魔方超级计算机应用环境下

基础科学研究用户

上机手册

上海超级计算中心
上海海计信息技术有限公司

2015 年 4 月

目 录

魔方超级计算机是基于集群概念设计的大型计算机系统，其整体计算能力理论峰值为 200T flops (1Tflops 即为每秒 10^{12} 浮点计算)。2009 年 8 月在上海超级计算中心完成安装并投入运行。自 2014 年 10 月起，魔方超级计算上只为基础科学研究用户提供计算服务，原工程与工业计算用户已迁移至“蜂鸟”集群。

本文主要介绍在魔方超级计算机上部署的基础科学类应用软件和机器的使用方法和环境。

1. 基本环境

为了方便管理和使用，全机系统被分成三个部分即 A、B、C 三个区域。原工程与工业计算用户使用的 A 区现已关闭，目前只有 B 区和 C 区提供计算服务。B 区和 C 区主要用于各种源代码类计算，B 区由 650 个刀片式服务器组成，每个服务器包含 4 颗 AMD Barcelona 1.9G Hz 四核处理器，64G 共享内存；C 区由 800 个刀片式服务器组成，每个服务器包含 4 颗 AMD Barcelona 1.9G Hz 四核处理器，其中 300 个刀片为 32G 共享内存，另外 500 个刀片为 64G 共享内存。有关于集群硬件的详细配置请参看附录 A。

魔方主机系统的存储分为两种：每个计算节点配备的本地硬盘；由存储节点建立的高速并行文件系统。其中**本地硬盘严禁普通用户使用**，仅供计算节点操作系统使用，用户的所有操作都应该在帐号所对应的$HOME（该$HOME 所在的位置为高速并行文件系统）下进行，用户登录时，会自动被引导到自己帐号的$HOME 下面。鉴于存储空间有限和数据安全的考虑，请用户务必做到及时下载计算结果文件并清理空间。

1.1. 系统软件环境

1.1.1. 操作系统

计算节点和前端接入节点的操作系统均为 64 位 SuSE Linux Enterprise Server (SLES) 10 SP2，提供标准的 64 位 Linux 操作环境，用户需要事先适当熟悉命令行方式的基本 Linux 操作，特别是文件目录操作，并应该会熟练使用一种编辑器(vi 或者 emacs 等)。相关的 Linux 操作命令，可以参看附录 B。

1.1.2. 作业调度系统

大规模超级计算机系统，为了有效利用众多处理器核心所提供的计算能力，需要有一个作业管理系统，统一地跟用户交互，接收提交的各类计算任务，合理分配计算资源，将用户作业指派到具体的节点上执行。对用户来说不需要关心计算具体是在哪里进行的，系统会自动按照最优化原则进行调度，这不仅方便了用户的使用，更提高了整个系统的利用效率。作业管理系统是整个超级计算机最重要的软件环境之一，目前在魔方超级计算机上使用的作业管理系统是 Platform 公司的 LSF (Load Sharing Facility) 作业管理系统。

1.1.3. 编译器和并行实现

魔方主机系统支持 OpenMP 和 MPI 两种并行方式。OpenMP 为**共享内存**方式，仅能在一个计算节点内并行，最大线程数不能超过该节点处理器核心数（在 B 区和 C 区为 16）；MPI 则是**分布式内存**并行，计算作业可以在一个或者若干个节点上进行，最大进程数仅受用户帐号所能调用的 CPU 总数限制。

共享内存的 OpenMP 并行方式通常由编译器来支持，目前 GNU (需要 4.0 版本以上)、Portland Group (PGI)、PathScale、Intel 的编译器软件均已实现了对该标准的支持，只是具体支持的标准版本有所不同。

分布式消息传递的 MPI 并行方式，其实是一个设计规范的标准，提供了大量用于消息传递和管理的函数，支持从 C/C++和 Fortran 语言编写的程序中调用，也可以绑定到其它一些编程语言。MPI 只是一个标准，遵从这一标准可以有很多不同的软件实现，但具体的应用程序应该不加修改就可以重新编译运行，这也是标准化带来的优点。目前常见的支持 InfiniBand 网络的 MPI 实现是 MVAPICH / MVAPICH2 和 OpenMPI (注意跟 OpenMP 的区别)。

