ubuntu 20.02 CPU 4 Core, Ram 8 GB

จากการได้ใช้งานดูตัว MicroK8S นั้นมีการใช้ทรัพยากรที่ไม่หนักมากแต่มีปัญหาด้านการตอบสนองของระบบเวลาทำงานไป และ ตัว VM นั้นมีปัญหาด้านการทำ Network และ เครื่องคอมพิวเตอร์ไม่รับภาระการทำงาน 3 เครื่องพร้อมกันไม่ไหว จึงได้ศึกษาค้นคว้าแนวทางใหม่ ๆ โดยมีตัวเลือกที่ศึกษามาไว้อยู่ 2 ตัวด้วยกันคือ

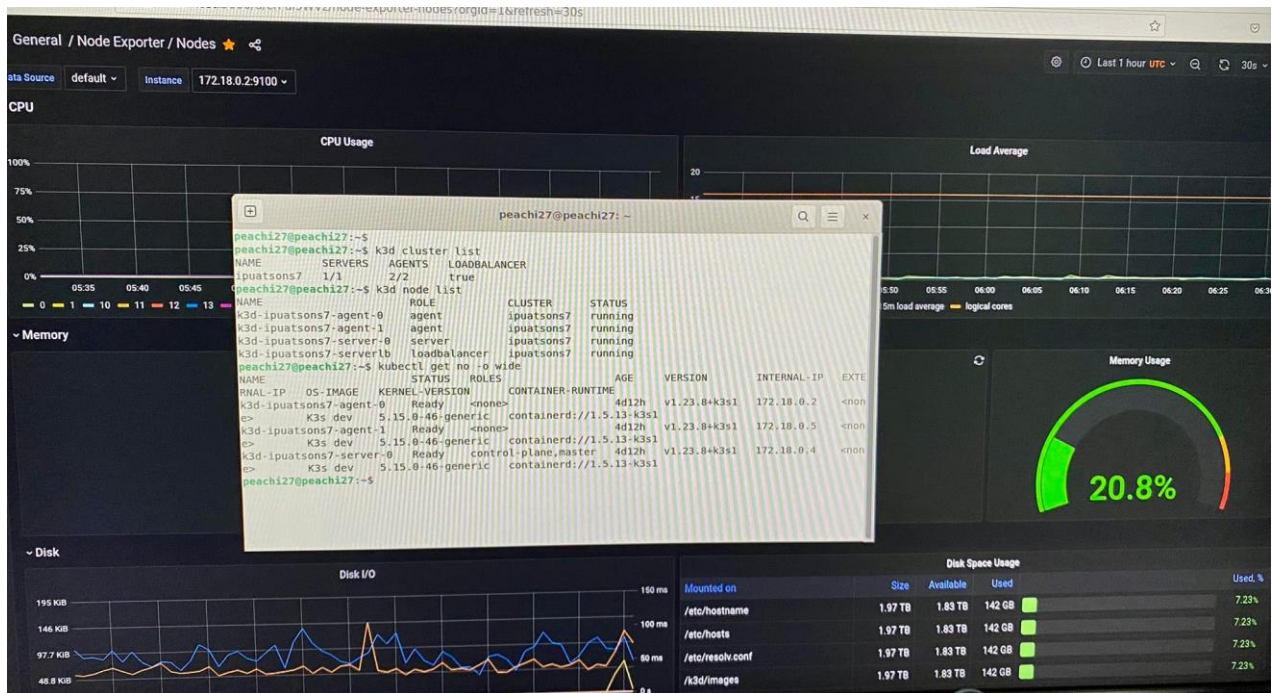
1. K3D
2. LXD

ทั้งสองตัวนี้เป็นตัวการจำลองเครื่องให้อยู่ในรูปแบบของ Container และ จำลอง Network ส่วนอื่น ๆ ได้ด้วยจึงได้ลองทดสอบดูและได้เลือกที่จะใช้ตัวของ K3D เนื่องจากมีการ Support และ สามารถปรับแต่ง ในการใช้งาน K3S เป็น Image ในการจำลองเครื่องมากกว่า หลังจากได้ติดตั้ง K3D และ ใช้ K3S เป็น Image ในการจำลองเครื่องก็ได้มีการปรับแต่งเพื่อให้ใช้ GPU ของเครื่องคอมพิวเตอร์เข้ามาใน Cluster ได้ด้วย

