并 行 计 算

Parallel Computing

实验指南

孙济洲，于策，肖健

天津大学 数字媒体实验中心

2015（修订）

目　录

实验简介

本实验指南是并行计算课程的实验教材，供选修本课程的硕士和博士研究生参考以完成所要求的实验，同时也是高年级本科生自学并行计算技术的参考资料。

并行计算是一门理论与实践并重的课程，共设置五次上机实验内容，与课堂教学进度紧密配合。其中前四次为指定内容及要求的实验，第五次实验则由选修本课程的同学结合自己的科研或者兴趣自行选择一个问题，使用合适的并行计算技术加以解决，并对结果做出评价。

指定内容的四次实验中，前两次实验所使用的并行计算环境为共享内存的结构，后两次是针对基于消息传递的并行计算环境。面向共享内存的计算环境中所采用的编程技术主要考虑多线程（Win32/Unix 多线程以及 Java 多线程）以及 OpenMP 技术；面向消息传递的计算环境中主要考察当前使用广泛的 MPI 以及 PVM。各次实验的难度循序渐进，对于硕士和博士研究生，实验报告的内容及格式要求相同。

第五次实验所解决的问题由学生自行解决，实验报告（论文）要求不同，对于硕士研究生，要求提交内容完备的实验报告；对于博士研究生，则要求以正式学术论文的形式完成对某一问题的定义、分析、解决、评价，最终完成一篇可发表学术论文的初稿。

实验前需掌握的知识：

C/C++程序设计

数据结构和算法

操作系统原理

Unix/Linux 系统基本操作

Unix/Linux 环境中程序编译及运行方法，makefile 文件的编写与修改

Linux 环境中 rsh 和 ssh 的配置

ftp、rcp、scp 等文件传输命令

计算机网络基本知识

X-win32 使用(可选，实验中可能需要远程启动 Linux 的图形程序)

Java 编程技术（可选）

实验报告要求：

一、实验报告的书写要求必须包括以下内容：

实验题目，作者，（同组人），时间，实验内容，原理，**程序流程图**，实现方法，结果（数据图表、效果图等），理论性能分析以及对实际结果的分析，**总结展望**（心得体会等）。

提交实验报告时，必须附上全部源代码，以及源代码的编译、运行、部署**详细说明**。如果是使用 VS.Net、Eclipse 等集成开发工具完成的程序，则必须提交**全部工程文件**以及相应说明。

二、实验报告及程序上传方法：

实验指导书下载(关于实验的通知也放在这里)

http://ibm.tju.edu.cn/learning/parallel/

作业上传

ftp://ibm.tju.edu.cn/learning/homework/parallel

将最终实验报告和程序**打包**后上传到相应目录下

压缩包命名规则：**实验学号＋姓名**

比如：2016216000 姓名.rar

每次报告和程序的上传时间截止到下一次实验课之前（两周内）

实验一　Win32 多线程及 Java 多线程程序设计

一、 实验环境

多核/多处理器计算环境，Win32 多线程在 Windows 环境中完成，Java 多线程程序使用 Windows 或 Unix/Linux 均可。服务器的使用方法见附录一。

Windows 环境：实验用 PC 机上可直接完成实验。

Unix 环境：RS6000。

Linux 环境：Dell 2950。

二、 实验内容

1、实现一个用多线程计算 π（圆周率）的程序，对比相应的串行程序运算速度。算法可选如下所附的积分法或随机数方法。

2、使用多线程实现矩阵乘算法。（要求矩阵阶数至少为 16，具体矩阵自己设定），并对比相应串行程序的运算速度。

三、 评分标准

实验 1	30 分，
实验 2	30 分，
实验报告	40 分。

附：

一、 参考资料列表

1、Intel 多核技术培训的讲稿及部分源代码。

2、关于共享内存并行程序设计的补充讲稿。

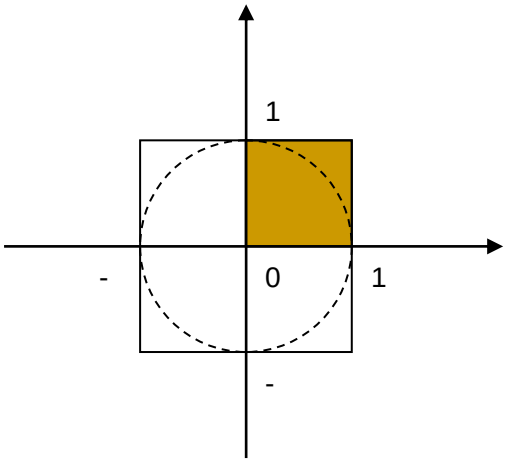
a) Java 多线程.ppt
b) Windows Threads.ppt
c) 共享内存及多线程.ppt
d) 内存系统对性能的影响.ppt

二、 计算 π 的两种方法

1、 积分法

$$\pi = \int_0^1 \frac{4}{1+x^2} dx \approx \sum_{0 \le i \le N} \frac{4}{1+\left(\frac{i+0.5}{N}\right)^2} \times \frac{1}{N}$$

2、 随机数方法（蒙特卡洛方法）



三、 性能评价的方法

1、 计算量相同，线程数不同

例如，N 取 1000,000，测试使用 1、2、3、4……个线程时所需要的时间。

2、 线程数相同，计算量不同

例如，只考察单线程和双线程的性能对比，N 分别取不同的数值。

实验二　Pthread 多线程及 OpenMP 程序设计

一、实验内容

1、 在 RS6000 上完成下列其中一个并行程序设计与实验:

（1）编写一个多线程程序来模拟两个人使用两个自动取款机来访问同一个共享帐号的情况。

（2）编写一个多线程程序来实现航空公司的机票预订系统，使其能够应付多个旅行社代理访问同一个空余机票源（存放在共享存储器中）的情况。

2、用 OpenMP 实现快速排序算法的并行程序。（使用随机数生成数组，至少 128 个元素）

3、完成实验报告，对程序并行化过程进行简单描述，以及对实验的体会

二、评分标准

实验 1　　30 分，

实验 2　　30 分，

实验报告　40 分。

附：

一、Pthread 程序编译命令

```
cc program.c -lpthread
```

二、OpenMP 程序编译命令

```
omcc program.c
```

三、参考资料列表：

- ✧ Pthread 目录
  - ✓ hello.c，pthread 多线程示例程序。
  - ✓ POSIXMultithreadProgrammingPrimer.pdf，教程。
  - ✓ pthreadTutorial.pdf，教程。
  - ✓ 例程和参考.doc，RS6000 中多进程程序设计示例。
- ✧ OpenMP 目录
  - ✓ omp_hello.c：示例程序
  - ✓ Omni-1.6.tar.gz：openmp 编译环境，可在 Linux 下安装

- ✓ Docs：(目录)实用参考文档

四、Omni-1.6 的安装。

1、获得 root 权限。

2、安装某个版本的 JDK

3、修改 /etc/profile 文件，在 export 之前添加(假定 JDK 安装到 /opt/j2sdk1.4.2_06)：

PATH=$PATH: /opt/j2sdk1.4.2_06/bin

4、重新登录

5、将 Omni-1.6.tar.gz 解压缩到硬盘某个目录。

6、进入解压缩后的目录，参照根目录下的 README 文件进行编译和安装。

五、支持 OpenMP 的编译器

Intel C++ Compiler9.1（可与 VS.Net 集成）。

GCC4.2。

具体编译方法请参考相应的使用说明。

实验三：PVM 环境构建及程序设计

一、 实验内容

1、PVM＆XPVM 并行环境的配置与测试。（程序及 Makefile 见 pi.zip）

2、mandelbrot 程序的并行化实现，并计算时间及加速比。（串行程序及编译说明 mandelbrot.zip）

3、完成实验报告，对程序并行化过程进行简单描述，以及对实验的体会。

二、评分标准：（满分 100）

1、mandelbrot 程序静态并行化实现　　90
2、mandelbrot 程序动态并行化实现　　100
3、mandelbrot 程序实现彩色效果　　＋10
4、无时间计算和加速比　　－10

