

集群软件环境

集群系统

计算节点和管理登录节点的操作系统均为64位Centos 7.2，提供标准的64位Linux操作系统环境。

集群管理

浪潮集群采用的是tsce4.0管理套件，可以有效的提高集群利用率。

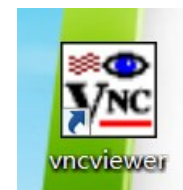
数学库

本集群采用intel 2015编译器自带的数学库，该数学库在经过intel优化编译后，在intel芯片上能跑出更为理想的结果。

集群使用

登陆集群

- 1、客户端与集群连通（网络）
- 2、客户端如果为linux可以直接ssh登录到管理节点；如果为windows系统，需要安装相应软件。



- 3、一个合法账号
- 4、相应操作知识积累

集群使用

登陆集群

1、客户端与集群连通（网络）

mu01 网口2 ip: 192.168.100.100

mu01 网口1 ip: 172.20.1.73

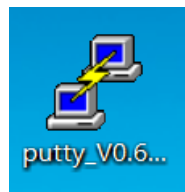
在校园网需要通过IP 172.20.1.73登录集群

集群使用

登陆集群

2、ssh工具

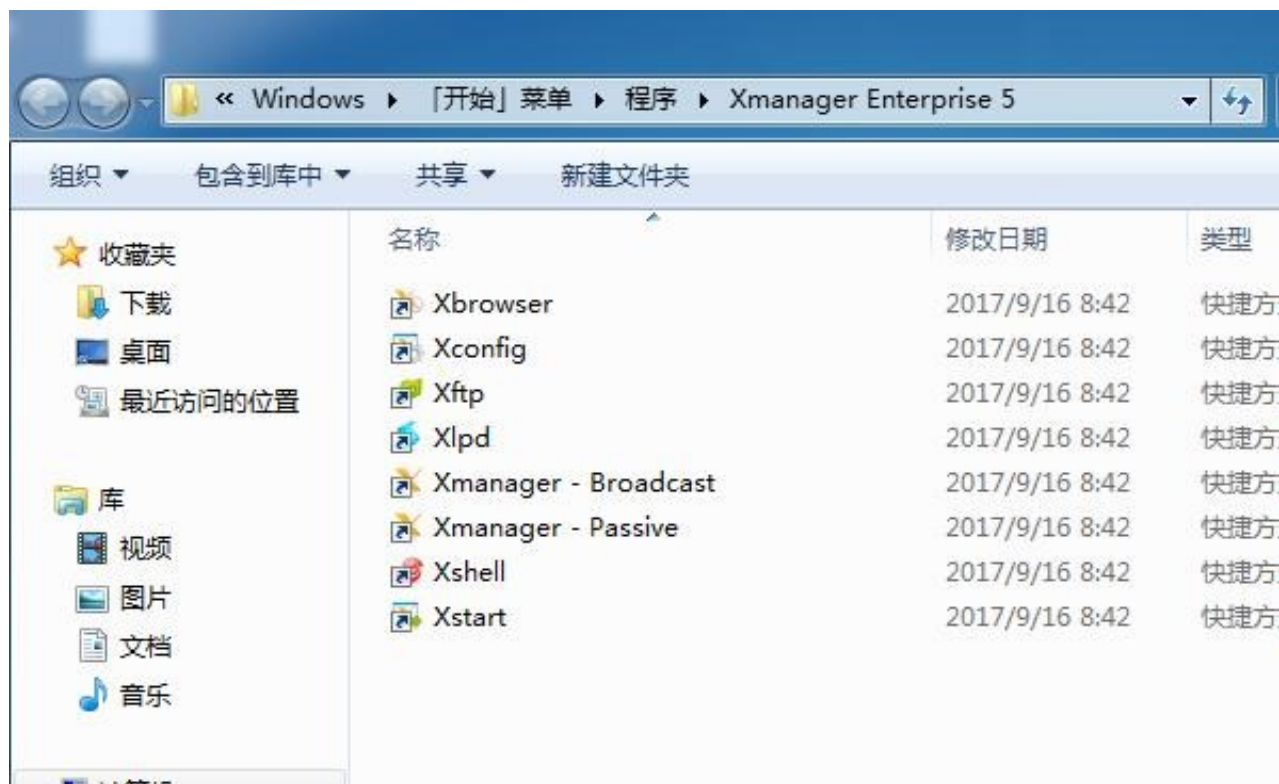
支持ssh协议的工具有很多，集群登陆不作具体限制。



集群使用

以使用Xmanage工具为例

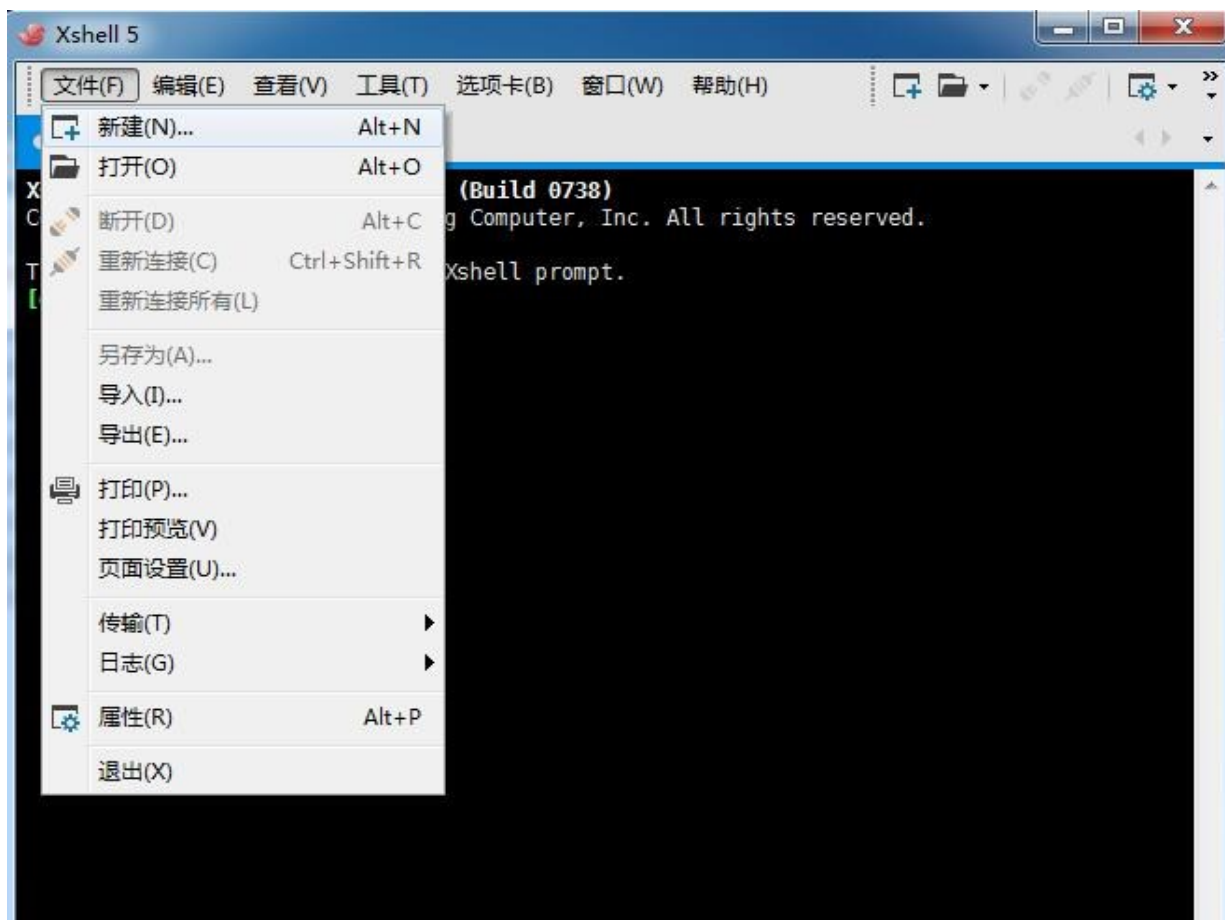
点击安装完后后的xmanage目录，打开后，选择Xshell文件



集群使用

