分布式系统第一次作业

q2文档部分

1752854

赵羿昕

# 代码环境：

Python3.7

# 使用库：

import mmap

import os

import time

import BytesWrite

import socketserver

import socket

import threading

from mpl\_toolkits.mplot3d import axes3d #画3d图

import matplotlib.pyplot as plt #画图用

from multiprocessing import Process #main.py使用多进程开启服务器和客户端

# 思路：

客户端和服务器端使用多线程发送和搜索单词，提高效率，客户端发送时将数据根据线程数分开发送，服务端接收到数据后根据发送数和线程数分配线程给数据，搜索结束后按接收顺序排序返回客户端，客户端接收消息后再按开始的数据顺序排序写入log中。

# 使用方法：

输入2个数：服务器端线程数和客户端线程数，使用回车隔开

输入1-5即对应线程数

如输入0\n0\n，则为客户端自动从1-5增加，服务器端每逢客户端到5线程增加线程，自动完成25次测试并返回3d图像

# 核心函数：

## 服务器端

1. 服务器初始化：

def init():

    """

    初始化服务端

    """

    global g\_socket\_server

    g\_socket\_server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  # 创建 socket 对象

    g\_socket\_server.bind(ip\_port)

    g\_socket\_server.listen(5)  # 最大等待数

    print("服务端已启动，等待客户端连接...")

2.接收新连接：

def accept\_client():

while True:

        if len(g\_conn\_pool) < thread\_max:

            client, \_ = g\_socket\_server.accept()  # 阻塞，等待客户端连接

            # 加入连接池

            g\_conn\_pool.append(client)

            # 给每个客户端创建一个独立的线程进行管理

            thread = threading.Thread(target=message\_handle, args=(client,))

            thread.setDaemon(True)

            thread.start()

        else:

            time.sleep(0.1)

1. 消息处理函数：

当收到消息时，先判断是否为关键词，再根据当前已被占用的线程数和总可用线程数，使用当前连接开启空余线程用来计算

def message\_handle(client):

    """

    消息处理

    """

    global thread\_max

    global thread\_count

    global flag

    while True:

        bytes = client.recv(1024)                       #客户端消息

        if bytes.decode(encoding='utf8') == '5':        #接收到客户端线程数达到5的消息

            thread\_max += 1                             #增加自己的线程上限

            rst = 'ok'

            #print(thread\_max)

            client.sendall(rst.encode(encoding='utf8'))

        elif bytes.decode(encoding='utf8') == 'end\n':    #接收到客户端的结束信号

            rst = 'end'

            client.sendall(rst.encode(encoding='utf8'))

            flag = 1                                    #结束进程

        elif len(bytes) == 0:

            client.close()

            # 删除连接

            g\_conn\_pool.remove(client)

            break

        else:

            data = bytes.split()

            while True:                                 #根据当前被占用线程数以及当前连接数

                if thread\_count >= thread\_max:

                    time.sleep(0.1)

                else:

                    myCount = thread\_max - thread\_count

                    break

            #print(myCount)

            bag = DivideWords(data, myCount)            #分解待搜索词

            mythread\_pool = []

            for i in bag:

                mythread\_pool.append(Search(i))

                thread\_count += 1

            msg = ''

            for i in mythread\_pool:

                i.start()

            for i in mythread\_pool:

                i.join()

                msg += i.get\_result()

            del mythread\_pool

            client.sendall(msg.encode(encoding='utf8'))

3.搜索线程类：

class Search(threading.Thread):

    def \_\_init\_\_(self, arg):

        super(Search, self).\_\_init\_\_()

        self.arg = arg

    def run(self):

        self.result = SearchWords(self.arg)

    def get\_result(self):

        return self.result

## 客户端：

1.发信函数：

def searchWords(message, full = '0'):

    sendMessage = ''

    for i in message:

        sendMessage += i

        sendMessage += '\n'

    tcp\_client = socket(AF\_INET,SOCK\_STREAM)

    tcp\_client.connect(ip\_port)

    if full == '0':

        tcp\_client.sendall(sendMessage.encode("utf-8"))

    else:

        sendMessage = full

        tcp\_client.sendall(sendMessage.encode("utf-8"))

    data = tcp\_client.recv(buffer\_size)

    if data.decode("utf-8") == 'end' or data.decode("utf-8") == 'ok':

        tcp\_client.close()

    else:

        tcp\_client.close()

        return data.decode("utf-8").strip('\n')

2.绘图函数：

def draw(paint):

    i = 1

    x = []

    y = [[] for i in range(5)]

    while i < 6:

        j = 1

        x.append([i, i, i, i, i])

        while j < 6:

            y[i - 1].append(j)

            j += 1

        i += 1

    fig = plt.figure()

    ax = fig.gca(projection='3d')

    ax.set\_title("server/client time")

    ax.set\_xlabel("threads - server")

    ax.set\_ylabel("threads - client")

    ax.set\_zlabel("time")

    figure1 = ax.plot(x[0], y[0], paint[0], c='g')

    figure2 = ax.plot(x[1], y[1], paint[1], c='b')

    figure3 = ax.plot(x[2], y[2], paint[2], c='y')

    figure4 = ax.plot(x[3], y[3], paint[3], c='orange')

    figure5 = ax.plot(x[4], y[4], paint[4], c='r')

    plt.savefig(module\_path + "out\\result.png")

plt.show()

3.客户端主函数：

def clientMain(ths=0):

    if ths == 0:

        n = 1

        count = 1

        paint = [[] for i in range(5)]

        while n < 6 and count < 6:

            li = []

            bag = DivideWords(msg, n)

            start = time.time()

            rst = ''

            for i in range(n):

                li.append(Search(bag[i], '0'))

            for i in li:

                i.setDaemon(True)

                i.start()

            for i in li:

                i.join()

                rst += i.get\_result()

            end = time.time()

            past = end - start

            l = open(pathLog, 'a+')

            print(rst, file= l)

            print(past, file= l)

            l.close()

            paint[count-1].append(past)

            print(n)

            print(past)

            if n == 5 and count < 25:

                searchWords('5', '5')

                n = 1

                count += 1

                print(count)

            elif n < 5 and count < 6:

                n += 1

            elif count >= 6:

                break

        searchWords(['end'], '0')

        draw(paint)

    else:

        li = []

        n = ths

        bag = DivideWords(msg, n)

        start = time.time()

        rst = ''

        for i in range(n):

            li.append(Search(bag[i], '0'))

        for i in li:

            i.setDaemon(True)

            i.start()

        for i in li:

            i.join()

            rst += i.get\_result()

        end = time.time()

        past = end - start

        searchWords(['end'], '0')

        l = open(pathLog, 'a+')

        print(rst, file= l)

        print(past, file= l)

        l.close()

# 执行结果（用自己的数据绘的图，根据新数据生成的图片在out文件夹下）：

