NIS2331《计算机组成与系统结构》

ARM 汇编程序实验作业 2

1. 实验目标

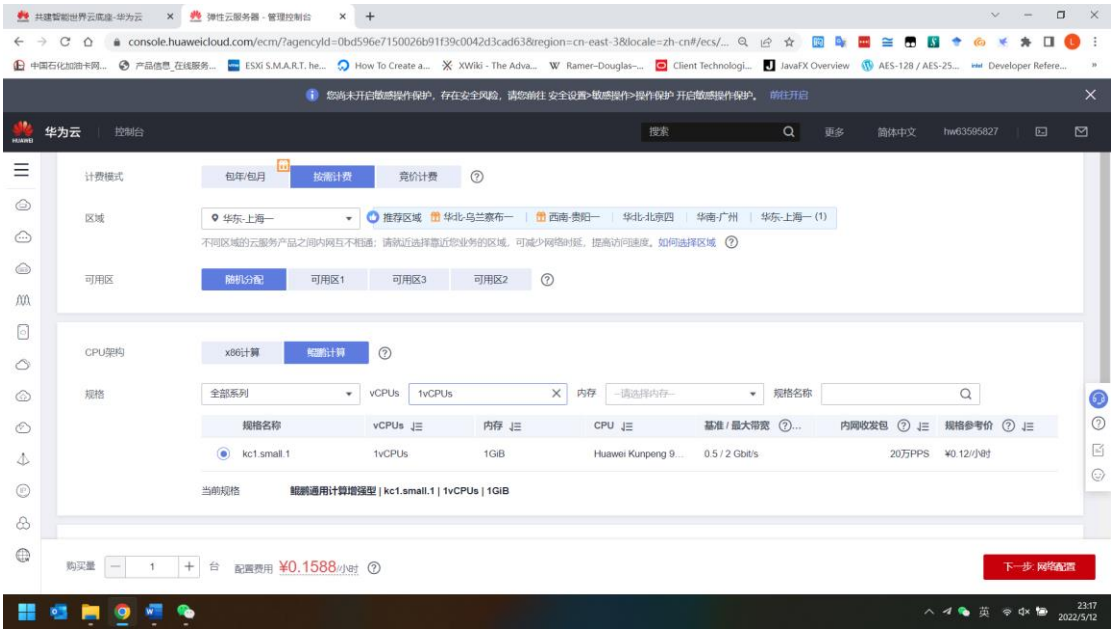
通过 ARMv8 NEON 汇编程序实现两个矩阵的乘法操作，观察 SIMD 指令集相对于基本 C 语言实现的性能提升。