附：

一、参考资料列表

1、《并行程序设计》　机械工业出版社　第三章及习题 7

2、PVM 配置测试说明.doc

3、XPVM 简要说明.doc

4、XPVM.ug.ps

5、pvm-book.pdf（PVM 编程参考）

6、其它一切可以参考的相关资料

二、“示例程序”目录

目录下有三个文件，分别是主程序、从程序和 makefile 文件。程序的算法是并行求解 PI。

三、“mandelbrot 串行程序”目录

目录下有三个文件，分别是 mandelbrot 的串行程序、编译这个串行程序用到的命

令，以及一个完成并行程序后可能会用到的 makefile 文件。

请注意：
mandelbrot 程序是一个图形界面程序，因此如果远程登录到集群系统上进行实验，需要将图形输出重定向到您的机器上。同时启动 XPVM 也需要图形支持。
方法：
1、安装一个 xwin32，运行起来。（在右下角出现一个小图标即可）
2、telnet 到集群的管理节点，执行 xterm -display ipaddress:0.0 （ipaddress 是您所用机器的 IP）
3、现在可以启动 XPVM。（输入命令 xpvm 即可）
4、同样再启动一个 xterm，运行您自己编写的并行化的 mandelbrot 程序。

如果是自己组建的 PVM 环境，并且 Linux 处于图形界面下，则不存在以上问题。

实验步骤说明：

实验环境

Cluster1350
管理节点 IP:59.67.33.245（192.168.5.6）
集群共有 16 个计算节点，为减少实验的重复步骤，在系机房上机时只使用其中两个计算结点(node01 和 node02)。
管理节点机器名为 managenode,计算节点分别命名为 node01~node08。
各节点的 hosts 文件都已配置好，不用修改。

实验步骤 0：准备

每个组分配一个用户名，两个人协调好，同一时间最多只允许有 1 人启动 PVM。
登录到 cluster1350，准备实验
登录方法 1 ：如果在 25 教学楼，直接 telnet 59.67.33.245
登录方法 2 ：如果不在 25 教学楼，则需通过 59.67.33.145 中传，步骤：
首先 telnet 59.67.33.145 ，用户名为 user01~uesr10 ，密码与用户名同。
然后，在 RS6000 的 shell 窗口中，telnet 59.67.33.245 使用本组的用户名与口令登录。具体登录方法参见附录一。

实验步骤 1：配置 shell 环境变量

修改所有三个节点上用户根目录下的 .bash_profile 文件
PATH=$PATH:$HOME/bin:/opt/csm/bin:/usr/share/pvm3/lib
PVM_ROOT=/usr/share/pvm3

export PATH PVM_ROOT MANPATH

修改所有三个节点上用户根目录下的 .bashrc 文件，添加以下两行

export PVM_ROOT=/usr/share/pvm3

export PVM_ARCH=LINUX

实验步骤 2：配置 rsh 环境

在各个节点的用户根目录下创建.rhosts 文件，内容如下（只是类似，假定用户 user01)：

managenode user01

node01 user01

node02 user01

将文件属性值改为 600：chmod 600 .rhosts

检验 rsh 是否正确配置的方法（以 node01 为例）

在 managenode 上，执行 rsh node01 date

实验步骤 3：确认 PVM 环境正常工作

登录到管理节点（59.67.33.245），输入命令：pvm

如果 PVM 环境正常工作，将显示：

pvm>

然后输入

pvm>add node01

如果等到类似输出结果则说明 node01 配置无误：

1 successful

HOST DTID

node01 80000

然后尝试是否可以将 node02 顺利加入到 PVM 中

如果成功，则 PVM 环境配置工作结束

实验步骤 4：创建 PVM 运行时环境

在各节点的用户根目录下创建目录：

~/pvm3/bin/LINUX

编译好的可执行的 PVM 程序要放到这个目录下

实验步骤 5：将示例程序拷贝到集群的管理节点

示例程序由三个文件组成：dtiming.cpp，dtiming_slave.cpp 和 Makefile.aimk

用 FTP 将三个文件上传到管理结点用户目录下（程序源文件所在位置没有特殊要求）

上传方法 1：如果在 25 教学楼，则直接 ftp 到 59.67.33.245 或者 192.168.5.6

上传方法 2：如果不在 25 教学楼，则通过 59.67.33.145 中传，步骤：
参见第二次实验的上传方法。

实验步骤 6：修改示例程序的 makefile 文件 Makefile.aimk

需要修改 Makefile.aimk 的以下两行

PVMDIR = ${PVM_ROOT}

SDIR = /home/yuce/pvm3/bin/LINUX/

将 PVMDIR 修改为 PVM 的安装目录，也就是环境变量 PVM_ROOT 。

将 SDIR 修改为源文件所在的目录

实验步骤 7：编译示例程序

在 Makefile.aimk 所在目录下，执行命令 aimk。生成两个可执行文件:dtime 和 dtime_slave。其中 dtime 是主程序，dtime_slave 是从程序。

实验步骤 8：部署示例程序

将编译生成的可执行文件拷贝到各个节点的 ~/pvm3/bin/LINUX 目录下

注：实际上，只是主程序所在的节点（managenode)上需要同时有这两个可执行文件，其余节点（node01,node02）只需要有从程序。

实验步骤 9：运行示例程序

启动 PVM 环境

pvm

pvm>add node01 node02

quit (只是退出了 PVM 的控制台，PVM 仍在运行）

在主程序所在目录下，执行 ./dtime 。

停止 PVM ，在 PVM 控制台输入 halt 。

FAQ

PVM 意外退出（比如没有响应而被迫使用 kill 命令终止进程），可能会导致无法再启动。

解决方法：到/tmp 目录下，删除所有你有权限删除的以 pvm 开头的文件。

rm /tmp/pvm*

在使用 XPVM 时，要用一个 shell 窗口启动 XPVM ，用另一个 shell 窗口启动你自己编写的 PVM 程序。

部署程序的方法（将可执行文件拷贝到各个节点）

ftp（注意文件传输完成后需要手工添加文件的可执行属性）

rcp（推荐使用，保留文件属性拷贝）

修改配置文件的方法

推荐用 vi

在 windows 环境下修改后，用部署程序的方法拷贝到各节点

实验四：MPI 环境构建及程序设计

一、 程序要求

1、编写一个使用分治方法的并行程序从一个存放在数组中整数数列中找出第一个 0。自己选择进程数和数组大小，但进程数最少 4 个，数组大小最小 64。（参见《并行程序设计》P109 习题 10）

2、热分布问题实现。

要求：

A.初始值设置。建议边缘设为 1，也可以在任意点设置任意大于 0 小于 1 的值（相当设置温度不同的热源）。

B.可视化显示。（建议使用 OpenGL）

提示：问题及算法的具体描述参见《并行程序设计》P145 的 6.3.2 节及《MPI 并行程序设计》中有关 Jacobi 迭代的有关算法。

二、 评分标准

程序 a35 分，程序 b55 分，报告 10 分。

如果程序 b 只实现算法而无可视化显示则最多只获得 90 分。

程序 b 可选的算法和可视化的方法很多，根据实现的技术含量及结果给予最多 10 分的加分。

三、 MPI 配置和运行

安装 MPICH 后请阅读 help 文档，有很多指导信息。

a) 安装

注意在安装时要选择安装所有组件，否则只能在本机上运行。

b) 在 VC＋＋6.0 中编译 MPI 程序

需要添加 MPI 的 include 和 lib 路径。

1.在 VC 的开发环境中，选择 Tools | Options | Directories

2.在 Include files 中添加 C:\PROGRAM FILES\ARGONNE NATIONAL LAB\MPICH.NT.1.2.3\SDK\INCLUDE

3. 在 Library files 中添加 C:\PROGRAM FILES\ARGONNE NATIONAL LAB\MPICH.NT.1.2.3\SDK\LIB

4.Alt+F7 | Link，在 Object/library modules:中添加 mpichd.lib。

c) 以各种方式运行 MPI 程序

具体请参考《MPI 并行程序设计》P89 第 10.2.3 节。

四、 参考资料

a) OpenGL 程序示例及编译说明（来自图形学课件）
b) 《MPI 并行程序设计》PDF 版
c) MPICH 软件的 winnt 版本
d) MPICH 自带的一个分析工具使用说明

实验步骤说明：

实验平台可以选择使用 Cluster 1350 或者 Windows.