K3D คือ เป็นตัวครอบคลุม K3S ไว้ในรูปแบบของ Docker Container ซึ่งมีการใช้ทรัพยากรที่น้อย และ ทำงานได้อย่างรวดเร็วในการสร้าง Node K3S ขึ้นมา และได้มีการรองรับการใช้งาน Rancher สำหรับเป็น Dashboard ที่รองรับการติดตั้ง Plugin เสริมได้อีกมากมาย แต่เนื่องจากการทดลองนี้มีการใช้ GPU ในการประมวลผลด้วย และ ใช้ Prometheus Grafana ในการ Monitor Metric ของทรัพยากรต่าง ๆ ภายในเครื่องจึงได้ทำการดัดแปลง Image ของ K3S

K3S นั้นได้มีการปรับแต่งตัว Image เพื่อให้รองรับ GPU ได้โดยได้ปรับแต่งตัว Base ของ Image เป็น Ubuntu 18.04 และใช้ Nvidia/CUDA 11.0.3 ในการทำงานของ GPU และได้ติดตั้ง nvidia-container-runtime ของ Ubuntu 18.04 เพื่อให้สามารถทำงานร่วมกับตัวคำสั่งต่าง ๆ ของ CUDA ได้ และได้ติดตั้งตัว Prometheus สำหรับ Monitor

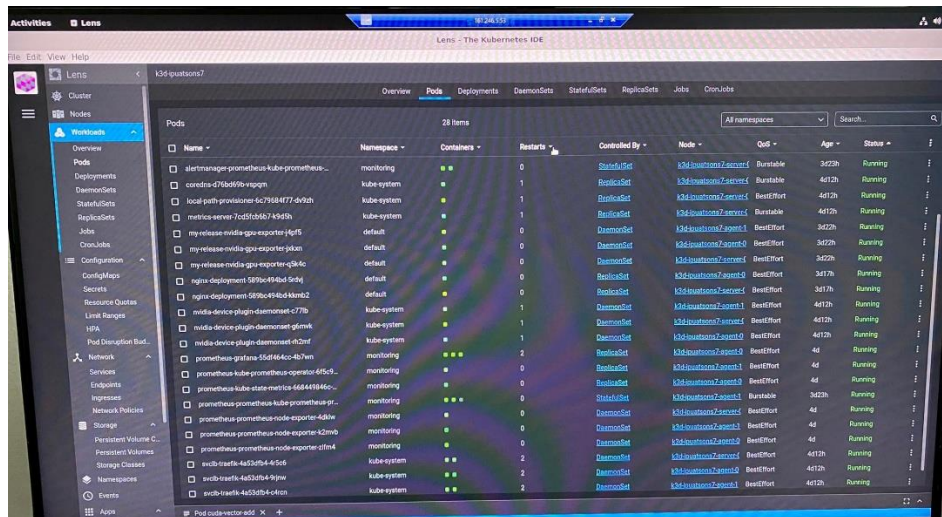
ในส่วนของ Monitor นั้นเป็น Prometheus Grafana ไว้สำหรับตรวจเช็คทรัพยากรในเครื่องในรูปแบบของ Metric เพื่อนำไปใช้งานในอนาคตต่อไป และได้ใช้ Kubernetes Lens สำหรับตรวจสอบงานต่าง ๆ ที่เข้ามา และ ตรวจเช็คว่า Node แต่ละตัวทำงานได้ปกติ



รูปภาพของหน้าจอ Prometheus Grafana และ K3D ที่ทำ Cluster และ สร้าง Node มา

จากการทำเครื่อง Cluster ที่มี Node อยู่ 3 ตัวโดยเป็น Server 1 ตัว และ Worker 2 ตัว โดยตัวของ Server จะมีหน้าที่ในการเป็น Control Plane ในการรับคำสั่ง และ ตัดสินใจเลือกเครื่องที่เหมาะสมจากทรัพยากร และ งานที่กำลังทำอยู่ให้แต่ละเครื่องไป ซึ่งตัวของ Server เองก็สามารถจะเลือกที่จะทำงานเองได้เช่นกัน และได้ติดตั้ง Kubernetes Lens เพื่อช่วยในการแสดงผลเป็น GUI ในการมองภาพรวมของ Node และ Pods ที่ได้สั่งงานไป หรือ กำลังทำงานอยู่ และ ตรวจเช็ค Network Service ต่าง ๆ ว่าได้ใช้ Port ตัวไหนบ้าง

