《并行计算》上机报告

姓名：	魏钊	学号：	PB18111699	日期：	2021.6.14
上机题目：	KungPeng 运算系统体验				

实验环境：
CPU：鲲鹏计算　　；　　规格：kcl.large.2 ;
服务器数：2　　；　　系统盘：40GB　；
操作系统：openEuler;　　软件平台：ubuntu;

一、算法设计与分析：

题目一:

为了完成实验，需要：

1. 购买并配置好服务器属性。
2. 使用 Ubuntu 通过 ssh 连接远程服务器
3. 连接上后对各服务器进行配置，安装实验软件，使能运行实验代码。
4. 编写各实验文档，能够进行并行的快速排序，完成准备工作。
5. 运行代码，完成实验。

二、核心代码：

代码已给出故此处不做过多说明。

三、结果与分析：

题目一:

```
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 1
1253 8000000
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 2
626 8000000
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 3
632 8000000
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 4
638 8000000
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 5
618 8000000
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 6
638 8000000
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 7
644 8000000
[ca@ecs-hw-0002 quicksort]$ bash run.sh quicksort 8
662 8000000
```

通过上述运行，可以看出快排算法程序已经在集群中并行运行起来。大致从整体

可以看出，随着进程数量的增加，耗时越来越少。从开始的 1250 减少到 630 左右。

四、备注（* 可选）：

有可能影响结论的因素：

总结：

本次实验难度不大，跟着实验文档做可以很快的完成。主要是通过在鲲鹏 cpu 上进行并行计算来体验其优化的效果。

附录（源代码）

算法源代码（C/C++/JAVA 描述）

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

#include <iostream>

#include "omp.h"

using namespace std;

void QuickSort(int *&array, int len) {
    if (len <= 1) return;
    int pivot = array[len / 2];
    int left_ptr = 0;
    int right_ptr = len - 1;
    while (left_ptr <= right_ptr) {
        while (array[left_ptr] < pivot) left_ptr += 1;
        while (array[right_ptr] > pivot) right_ptr -= 1;
        if (left_ptr <= right_ptr) {
            swap(array[left_ptr], array[right_ptr]);
            left_ptr += 1;
            right_ptr -= 1;
        }
    }
    int *sub_array[] = {array, &(array[left_ptr])};
    int sub_len[] = {right_ptr + 1, len - left_ptr};

#pragma omp task default(none) firstprivate(sub_array, sub_len)
    { QuickSort(sub_array[0], sub_len[0]); }
#pragma omp task default(none) firstprivate(sub_array, sub_len)
    { QuickSort(sub_array[1], sub_len[1]); }
    // for (int i = 0; i < 2; i++) QuickSort(sub_array[i], sub_len[i]);
}
```

```
int main(int argc, char *argv[]) {
    srand(time(NULL));
    if (argc != 3) {
        cout << "Usage: " << argv[0] << " thread-num array-len\n";
        exit(-1);
    }
    int t = atoi(argv[1]);
    int n = atoi(argv[2]);
    int *array = new int[n];
    omp_set_num_threads(t);

    unsigned int seed = 1024;
#pragma omp parallel for
    for (int i = 0; i < n; i++) array[i] = rand_r(&seed);

    struct timeval start, stop;
    gettimeofday(&start, NULL);

#pragma omp parallel default(none) shared(array, n)
    {
#pragma omp single nowait
        { QuickSort(array, n); }
    }
    gettimeofday(&stop, NULL);

    double elapse = (stop.tv_sec - start.tv_sec) * 1000 +
                    (stop.tv_usec - start.tv_usec) / 1000;
    cout << elapse << " " << n << endl;

    for (int i = 0; i < n - 1; i++) {
        if (array[i] > array[i + 1]) {
            cerr << "quick sort fails! \n";
            break;
        }
    }
    return 0;
}
```