一、Cluster 上 MPI 环境配置 ：ssh

仍是使用 Managenode、node01、node02，假设要在 Managenode 上启动 MPI 程序。

只需要配置 SSH

方法：

登录到 Managenode， 输入命令：

ssh-keygen -t rsa

输入三次回车

这将在用户的根目录下生成一个 .ssh 的 目录，里面有三个文件。

在 node01 和 node02 对应的用户根目录下创建 .ssh 目录

将 Managenode 节点上用户根目录下 .ssh 目录中的 id_rsa.pub 文件拷贝到上一步骤创建的目录下，并更名为 authorized_keys

注：第一次进行 ssh 登录时会系统会有一次询问，此时要输入“yes”，而不能输入“y”。

Cluster 1350 中 MPI 程序编译

方法 1：直接使用 mpiCC/mpicc，此方法使用于编译比较简单的程序

方法 2：编写 makefile，请参考 MPI 环境自带的例子。

/opt/mpich/share/examples

Cluster 1350 中 MPI 程序运行

方法 1、最简单的运行命令

mpirun –np N program

此时 MPI 所使用的默认的配置文件

方法 2、自定义配置文件，指明所用机器

mpirun –machinefile hosts –np 6 cpi

machinefile 文件中只需列出机器名

方法 3、使用更为详细的配置文件，指定所用的节点、可执行程序所在位置、相

应节点上启动的进程数。

详见《**MPI 并行程序设计** 》

二、Windows 环境下 MPICH 安装

注意在安装时要选择安装所有组件，否则只能在本机上运行。

Windows 环境下 MPI 编程

1.在 VC 的开发环境中，选择

Tools | Options | Directories

2.在 Include files 中添加

安装目录(此处替换为实际的路径)\SDK\INCLUDE

3.在 Library files 中添加

安装目录(此处替换为实际的路径) \SDK\LIB

4.Alt+F7 | Link，

在 Object/library modules:中添加 mpichd.lib。

Windows 环境下 MPI 程序的运行

同样有多种方法可以运行程序

推荐使用：

mpirun configureFile

configureFile 是格式如下的配置文件：

```
exe I:\yuce\Study\parallel\MPI_1\Jacobi\Debug\jacobi.exe
hosts
node01 3
node02 2    e:\yuce\jacobi.exe
```

其余方法见《MPI 并行程序设计》。

三、VC 环境下实现热分布问题可视化注意事项

为了使用 OpenGL，需要在项目中加入三个相关的 Lib 文件：glu32.lib、glaux.lib、opengl32.lib，这三个文件位于 c:\program files\devstudio\vc\lib 目录中。

选中菜单 Project->Add To Project->Files 项（或用鼠标右键），把这三个文件加入项目，在 FileView 中会有显示。这三个文件请务必加入，否则编译时会出错。或者将这三个文件名添加到 Project->Setting->Link->Object/library Modules 即可。

实验五：自主设计实验

一、内容及要求：

自行选定题目，运用课堂上所讲到的理论与技术解决之，并提交完整的实验程序及报告。

二、选题要求：

建议与自己的研究课题相结合。也可选择其他类型的题目，但如果题目的复杂度过低，会影响本次实验的成绩。

三、需要提交的材料：

需提交源程序及报告文档。

报告文档要求至少包括以下内容：

对题目的描述、选题背景、当前已有的解决方法、自己提出或设计的解决方法、实现、对实现结果的评价与总结。

附录一：实验设备及使用方法

实验设备：

实验用终端机：

Dell PC 机，双核处理器，Windows 操作系统，可在本机直接进行 Win32 多线程、Java 多线程、MPI 实验。

服务器：

RS6000，IBM 小型机，双 POWER 处理器。可进行 Pthread 实验。

Dell2950 服务器，双四核处理器，Linux 操作系统，可进行 Pthread、Java 多线程实验。

Cluster1350，IBM 集群系统，1 管理节点+16 计算节点，可进行 OpenMP、PVM、MPI 实验。

服务器使用方法：

25 教学楼内登录方法：

服务器	IP	用户名/口令	备注
RS6000	59.67.33.145	user01~user10	telnet, ssh, ftp, scp
Dell2950	59.67.33.248	user01~user10	ssh, scp
Cluster1350	59.67.33.245	user01~user20	telnet, ssh, ftp, scp

