上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

Serverless Computing Labs

蔡子诺

上海交通大学

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录

01

阿里云无服务器计算平台学习

https://help.aliyun.com/product/50980.html

02

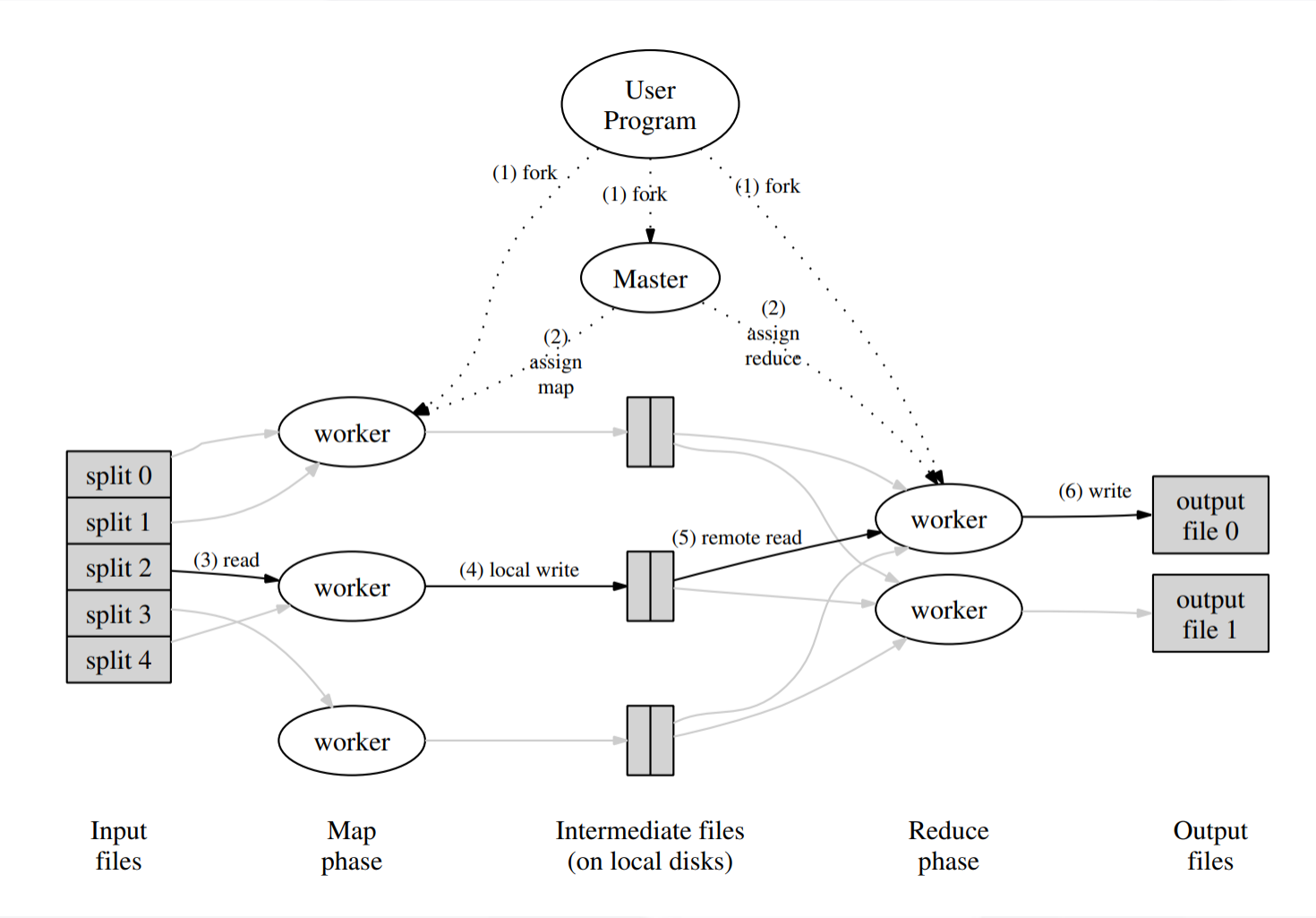
基于 serverless 平台的 MapReduce

- MapReduce

MapReduce Introduction

- MapReduce: Simplified Data Processing on Large Clusters (OSDI)
- https://www.usenix.org/legacy/events/osdi04/tech/full_papers/dean/dean.pdf
- MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a _map_ function that processes a key/value pair to generate a set of intermediate key/value pairs, and a _reduce_ function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.
- Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.
- Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

