k8s-work-queue-video-coding

This project deploys a distributed video encoding infrastructure capable of utilizing both GPU-supported and CPU clusters. Depending on the available resources, the deployment can be optimized for hardware with GPU capabilities or solely rely on CPU power. These artifacts are the same as those used to obtain the results presented in our Euro-Par 2024 paper.

Each encoding job encapsulated in a JSON message and is sent to RabbitMQ (that is also deployed inside Kubernetes). The message contains information about the videos to be downloaded and where the encoded videos and files with times should be uploaded. The messages from the RabbitMQ queue are consumed by Pods (managed by a deployment) that perform the encoding.

Please, be aware that the YAML files contains credentials that should be changed.

Table of Contents

Prerequisites

Before deploying the infrastructure, ensure that you meet the following prerequisites:

- **Kubernetes**: You need a Kubernetes cluster ready for deployment. Ensure that your Kubernetes is at least version 1.21, as this is required to support newer APIs and stability improvements. Install Kubernetes.
- **Nvidia Operator**: For GPU encoding, ensure that your Kubernetes cluster is equipped with an operational Nvidia GPU operator. This operator simplifies the management of GPU resources in your Kubernetes cluster. Make sure you have at least version 1.8 of the Nvidia GPU operator installed. Install Nvidia GPU Operator.
- **Python**: The deployment scripts and test scripts require Python. Ensure that you have Python 3.8 or newer installed, as it includes improvements and features necessary for modern applications. Install Python.

It is recommended to verify the compatibility of your system and the required software before proceeding with the installation and deployment of the infrastructure.

Installation

1. **Clone the repository**:

```
git clone github.com:cloudmedialab-uv/k8s-work-queue-video-coding.git
cd k8s-work-queue-video-coding
```

2. **Install Python dependencies**: Ensure Python and necessary libraries are installed by running:

```
pip install -r test/requirements.txt
```

Running the Infrastructure

Depending on your cluster's capabilities (GPU or CPU), apply the appropriate Kubernetes YAML files:

- **For GPU-supported clusters**:

```
kubectl apply -f deploy/gpu/infrastructure.yml
```

- **For CPU-only clusters**

```
kubectl apply -f deploy/cpu/infraestructure.yml
```

Testing

After deploying the appropriate infrastructure, proceed with the following steps:

1. **Obtain the Kubernetes cluster IP**: Update the IP address in the testing scripts to point to your cluster.
2. **Run the test scripts**:

- For GPU:

```
python test/gpu.py <kubernetes_cluster_ip>
```

- For CPU:

```
python test/cpu.py <kubernetes_cluster_ip>
```

3. **Verify the output**: Check the upload container within the infrastructure to ensure that the video has been encoded and uploaded successfully.

Full Example using minikube

This section provides a complete example of setting up and running the CPU-only video encoding infrastructure using Minikube.

Prerequisites for this Example

Before proceeding with the example, ensure you have the following tools installed and configured:

- **Minikube**: Used to create a local Kubernetes cluster for this example. Install Minikube.
- **Python**: Required to run the test scripts. Make sure Python is installed along with pip to handle package installations. Install Python.

1. Start Minikube:

This command starts your Minikube environment.

```
minikube start
```

2. Deploy the infrastructure:

Apply the CPU infrastructure YAML to set up the environment.

```
minikube kubectl -- apply -f deploy/cpu/infraestructure.yml
```

3. Wait until deploy end:

Check the deploy status and wait util all the deployments are ready

```
minikube kubectl -- get deployments --namespace cpu-video-coding
```

Expected output

```
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
ffmpeg-fn             1/1     1            1           5m49s
rabbitmq-deployment   1/1     1            1           5m49s
upload-deployment     1/1     1            1           5m48s
```

4. Install Python dependencies:

Install the required Python libraries needed for the test script.

```
pip install -r test/requirements.txt
```

5. Run the test script:

Execute the Python test script using the IP address of your Minikube cluster (or put the IP of your Kubernetes cluster). The script sends a message to RabbitMQ to encode a video. See the source code of test/cpu.py to see the details of the JSON message.

```
python3 test/cpu.py $(minikube ip)
```

6. Check status:

Run to check the video codify status by accessing to container logs

```
minikube kubectl -- logs deployment/ffmpeg-fn \
    --namespace cpu-video-coding --container ffmpeg-fn
```

Expected output that shows the encoding perfomed

```
024/05/13 13:34:14 /shared/1715607241928
2024/05/13 13:34:14 Job: {-i bbb_30fps_640x360_800k.mp4 -c:v libx264 out.mp4}
2024/05/13 13:34:14 Performing ffmpeg job...
2024/05/13 13:34:14 ffmpeg command: ffmpeg -hide_banner -v quiet -nostats -i bbb_30fps_640x360
```

7. List files in upload container:

Connect to the upload container where the encoded video is stored.an list files in /app/upload folder.

```
minikube kubectl -- exec -it deployment/upload-deployment \
    --namespace cpu-video-coding -- ls /app/upload
```

8. See contents of times file

```
minikube kubectl -- exec deployment/upload-deployment \
    --namespace cpu-video-coding -- cat /app/upload/times.json
```

Sample Output

This is a formated output (using jq):

```
{
  "job": {
    "downloadFiles": [
         "https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps_640x360_800k.mp4"
    ],
    "uploadFiles": [
      "out.mp4"
    ],
    "uploadUrl": "http://MINIKUBE_IP:31808/upload",
    "timesFile": "times.json"
  },
  "tsEventGeneratedTime": 1715607241928,
  "tsReceivedTime": 1715607241931,
  "tsFinalTime": 1715607422020,
  "downloadTime": 12070,
  "processTime": 165935,
  "uploadTime": 2084,
  "container": "ffmpeg-fn-5dfc678bf7-lx8df"
}
```


Citing

If you use this infrastructure please cite it as follows:


```
@inproceedings{Salcedo-Navarro-2024a,
authors={Salcedo Navarro, Andoni and Peña-Ortiz, Raúl and
Claver, José M., and Garcia-Pineda, Miguel and Gutiérrez Aguado, Juan},
title={Cloud-native GPU-enabled architecture for parallel video encoding}
booktitle = {30th International European Conference on Parallel and
Distributed Computing (Euro-Par)},
year = {2024}
}
```