

MPICH&PBS 使用指南

一、MPI 编程	1
二、MPICH 下编译和运行	3
三、PBS 环境下运行	4

一、MPI 编程

1、MPI 编程函数介绍

MPI 实际上是一个提供并行程序消息传递机制的函数库，有 40 多个函数，常用的有 6 个基本函数。下面以 C 语言为例简单介绍这些函数。

(1) MPI_Init 函数

定义: `int MPI_Init(int *argc, char ***argv)`

功能: 用命令行参数初始化 MPI 环境

输入: `argc`、`argv`—表示命令行参数，同 C 语言的 `main()` 函数参数格式，`argv` 中包含欲并行运行的进程数

输出: 返回值—非零/零表示初始化是否成功

说明: 该函数必须为程序中第一个调用的 MPI 函数

示例: `MPI_Init(&argc, &argv);` // `argc`、`argv` 引用的是 `main()` 函数的参数

(2) MPI_Finalize 函数

定义: `int MPI_Finalize (void)`

功能: 结束 MPI 程序的运行，指结束 MPI 环境的使用

输入: 无

输出: 返回值—非零/零表示结束 MPI 环境是否成功

说明: 该函数必须为程序中最后一个调用的 MPI 函数

示例: `MPI_Finalize ();`

(3) MPI_Comm_size 函数

定义: `int MPI_Comm_size(MPI_Comm comm, int *size)`

功能: 得到总进程数

输入: `comm` 通信域句柄（系统默认的为 `MPI_COMM_WORLD`，也可自己定义）

输出: `size`，即通信域 `comm` 内包括的进程数整数

(4) MPI_Comm_rank 函数

定义: `int MPI_Comm_rank(MPI_Comm comm, int *rank)`

功能：得到本进程的进程号

输入：comm，该进程所在的通信域句柄

输出：rank，调用进程在 comm 中的标识号

(5) MPI_Send 函数

定义：int MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

功能：发送消息给特定的进程

输入：buf 发送缓冲区的起始地址(可选类型)

count 将发送的数据的个数(非负整数)

datatype 发送数据的数据类型(句柄)

dest 目的进程标识号(整型)

tag 消息标志(整型)

comm 通信域(句柄)

输出：无

(6) MPI_Recv 函数

定义：int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)

功能：接受别的进程发过来的消息

输入：count 最多可接收的数据的个数(整型)

datatype 接收数据的数据类型(句柄)

source 接收数据的来源即发送数据的进程的进程标识号(整型)

tag 消息标识与相应的发送操作的表示相匹配相同(整型)

comm 本进程和发送进程所在的通信域(句柄)

输出：buf 接收缓冲区的起始地址(可选数据类型)

status 返回状态 (状态类型 MPI_Status)

2、MPI 程序示例

MPI 程序中必须包含 MPI 库的头文件，C 语言头文件名为 mpi.h，FORTRAN 语言头文件名为 mpif.h。

/* helloworld.c 程序清单*/

```
#include "mpi.h"
```

```
main(int argc, char **argv)
```

```
{
```

```
    int numprocs,myrank,i,j,k;
```

```
    MPI_Status status;
```

```
    char msg[20];
```

```
    MPI_Init(&argc,&argv);
```

```
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs); //
```

```
    MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
```

```
    if(myrank == 0)
```

```
    {
```

```

        strcpy(msg,"Hello World");
        MPI_Send(msg,strlen(msg)+1, MPI_CHAR,1,99, MPI_COMM_WORLD);
    }
    else if(myrank ==1)
    {
        MPI_Recv(msg,20,MPI_CHAR,0,99,MPI_COMM_WORLD,&status);
        printf("Receive message = %s\n",msg);
    }
    MPI_Finalize();
}

```

二、MPICH 下编译和运行

1、编译

MPICH 环境下，程序的编译采用命令行方式，即在 Linux 命令行下用 mpicc 命令进行编译。

编译命令格式：mpicc -o 可执行文件名 源程序文件名

命令参数说明：(1) 源程序文件经过 mpicc 编译连接以后得到可执行文件；

(2) 此处的 mpicc 命令就是 /usr/local/topspin/mpi/mpich/bin/mpicc，下文中的 mpirun_rsh 也是在此目录下。

程序编译示例：mpicc -o helloworld helloworld.c

2、运行

MPICH 环境下，程序的运行亦采用命令行方式，即在 Linux 命令行下用 mpirun_rsh 命令运行程序。

运行命令格式：mpirun_rsh -np 进程数 -hostfile 节点列表文件 ./可执行文件名

命令参数说明：节点列表文件(如：hosts)为手工建立的一个文本文档，内容为：

```

iblade01
iblade02
iblade03
iblade04
...