以使用Xmanage工具为例

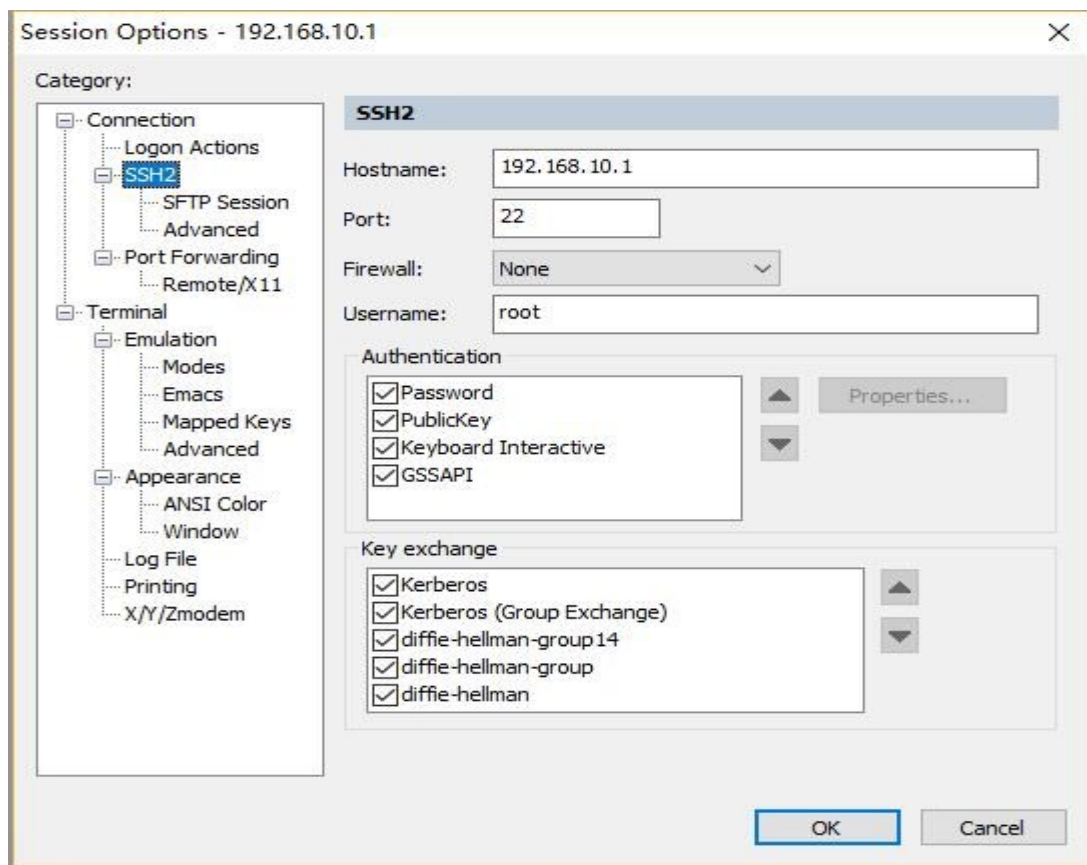
打开Xshell文件后，会出现如下界面，点击“新建”进行创建登录文件



集群使用

以使用Xmanage工具为例

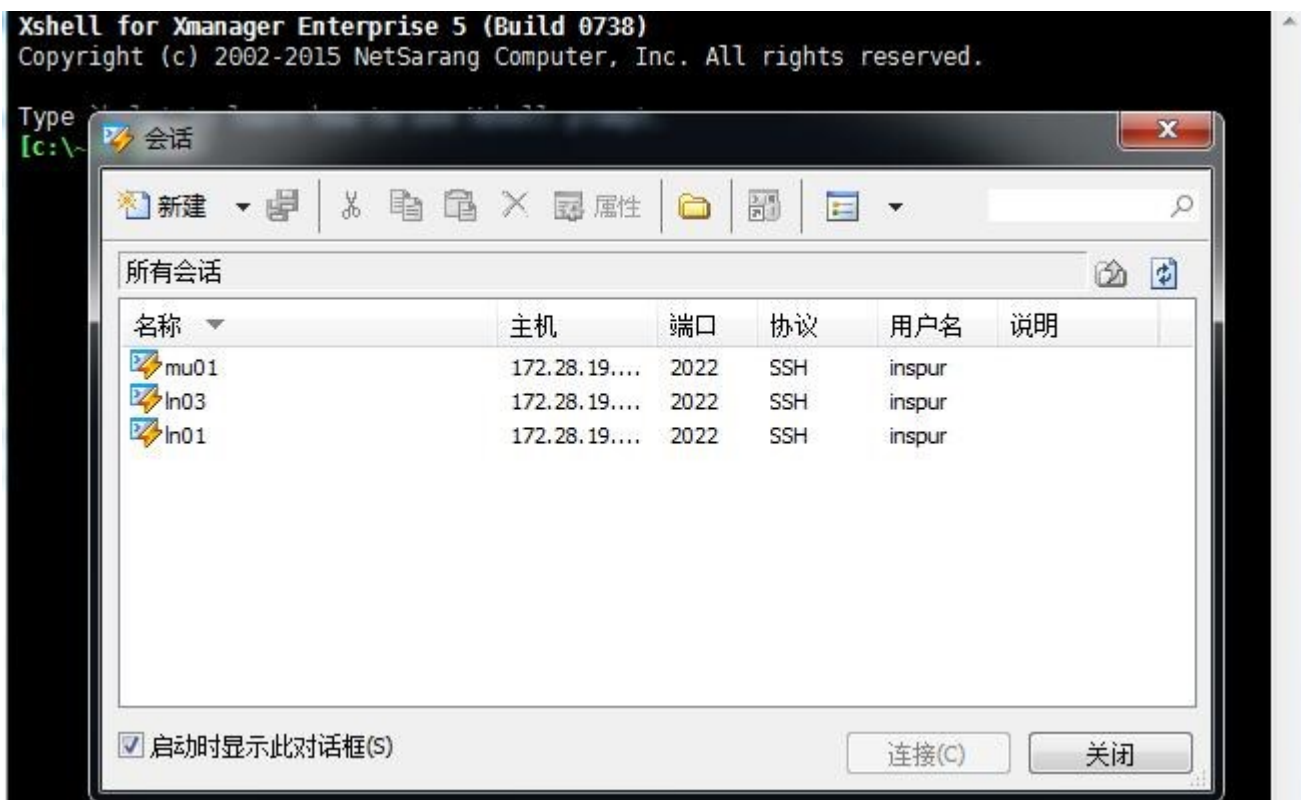
填写名称、主机、端口号等信息，其中名称可自行填写，主机为需要登录节点的IP地址，端口号为22。可以将名称和IP对应起来，方便记录



集群使用

以使用**Xmanage**工具为例

可以将所有节点一次性创建完成，后期可以点击使用



集群使用

以使用Xmanage工具为例

需要连接时，点击要连接的节点名称，进行连接，会提示输入用户名，输入自己的用户名，点击“确定”