MapReduce Framework



Programming Model

map_reduce.py > ...

```python
import minio


def main(params):
    role = params['role']

    if role == 'map':
        do_map(params)
    elif role == 'reduce':
        do_reduce(params)
    elif role == 'master':
        do_master(params)
    else:
        print('error role')

    return {}


def do_map(params):...


def do_reduce(params):...


def do_master(params):...
```

Requirements

- Implement a map-reduce framework to handle **word frequency** task.
- How many **map and reduce functions** do you set? How do they affect the completion time?
- Can you make a summary of the function execution time and communication time ratio?
- What is the memory consumption of each function?
- How the **memory/cpu** parameter influences the execution time?
  - Design an automatic algorithm to find the minimal cost: 贝叶斯优化、机器学习、启发式算法

03

基于 serverless 平台的高维矩阵运算

请在此输入文字说明

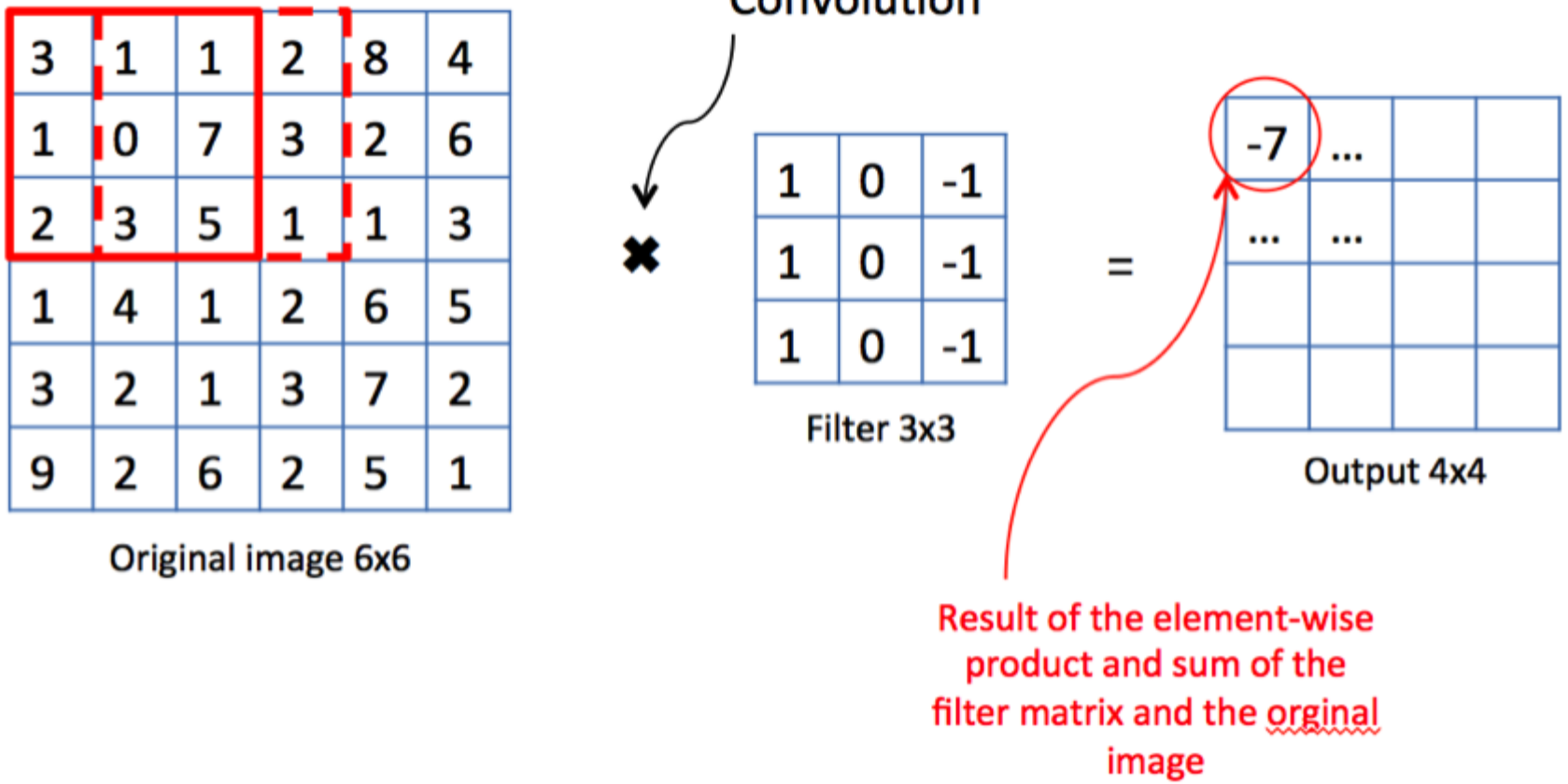
矩阵运算

Matrix multiplication

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

矩阵计算

Convolution / Pooling



Programming Model

```python
import numpy as np


def main(params):
    pass


def worker(params):
    pass


def master(params):
    pass
```