```

有多少个进程就需要写多少行，因节点为双 CPU，故节点名可重复一次。(下例只需这四行即可)

程序运行示例：mpirun_rsh -np 4 -hostfile hosts ./helloworld

运行结果：

```

Hello World!Process 0 of 4 on blade01.cluster.seu.edu.cn
Hello World!Process 1 of 4 on blade02.cluster.seu.edu.cn
Hello World!Process 2 of 4 on blade03.cluster.seu.edu.cn
Hello World!Process 3 of 4 on blade04.cluster.seu.edu.cn

```

强烈建议：不采用该方式，而采用 PBS 环境下运行方式，因为所列节点可能有故障。

三、PBS 环境下运行

PBS 是开源代码的作业管理系统，在此环境下运行，用户不需要指定程序在哪些节点上运行，程序所需的硬件资源由 PBS 管理和分配。

1、PBS 命令

PBS 提供 4 条命令用于作业管理。

(1) qsub 命令—用于提交作业脚本

命令格式：

```
qsub [-a date_time] [-c interval] [-C directive_prefix]
      [-e path] [-I] [-j join] [-k keep] [-l resource_list] [-m mail_options]
      [-M user_list][-N name] [-o path] [-p priority] [-q destination] [-r c]
      [-S path_list] [-u user_list][-v variable_list] [-V]
      [-W additional_attributes] [-z]
      [script]
```

参数说明：因为所采用的选项一般放在 pbs 脚本中提交，所以具体见 PBS 脚本选项。

例：# qsub aaa.pbs 提交某作业，系统将产生一个作业号

(2) qstat 命令—用于查询作业状态信息

命令格式：qstat [-f][-a][-i] [-n][-s] [-R] [-Q][-q][-B][-u]

参数说明：

- f jobid 列出指定作业的信息
- a 列出系统所有作业
- i 列出不在运行的作业
- n 列出分配给此作业的结点
- s 列出队列管理员与 scheduler 所提供的建议
- R 列出磁盘预留信息
- Q 操作符是 destination id，指明请求的是队列状态
- q 列出队列状态，并以 alternative 形式显示
- au userid 列出指定用户的所有作业
- B 列出 PBS Server 信息
- r 列出所有正在运行的作业
- Qf queue 列出指定队列的信息
- u 若操作符为作业号，则列出其状态。

若操作符为 destination id，则列出运行在其上的属于 user_list 中用户的作业状态。

例：# qstat -f 211 查询作业号为 211 的作业的具体信息。

(3) qdel 命令—用于删除已提交的作业

命令格式: qdel [-W 间隔时间] 作业号

命令行参数:

例: # qdel -W 15 211 15 秒后删除作业号为 211 的作业

(4) qmgr 命令—用于队列管理

```
qmgr -c "create queue batch queue_type=execution"
qmgr -c "set queue batch started=true"
qmgr -c "set queue batch enabled=true"
qmgr -c "set queue batch resources_default.nodes=1"
qmgr -c "set queue batch resources_default.walltime=3600"
qmgr -c "set server default_queue=batch"
```

2、PBS 脚本文件

PBS 脚本文件由脚本选项和运行脚本两部分组成。

(1) PBS 作业脚本选项 (若无-C 选项, 则每项前面加 '#PBS')