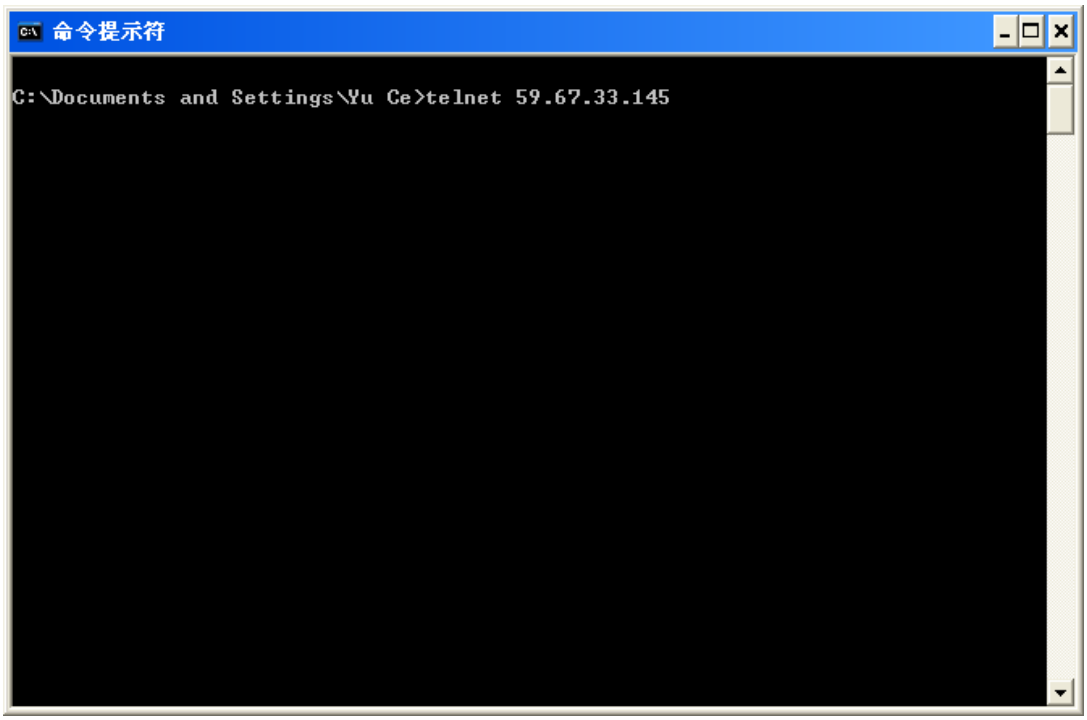
25 教学楼外登录及文件传输方法：

RS6000，直接访问 59.67.33.145 (ibm.tju.edu.cn) 即可。

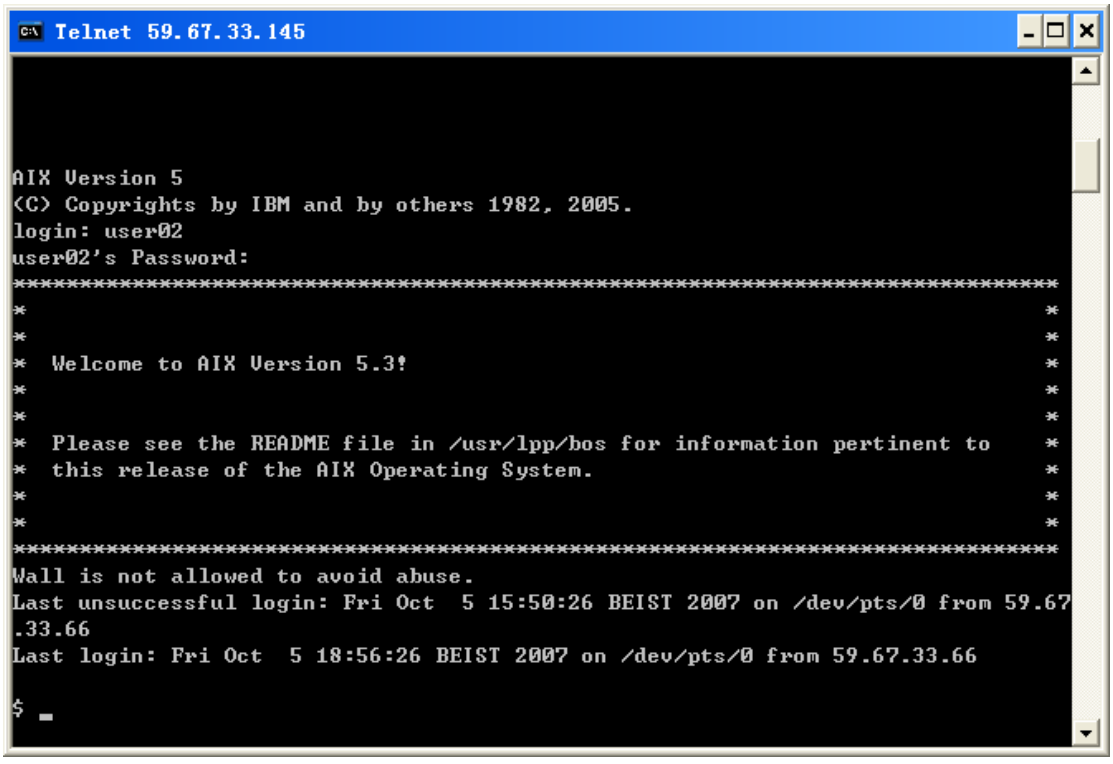
登录 Dell2590 及 Cluster1350 需通过 RS6000 中转：