ในส่วนการทดสอบ Cluster เนื่องจาก Node อยู่ในรูปแบบของ Container ทรัพยากรจึงใช้ร่วมกัน แต่เนื่องจากยังเป็นขั้นทดสอบจึงใช้รูปแบบนี้ไปก่อน โดยได้มีการทดลองสั่งงานต่าง ๆ เช่น คำนวณค่า Pi หรือ นับถอยหลังเลข และ ทดสอบว่า GPU ใน Node นั้นสามารถทำงานได้หรือไม่



รูปภาพของหน้าจอ Kubernetes Lens ที่ไว้ตรวจสอบเช็คสภาพ Node และ Pods ที่กำลังทำงาน หรือ ทำงานเสร็จสิ้นแล้ว

ในส่วนของการพัฒนา web application ผู้เรียนได้ทำการกำหนด version ของ library และ framework ที่ต้องการใช้เรียบ เพื่อให้เกิดความคล่องตัวในการทำงานร่วมกันของแต่ละ library และ framework โดยทางผู้เรียนได้กำหนด version หลักของ framework ให้ใช้ดังนี้

- Vue JS version 5.0.8 ในการพัฒนาส่วน Front end
- Django version 4.0.6 .ในการพัฒนาส่วน Back end

โดยที่ทางผู้เรียนได้ทำการเชื่อมต่อส่วนของ Front end และ Back end ด้วย API และทำการเชื่อมต่อกับ Database ที่เป็นของ MongoDB จนทำงานได้เป็นปกติ และเริ่มการพัฒนาส่วน Front end ของหน้า Login และ Sign up เอาไว้เพื่อทำการทดลองส่งข้อมูลกันระหว่าง Front end และ Back end รวมไปถึงมีการทดสอบ Function ในการทำงานส่วนการสั่งทำงานด้วยไฟล์รูปแบบ Kubernetes nginx configuration deployment file ที่เป็นไฟล์นามสกุล .yaml เพื่อรองรับการสั่งทำงาน Image processing application ในส่วนต่อไป

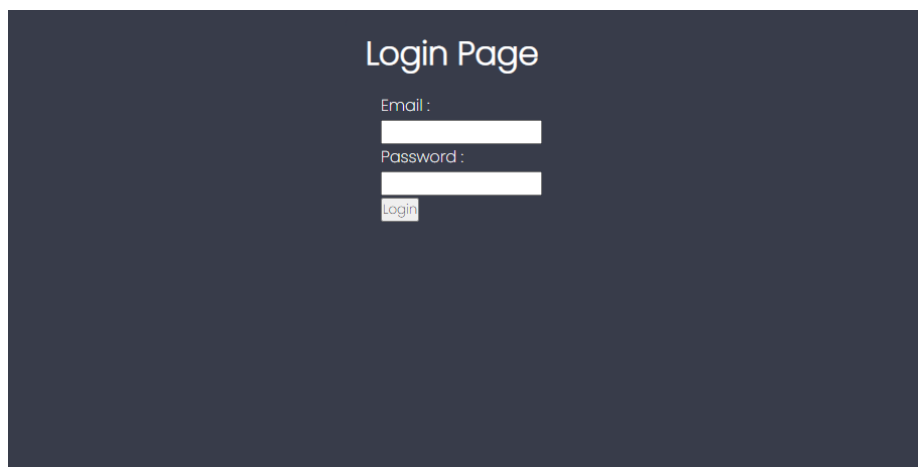
```
apiVersion: batch/v1
kind: Job
metadata:
  name: my-job-id
  namespace: jobdemonamespace
  labels:
    job_name: my-job-id
spec:
  template:
    metadata:
      labels:
        app: my-job-pod-id
        name: my-job-pod-id
    spec:
      containers:
        - image: "shuffler:latest"
          imagePullPolicy: Never
          name: "shuffler"
          command:
            - python3
            - -u
            - ./test.py "userID" "jobID" "appID" "Folder" "Fileseleted"
          args:
            - "Kubernetes"
          restartPolicy: Never
```