- a date_time : date_time 格式为: [[[CC]YY]MM]DD]hhmm[.SS]
表示经过 date_time 时间后作业才可以运行。
- c interval : 定义作业的检查点间隔, 如果机器不支持检查点, 则忽略此选项。
- C directive_prefix : 在脚本文件中以 directive_prefix 开头的行解释为 qsub 的命令选项。(若无此选项, 则默认为'#PBS')
- e path : 将标准错误信息重定向到 path
- I : 以交互方式运行
- j join : 将标准输出信息与标准错误信息合并到一个文件 join 中去。
- k keep : 定义在执行结点上保留标准输出和标准错误信息中的哪个文件。
keep 为 o 表示保留前者, e 表示后者, oe 或 eo 表示二者都保留,
n 表示皆不保留。若忽略此选项, 二者都不保留。
- l resource_list : 定义资源列表。以下为几个常用的资源种类。
 - cpus=N : 请求 N 秒的 CPU 时间; N 也可以是 hh:mm:ss 的形式。
 - mem=N[K|M|G][B|W]: 请求 N {kilo|mega|giga}{bytes|words} 大小的内存。
 - nodes=N:ppn=M : 请求 N 个结点, 每个结点 M 个处理器。
- m mail_options : mail_option 为 a: 作业 abort 时给用户发信; 为 b: 作业开始运行发信; 为 e: 作业结束运行时发信。若无此选项, 默认为 a。
- M user_list : 定义有关此作业的 mail 发给哪些用户。
- N name : 作业名, 限 15 个字符, 首字符为字母, 无空格。
- o path : 重定向标准输出到 path。
- p priority : 任务优先级, 整数, [-1024, 1023], 若无定义则为 0。
- q destination : destination 有三种形式: queue, @server,queue@server。
- r y/n : 指明作业是否可运行, y 为可运行, n 为不可运行。
- S shell : 指明执行运行脚本所用的 shell, 须包含全路径。

- u user_list : 定义作业将在运行结点上以哪个用户名来运行。
- v variable_list : 定义 export 到本作业的环境变量的扩展列表。
- V : 表明 qsub 命令的所有环境变量都 export 到此作业。
- W additional_attributes : 作业的其它属性。
- z : 指明 qsub 命令提交作业后, 不在终端显示作业号。

(2) 运行脚本同 LINUX 下一般的运行脚本文件。

[注]: 脚本文件中的 mpirun_rsh 命令行中的节点列表文件要用环境变量表示

\$PBS_NODEFILE, 这个环境变量表示由 pbs 自动分配给作业的节点列表;

节点数为命令行中指定的进程数。

格式如下:

```
mpirun_rsh -np 进程数 -hostfile $PBS_NODEFILE 可执行程序名
```

3、PBS 环境下运行示例

(1)脚本文件编辑示例

实例 1: 运行 mpi 程序

命令行: #vi aaa.pbs

编辑的内容:

```
#PBS -N myjob
#PBS -o /home/jz/my.out
#PBS -e /home/jz/my.err
#PBS -l nodes=2:ppn=2
cd 目录 (你们原来直接在节点上运行时所在的目录)
mpirun_rsh -np 4 -hostfile $PBS_NODEFILE /home/jz/helloworld
```

解释: 原先大家都是在中断输入 mpirun_rsh.....这些命令执行程序, 现在只要把这些提交命令放在.pbs 配置文件的最后, 由 PBS 来调度执行 (自动分配节点和其它资源)。

Myjob 是为你此次要运行的程序起的任务名, 可以改成你自己想要的名字

原先输出信息都是直接在屏幕上显示的, 现在屏幕上的显示全部输出到文件中, 上例中输出文件是/home/jz/my.out 文件, 大家可以根据自己的需要修改 (目录, 文件名)。程序运行时遇到的一些错误会记录在.err 文件中。**好处:**因为对每个任务都设定了不同的输出文件, 所以看结果只要打开相应文件看就可以了, 不需要开多个终端, 而且里面有任务的详细信息, 比如实际分配的是哪些节点计算, 运行时间等。

#PBS -l nodes=2:ppn=2, 你们程序需要几个节点只要修改 nodes 后的数字就可以了, ppn=2 保持不变, 因为我们的机器每个节点都是双 cpu 的。

```
mpirun_rsh -np 4 -hostfile $PBS_NODEFILE /home/jz/helloworld
```

此例中 -np 后的 4 是并行数 (2×2=4 个 cpu), -hostfile \$PBS_NODEFILE 不需要改变。

/home/jz/helloworld 是你编译好的可执行文件名, 需修改。

对于每个你要运行的 mpi 程序都需要这样一个.pbs 配置文件

也就是说大家原来的操作是：mpirun.....