集群使用

以使用Xmanage工具为例

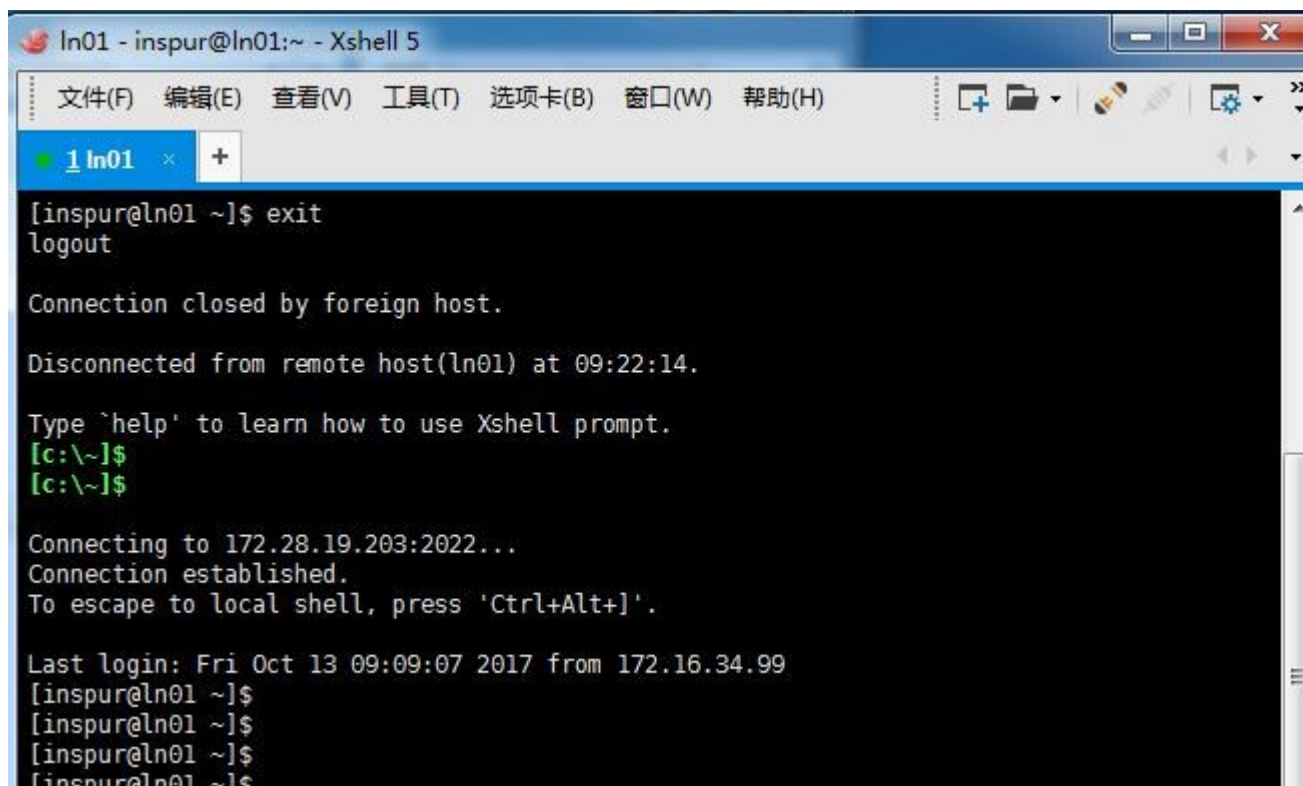
会提示输入用户密码，填写后，点击“确认”完成登录



集群使用

以使用**Xmanage**工具为例

完成登录，可进行操作

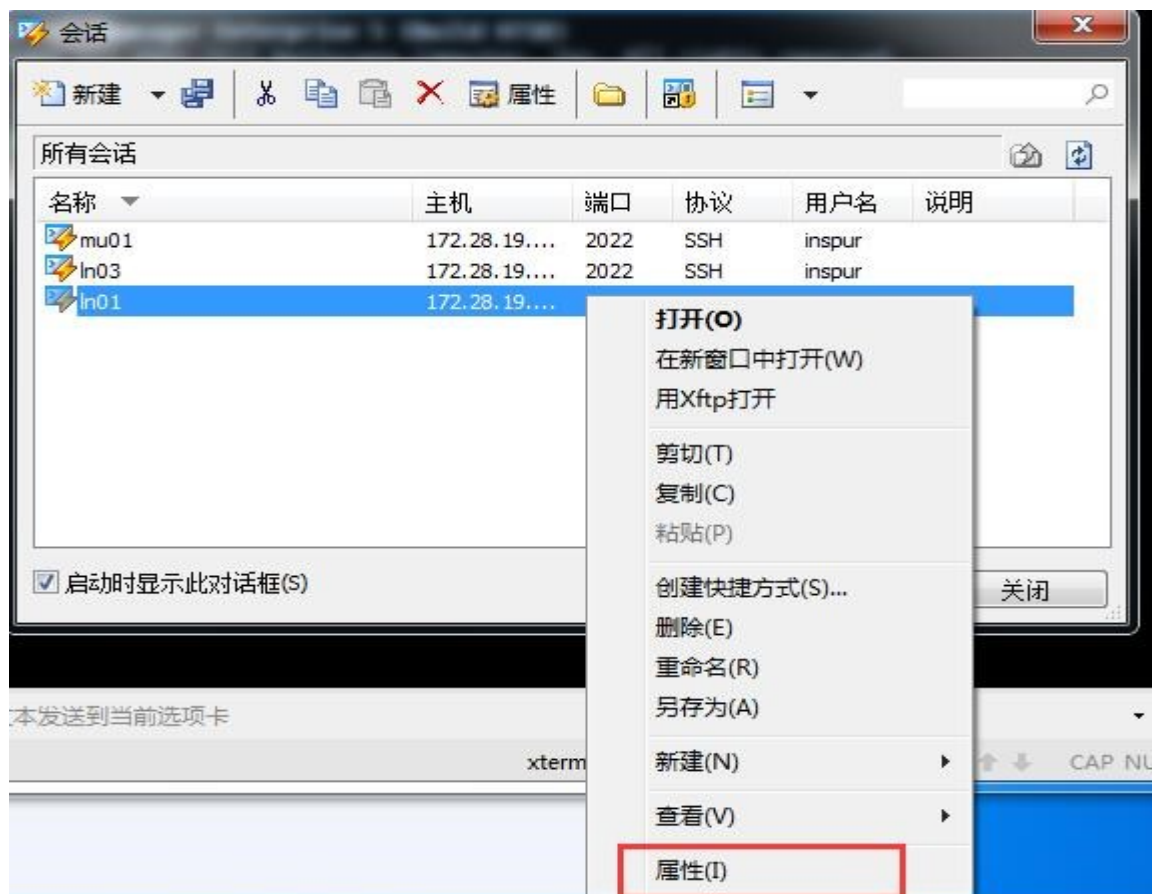


```
ln01 - inspur@ln01:~ - Xshell 5
文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(B) 窗口(W) 帮助(H)
1 ln01 x +
[inspur@ln01 ~]$ exit
logout
Connection closed by foreign host.
Disconnected from remote host(ln01) at 09:22:14.
Type `help` to learn how to use Xshell prompt.
[c:\~]$
[c:\~]$
Connecting to 172.28.19.203:2022...
Connection established.
To escape to local shell, press 'Ctrl+Alt+J'.
Last login: Fri Oct 13 09:09:07 2017 from 172.16.34.99
[inspur@ln01 ~]$
[inspur@ln01 ~]$
[inspur@ln01 ~]$
[inspur@ln01 ~]$
```