注：矩阵的行与列规模均为 4 的整数倍。

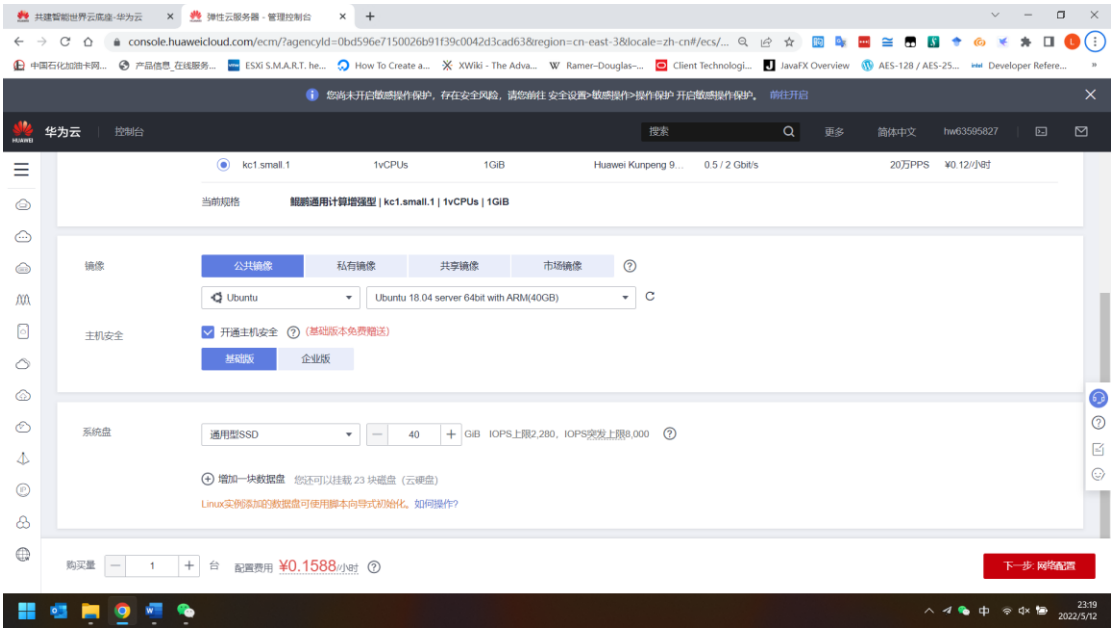
2. 实验环境

鲲鹏云弹性云服务器，Ubuntu 18.04 公共镜像操作系统。具体说明如下：

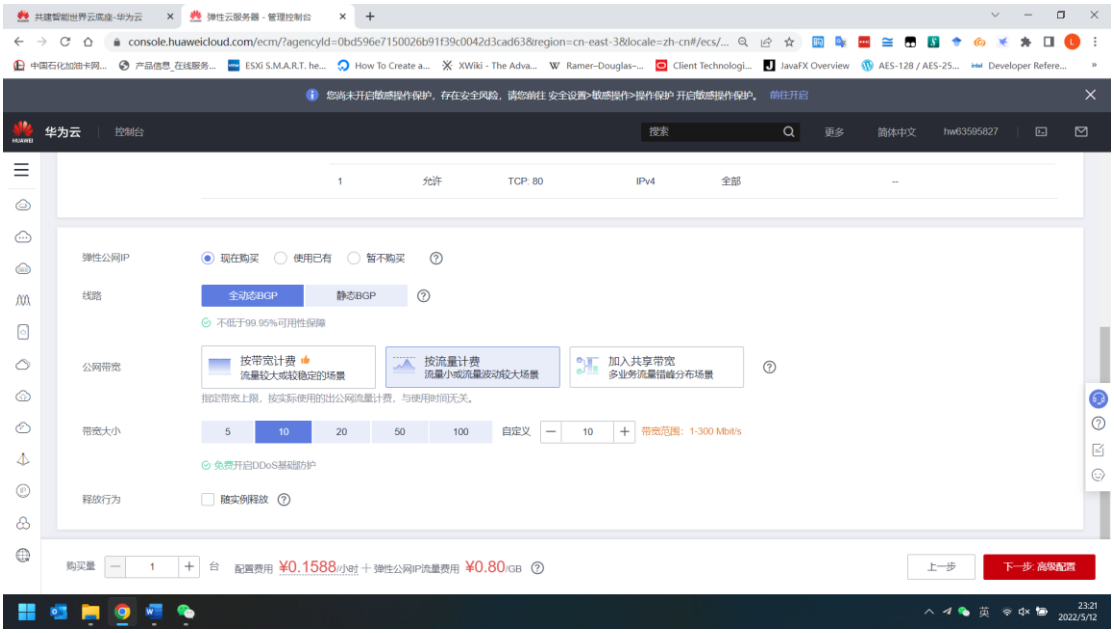
1) 访问华为云 https://www.huaweicloud.com/，注册华为用户，登录，并在账户中心处完成实名认证；
2) 扫描华为云资源预申请二维码，使用实名认证的华为账户登录，并填写问卷；
3) 华为云资源代金券会在审核后发放至申请的华为账户；
4) 收到代金券后，在弹性云服务器菜单下，创建鲲鹏云服务器；配置方面，选择“按需分配”，“鲲鹏计算”，“kc1.small.1”



