代码

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char **argv)
{
    // 描述通信子集合大小和当前进程序号
    int rank, size;
    char data = 0;

    // 记录时间
    double start_time, stop_time, cost_time;

    // MPI 初始化
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    // 记录开始时间
    start_time = MPI_Wtime();

    // 向后发送消息，tag = 0
    MPI_Send(&data, 1, MPI_INT, (rank + 1) % size, 0, MPI_COMM_WORLD);
    // Recv 是阻塞的，接收到消息时回到运行
    MPI_Recv(&data, 1, MPI_INT, (rank - 1 + size) % size, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);

    stop_time = MPI_Wtime();
    cost_time = stop_time - start_time;

    printf("Process %d: spend  %f seconds for one full circle of transmission\n",
rank, cost_time);

    MPI_Finalize();
    return 0;
}
```

执行脚本

```
#!/bin/bash
#SBATCH -J hpc_homework1_test_1   #作业名
#SBATCH -p cpu-quota
#SBATCH -N 3            #3个节点
#SBATCH -n 28           #28个cpu
#SBATCH -o homework1.out # 将屏幕的输出结果保存到当前文件夹的test_hpc_1.out
#SBATCH -e homework1.err # 将屏幕的输出结果保存到当前文件夹的test_hpc_1.err

srun hostname | sort > machinefile.${SLURM_JOB_ID}
NP=`cat machinefile.${SLURM_JOB_ID} | wc -l`
module load intel/19.0.5.281
export I_MPI_HYDRA_TOPOLIB=ipl
```

```
mpirun -genv I_MPI_FABRICS shm:dapl -np ${NP} -f ./machinefile.${SLURM_JOB_ID}
./tes
 sed -i 's/\r$//' test.sh
```

执行结果