Requirements

- Implement a matrix multiplication and convolution framework.
- How many worker functions do you set? How do they affect the completion time?
- Can you make a summary of the function execution time and communication time ratio?
- What is the memory consumption of each function?
- How the memory/cpu parameter influences the execution time?

04

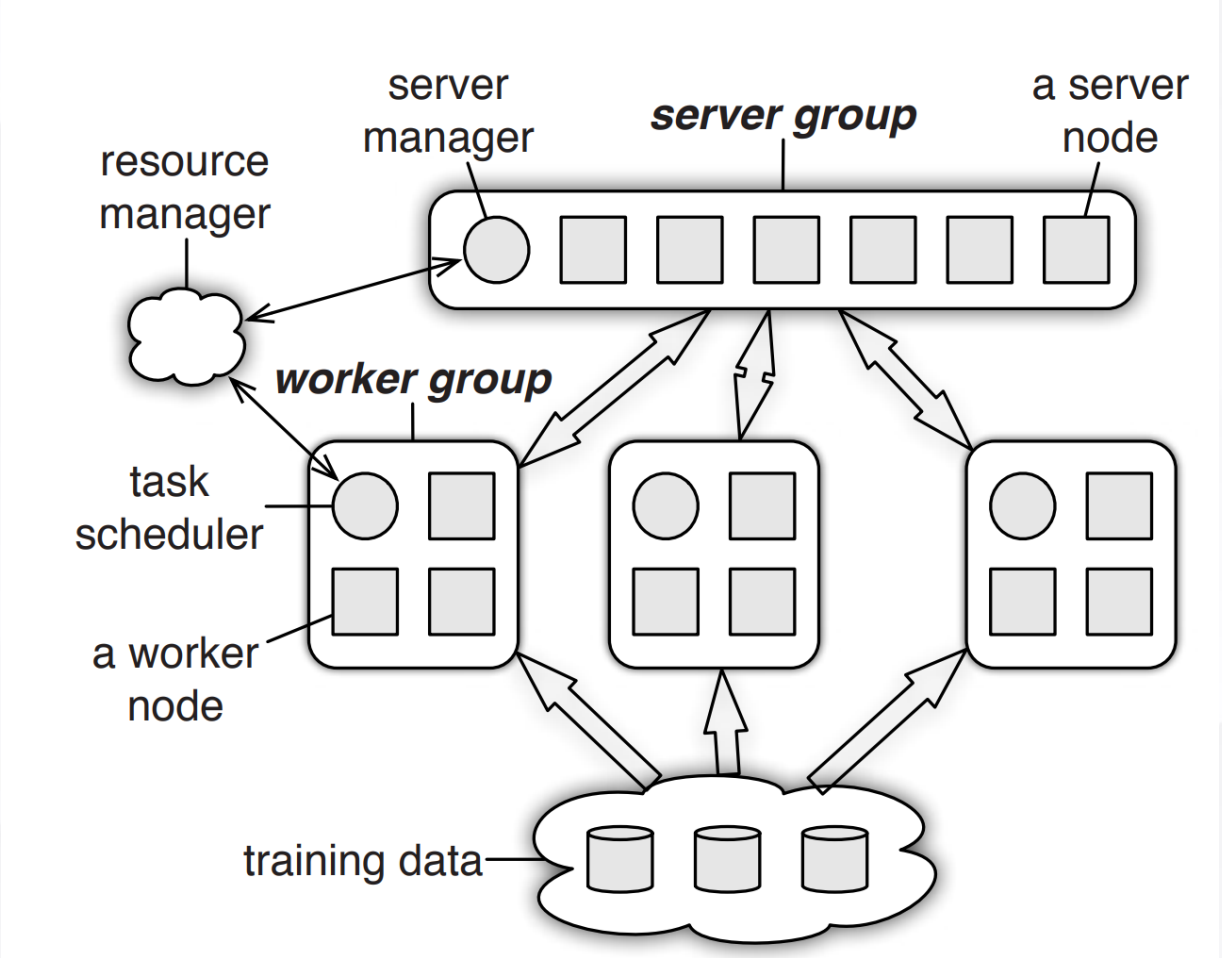
基于 serverless 平台的分布式机器学习训练

请在此输入文字说明

Parameter Server

- Scaling Distributed Machine Learning with the Parameter Server
- https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-li_mu.pdf
- We propose a parameter server framework for distributed machine learning problems. Both data and workloads are distributed over worker nodes, while the server nodes maintain globally shared parameters, represented as dense or sparse vectors and matrices. The framework manages asynchronous data communication between nodes, and supports flexible consistency models, elastic scalability, and continuous fault tolerance.
