

CTW ML assignment

Solution:

Delivery:

Input/Output:

Scalable architecture

Inference performance

Efficient CPU/GPU utilization

Solution:

1. Utilize "ray" to support the backend model inference computing which can be extended to multi-core /distributed computing
2. It is easy to set the num_replicas/nums_cpu/nums_gpu by ray serve in order to utilize the CPU/GPU resource.
3. Split the translate service through 'ray serve' , like this "serve run -h 0.0.0.0 -p 9527 translate_service:translator"

Delivery:

- **app/Dockerfile:** ml-assignment/app/Dockerfile

Containers can be launched by following commands.

- `docker build . -t ensalty/ml-assignment -f ./ml-assignment/app/Dockerfile`
- `docker run -p 127.0.0.1:8265:8265 -p 127.0.0.1:9527:9527 --cpus=4 --shm-size=2.47gb -it ensalty/ml-assignment:latest /bin/bash`

- **k8s/deployment.yaml:** ml-assignment/k8s/deployment.yaml

container can be deployed on k8s by following command

- `kubectl create -f ml-assignment/k8s/deployment.yaml`

- Other necessary code

Input/Output:

The screenshot shows the Kubernetes dashboard interface. On the left, there's a sidebar with navigation links like Workloads, Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets, Service, Ingresses, and Ingress Classes. The main area displays a 'Shell' session for a pod named 'translator-deployment-5df74449bd-8c7kp'. The terminal output shows a curl command being executed, which sends a POST request to 'http://127.0.0.1:9527/translation' with a JSON payload. The response is a JSON object containing a translated text in Japanese.

```
[root@docker-desktop service]# curl --location --request POST 'http://127.0.0.1:9527/translation' --header 'Content-Type: application/json' -d '{
  "payload": {
    "fromLang": "en",
    "records": [
      {
        "id": "123",
        "text": "Life is like a box of chocolates."
      }
    ],
    "toLang": "ja"
  }
}'
{"result": [{"id": "123", "text": "人生はチョコレートのようなものだ。"}]}[root@docker-desktop service]#
```

Scalable architecture

1. Backend service can be extended to multiple-cores or multiple hosts by configuring params.
2. Ray cluster can be extended by distributed nodes.
3. the number of CPUs/GPUs can be exactly assigned to the specific function.

```
@serve.deployment(num_replicas=1, ray_actor_options={"num_cpus": 1, "num_gpus": 0}, route_prefix='/translation')
class Translator:
    """
    num_replicas determines how many copies of our deployment process run in Ray.
    Requests are load balanced across these replicas, allowing you to scale your deployments horizontally.
    """
```

Inference performance

1. LightSeq can be considered to optimize performance
2. `self.accelerator = Accelerator(split_batches=True, dispatch_batches=False)`

Efficient CPU/GPU utilization

CPU resource can be fully used by setting the `nums_cpu`.

NODES




JOBS


ACTORS

LOGS

EVENTS

METRICS





Node Statistics

TOTAL x 1ALIVE x 1

Node List

HostIPStatePage SizeSort ByReverse:TABLECARD

<1>

Host / Cmd Line	State	ID	IP / PID	CPU Usage	Memory	GPU	GRAM
—docker-desktop	ALIVE	6c86d...	192.168.65.3	26.5%	4.46GB/11.45GB(38.9%)	N/A	N/A
ray::ServeController.listen_for_change()	ALIVE	54354...	239	0%	106.11MB/11.45GB(0.0%)	N/A	N/A
ray::HTTPProxyActor	ALIVE	70c97...	268	0%	106.46MB/11.45GB(0.0%)	N/A	N/A
ray::ServeReplica.Translator.handle_request()	ALIVE	ab70d...	305	0%	2.23GB/11.45GB(0.2%)	N/A	N/A