在高级编程语言支持方面，主要可以使用 GNU、Portland Group (PGI)、Intel、PathScale 还有 Open64 的 C/C++和 Fortran77/90 编译器。具体语言编写的程序如何编译，调用的相应命令可参看附录 C 中的表格。表里也列出了 MPI 对应各种语言的标准编译命令。在 Linux 操作系统下一般用 make 工具来组织管理源代码编译，而不是直接调用这些编译命令。

1.1.4. 数学库

开放源码程序往往要调用大量的数学函数进行各种计算，经过长期积累，已经有一些比较成熟的标准化的数学库，其中最常见的诸如线性代数方面的 ***BLAS***、***LAPACK***、***ScaLAPACK*** 和快速傅立叶变换 ***FFT*** 等等。通常情况下推荐使用 AMD 官方的 ***ACML*** 数学库(AMD Core Math Library)，魔方超级计算机上部署的 ACML 数学库的位置为 /home/compiler/pgi/linux86-64/7.0/lib/libacml.a，该库为 PGI-7.0 版本编译器所匹配的数学库，库

内的数学函数针对处理器进行了优化，能够获得更高的性能。对线性代数计算也可以用 ***GotoBLAS***，傅立叶变换则推荐 ***FFTW*** 库。如果使用 intel 编译器，则可以使用相关的 ***MKL*** 数学库。魔方超级计算机在/home/compiler/intel/mkl 目录下部署了两个版本的数学库，其中 MKL8.1.1 版主要和 intel-9.1 版本的编译器相匹配，MKL10.2.1 主要和 intel-11.1 版本的编译器相匹配。（目前魔方机器的编译器和相关数学库仅为测试版）

2. 应用部署

目前，在魔方超级计算机系统上已经测试并部署了大量的源代码软件。

2.1. 源代码软件

源代码软件的测试和部署主要是在刀片式服务器的 B 区和 C 区完成的。测试过的软件包括 abinit、cpmd、gamess、nwchem、vasp、amber、espresso、gromacs、siesta、WIEN2k、cp2k、gaussian03、lammps、smeagol、namd、DL_POLY、meep 等，其中部分软件由用户协助测试完成。

2.2. 计算队列分布

源代码计算队列主要分布在 B 区和 C 区。节点采用字母加数字的方式来进行编组，其中，B 区的节点编号以 a、b 开头，C 区的节点编号以 c、d、e 开头，后三位为一定规律的数字。

在每个区内，都开放了两个公共队列。

B 区包括 **snode** 和 **score** 两个队列，其中，**snode** 队列包括 **8000** 颗核，每个节点 64G 内存，只能提交 16 的整数倍的 CPU 数目的作业，作业完全按照提交时间的先后顺序运行，实行资源预约机制，不限制作业运行时间。score 队列包括 **2080** 颗核，每个节点 64G 内存，可提交任意 CPU 数目的作业，并优先将 CPU 分配给第一个满足下列要求的作业：即作业所需使用的 CPU 数目小于或等于队列里空闲的可使用的 CPU 数目。不限制作业运行时间。

C 区包括 csnode 和 cscore 两个队列，其中，**csnode** 队列包括 **8000** 颗核，每个节点 64G 内存，只能提交 16 的整数倍的 CPU 数目的作业，作业完全按照提交时间的先后顺序运行，实行资源预约机制，不限制作业运行时间。cscore 队列包括 **3200** 颗核，每个节点 32G 内存，可提交任意 CPU 数目的作业，并优先将 CPU 分配给第一个满足要求的作业：即作业所需使用的 CPU 数目小于或等于队列里空闲的可使用的 CPU 数目。不限制作业运行时间。

3. 上机操作

魔方系统内部有着复杂的网络系统来实现大规模集群系统的各类功能，为了安全起见，从外部公网只能通过 VPN 访问魔方集群。

要使用魔方超级计算机，必须先登录魔方超级计算机，通过作业调度系统进行作业提交、管理、监控、删除等操作。所有作业提交均通过提交作业脚本的方式来进行。无论 B 区还是 C 区，都分配有各自独立 telnet 登录节点和 FTP 文件传输节点，这些节点分配有独立的 IP 地址，**禁止在登录节点运行任何大规模程序和编译任何程序**，只可以进行简单的文本操作。用户可以到编译节点编译程序，运行小规模的测试。