- To demonstrate the scalability of the proposed framework, we show experimental results on petabytes of real data with billions of examples and parameters on problems ranging from Sparse Logistic Regression to Latent Dirichlet Allocation and Distributed Sketching.

Framework


server
manager
server group
a server
node
resource
manager
worker group
task
scheduler
a worker
node
training data

Serverless Framework

- https://arxiv.org/abs/2105.07806

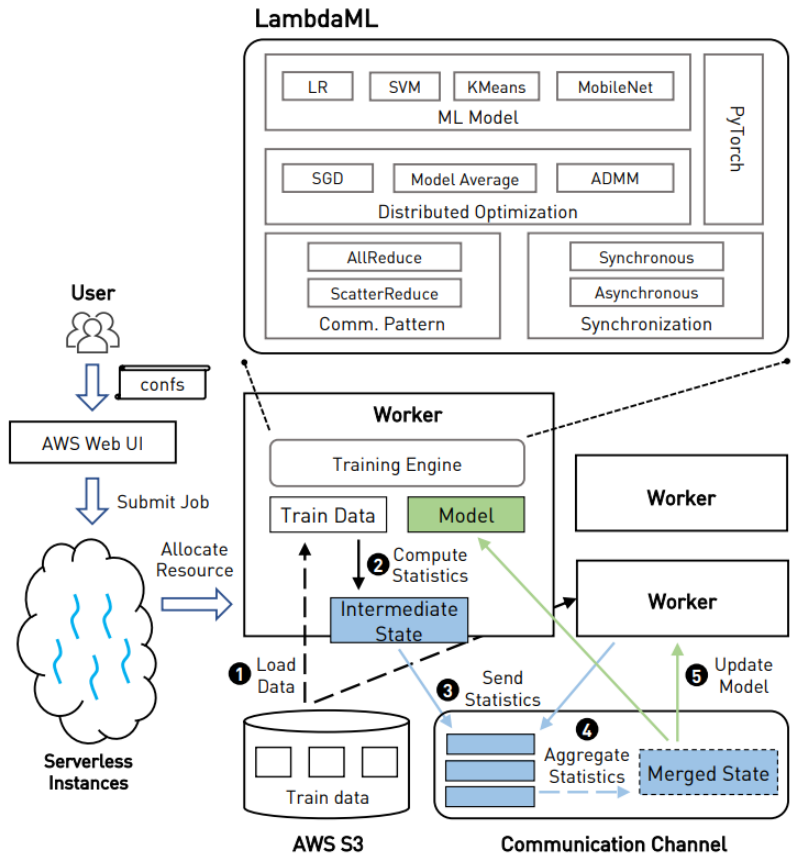


Figure 2: Framework of LambdaML.

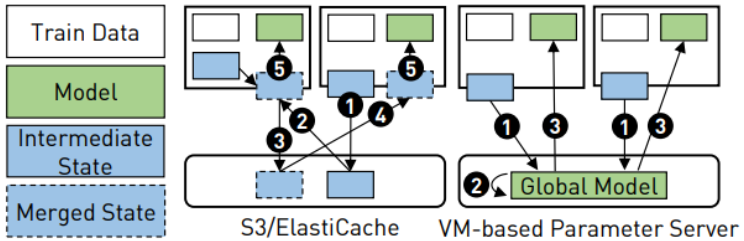


Figure 3: An FaaS-based data aggregation.

Requirements

- Implement a serverless distributed training framework.
- How many worker functions do you set? How do they affect the completion time?
- Can you make a summary of the function execution time and communication time ratio?
- What is the memory consumption of each function?
- How the memory/cpu parameter influences the execution time?