1、 访问 Dell 2950

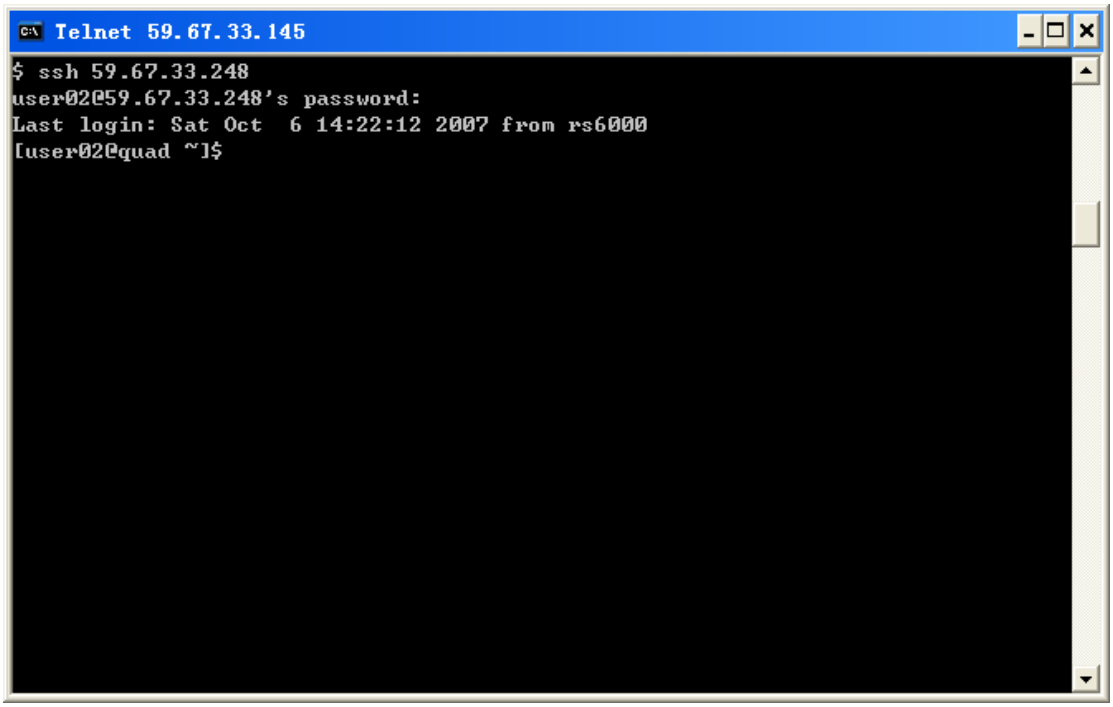
1) 登录到 RS6000（59.67.33.145）



2) 输入用户名及口令。用户名为 user01~uesr10，密码与用户名同。下图所示用户名为 user02。



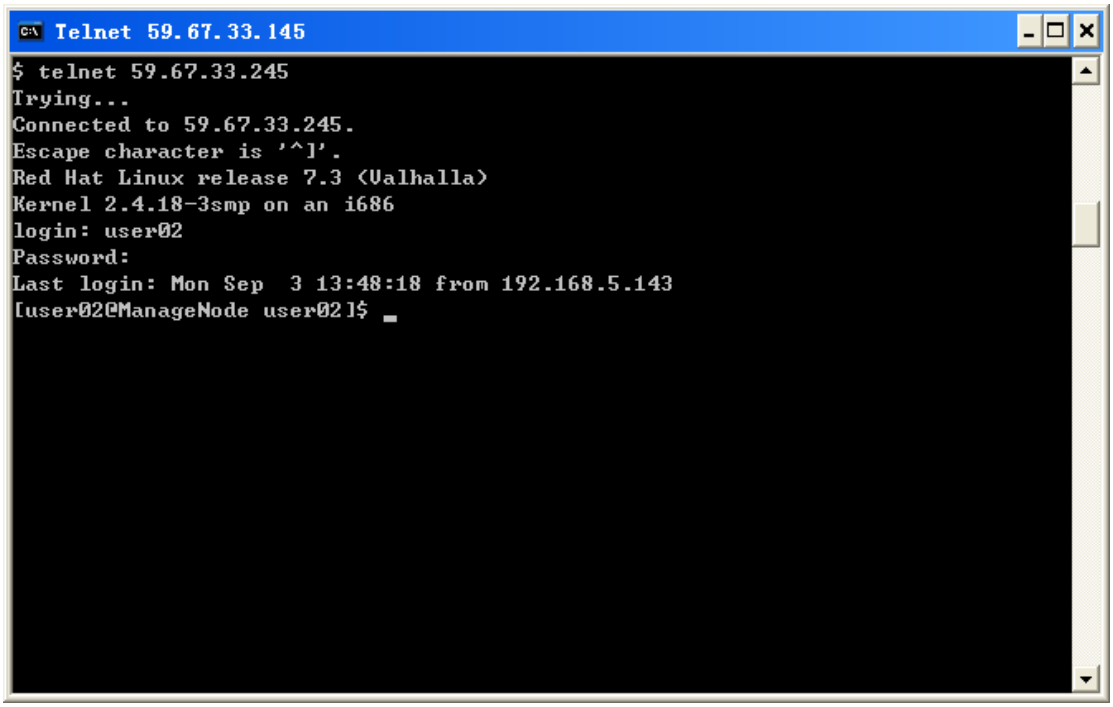
3) 使用 ssh 连接到 Dell2950



然后便可以开始在 Dell 2950 上进行操作。

2、 访问 Cluster1350

登录到 RS6000 后（见访问 Dell2950 的第一、二步骤），可以通过 telnet 或者 ssh 连接到 Cluster1350 的管理节点。



注意：如要返回到 RS6000，请在 shell 窗口中输入“exit”，则可直接返回。不要以“telnet 59.67.33.145”的方式返回。

3、 访问 Cluster1350 的计算节点

Cluster1350 共有 17 个节点，其中 1 个为管理节点，其余 16 个为计算节点。

登录到 Cluster1350 的管理节点后，方可访问计算节点。实验中开放了两个节点：node01 和 node02。

通过管理节点访问计算节点方式，与通过 RS6000 访问 Dell2950 或 Cluster1350 管理节点的方式相同，可使用 telnet 或者 ssh。

4、 文件传输

建议使用 scp 完成文件的传输。

scp 的语法为 “scp 文件名 用户名@目标主机:路径”。

如果目标主机中用户名与当前用户名相同，则可省略命令中的“用户名”。

假设当前用户（user02）在 RS6000 上的某个文件（sample.c）需要传输到 Dell2950。

1) 确认用户当前登录于 RS6000，且文件名无误。

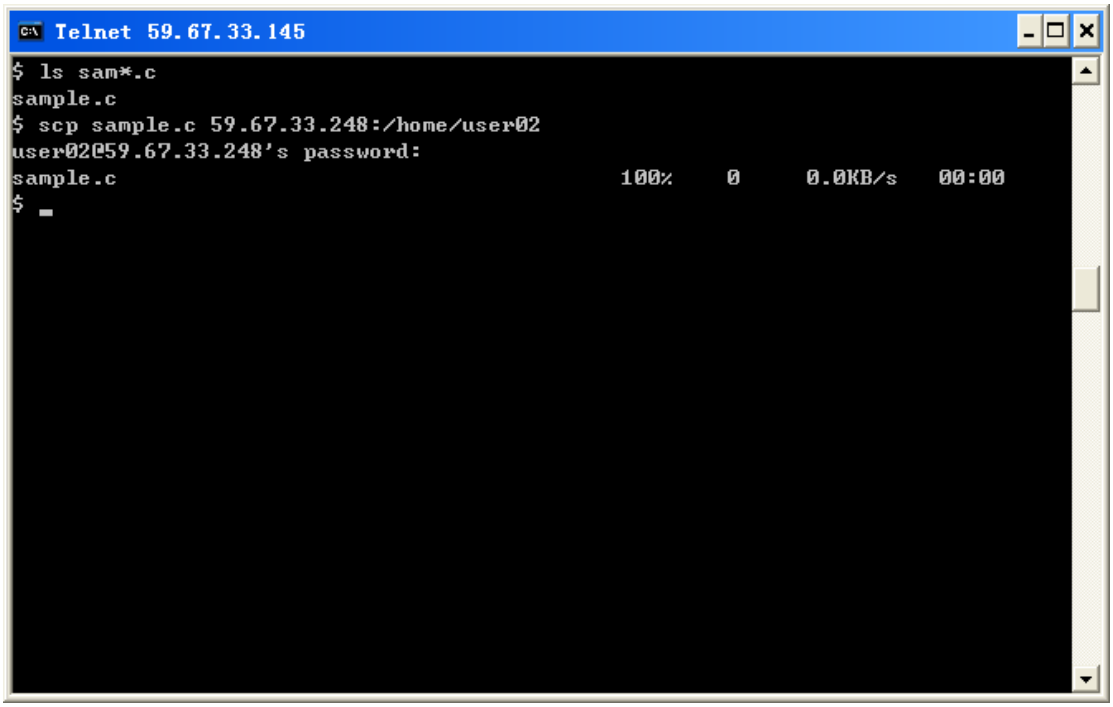
2) 使用如下命令传输：

scp sample.c 59.67.33.248:/home/user02

3) 若当前用户为 user02，而需要传输文件到 user03 的目录下：

scp sample.c user03@59.67.33.248:/home/user02

如下图所示：



附录二：RSH 的配置与应用

RSH 是 Remote Shell 的简写，意思是“远程 shell”，也就是说，通过 RSH 可以在本地系统中执行远程机器上的任何 shell 命令。如图 1 所示：

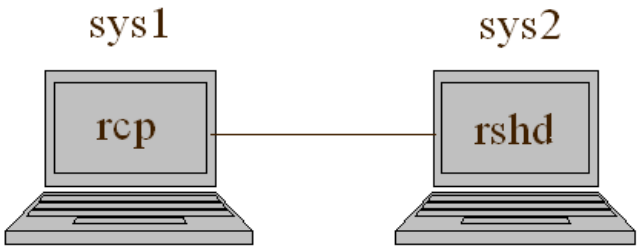


图 1 RSH 结构示意图

其中 sys2 系统中运行 rshd，即 RSH 的服务器端。sys1 系统为客户机。在这种情况下，如果有适当的配置，并且两台机器之间的网络正常，那么在 sys1 系统中就可以通过 RSH 远程执行 sys2 系统中的 shell 指令。一些可能的命令格式示例见清单 1。

```
sys1>whoami
                team02
sys1>rsh sys2 date
sys1>rsh sys2 –l tem04 date
sys1>rsh sys2
```

清单 1。 利用 RSH 执行远程 shell 命令

rsh sys2 date ：在 sys2 上执行 date 命令；

rsh sys2 –l tem04 date：以 tem04 身份在 sys2 上执行 date 命令；

rsh sys2：通过 RSH 登录到 sys2。(这种方式可以代替传统的 telnet)

当然，安全起见，运行 RSH 服务的系统必然会对试图远程执行 shell 命令的用户进行验证，验证的流程如图 2 所示。

由图中可以看出，当 RSH 服务器得到一个来自其他机器 (也可以是本机) 的 rsh 命令，触发验证流程。首先，服务器先查看执行远程命令的用户是不是在本机的 /etc/passwd 文件中，也就是说，先看此用户是否在本机上有相应的合法的用户，如果没有，则直接返回失败。如果此步验证通过，则判断此用户是不是为 root，等等，依次进行验证。