集群使用

以使用Xmanage工具为例

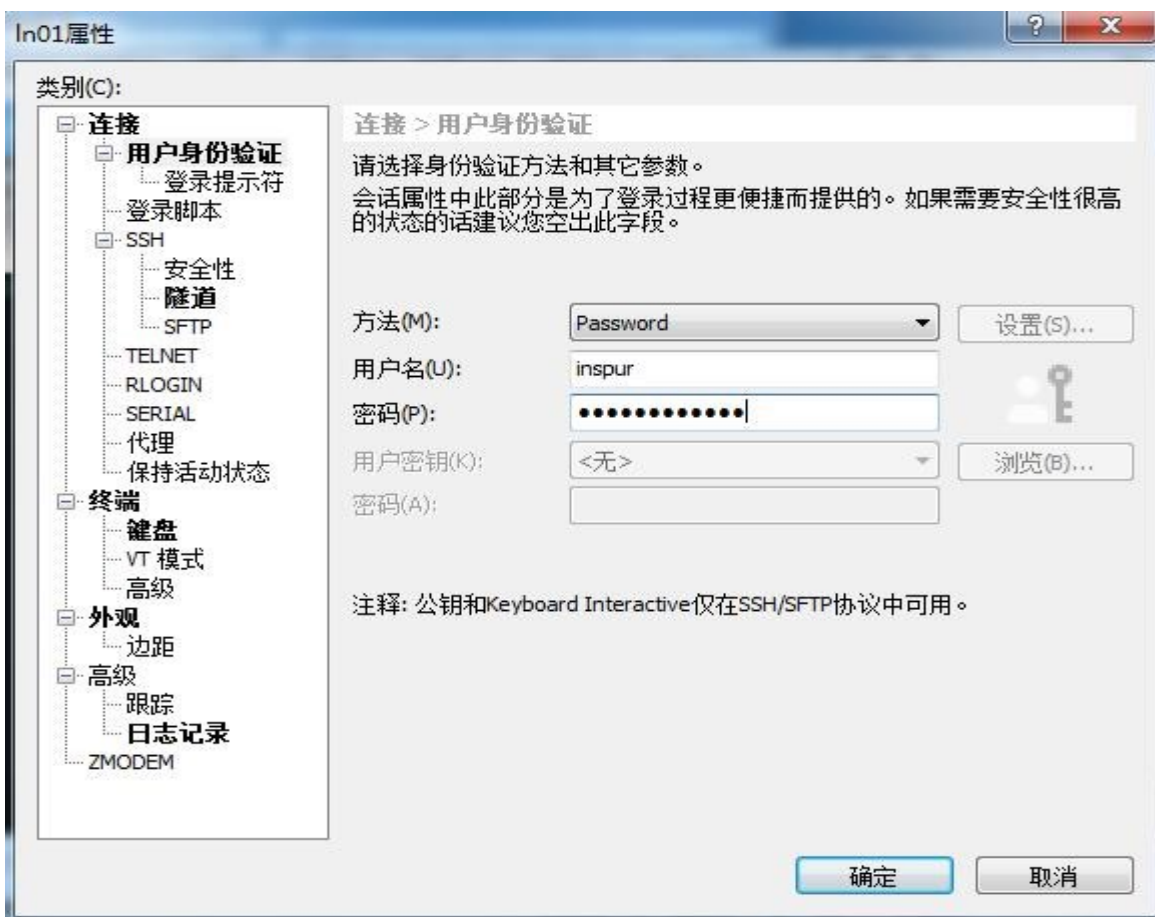
小技巧：在创建登录文件时候，可以在“用户身份验证”菜单项下设置登录用户名和密码，这样省去每次登录手动输入



集群使用

以使用Xmanage工具为例

小技巧：在创建登录文件时候，可以在“用户身份验证”菜单项下设置登录用户名和密码，这样省去每次登录手动输入



集群使用

VNC的使用

首先在系统下执行vncserver命令

```
Last login: Fri Oct 13 09:11:20 2017 from 172.16.34.99
[inspur@ln01 ~]$ vncserver

You will require a password to access your desktops.

Password:
Verify:

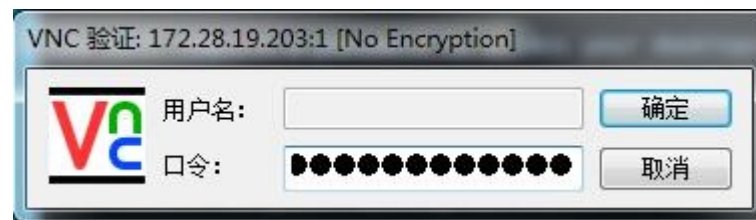
New 'ln01:1 (inspur)' desktop is ln01:1

Creating default startup script /home/inspur/.vnc/xstartup
Starting applications specified in /home/inspur/.vnc/xstartup
Log file is /home/inspur/.vnc/ln01:1.log
```