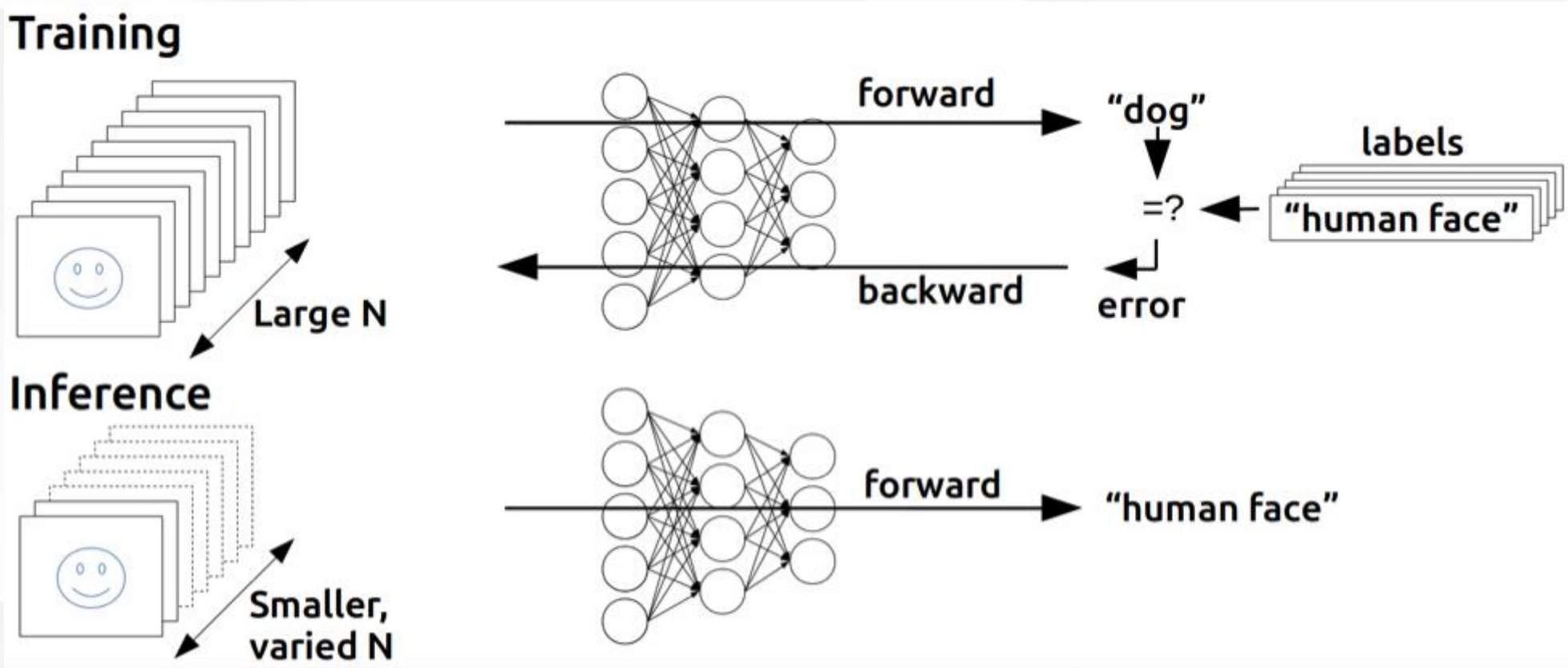
05

基于 serverless 平台的机器学习推理任务

请在此输入文字说明

ML Inference

- Configure memory size to compare performance
- Add more workloads to observe the scaling



Gillis: Serving Large Neural Networks in Serverless Functions with Automatic Model Partitioning