B 区编译节点为 a110 和 a111，C 区的编译节点为 c110 和 c111。

一个正常作业基本步骤如下：

（1） 模型准备——用户准备模型数据文件和作业脚本文件。

（2） 模型上传——通过 FTP 工具将模型数据文件和脚本文件上传至 FTP server。

（3） 作业提交——利用 Telnet 或者 Putty 工具登陆魔方超级计算机，用 dos2unix 命令处理上传的文本文件后，用作业提交命令提交脚本文件进行计算。

（4） 作业监控——通过 Telnet 或者 Putty 工具方式登录魔方机，采用作业管理命令监控作业的执行情况。

（5） 结果下载——计算完成后，通过 ftp 工具从 FTP Server 下载结果文件。

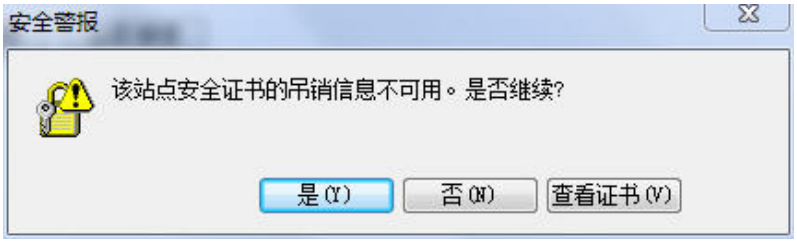
下面分别介绍上机操作的相关内容。

3.1. 登录和传输文件

3.1.1. VPN 验证

为了确保用户登录的安全性，自 2010 年 4 月起，超算中心采用 VPN 软件对用户的登录进行管理。用户首先需要使用浏览器访问 https://vpn.ssc.net.cn，在该页面中输入自己的 VPN 用户名和口令，可以对该页面进行访问。

注意：在登录网站时，会出现 Windows 安全警报，选择“是”，在接下来的页面，点击“继续浏览该网站”，即可进入上海超算中心 VPN 主页。



进入到超算中心 VPN 主页后，请根据自己的操作系统阅读对应的 SSL VPN 用户使用说明（如下图所示）。其中对于 Windows 系统，提供了 web 和客户端两种登陆方式的使用说明，并提供了适用于 windows 32 位和 windows 64 位的客户端下载链接。Linux 使用客户端的方式登陆，网页上提供了《SSL VPN 用户使用说明（linux）》以及 SSL VPN 客户端软件（linux）的下载链接。

在 windows 系统下以网页方式登陆 VPN：可以在进入上海超算中心 VPN 主页以后，选择“主机应用”标签。按照系统提示安装标题为“Array Networks，Inc 中的 Array Networks Clients Software”的控件后，点击“开启 VPN 客户端”按钮，即开始尝试 VPN 连接，连接成功后，会显示分配了一个 192.168 开头的内网 IP。

在 windows 系统下以客户端方式登陆 VPN：根据本地系统类别下载对应的 SSL VPN 客户端软件，客户端软件的安装使用说明可参照《SSL VPN 用户使用说明（Windows 客户端方式）》。按照提示安装好客户端软件后，从“Profile”菜单选择 New，按照下图格式输入网址、用户名和密码，点击“Save”按钮保存。双击生成的 Profile，即可开启 VPN 连接。

当通过 web 或者客户端连接成功后，客户可以使用内部网址访问魔方超级计算机的 B、C 分区，在没有切断连接前，不需要重复登录。另外，在停止访问魔方主机前，不要关闭 VPN 客户端或者 VPN 的浏览器访问。

注意：如果您使用的是 win8.0/win8.1 系统，或者在 64 位 Win vista/Win7/Win8.1 系统下，使用 64 位浏览器登录 VPN，请打开“官网最新客户端软件下载链接”并参照其中的说明。