操作系统选择“公共镜像”下的“Ubuntu 18.04 Server 64bit with ARM”，系统盘为 40GB 通用型 SSD：



按流量计费，10Mbit/s 带宽：



5) 创建成功后，远程登录到云服务器上进行实验操作。

3. 实验方案

3.1 前置

在远程 ubuntu 系统上安装 gcc 和 make 包。

3.2 文件说明

实验提供的代码文件包括 main.c、matrix.h、matrix_mul.c、matrix_mul_asm.s 和 makefile。

3.1.1. matrix_mul.c

为 C 语言程序，提供了一个 C 语言实现的矩阵乘法函数：

```
int matrix_mul(Matrix *results, Matrix *source1, Matrix *source2);
```

参数及返回值说明：

a) 参数 source1：源矩阵 1 结构体；

b) 参数 source2：源矩阵 2 结构体；

c) 参数 results：结果矩阵结构体；

d) 返回值：0 表示执行成功；1 表示执行失败。

3.1.2. matrix.h

定义了矩阵规模以及矩阵结构体 Matrix；

```
typedef struct Matrix {
        int         row;            //Number of rows
        int         column;         //Number of columns
        const int*      data;       //Address of matrix data
}Matrix;
```

这里 data 成员指针指向一个二位数组的地址。

3.1.3. main.c

为 C 语言程序，程序入口。其 main 函数说明如下：

a) 定义了两个源矩阵结构体 source1 和 source2，使用 matrix.h 中定义的规模，并为其附随机数值；

b) 定义了结果矩阵结构体 results1 和 results2，使用 matrix.h 中定义的规模，并预分配空间；

c) 将结果矩阵数据部分内存空间清 0；

d) 调用 matrix_mul 使用 C 语言实现的对两个源矩阵进行乘法操作，结果存放在结果矩阵 results1 中；

e) 调用 matrix_mul_asm.s 中使用 ARMv8 NEON 实现的两个源矩阵进行乘法操作，结果存放在结果矩阵 results2 中；

f) 显示上述两种矩阵乘法的操作时间；

g) 比较 results1 和 results2，如果相同则说明两个矩阵相同，即在该轮测试中，结果正确；如果不同，则说明我们实验所实现的汇编代码有误，请调试。

matrix_mul_asm.s：实现一个矩阵乘法函数 matrix_mul_asm；

1) 定义了代码区域(.text)，对外导出了 matrix_mul_asm 符号(.global matrix_mul_asm)。

2) 提供了函数返回。

矩阵乘法功能待实验开发。

3.3 实验流程

1) 在 matrix_mul_asm.s 的代码区域实现使用 ARMv8 NEON 的矩阵乘法函数 matrix_mul_asm；

2) 使用命令 make 进行编译和链接，成功后运行可执行程序 matrix_mul，查

看矩阵乘法是否正确，以及性能提升程度：

```
longsky@ecs-441b:~/ARMLearning/matrix_mul$ ./matrix_mul
C multiplication took 159829 clock.
SIMD multiplication took 50458 clock.
SIMD implementation provides correct results.
```

从上述运行结果来看，SIMD 实现将矩阵乘法执行时间从近 16 万 clock 缩减至 5 万 clock；矩阵乘法结果正确(provides correct results)。

3) 提交实现了矩阵乘法的 matrix_mul_asm.s。

3.4 调试

代码编译通过后，可通过 gdb 进行通过观察寄存器值，内存值，单步跟踪等手段进行调试排错（GDB 调试方法见参考文献）。

3.5 参考文献

Canvas 中，“文件”→“ARM Programming”目录下：

ARM® Cortex®-A Series Version 1.0 Programmer's Guide for ARMv8-A.pdf 文件中给出了 ARMv8 Cortex-A 的开发人员指引，包含体系结构、寄存器、指令集（包括 NEON 指令集）等内容的说明，可以参考该文献，使用更为丰富的指令集完成本实验。

Coding for Neon.pdf 文件给出了 ARMv8 下 NEON 开发的一个 Tutorial，容易上手学习，同时也给出了使用 NEON 进行矩阵乘法的一个思路。

Debugging with gdb.pdf 文件给出了 GDB 调试的完整手册。

https://azeria-labs.com/debugging-with-gdb-introduction/提供了一个 GDB 调试的上手 Tutorial，容易上手学习。