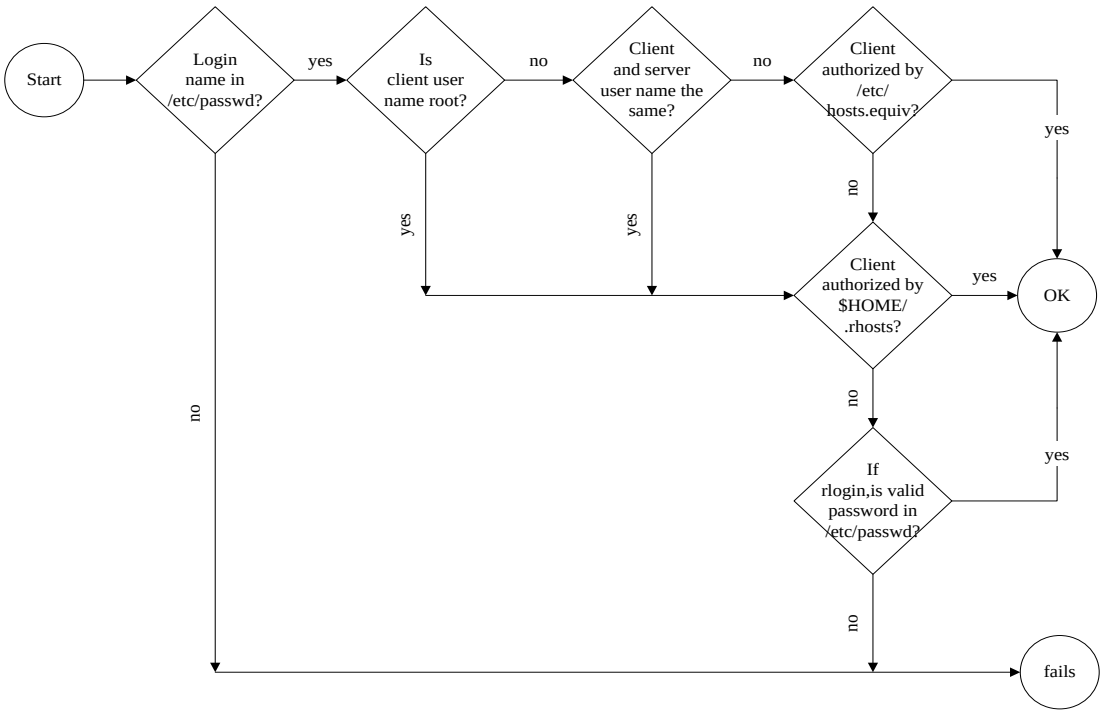


图 2 RSH 用户权限验证流程图

最终我们可以看出，在多种条件下通过 RSH 都可以成功地运行远程 shell 命令。不过要注意的是，通常系统不会允许以 root 身份通过 RSH 执行远程 shell 命令。最常见的一种情况是，两个系统中有相同名字的用户，且在 RSH 服务器端用户的主目录下有 .rhost 文件 (文件属性为 600) 指明对此用户放权，这样在 sys1 中这个用户就可以直接以 rsh sys2 *** 的形式远程执行 shell 命令，较为方便。

服务器系统中用户主目录下的 .rhost 格式很简单，如清单 2 所示。

```
sys2        temp02
sys3        temp02
sys1        temp02
```

清单 2。.rhost 文件格式

其中，sys1 一列为信任的主机名，temp02 为信任的用户。注意，此文件的属性必须设置为 600 (也就是 rw-------)，系统才会认为这个文件有效。

与 rsh 命令相关的命令还有 rcp 和 rlogin。其中 rcp 是用来在两个系统之间拷贝文件，rlogin 是相当于 telnet 的一个命令，可以登录到另一个系统。由于 rlogin 与构建 Linux 集群关系相对不是特别密切，因此不再详细描述。rcp 可能的使用方法见清单 3。

```
sys1>whoami
                team02
sys1>rcp filea sys2:fileb
sys1>rcp filea team04@sys2:fileb
sys1>rcp –p –r sys2:dir sys4:dir
```

清单 3。可能的 rcp 使用方式

rcp filea sys2:fileb：将本地系统中当前目录下的文件 filea 拷贝到 sys2 上同名用户主目录下，并重命名为 fileb；

rcp filea team04@sys2:fileb：将本志系统中当前目录下的文件 filea 以用户 team04 身份拷贝到 sys2 上 team04 用户的主目录下，并重命名为 fileb；

rcp –p –r sys2:dir sys4:dir：假定用户在 sys2 和 sys4 上都有权限执行远程 shell 命令，则此命令将 sys2 上与本地系统当前用户同名的用户主目录下 dir 目录拷贝为 sys4 上同名的 dir 目录，其中 –p 参数是保持当前各文件属性不变，-r 参数是为了拷贝整个目录。

rcp 的权限验证流程与 rsh 相同。

RSH 环境的配置步骤举例。假定已有如下条件：三台节点机，机器名分别为：managenode、node01、node02。网络工作正常。三台节点机上具有相同的用户 user01。各节点机的 /etc/hosts 文件内容如下：

```
192.168.0.1  Managenode
192.168.0.2  node01
192.168.0.3  node02
```

如果我们现在希望在 managenode 上通过 RSH 在 node01 和 node02 上远程执行 shell 命令 (或者执行其他 RSH 命令)，只需要在 node01 和 node02 节点机上用户 user01 的根目录下创建 .rhosts 文件，内容如下：

```
managenode user01
```

并将文件属性改为 600。

测试：以 user01 身份登录到 managenode，执行：

rsh node01 date

如果 RSH 环境已经配置成功，则应显示出 node01 的当前日期和时间。

不过，这只是实现了在 managenode 上可以以用户 user01 的身份在 node01 和 node02 上远程执行 shell 命令。如果需要在任何一个节点上都能在其他所有节点上都能远程执行 shell 命令，则需要重复进行以上的步骤，配置各节点的 .rhosts 文件。

附录三：SSH 的配置与应用

ssh (Secure Shell) 是安全的远程执行 shell 命令的工具。SSH 的 s 命令是为了用来替代 r 命令而设计的。s 命令的用法和命名与 r 命令一致，用户更容易掌握。当 SSH 正确安装和配置以后，s 命令提供了对用户透明的安全特性。与伯克利版本提供的服务不同，SSH 的命令仅仅使用了一个守护进程 sshd 和一个 TCP 端口,由于只用了一个进程来管理服务，使得 SSH 易于监控和配置。

SSH 提供的客户端命令

ssh 安全远程 shell

slogin 安全远程登录

scp 安全远程拷贝

sftp 安全文件传输(只有 SSH2 提供了 sftp 客户端。)

由于 SSH 的实现原理很复杂，因此我们在这里不详细分析其验证机制和加密方法，而只是介绍一下如何配置并使用 SSH 环境。SSH 相关命令的语法与 RSH 中的相应命令基本相同。

SSH 环境配置步骤举例。假定已有如下条件：三台节点机，机器名分别为：managenode、node01、node02。网络工作正常。三台节点机上具有相同的用户 user01。各节点机的 /etc/hosts 文件内容如下：

```
192.168.0.1  Managenode
192.168.0.2  node01
192.168.0.3  node02
```

如果我们现在希望在 managenode 上通过 SSH 在 node01 和 node02 上远程执行 shell 命令 (或者执行其他 SSH 命令)，需要进行以下步骤。

1、以 user01 身份登录到 managenode，在用户主目录下输入

ssh-keygen -t rsa

输入三次回车，注意，当系统提示输入信息时，不要输入任何内容，只需要直接回车即可。

这将在用户的根目录下生成一个 .ssh 的目录，里面有三个文件。

2、在 node01 和 node02 对应的用户根目录下创建 .ssh 目录。

将 Managenode 节点上用户根目录下 .ssh 目录中的 id_rsa.pub 文件拷贝到刚创建的目录下，并更名为 authorized_keys。

3、在 managenode 节点机上以 user01 身份执行：

ssh node01 date

如果 SSH 已经配置成功，则应打印出 node01 节点机当前的日期和时间。

与 RSH 相同，这样只是在 managenode 上可以以用户 user01 的身份在 node01 和 node02 上远程执行 shell 命令。如果需要在任何一个节点上都能在其他所有节点上都能远程执行 shell 命令，则需要重复进行类似的步骤，对名节点进行配置。