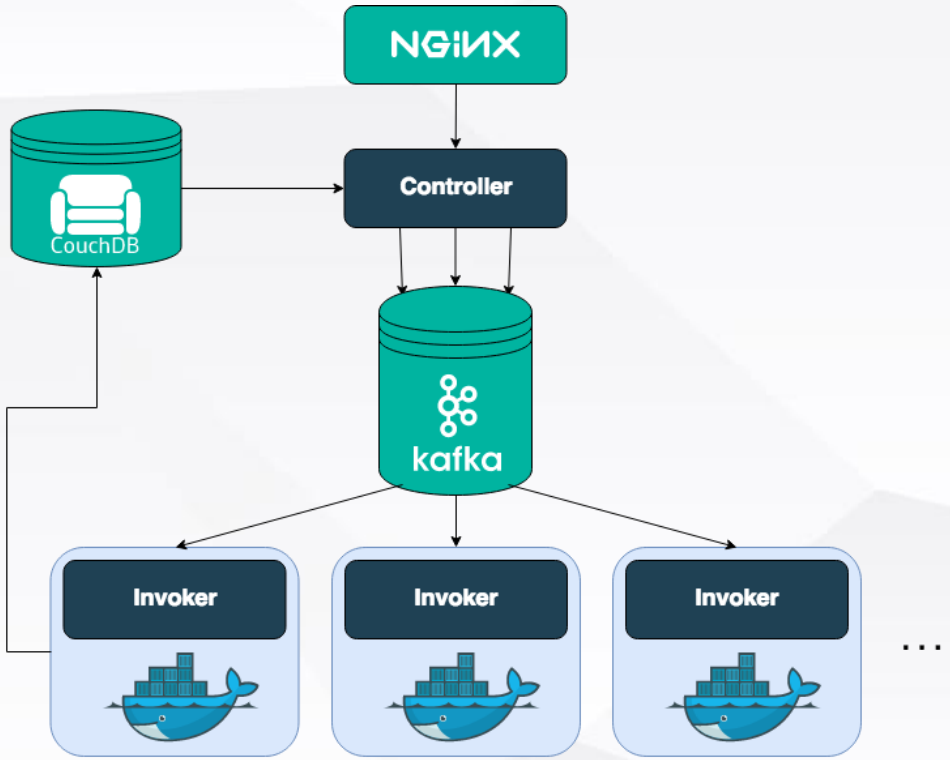
- The increased use of deep neural networks has stimulated the growing demand for cloud-based model serving platforms. Serverless computing offers a simplified solution: users deploy models as serverless functions and let the platform handle provisioning and scaling. However, serverless functions have constrained resources in CPU and memory, making them inefficient or infeasible to serve large neural networks-which have become increasingly popular. In this paper, we present Gillis, a serverless-based model serving system that automatically partitions a large model across multiple serverless functions for faster inference and reduced memory footprint per function. Gillis employs two novel model partitioning algorithms that respectively achieve latency-optimal serving and cost-optimal serving with SLO compliance. We have implemented Gillis on three serverless platforms-AWS Lambda, Google Cloud Functions, and KNIX-with MXNet as the serving backend. Experimental evaluations against popular models show that Gillis supports serving very large neural networks, reduces the inference latency substantially, and meets various SLOs with a low serving cost.

Requirements

- Implement a serverless distributed inference framework.
- How many worker functions do you set? How do they affect the completion time?
- Can you make a summary of the function execution time and communication time ratio?
- What is the memory consumption of each function?
- How the memory/cpu parameter influences the execution time?


Q&A
饮 水 思 源 爱 国 荣 校

OpenWhisk




```
[zhangrenjun@Octopus ~]$ wsk

Usage:
  wsk [command]

Available Commands:
  action      work with actions
  activation  work with activations
  api         work with APIs
  help        Help about any command
  list        list entities in the current namespace
  namespace   work with namespaces
  package     work with packages
  project     The OpenWhisk Project Management Tool
  property    work with whisk properties
  rule        work with rules
  sdk         work with the sdk
  trigger     work with triggers

Flags:
      --apihost HOST         whisk API HOST
      --apiversion VERSION   whisk API VERSION
  -u, --auth KEY             authorization KEY
      --cert string          client cert
  -d, --debug                debug level output
  -h, --help                 help for wsk
  -i, --insecure             bypass certificate checking
      --key string           client key
  -v, --verbose              verbose output
```

wsk action

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk action -h
work with actions
Usage:
  wsk action [command]

Available Commands:
  create      create a new action
  delete      delete action
  get         get action
  invoke      invoke action
  list        list all actions in a namespace or actions contained in a package
  update      update an existing action, or create an action if it does not exist

Flags:
  -h, --help   help for action

Global Flags:
      --apihost HOST         whisk API HOST
      --apiversion VERSION   whisk API VERSION
  -u, --auth KEY             authorization KEY
      --cert string          client cert
  -d, --debug                debug level output
  -i, --insecure             bypass certificate checking
      --key string           client key
  -v, --verbose              verbose output