3.1.2. 终端登录

在使用 VPN 成功登录超算中心后，可以使用 Telnet 工具以内部地址登录魔方超级计算机 B、C 区域。

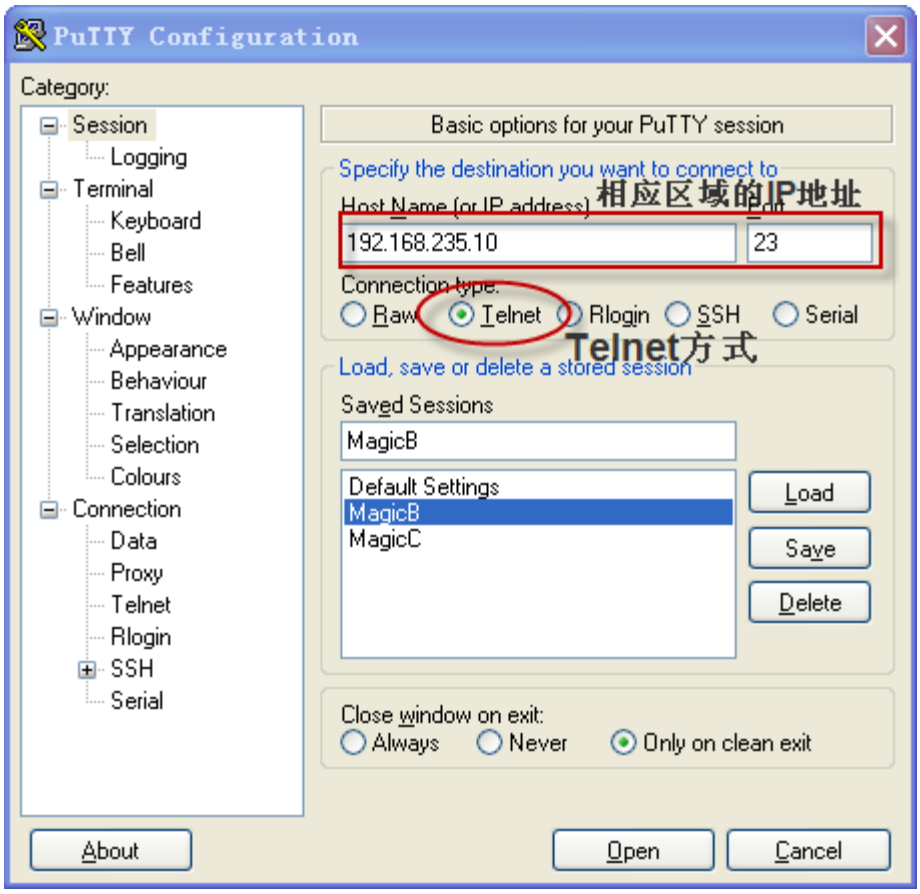
魔方超级计算机的内部 IP 地址为：

主机	服务	内网地址
魔方 B 区	telnet	192.168.235.10

	telnet	192.168.235.20
	ftp	192.168.235.50
魔方 C 区	telnet	192.168.235.30
	telnet	192.168.235.40
	ftp	192.168.235.70

在 windows 系统下，我们建议用户使用 telnet 工具进行登录。例如免费 telnet 软件 Putty，其官方下载地址为：http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html，也可以通过华军等下载网站获得。

下面以 putty 为例演示魔方超级计算机的登录过程。登录 B 区的过程如下：填写 IP address 192.168.235.10，填写 Port 为 23，选择连接方式为 Telnet。



然后点击“Open”或回车后，会进入提示，需要输入用户口令和密码，输入正确后会出现提示：

Welcome to MagicCube Scientific Computing Center of SSC

Have a nice day!