注：第一次使用 ssh 命令时会系统会有一次询问，是否要将目标主机加入已知主机列表，此时必须要输入“yes”，而不能输入“y”。

附录四：构建 PVM 集群

一、PVM 简介

PVM (Parallel Virtual Machine) 的开发始于 1989 年夏天，当前它的开发队伍包括美国橡树岭国家实验室（ORNL）、Tennessee 大学、Emory 大学以及 CMU 等单位，并得到了美国能源部、国家科学基金以及田纳西州的资助。

PVM 是一套并行计算工具软件，支持多种体系结构的计算机，比如工作站、并行机以及向量机等，通过网络将它们连起来，为用户提供一个功能强大的异构分布存储计算机系统。PVM 支持 C 和 Fortran 两种语言，到现在为止最新版本为 3.4.4。PVM 是免费的。

PVM 支持构建异构的集群，但本文的内容仅限于 PVM 在 Linux 集群上的安装与配置。PVM 编程模型见图 3。

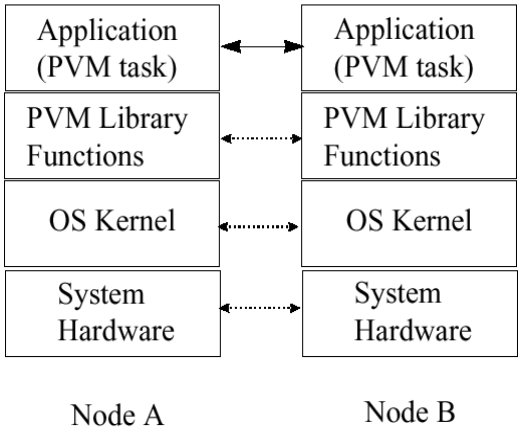


图 3. PVM 编程模型

在这个编程模型中，也体现出了 PVM 构建高性能并行计算环境的方式。基于 PVM 的并行应用程序只需要去将计算分解设计为一组 PVM 任务，这些任务可以通过标准接口例程库访问 PVM 资源。任务进程的控制、通信、同步等工作由 PVM 的标准例程来实现。可以认为，PVM 是在真正的操作系统的上层又构建了一个“虚拟”的、对用户看来唯一的计算平台。

实际上，为了管理进程，PVM 在启动时会在每个节点上同时启动一个 PVMD 后台进程，所有进程间的通信、同步及进程的派生、停止都是由 PVMD 完成的。

二、PVM 集群构建测试过程

环境说明：

三台节点机，机器名分别为：managenode、node01、node02。网络工作正常。

步骤 1：在各个节点机上安装 PVM 软件。

方案一，编译源代码方式。

1．自 http://www.netlib.org/pvm3/index.html 网站上下载 PVM 源代码，并拷贝到要安装的节点机，解压缩。

2．设置环境变量 PVM_ROOT 和 PVM_ARCH，假定用的是 bash，修改用户根目录下的 .bash_profile 文件，添加如下两行：

export PVM_ROOT=$HOME/pvm3

export PVM_ARCH=LINUX

3．然后执行 make 进行编译。

注：这种编译方式只是为当前用户安装了 PVM 环境，如果想安装为系统的软件，让所有用户都可以，那么应该以 root 身份进行安装，并执行 make install。

方案二，rpm 方式。

1．自网络上下载或者自 Red Hat Linux 安装池光盘获得 PVM 的 rpm 包

pvm-3.4.*.i386.rpm

pvm-gui-3.4.*.i386.rpm

2．以 root 身份执行

rpm –ivh pvm*.rpm

安装即可完成。

注：如果安装了 pvm-gui 软件包，则可以使用 XPVM。XPVM 是 PVM 的可视化控制、管理和统计工具，但是需要在窗口模式下才可以运行。

步骤 2：各个结点上设置一个相同的用户 (比如 user01)，密码也相同。

步骤 3：修改各个节点的 /etc/hosts 文件，配置主机名与 IP 地址的映射。

加入如下三行

192.168.0.1 managenode

192.168.0.2 node01

192.168.0.3 node02

步骤 4：配置用户的 shell 环境变量

修改所有三个节点上用户根目录下的 .bash_profile 文件

PATH=$PATH:$HOME/bin:/opt/csm/bin:/usr/share/pvm3/lib

PVM_ROOT=/usr/share/pvm3

export PATH PVM_ROOT MANPATH

修改所有三个节点上用户根目录下的 .bashrc 文件，添加以下两行

export PVM_ROOT=/usr/share/pvm3

export PVM_ARCH=LINUX

确认新的环境变量已经生效。

步骤 5：配置 RSH 环境 (或者 SSH 环境)

在各个节点的用户根目录下创建.rhosts 文件，内容如下（只是类似，假定用户 user01)：

managenode user01

node01 user01

node02 user01

将文件属性值改为 600

chmod 600 .rhosts

检验 rsh 是否正确配置的方法（以 node01 为例）

在 managenode 上，执行

rsh node01 date

步骤 6：确认 PVM 环境正常工作

登录到 managenode，输入命令：pvm

如果 PVM 环境正常工作，将进入 PVM 控制台，显示：

pvm>

然后输入

pvm>add node01

如果等到类似输出结果则说明 node01 配置无误：

1 successful

HOST DTID

node01 80000

然后尝试是否可以将 node02 顺利加入到 PVM 中

如果成功，则 PVM 环境配置工作结束

步骤 7：创建 PVM 运行时环境

在各节点的用户根目录下创建目录：

~/pvm3/bin/LINUX

默认情况下，PVM 将到此目录下查找编译好的可执行的 PVM 程序

现在就可以编译运行自己的 PVM 程序了。

步骤 8：编写 hello, world 程序

主程序：hello.c

```
/* hello.c */
#include <stdio.h>
#include "pvm3.h"
main()
{
    int cc, tid;
    char buf[100];
```

```
    printf("i'm t%x\n", pvm_mytid());
    cc = pvm_spawn("hello_other", (char**)0, 0, "", 1, &tid); /*  派生子程序*/
    if (cc == 1) {
        cc = pvm_recv(-1, -1);
        pvm_bufinfo(cc, (int*)0, (int*)0, &tid);
        pvm_upkstr(buf);
        printf("from t%x: %s\n", tid, buf);
    } else
        printf("can't start hello_other\n");
    pvm_exit();
exit(0);
}
```

从程序：hello_other.c

```
/*hello_other.c */
#include "pvm3.h"
main()
{
    int ptid;
    char buf[100];
    ptid = pvm_parent();
    strcpy(buf, "hello, world from ");
    gethostname(buf strlen(buf), 64);
    pvm_initsend(PvmDataDefault);
    pvm_pkstr(buf);
    pvm_send(ptid, 1);
    pvm_exit();
    exit(0);
}
```