现在改成 2 步走：1) 写个 pbs 配置文件(比如 xxx.pbs); 2) 向 pbs 提交 (qsub xxx.pbs)

实例 2: 运行非 mpi 程序

有些用户并不是自己编写 mpi 程序，同样也可以用 pbs 提交。

比如物理系运行程序时一般输入的命令是 RunDMol3.sh TiFeCp2-pbe-dspp-m=1-opt ，那么配置文件可以这样写：

命令行：#vi job.pbs

编辑的内容：

#PBS -N physics_job

#PBS -o /home/physics/physics_job.out

#PBS -e /home/physics/physics_job.err

#PBS -l nodes=1:ppn=2

#PBS -r y

cd 目录（你们原来直接在节点上运行时所在的目录）

RunDMol3.sh TiFeCp2-pbe-dspp-m=1-opt

解释：也就是说把原来在终端直接输入的命令 RunDMol3.sh TiFeCp2-pbe-dspp-m=1-opt 放到 pbs 配置文件中，因为你们只要一个节点，所以 nodes=1，至于用哪个节点系统自动分配，你们肯定很关心是分配了哪个节点给你们，那么可以用 qstat 命令查询（比如 qstat -n）。

(2) 提交作业示例

命令行：#qsub aaa.pbs

显示结果：

```
[jz@managenode jz]$ qsub aaa.pbs
211.managenode.cluster.seu.edu.cn
```

(3) 作业状态查询示例

Qstat 后加不同参数可以查看不同的信息（各参数的意思，上面有详细的说明，你们可以一个个试验一下，以后就知道查看哪些信息，需要哪些参数）

实例：

命令行：#qstat -a （查看作业的状态）

显示结果：

```
[jz@managenode jz]$ qstat
Job id          Name          User          Time Use S Queue
-----
211.managenode myjob         jz            00:00:00 R batch
```

解释：Job id 211 是给你提交的任务分配的任务号，S（任务状态，R 表示正在运行，Q 表示正在排队等候调度）

命令行: #qstat -n (查看作业使用的节点)

显示结果:

```
[physics@managenode physics]# qstat -n
managenode.cluster.seu.edu.cn:

Job ID          Username Queue   Jobname   SessID  NDS  TSK  Req'd  Req'd  Elap
-----  -----  -----  -----  -----  ---  ---  ---  ---  ---
580.managenode.physics batch   physics_jo  4124    1  --   --   144:0 R   --
      blade32/1+blade32/0
```

解释: blade32 就是分给你这个任务的节点

命令行: #qstat -f 211 (查看有关作业运行具体信息)

显示结果:

```
[jz@managenode jz]# qstat -f 211
Job Id: 211.managenode.cluster.seu.edu.cn
Job_Name = myjob
Job_Owner = jz@managenode.cluster.seu.edu.cn
resources_used.cput = 00:00:00
resources_used.mem = 716kb
resources_used.vmem = 9436kb
resources_used.walltime = 00:00:00
job_state = R
queue = batch
server = managenode.cluster.seu.edu.cn
Checkpoint = u
ctime = Sat May 20 14:00:11 2006
Error_Path = managenode.cluster.seu.edu.cn:/home/jz/my.err
exec_host = blade32/1+blade32/0+blade31/1+blade31/0
Hold_Types = n
Join_Path = n
Keep_Files = n
Mail_Points = a
mtime = Sat May 20 14:00:13 2006
Output_Path = managenode.cluster.seu.edu.cn:/home/jz/my.out
Priority = 0
qtime = Sat May 20 14:00:11 2006
Rerunnable = True
Resource_List.nodect = 2
Resource_List.nodes = 2:ppn=2
Resource_List.walltime = 01:00:00
Variable_List = PBS_O_HOME=/home/jz,PBS_O_LANG=en_US.UTF-8,PBS_O_LOGNAME=jz,
  PBS_O_PATH=/usr/local/topspin/mpi/mpich/bin:/usr/local/topspin/mpi/mp
  ich/bin:/opt/xcsm/bin:/usr/kerberos/bin:/opt/csm/bin:/usr/local/bin:/b
  in:/usr/bin:/usr/X11R6/bin:/usr/local/pbs/x86_64/bin:/usr/local/pbs/x86
  _64/sbin:/usr/local/maui/bin:/home/jz/bin,
  PBS_O_MAIL=/var/spool/mail/jz,PBS_O_SHELL=/bin/bash,
  PBS_O_HOST=managenode.cluster.seu.edu.cn,PBS_O_WORKDIR=/home/jz,
  PBS_O_QUEUE=batch
etime = Sat May 20 14:00:11 2006
```

解释: exec_host 显示的是实际执行该任务的节点