并行程序示例

上海交通大学高性能计算中心

http://hpc.sjtu.edu.cn

2013 年 11 月 8 日更新

目录

1 OpenMP 示例

2 MPI 示例

这部分演示如何使用 Intel 编译器编译一个名为`mpihello`的 MPI 程序，然后在本地测试运行，最后向 LSF 作业管理系统提交正式作业。使用 Environment Module 可以简化环境参数设定，我们也会给出使用 module 的过程。

2.1 编译源代码

`mpihello`程序的源文件为`mpihello.c`，内容如下：

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

#define MAX_HOSTNAME_LENGTH 256

int main(int argc, char *argv[])
{
    int pid;
    char hostname[MAX_HOSTNAME_LENGTH];

    int numprocs;
    int rank;

    int rc;

    /* Initialize MPI. Pass reference to the command line to
     * allow MPI to take any arguments it needs
     */
    rc = MPI_Init(&argc, &argv);

    /* It's always good to check the return values on MPI calls */
    if (rc != MPI_SUCCESS)
    {
        fprintf(stderr, "MPI_Init failed\n");
        return 1;
```

```
    }

    /* Get the number of processes and the rank of this process */
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    /* let's see who we are to the "outside world" - what host and what PID */
    gethostname(hostname, MAX_HOSTNAME_LENGTH);
    pid = getpid();

    /* say who we are */
    printf("Rank %d of %d has pid %5d on %s\n", rank, numprocs, pid, hostname);
    fflush(stdout);

    /* allow MPI to clean up after itself */
    MPI_Finalize();
    return 0;
}
```

我们使用 Intel MPI 并行开发环境提供的`mpiicc`对其编译, `mpiicc`会自动调用后端编译器`icc`。使用 icc 前需要运行脚本使环境变量生效。

```
$ source /lustre/utility/intel/composer_xe_2013.3.163/bin/compilervars.sh intel64
$ source /lustre/utility/intel/mkl/bin/intel64/mklvars_intel64.sh
$ source /lustre/utility/intel/impi/4.1.1.036/bin64/mpivars.sh
$ mpicc -o mpihello mpihello.c
```

2.2 使用`mpirun`在本地测试运行

`mpirun`用于启动 MPI 并行程序。下面的命令启动`mpihello`并行程序，分配 4 个线程。*注意：在 π 集群中，`mpirun`只能用作小于 4 线程的测试，大规模正式作业必须提交到 LSF。*

```
$ mpirun -np 4 ./mpihello
Rank 0 of 4 has pid 90037 on mu05
Rank 1 of 4 has pid 90038 on mu05
Rank 2 of 4 has pid 90039 on mu05
Rank 3 of 4 has pid 90040 on mu05
```

2.3 提交到 LSF 作业管理系统

大规模正式作业必须提交到 LSF 作业管理系统。
作业控制脚本`mpihello.lsf`内容如下：

```
#BSUB -J HELLO_MPI
#BSUB -o job.out
#BSUB -e job.err
#BSUB -n 256

source /lustre/utility/intel/composer_xe_2014.3.163/bin/compilervars.sh intel64
source /lustre/utility/intel/mkl/bin/intel64/mklvars_intel64.sh
source /lustre/utility/intel/impi/4.1.1.036/bin64/mpivars.sh

MPIRUN=`which mpirun`
EXE="./mpihello"

CURDIR=$PWD
cd $CURDIR
rm -f nodelist nodes >& /dev/null
touch nodelist
touch nodes
NP=0

for host in `echo $LSB_MCPU_HOSTS |sed -e 's/ /:/g'|  sed 's/:n/\nn/g'`
do
echo $host >> nodelist
echo $host | cut -d ":" -f1 >> nodes
nn=`echo $host | cut -d ":" -f2`
NP=`echo $NP+$nn | bc`
done
```

将作业提交到名为`cpu`的队列上：

```
$ bsub -q cpu < mpihello.lsf
```

作业运行结束后，查看当前目录下的输出结果。

2.4 使用 module 简化环境设置

1. 编译源程序。

```
$ module load icc/13.1.1
$ module load mkl/11.0.3
$ module load impi/4.1.1.036
$ mpiicc -o mpihello mpihello.c
```

2. 在本地测试运行。

```
$ mpirun -np 4 ./mpihello
```

3. 提交到 LSF 作业管理系统，所用的作业控制脚本如下。注意，增加了*BSUB -L*、*MODULEPATH*、*module*若干行。

```
#BSUB -J HELLO_MPI
#BSUB -L /bin/bash
#BSUB -o job.out
#BSUB -e job.err
#BSUB -n 256

MODULEPATH=/lustre/utility/modulefiles:$MODULEPATH
module unload icc/13.1.1
module load mkl/11.0.3
module load impi/4.1.1.036

MPIRUN=`which mpirun`
EXE="./mpihello"

CURDIR=$PWD
cd $CURDIR
rm -f nodelist nodes >& /dev/null
touch nodelist
touch nodes
NP=0

for host in `echo $LSB_MCPU_HOSTS |sed -e 's/ /:/g'|  sed 's/:n/\nn/g'`
do
echo $host >> nodelist
```

```
echo $host | cut -d ":" -f1 >> nodes
nn=`echo $host | cut -d ":" -f2`
NP=`echo $NP+$nn | bc`
done
```

3 CUDA 示例

4 参考资料

- “LLNL Tutorials: Message Passing Interface (MPI)” https://computing.llnl.gov/tutorials/mpi/
- “mpihello by ludwig Luis Armendariz” https://github.com/ludwig/examples