Use "wsk action [command] --help" for more information about a command.
```

wsk action create

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk action create -h
create a new action
Usage:
  wsk action create ACTION_NAME ACTION [flags]

Flags:
  -a, --annotation KEY VALUE   annotation values in KEY VALUE format
  -A, --annotation-file FILE   FILE containing annotation values in JSON format
  -c, --concurrency LIMIT      the maximum intra-container concurrent activation LIMIT for the action (default 1)
      --copy                   treat ACTION as the name of an existing action
      --docker string          use provided docker image (a path on DockerHub) to run the action
  -h, --help                   help for create
      --kind KIND              the KIND of the action runtime (example: swift:default, nodejs:default)
  -l, --logsize LIMIT          the maximum log size LIMIT in MB for the action (default 10)
      --main string            the name of the action entry point (function or fully-qualified method name when applicable)
  -m, --memory LIMIT           the maximum memory LIMIT in MB for the action (default 256)
      --native                 treat ACTION as native action (zip file provides a compatible executable to run)
  -p, --param KEY VALUE        parameter values in KEY VALUE format
  -P, --param-file FILE        FILE containing parameter values in JSON format
      --sequence               treat ACTION as comma separated sequence of actions to invoke
  -t, --timeout LIMIT          the timeout LIMIT in milliseconds after which the action is terminated (default 60000)
      --web string             treat ACTION as a web action, a raw HTTP web action, or as a standard action; yes | true = web action, raw = raw HTTP
 web action, no | false = standard action
      --web-secure SECRET      secure the web action. where SECRET is true, false, or any string. Only valid when the ACTION is a web action
```

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk action create hello_world hello_world.py -i
ok: created action hello_world
```

wsk action invoke

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk action invoke -h
invoke action
Usage:
  wsk action invoke ACTION_NAME [flags]

Flags:
  -b, --blocking           blocking invoke
  -h, --help               help for invoke
  -p, --param KEY VALUE    parameter values in KEY VALUE format
  -P, --param-file FILE    FILE containing parameter values in JSON format
  -r, --result             blocking invoke; show only activation result (unless there is a failure)
```

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk action -i  invoke hello_world -p name zinuo
ok: invoked /_/hello_world with id c80ef2a5252842e58ef2a5252852e5f5
```

wsk activation

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk activation -h
work with activations
Usage:
  wsk activation [command]

Available Commands:
  get         get activation
  list        list activations
  logs        get the logs of an activation
  poll        poll continuously for log messages from currently running actions
  result      get the result of an activation

Flags:
  -h, --help   help for activation
```

wsk activation list

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk activation list -h
list activations
Usage:
  wsk activation list [NAMESPACE or NAME] [flags]
```

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk activation list -i
Datetime            Activation ID                  Kind      Start Duration   Status          Entity
2022-07-23 20:13:44 08c7ec04c75f4e6087ec04c75f7e6019 python:3 cold  97ms       success         guest/hello_world:0.0.1
2022-07-23 20:09:30 c80ef2a5252842e58ef2a5252852e5f5 python:3 warm  0s         internal error guest/hello_world:0.0.1
```

wsk activation get

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk activation get -h
get activation
Usage:
  wsk activation get (ACTIVATION_ID | --last) [FIELD_FILTER] [flags]
```

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk activation get 08c7ec04c75f4e6087ec04c75f7e6019 -i
ok: got activation 08c7ec04c75f4e6087ec04c75f7e6019
{
    "namespace": "guest",
    "name": "hello_world",
    "version": "0.0.1",
    "subject": "guest",
    "activationId": "08c7ec04c75f4e6087ec04c75f7e6019",
    "start": 1658578424894,
    "end": 1658578424991,
    "duration": 97,
    "statusCode": 0,
    "response": {
        "status": "success",
        "statusCode": 0,
        "success": true,
        "result": {
            "greeting": "hello, zinuo"
        }
    },
    "logs": [],
```

success

wsk activation get

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ wsk activation get c80ef2a5252842e58ef2a5252852e5f5 -i
ok: got activation c80ef2a5252842e58ef2a5252852e5f5
{
    "namespace": "guest",
    "name": "hello_world",
    "version": "0.0.1",
    "subject": "guest",
    "activationId": "c80ef2a5252842e58ef2a5252852e5f5",
    "start": 1658578170123,
    "end": 1658578170123,
    "duration": 0,
    "statusCode": 3,
    "response": {
        "status": "whisk internal error",
        "statusCode": 0,
        "success": false,
        "result": {
            "error": "Failed to run container with image 'openwhisk/action-python-v3.7:1.17.0'."
        }
    },
    "logs": [],
```

failure

基于容器的开发环境

Build Docker Image

