单机与集群MPI编译环境安装与简单使用--MPICH

单机MPI环境配置及其测试

单机环境下也能安装MPI，并进行相应的代码测试，本小节将给出安装及其测试细节。MPI实际上时通过gcc编译的，但是需要安装一些工具，如numa、openmpi等。

环境

项目	版本
操作系统	Ubuntu 16.04.4
gcc	gcc 5.4.0
内核版本	4.15.0-34-generic

安装MPICH

- 首先安装libnuma.so依赖工具 https://github.com/numactl/numactl
- 接着编译安装MPICH https://www.mpich.org/downloads/

说明: 上面两个工具的安装全部按照Linux三段论来做：

```
./configure
make -j 40
```

```
sudo make install -j 40
sudo ldconfig      #记住，所有的共享链接库安装完成以后，都要执行该命令，使得共享库可以用
```

测试代码

```
//helloMpi.c
#include <stdio.h>
#include <mpi.h>
int main( int argc, char *argv[] )
{
  MPI_Init(&argc, &argv);
  printf("Hello World!\n");
  MPI_Finalize();
}
```

- 编译 `mpicc -o test helloMpi.c`
- 运行 `mpirun -np 4 ./test`

注意：在mpirun时一定要在可执行文件test前加 `./` ，否则会报错 `Primary job terminated normally, but 1 process returned a non-zero exit code. Per user-direction, the job has been aborted.` ，提示job已经被放弃。

集群MPI环境配置及其测试

环境

项目	版本
操作系统	Ubuntu 16.04.4
gcc	gcc 5.4.0
内核版本	4.15.0-34-generic

建立三节点的MPI集群，并且在每个节点的 `/etc/hosts` 下添加如下信息：

```
127.0.1.1       localhost    #**注意：**一定要有localhost，一定不能有127.0.0.1 master等这样的地址，如果报错，请参见参考文献2和参考文献3
```

```
219.228.135.26   server
219.228.135.111  master
```

注意：集群中不同节点的用户名一定要相同，否则是不能用ssh密码登录，这样会导致集群之间不能通信。

配置成功后，请在不同的机器上检查，是否能够成功的ping通每一台机子，所有的分布式环境都需要这样搞，比如 `hadoop` 集群。

配置步骤

- 配置ssh免密码登录

首先在分布式系统中任意一节点（本文中选用master节点）上执行下列操作：

```
ssh master
cd ~/.ssh/  #若没有该目录，请先执行一次ssh localhost
ssh-keygen -t rsa #会有提示，都按回车就可以
cat id_rsa.pub >> authorized_keys #加入授权
chmod 600 ./authorized_keys  #修改文件的权限
```

接着需要在其它节点节点执行下列步骤：

```
ssh server
cd ~/.ssh/  #若没有该目录，请先执行一次ssh localhost
ssh-keygen -t rsa #会有提示，都按回车就可以
cat id_rsa.pub >> authorized_keys #加入授权
chmod 600 ./authorized_keys  #修改文件的权限
scp ./id_rsa.pub lab@master:~/.ssh/server_id_rsa.pub   #将server节点的授权文件传入到master节点上
```

接着在master机器上将公钥加入到授权文件中：

```
cat ~/.ssh/server_id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ./authorized_keys  #修改授权文件的权限
rm ~/.ssh/server_id_rsa.pub
```

接着将master机器上的授权文件传给其它的节点(在本文中传给server节点)

```
scp ./authorized_keys lab@server:~/.ssh/authorized_keys
```

- 检测每台MPI机子之间是否可以免密码登录

```
ssh master
ssh server
```

- 在每台MPI机子上安装MPICH，安装步骤同单机。安装之后在集群中各个节点的环境变量 `.bashrc` 中写入下面内容：

```
PATH=$PATH:/usr/local/bin
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

紧接着在shell中执行 `source ~/.bashrc` 。

测试

- 在集群中某个节点上创建文件

在任意目录下执行下列命令:

```
mkdir HelloMPI
cd HelloMPI
```

在该文件夹中创建两个文件，分别是

```
//helloMpi.c  测试代码
#include<mpi.h>
#include<stdio.h>
#include<string.h>
#include <unistd.h>
#include <netdb.h>
```

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

void get(char *hname){
    struct hostent *hent;
    gethostname(hname, 128);//sizeof(hname)
    //hent = gethostent();
    hent = gethostbyname(hname);
    //printf("hostname: %s/naddress list: ", hent->h_name);
}

int main(int argc,char *argv[])
{
    int my_rank;/*进程序号*/
    int p; /*进程总数*/
    int source;/*发送者序号*/
    int dest;/*接受者序号*/
    int tag=0;/*消息标签*/
    char message[100];/*消息储存*/
    MPI_Status status;/*return status for*/

    /*receive*/
    /*启动MPI*/
    MPI_Init(&argc,&argv);
    /*查找进程号*/
    MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);
    /*查找进程总数*/
    MPI_Comm_size(MPI_COMM_WORLD,&p);

    if(my_rank!=0)
    {/*创建消息*/
        //得到本机的ip地址
        char hostname[128];
        get(hostname);
        sprintf(message,"Greeting from process %d,and ip is
%s!",my_rank,hostname);
        dest=0;
        /*Use strlen+1 so that \0 gets transmitted*/

 MPI_Send(message,strlen(message)+1,MPI_CHAR,dest,tag,MPI_COMM_WORLD)
;
```

```
    }else{/*my rank ==0*/
        for(source=1;source<p;source++){

 MPI_Recv(message,100,MPI_CHAR,source,tag,MPI_COMM_WORLD,&status);
            printf("%s\n",message);
        }
        char hostname[128];
        get(hostname);
        printf("Greeting from process %d,and ip is
%s!\n",my_rank,hostname);
    }
    /*关闭MPI*/
    MPI_Finalize();
}/*主函数结束*/
```

```
# hosts
master:4 #运行4个进程
server:4 #运行4个进程
```

- 编译并运行

```
mpicc -o test helloMpi.c   #编译后将可执行文件使用scp命令传递到集群中其它节点的同一路径下，一定要注意：必须传递到同一路径下，为了避免多次传递，请使用NFS文件服务器共享文件
mpiexec -f ./hosts -np 8 ./test   #如果回写的结果中既有server，又有master，说明整个配置过程是正确的
```

```
lab@master:~$ mpiexec -f hosts -np 8 ./test
Greeting from process 1,and ip is master!
Greeting from process 2,and ip is master!
Greeting from process 3,and ip is master!
Greeting from process 4,and ip is server!
Greeting from process 5,and ip is server!
Greeting from process 6,and ip is server!
Greeting from process 7,and ip is server!
Greeting from process 0,and ip is master!
```

参考资料

1. http://cugxuan.coding.me/2017/11/17/Openmpi/openmpi%E9%9B%86%E7%BE%A4%E6%90%AD%E5%BB%BA/
2. https://stackoverflow.com/questions/36577630/mpi-communication-error-with-rank-1-connection-refused
3. https://blog.csdn.net/yhsweetlife/article/details/46654181