步骤 9：编译 hello, world 程序

编写 makefile.aimk 文件

```
SHELL     =   /bin/sh
PVMDIR        =    ${PVM_ROOT}
SDIR      =     /home/user01/pvm3/bin/LINUX/
BDIR      =   $(SDIR)
XDIR      =   $(BDIR)
CFLOPTS   =   -g
CFLAGS        =    $(CFLOPTS) -I$(PVMDIR)/include $(ARCHCFLAGS)
PVMLIB        =    -lpvm3
PVMHLIB   =   -lpvm3
THREADLIB     =      -lpthread     -D_REENTRANT
#######################################################################
#######
LIBS      =   $(PVMLIB) $(ARCHLIB) $(THREADLIB)
```

```
HLIBS       =   $(PVMHLIB) $(ARCHLIB)
GLIBS       =   -lgpvm3
CC      =   gcc
GCC         =   g++
LFLAGS          =   $(LOPT) -L$(PVMDIR)/lib/$(PVM_ARCH)
CPROGS          =   de$(EXESFX)
default:  hello$(EXESFX) hello_other$(EXESFX)
clean:
    rm -f *.o $(CPROGS)
$(XDIR):
    - mkdir $(BDIR)
    - mkdir $(XDIR)
GPC_PVM_SPMD= $(SDIR)/gpc_pvm_spmd
HELLO= $(SDIR)/ hello
hello $(EXESFX):   $( HELLO).cpp $(XDIR)
    $(GCC) $(CFLAGS) -o $@ $( HELLO).cpp $(LFLAGS) $(LIBS)
    mv $@ $(XDIR)
hello_other $(EXESFX):   $(SDIR)/ hello_other.cpp $(XDIR)
    $(GCC) $(CFLAGS) -o $@ $(SDIR)/ hello_other.cpp $(LFLAGS) $(LIBS)
    mv $@ $(XDIR)
```

然后将 hello.c、hello_other.c 和 makefile.aimk 三个文件都拷贝到 makefile.aimk 文件中 SDIR 指定的目录中，执行指令 aimk 进行编译。编译完成后会在同一目录下生成两个可执行文件：hello 和 hello_other。

步骤 10：部署示例程序

将编译生成的可执行文件拷贝到各个节点的 ~/pvm3/bin/LINUX 目录下，可以通过 ftp 或者 rcp 实现，不过值得注意的是，如果通过 ftp 传输文件，传输完成后文件的可执行属性会失去，因此需要手工再进行设置。推荐用 rcp 命令或者集群系统提供的文件部署工具来部署示例程序。

注：实际上，只是主程序所在的节点（managenode)上需要同时有这两个可执行文件，其余节点（node01,node02）只需要有从程序。

步骤 11：运行示例程序

启动 PVM 环境

pvm

pvm>add node01 node02

quit

(只是退出了 PVM 的控制台，PVM 仍在运行）

在主程序所在目录下，执行 ./hello。

停止 PVM，在 PVM 控制台输入 halt。

附录五：XPVM 简要说明

一、修改关于节点机的配置文件

在用户的根目录下修改（新建）.xpvm_hosts 文件，示例如下：

```
pvm01
&pvm02
&pvm03 lo=curtis
```

如果机器名前面没有＆符号，则本机的 xpvm 启动以后自动将列表中的节点加入；如果有＆符号，则机器名只是出现在 xpvm 的 Hosts 列表中，并不自动启动。Lo=curtis 暂时不用。

二、启动

$xpvm　回车即可

三、添加（删除）节点机

单击菜单中　Hosts ，可以看到在.xpvm_hosts 中所列的节点机，要添加或删除某个节点，单击节点机名即可。（节点机名前矩形为深色说明节点在当前虚拟机中）。

四、执行程序

单击菜单中　Tasks | SPAWN，在 Command 文本框中输入要执行的程序名称。（确认要执行的程序已经在 PATH 路径中）

五、退出和停止虚拟机

File | Quit XPVM 仅退出 XPVM 控制台，虚拟机仍在运行。

File | Halt PVM　PVM 虚拟机停止。

附录六：构建 MPI 集群

一、MPI 简介

MPI（消息传递接口）是世界各地的工业、政府和科研部门等许多组织联合推出一个标准，用于并行程序间的消息传递，第一版标准 MPI 1.0 本于 1994 年 6 月推出，目前最新的为 MPI 2.0 版，于 1998 年年底推出。

MPI 的具体实现主要有三个：MPICH、LAMMPI 和 CHIMP，目前均已实现 MPI 1.2 版标准，可以应用于任何并行计算平台。部分 MPI 2.0 标准也已经得到了实现。本文中用到的是 MPICH。

MPICH 是一个与 MPI 规范同步发展的版本，主要由 Argonne 国家试验室和 MSU 完成。可以从 http://www-unix.mcs.anl.gov/mpi/mpich 免费下载得到。

MPI 是一个标准，各种 MPI 实现都是程序库，MPI 不是一门语言；MPI 所支持的编程模型也是消息传递模型 (同 PVM)。

二、MPI 集群构建测试过程

MPICH 只是提供了一套用于通信的接口，并不需要像 PVM 一样在后台运行守护进程。在默认情况下，以 rpm 方式安装的 MPICH 使用 SSH 进行节点间的互操作。

步骤 1：安装 MPICH

方案一：编译源代码方式。

1. 自 http://www-unix.mcs.anl.gov/mpi/mpich/ 下载 MPICH 某一版本的压缩包，解压缩。
2. 配置：

   切换到解压缩生成的目录第一层，执行

   ./configour

   如果需要安装到指定目录，在此步骤执行

   ./configure –prefix=/usr/local/mpich-1.x.x
3. 编译：执行

   make

   如果需要安装到前面指定的目录，则需要执行

   make install

方案二：rpm 方式

1. 下载或者自 SUSE Linux 的系统安装光盘上获得 MPICH 的 rpm 包。

   mpich-1.*.*.rpm

mpich-delel.1.*.*.rpm

2．以 root 身份执行

rpm –ivh mpich*.rpm

注：MPICH 的 rpm 包有两个，其中 mpich-1.*.*.rpm 只是运行环境，只能支持运行 MPI 程序，不能编译。若需要开发编译 MPI 程序，则必须要安装 mpich-delel.1.*.*.rpm

步骤 2：各个结点上设置一个相同的用户 (比如 user01)，密码也相同。

步骤 3：修改各个节点的 /etc/hosts 文件，配置主机名与 IP 地址的映射。

加入如下三行

192.168.0.1 Managenode

192.168.0.2 node01

192.168.0.3 node02

步骤 4：配置 SSH 环境。

步骤 5：编写 hello, world 程序

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

步骤 6：编译程序

方法 1：直接使用 mpiCC/mpicc，用法与 cc/gcc 相同。

mpiCC –o hello hello.c

此方法使用于编译比较简单的程序。

方法 2：编写 makefile，请参考 MPI 环境自带的例子。

/opt/mpich/share/examples。

步骤 7：部署程序

将上一步编译生成的可执行文件复制到各个节点，注意，如果使用 FTP 方式复制需要手工修改文件的属性使之可以执行。各个节点上可执行程序存放的位置，见下一步运行程序的要求。

步骤 8：运行程序

方法 1、最简单的运行命令

mpirun –np N hello

此时 MPI 所使用的默认的配置文件，N 为启动的进程数。使用这种

方法时，各节点上要运行的可执行程序必须放在相同的路径下。比如，在 managenode 节点上的 /home/user01/test 目录下运行 mpirun –np 3 hello，那么在 node01 和 node02 节点上，可执行程序 hello 必须也位于 /home/user01/test 目录下。

方法 2、自定义配置文件，指明所用机器

mpirun –machinefile hosts –np 6 hello

machinefile 文件中只需列出机器名，格式如下：

```
managenode
node01
node02
```

使用这种方法，也要求各节点上要运行的可执行程序在相同路径下，同上。

方法 3、使用更为详细的配置文件，指定所用的节点、可执行程序所在位置、相应节点上启动的进程数。命令格式为

mpirun –p4pg pgfile hello

其中 pgfile 的一种可能格式如下

```
managenode      0    /home/user01/test/hello
node01          1    /home/user01/test/hello
node02          1    /home/user01/test/hello
```

第一行中的“0”并不是说在 managenode 上没有进程运行，而是指在 managenode 上启动 MPI 程序的运行。