```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ docker build -t myenv:v1 .
Sending build context to Docker daemon  122.9kB
Step 1/22 : FROM golang:1.15 AS builder_source
 ---> 40349a2425ef
Step 2/22 : RUN go env -w GO111MODULE=on && go env -w GOPROXY="https://goproxy.io,direct"
 ---> Using cache
 ---> 0a459053855d
```

Kind load docker-image

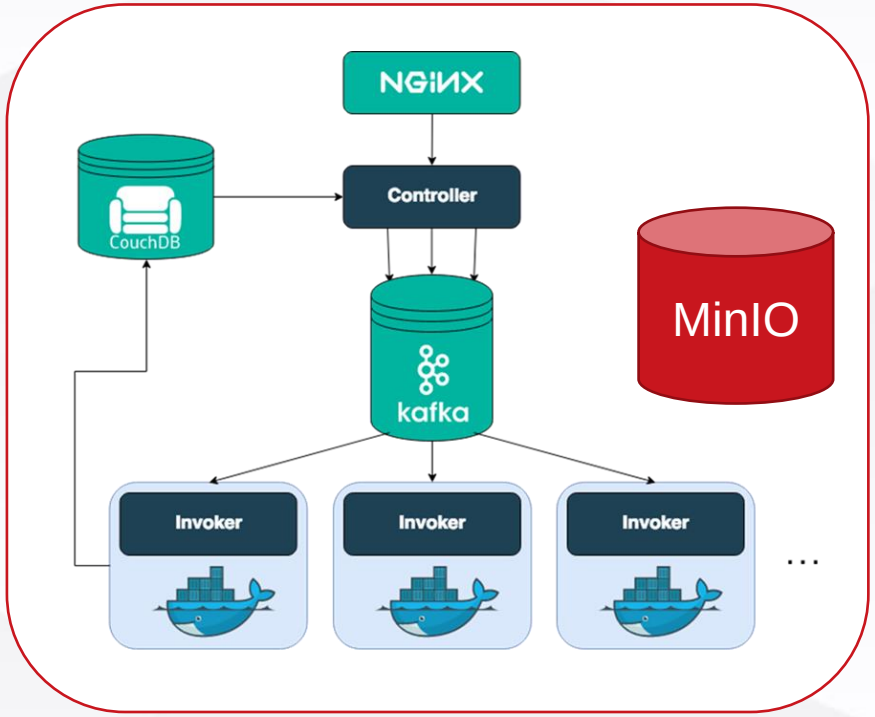
```
dashuju@iZuf6dmz3aabl13vvtg3myZ:~/caizinuo$ kind load docker-image myenv:v1
Image: "myenv:v1" with ID "sha256:c487ab5216b53320b1f681eebc3add99ee115219cf399d0d9bd90c
kind-worker", loading...
Image: "myenv:v1" with ID "sha256:c487ab5216b53320b1f681eebc3add99ee115219cf399d0d9bd90c
kind-worker2", loading...
Image: "myenv:v1" with ID "sha256:c487ab5216b53320b1f681eebc3add99ee115219cf399d0d9bd90c
kind-control-plane", loading...
```

Create function with the specified image

Invoke the function

数据交互

- export POD_NAME=$(kubectl get pods --namespace openwhisk -l "release=minio-1658640199" -o jsonpath="{.items[0].metadata.name}")
- kubectl port-forward $POD_NAME 9000 --namespace openwhisk
- IP: 127.0.0.1:9000 (out of cluster) / 10.96.136.135:9000
- Access key: `AKIAIOSFODNN7EXAMPLE`
- Secret key: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`



数据交互

```python
from minio import Minio


def main(params):
    minioClient = Minio('10.96.136.135:9000',
                        access_key='AKIAIOSFODNN7EXAMPLE',
                        secret_key='wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY',
                        secure=False)

    buckets = minioClient.list_buckets()

    for bucket in buckets:
        print(bucket.name, bucket.creation_date)

    if minioClient.bucket_exists('test'):
        print('bucket test already exists')
    else:
        minioClient.make_bucket('test')
        print('make bucket test')

    minioClient.fput_object('test', 'hello_world', 'hello_world.py')

    return {'greeting': 'hello, world'}
```

总结

- **OpenWhisk**
- **Wsk action**
  - Create
  - Invoke
- **Wsk activation**
  - List
  - Get
- **基于容器的开发环境**
- **数据交互**