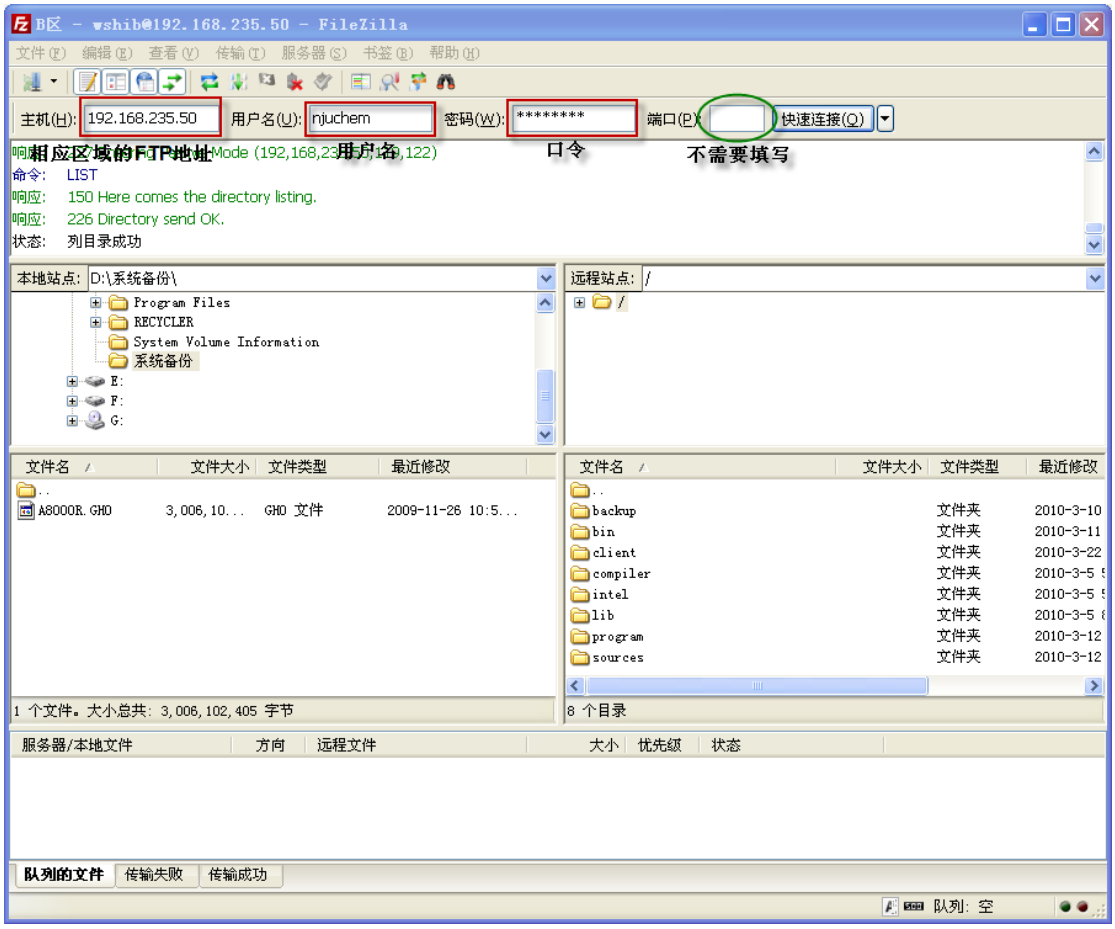
此时可执行 Linux 命令，例如登录到 B 区编译节点，可执行命令：***ssh a110***。

3.1.3. 数据传输

用户必须通过 FTP 进行文件的传输。登录时，需使用主机用户名和主机密码。**注意 FTP 服务是全天 24 小时可以连接的，但需提前使用 VPN 登录魔方集群。**

在 Windows 系统下，推荐用户使用免费 ftp 软件 Filezilla，下载的官方地址为：http://sourceforge.net/projects/filezilla/免费下载，或者通过华军下载等国内知名网站获得。

安装 Filezilla 软件后，可以双击打开该软件，按照下图进行设置，完成后单击快速链接或者回车可以登录 ftp 站点。在文件菜单中，可以选择“站点管理器”保存 ftp 地址、用户名和密码，下次登录时只需要从“站点管理器”中选择相对应的 FTP 地址即可直接登录。



此外，也可以使用商业软件 cuteftp、flashfxp 等进行登录。在有大量文件需要下载时，建议使用 tar 命令进行打包，以加快下载速度，减少出错几率。该命令为：**tar –zcvf file.tar.gz file**，其中 file 为需要打包压缩的文件或目录，file.tar.gz 为打包后的文件。

注意：传输文件一般是用二进制方式，但文本文件包括提交作业的脚本在上传结束后，必须用 dos2unix 命令进行转换（主要是转换回车换行符号），否则有些计算程序可能会在读取时出错。

3.2. 编译

成功登录后，首先进入的是登录节点。用户可以在登录节点查看目录、编辑文件、查看作业、查看资源使用情况等。但是用户**不允许在登录节点**运行计算程序或前后处理程序，也不允许进行程序编译。

用户可以从登录节点转移到编译节点进行程序编译。B 区的编译节点为 a110 和 a111；C 区的编译节点为 c110 和 c111。转移登录的命令为 ssh，例如用户可以使用命令 ssh a110 登录到编译节点 a110。关于魔方超级计算机上编译器的配置和使用方法，可参考前面的 1.1.3 节

3.3. 作业提交和软件使用

3.3.1. 作业提交

3.3.1.1. LSF 脚本

计算任务是通过脚本文件提交到作业管理系统的，脚本文件是一个常规文本文件，具有执行权限，可以直接在登入节点使用 vi 编辑器编写，也可异地编写上传至用户作业工作目录，但要注意 dos2unix 转换一下。