集群使用

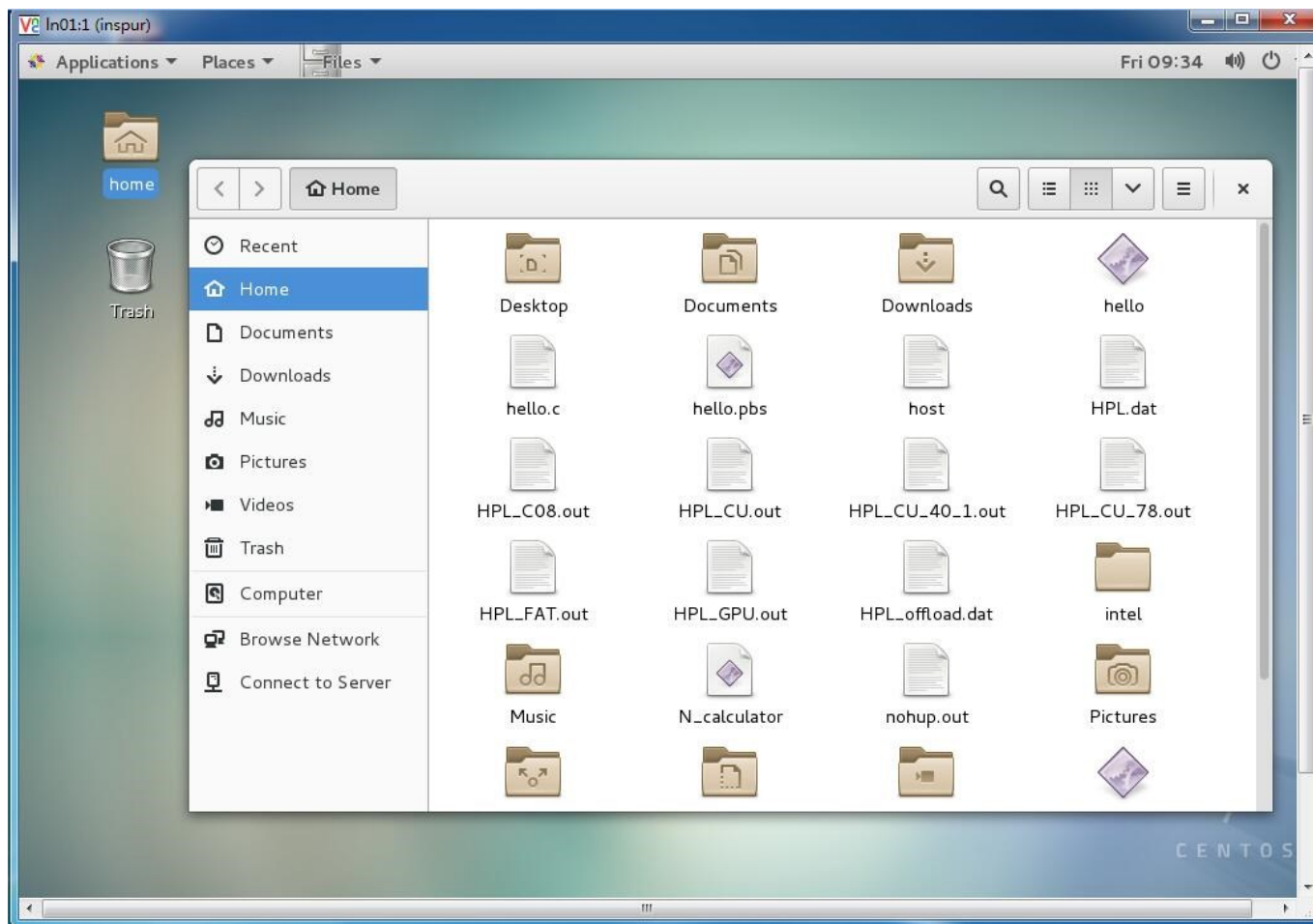
VNC的使用

打开windows端 vnc软件



集群使用

VNC的使用



集群使用

登陆集群

3、合法的账号

本集群默认账号设置（现阶段）

环境变量：intel环境变量（编译器、mkl、mpi）

用户第一次登陆后需要做：

a) 修改密码，在任意一节点执行：yppasswd，按照提示输入自己的旧密码和新密码即可。请注意新密码的复杂度，防止被盗用。

```
[inspur@ln01 ~]$  
[inspur@ln01 ~]$ yppasswd  
Changing NIS account information for inspur on mu01.  
Please enter old password:  
  
Changing NIS password for inspur on mu01.  
Please enter new password:  
  
Please retype new password:  
  
The NIS password has been changed on mu01.  
[inspur@ln01 ~]$  
[inspur@ln01 ~]$ █
```

集群使用

登陆集群

3、合法的账号

b) 修改环境变量（ ~/.bashrc 文件）

intel2015编译器（按需添加）：

```
source /opt/intel/composer_xe_2015/bin/compilervars.sh intel64  
source /opt/intel/mkl/bin/intel64/mklvars_intel64.sh  
source /opt/intel/impi/5.0.2.044/bin64/mpivars.sh
```

