**并行计算第一次作业**

## 实验的方法

* + 号处理器开始计时后向其后继发送一个消息。
  + 号处理器等待接收其前驱发送来的消息，接收完成后停止计时。
  + 非号处理器等待接收其前驱发送来的消息，接收完成后向其后继发送消息。
  + 从而信息从号处理器传出，以环型方式依次传输给所有的处理器，最后回到号处理器。

## 注释后的实验程序

**#include <mpi.h>**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**/\* 记录传送的消息的大小 \*/**

**#define MESSAGE\_LEN 64**

**/\* 用于条件编译，存在此宏定义时会输出到达信息 \*/**

**// #define \_\_DISPLAY\_TRACE\_\_**

**/\* 宏函数，用于获得模 total 意义下， id 的后继\*/**

**#define NEXT(id, total) (id + 1) % total**

**/\* 宏函数，用于获得模 total 意义下， id 的前驱\*/**

**#define LAST(id, total) (id + total - 1) % total**

**/\* 记录出错数，每次接收后会进行检查并更新 \*/**

**int errors = 0;**

**/\* 记录运行次数，用于多次运行取平均 \*/**

**const int rounds = 16;**

**int main(int argc, char\* argv[]) {**

**int id, total;**

**MPI\_Status status;**

**/\* 存放要传输的数据 \*/**

**char message[MESSAGE\_LEN];**

**/\* 初始化 MPI ，获得当前节点 id 和总节点数 \*/**

**MPI\_Init(&argc, &argv);**

**MPI\_Comm\_rank(MPI\_COMM\_WORLD, &id);**

**MPI\_Comm\_size(MPI\_COMM\_WORLD, &total);**

**/\* 存放总运行时间 \*/**

**double total\_time = 0;**

**for (int i = 0; i < rounds; i++) {**

**/\* 每次运行开始时同步一次，保证计时准确 \*/**

**MPI\_Barrier(MPI\_COMM\_WORLD);**

**if (id == 0) {**

**/\* 只在 0 号节点上计时 \*/**

**double start\_time = MPI\_Wtime();**

**/\***

**\* 设置要发送的信息**

**\* 发送的信息为字符串的形式**

**\* 包含当前节点的 id 和目标节点的 id**

**\*/**

**sprintf(message, "%d -> %d", id, NEXT(id, total));**

**/\* 向后继节点发送此消息 \*/**

**MPI\_Send(message, (int)strlen(message) + 1, MPI\_CHAR,**

**NEXT(id, total), 99, MPI\_COMM\_WORLD);**

**/\***

**\* 接收前驱节点的消息**

**\* 此前驱节点是环的意义上的，实质是 total - 1 号节点**

**\*/**

**MPI\_Recv(message, MESSAGE\_LEN, MPI\_CHAR,**

**LAST(id, total), 99, MPI\_COMM\_WORLD, &status);**

**/\***

**\* 解析接收到的消息，提取其中“当前节点的 id 和目标节点的 id ”并检查：**

**\* 接收到消息的“当前节点的 id ”是否为本节点前驱的 id**

**\* 接收到消息的“目标节点的 id ”是否为本节点的 id**

**\* 若存在不一致，更新 errors**

**\*/**

**char buf1[10], buf2[10];**

**/\* 在调试时可选输出到达信息 \*/**

**#ifdef \_\_DISPLAY\_TRACE\_\_**

**printf("reached node %d\n", id);**

**#endif**

**sscanf(message, "%s -> %s", buf1, buf2);**

**if (strtol(buf1, NULL, 10) != LAST(id, total)**

**|| strtol(buf2, NULL, 10) != id) {**

**errors++;**

**}**

**/\* 结束计时并累加 \*/**

**double finish\_time = MPI\_Wtime();**

**total\_time += finish\_time - start\_time;**

**}**

**else {**

**/\* 类似的，接收前驱节点的消息 \*/**

**MPI\_Recv(message, MESSAGE\_LEN, MPI\_CHAR,**

**LAST(id, total), 99, MPI\_COMM\_WORLD, &status);**

**/\***

**\* 同样的，解析接收到的消息，提取其中“当前节点的 id 和目标节点的 id ”并检查：**

**\* 接收到消息的“当前节点的 id ”是否为本节点前驱的 id**

**\* 接收到消息的“目标节点的 id ”是否为本节点的 id**

**\* 若存在不一致，更新 errors**

**\*/**

**char buf1[10], buf2[10];**

**/\* 同样的，在调试时可选输出到达信息 \*/**

**sscanf(message, "%s -> %s", buf1, buf2);**

**if (strtol(buf1, NULL, 10) != LAST(id, total)**

**|| strtol(buf2, NULL, 10) != id) {**

**errors++;**

**}**

**/\***

**\* 同样的，设置要发送的信息**

**\* 发送的信息为字符串的形式**

**\* 包含当前节点的 id 和目标节点的 id**

**\*/**

**sprintf(message, "%d -> %d", id, NEXT(id, total));**

**/\* 同样的，向后继节点发送此消息 \*/**

**MPI\_Send(message, (int) strlen(message) + 1, MPI\_CHAR,**

**NEXT(id, total), 99, MPI\_COMM\_WORLD);**

**}**

**}**

**/\* 输出总运行时间及轮平均时间等信息 \*/**

**if (id == 0) {**

**printf("program finished with error count %d\n", errors);**

**printf("total node is %d\n", total);**

**printf("total time is %lf\n", total\_time);**

**printf("total round is %d\n", rounds);**

**printf("average time is %lf\n", total\_time / rounds);**

**}**

**/\* 结束程序 \*/**

**MPI\_Finalize();**

**return 0;**

**}**

## 并行机上运行程序

**program finished with error count 0**

**total node is 32**

**total time is 8.962051**

**total round is 16**

**average time is 0.560128**

## 实验结果

提交的为。