รูปของไฟล์คำสั่ง Kubernetes nginx configuration deployment file ที่สร้างออกมาได้

```
def writeConfig(job_id_name, **kwargs):
    template = """apiVersion: batch/v1
kind: Job
metadata:
  name: {job_id}
  namespace: jobdemonamespace
  labels:
    job_name: {job_id}
spec:
  template:
    metadata:
      labels:
        app: my-job-pod-id
    name: my-job-pod-id
    spec:
      containers:
        - image: "shuffler:latest"
          imagePullPolicy: Never
          name: "shuffler"
          command:
            - python3
            - -u
            - ./test.py "{user_id}" "{job_id}" "{app_id}" "{path}" "{img_selected}"
          args:
            - "Kubernetes"
          restartPolicy: Never"""
    with open('yaml_file/'+job_id_name+'.yaml', 'w') as yfile:
        yfile.write(template.format(**kwargs))
```

รูปส่วนของ Source code ที่ใช้ในการสร้าง Kubernetes nginx configuration deployment file

และได้ทำการพัฒนาส่วน Back end ของหน้า Login ขึ้นมาเพื่อใช้ในการทดสอบการทำงานของ API ในด้านความปลอดภัย และทำการสร้างส่วน Front end ของหน้า Login ขึ้นมาด้วยเช่นกัน



รูปของส่วน Front end ของหน้า Login

4. ปัญหาที่เกิดขึ้นและแนวทางการแก้ไข

ปัญหาที่เกิดขึ้นในส่วนการส่วนของ Server และ Kubernetes (MicroK8S) คือ ทรัพยากรของเครื่อง ๆ เดียวยังทำงานได้ไม่เหมาะสมจึงได้หาแนวทางการจำลองเครื่องแบบอื่น ๆ เพื่อให้สามารถนำมาทดสอบ และ ใช้งานได้ในการทำ Project นี้ โดยใช้ K3S เป็น Kubernetes แบบ Lightweight ที่ใช้ทรัพยากรของเครื่องน้อย และ ทำงานได้เหมือนกับตัว Kubernetes ตัวปกติ และได้ใช้ K3D ในการจำลองเครื่องให้อยู่ในรูปของ Container เพื่อจะได้แบ่งเป็น 3 เครื่องเพื่อทดสอบระบบการแบ่งงานให้แต่ละเครื่องได้

ปัญหาที่เกิดขึ้นในส่วนของพัฒนา web application คือการที่ API นั้นตอบสนองได้ค่อนข้างช้าและสิ่งที่ใช้ในการเรียก API มาใช้นั้นค่อนข้างมีความซับซ้อนจริงต้องมีการศึกษาเพิ่มเติมเพื่อเพิ่มความเร็วในการตอบสนองเอง รวมไปถึง ด้านความปลอดภัยที่สามารถทำได้เพียงการเข้ารหัส email และ password ที่ส่งไปทาง API ซึ่งสามารถถูกดักไปได้หากนำไปใช้จริง

5. สิ่งที่จะดำเนินการต่อไป

สิ่งที่จะดำเนินการต่อไปคือ

- เขียนโปรแกรม Docker Image สำหรับแอปพลิเคชัน Image Processing
- ทดสอบแอปพลิเคชัน Image Processing ที่ได้เขียนมาให้ทำงานอยู่ในรูปของ Container
- ทดสอบระบบ Web Service
- ติดตั้ง Database บนระบบ Cluster
- เพิ่มความปลอดภัยของ API ที่ใช้ในการติดต่อกัน ระหว่าง Front end และ Back end
- ปรับปรุงฐานข้อมูลเพื่อให้ตัว Model ข้อมูลนั้นเหมาะสมกับการใช้งานมากขึ้น
- ตกแต่งหน้า Login เพิ่มเติมด้วย CSS
- พัฒนา Web application ในส่วน User management