可将该环境变量保存到用户.bashrc文件下，实现每次切换用户均生效该环境变量

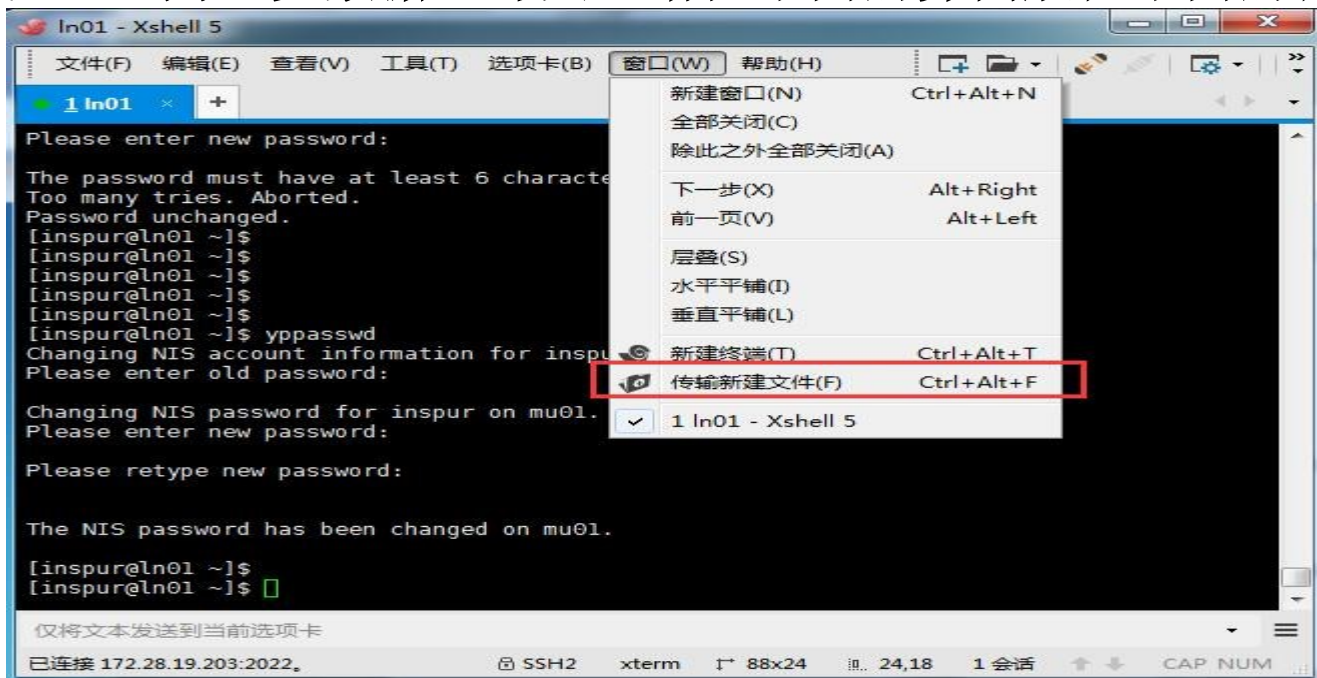
集群使用

登陆集群

3、合法的账号

c) 上传文件（以xftp为例）

注意：为避免数据些冲突，请把不同的算例放在不同的目录下



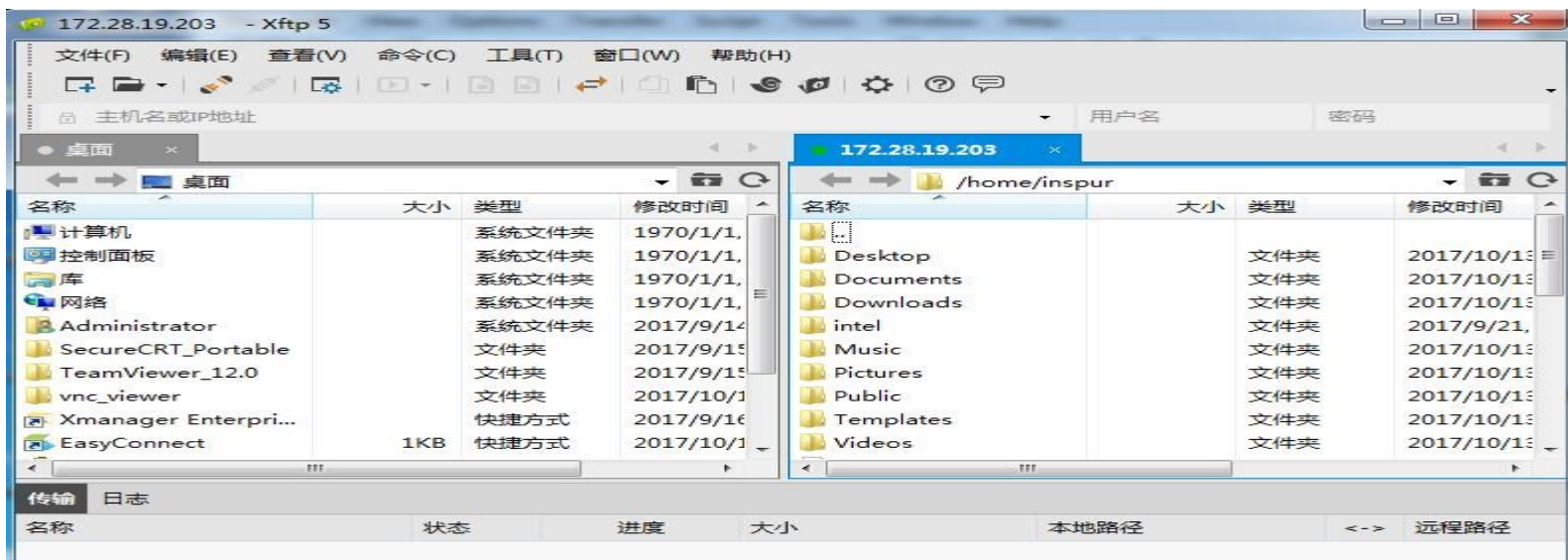
集群使用

登陆集群

3、合法的账号

c) 上传文件（以xftp为例）

打开后左边为本地空间，右边为服务器目录，由于权限设置，用户只能在自己的家目录下，或者具有读写权限的目录下进行文件上传下载操作



集群使用

登陆集群

4、相关操作知识（存储空间）

查询当前目录使用量（当前目录）

```
du --max-depth=1 -h
```

```
[inspur@ln01 ~]$ du --max-depth=1 -h
48K    ./vnc
408K   ./local
6.1M   ./cache
36K    ./ssh
4.0K   ./Music
8.0K   ./redhat
4.0K   ./Downloads
28K    ./intel
4.0K   ./Documents
12K    ./mozilla
4.0K   ./Public
4.0K   ./Templates
168K   ./config
4.0K   ./Videos
4.0K   ./Desktop
4.0K   ./Pictures
25M    .
[inspur@ln01 ~]$
```

集群使用

登陆集群

4、相关操作知识（存储空间）

查询当前存储使用总量（当前目录）

`du -sh`

```
[inspur@ln01 ~]$  
[inspur@ln01 ~]$ pwd  
/home/inspur  
[inspur@ln01 ~]$ du -sh  
25M  .  
[inspur@ln01 ~]$  
[inspur@ln01 ~]$
```

集群使用

登陆集群

4、相关操作知识（常用命令）

ls 、 cd 、 ssh 、 su 、 chmod 、 chown 、 tar 、 rm 、 touch 、 mkdir 、 mv 、 cp 等命令，均为linux系统常用命令，均需要熟练掌握。

查看cpu核心数、主频等 `cat /proc/cpuinfo`

查看内存使用量 `free -g`

查看逻辑磁盘 `fdisk -l`

查看磁盘挂载用量 `df -h`

查看gpu信息 `nvidia-smi`

集群使用

资源查询

在运行命令之前，首先查看下该节点资源是否充足。

1) 使用top命令进行

查询%id的比例为，%id为当前剩余CPU的资源比例

```
top - 09:52:48 up 22:09, 1 user, load average: 0.00, 0.01, 0.05
Tasks: 366 total, 1 running, 365 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.1 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 13158212+total, 12909209+free, 1680172 used, 809800 buff/cache
KiB Swap: 32767996 total, 32767996 free, 0 used. 12953208+avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3084	root	20	0	55248	1616	1128	S	0.7	0.0	9:53.44	tsced
4728	inspur	20	0	150524	2388	1516	R	0.3	0.0	0:00.02	top
1	root	20	0	189740	4824	2416	S	0.0	0.0	0:06.52	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0.0	0.0	0:00.11	kworker/u33:0
8	root	rt	0	0	0	0	S	0.0	0.0	0:00.23	migration/0
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/0
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/1
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/2
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/3
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/4

MPI使用

mpi是并行基础，在安装完intel编译器后，需要添加完环境变量。测试mpi是否生效：

```
[inspur@mu01 ~]$ which mpirun  
/opt/intel/impi/5.0.2.044/intel64/bin/mpirun
```

简单案例

使用C语言编写一个简单的程序，hello.c:

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

int main(int argc, char** argv) {
    MPI_Init(NULL, NULL);

    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    printf("Hello world from processor %s, rank %d"
           " out of %d processors\n",
           processor_name, world_rank, world_size);

    MPI_Finalize();
}
```

简单案例

使用mpicc进行编译，编译成可执行文件：
直接执行的输出如下：

INTEL编译器包括C，C++，Fortran编译器

C	C++	Fortran
icc	icpc	ifort

```
[inspur@mu01 ~]$ mpiicc -o hello hello.c
[inspur@mu01 ~]$ ls
hello hello1 hello.c hello.pbs host hpl-2.0_FERMI_v15 hpl-2.0-gpu.
[inspur@mu01 ~]$ ./hello
Hello world from processor mu01, rank 0 out of 1 processors
[inspur@mu01 ~]$
```

简单案例

使用mpirun，单机跑多核测试：

```
[inspur@mu01 ~]$ mpirun -np 8 hello  
Hello world from processor mu01, rank 3 out of 8 processors  
Hello world from processor mu01, rank 1 out of 8 processors  
Hello world from processor mu01, rank 2 out of 8 processors  
Hello world from processor mu01, rank 4 out of 8 processors  
Hello world from processor mu01, rank 5 out of 8 processors  
Hello world from processor mu01, rank 6 out of 8 processors  
Hello world from processor mu01, rank 7 out of 8 processors  
Hello world from processor mu01, rank 0 out of 8 processors
```

简单案例

使用mpirun，并行跑测试，首先定义host，指定哪些节点参与；

-genv I_MPI_DEVICE rdma

指定跑ib网络，rdma可换成rdssm，换成ssm表示跑以太网

-machinefile host 指定host内的这些节点参与

```
[inspur@mu01 ~]$ cat host
cu01
cu02
cu03
cu04
[inspur@mu01 ~]$ mpirun -genv I_MPI_DEVICE rdma -machinefile host ./hello
Hello world from processor cu03, rank 2 out of 4 processors
Hello world from processor cu01, rank 0 out of 4 processors
Hello world from processor cu02, rank 1 out of 4 processors
Hello world from processor cu04, rank 3 out of 4 processors
```