脚本文件名无特殊规定，起一个可识别的名字即可。编辑完成脚本文件后，将脚本赋予可执行权限，然后提交。例如对一个名称为 test.lsf 的作业脚本文件，编辑完成后，需要执行命令 chmod 755 test.lsf 赋予执行权限，然后使用命令 bsub ./test.lsf 来提交。**注意，**提交命令中，在 test.lsf 前面有./指定 test.lsf 脚本的位置。

作业脚本范例：

APP_NAME=snode	指定计算任务所要进入的队列
NP=128	计算所需的全部 core 数
NP_PER_NODE=16	每节点所需 core 数（最多不超过 16） 本参数可以省略，对 score 和 cscore 队列会根据空余情况将作业分配到节点上，对 snode 和 csnode 队列则默认全部是 16
RUN=”/home/user/test/bin/mdrun”	可执行文件的全路径

脚本参数含义：

APP_NAME

指定作业运行使用的队列名称，具体可用队列可参照前面 2.3 节。

NP

指定作业运行需要的 CORE 数，如果该参数不指定，则缺省值为 1 个 CORE。

NP_PER_NODE

指定每个节点上最多分配给作业运行的 core 数。NP_PER_NODE 必须小于或等于 16。**建议不指定此参数**。如果不指定此参数，系统默认会对 snode 队列以 NP_PER_NODE=16 来分配资源，对提交到 score 队列中的计算作业，则会搜寻所有可用的资源，每节点上的 core 数有可能不相等。

RUN

指定具体执行的命令参数，**必须用双引号" "**将命令行引起来。例如，手动运行一个 mpi 程序的命令为 mpirun –np 4 –machinefile machinefile /home/user/test/bin/mdrun，此时脚本里面就要写成 RUN=”/home/user/test/bin/mdrun”，系统会自动加载前面的 mpi 参数。

注意：严禁使用任何前台或者后台方式直接由用户运行程序，所有的计算都必须作为任务提交到 LSF 系统然后统一调度执行，否则影响到主机正常运行。

3.3.1.2. 典型作业脚本

魔方超级计算机上最常见的用户作业可以分为几类，一类是普通串行程序，只需使用一个 core 即可；一类是必须在同一个节点内才能运行的 OpenMP 或者基于线程(threads)的共享内存并行程序；最后一类是消息传递方式的 MPI 并行程序，当然还有更加复杂的 OpenMP+MPI 混合并行程序，以及流程驱动调用多个计算程序的复杂脚本作业。下面针对这几种类型分别介绍如何向作业管理系统提交任务。

3.3.1.2.1. 普通串行计算

APP_NAME=**score**	指定计算任务所要进入的调度队列
NP=1	任务所需 core 数
RUN=”a.out”	执行计算命令，如果不在系统查找目录里则给出全路径名

这里 RUN 中给出的计算命令，可以是一个可执行的计算程序，也可以是一个执行一系列计算的脚本，两者跟通常在 bash 等 shell 中一样，不加太多区分地执行。

3.3.1.2.2. 共享内存并行作业

这类作业既包括 OpenMP 并行方式的，也包括不使用 OpenMP 而是通过 POSIX 等系统底层线程库所编写的多线程程序，概言之，任何多线程程序都通过下面方式提交计算。最典型的就是 Gaussian 量化软件。注意对这类程序，性能随着 core 数的增加，不一定会线性增加甚至会下降。通常用 4 个进程即可，可以通过典型算例在 2、4、8、16 个线程下的计算速度来确定。

#!/bin/sh	
APP_NAME=score	指定计算任务进入 score 调度队列
NP=4	任务需四个 core 执行亦即四个进程
NP_PER_NODE=4	每节点要求有四个 core 可用
RUN=”RAW”	固定格式说明
export OMP_NUM_THREADS=$NP	设置 OpenMP 线程数环境变量
/path/to/g03 test.com	执行计算命令，如果不在系统查找目录里则给出全路径名

注意在这里，NP、NP_PER_NODE 和 OMP_NUM_THREADS 必须一致，而且不能超过每节点 16 个核心的限制。同时还要注意修改计算软件的输入文件，比方 Gaussian 软件输入文件中的%nproc 参数，其应该与 NP 一致。

如果可执行文件是一个调用 OpenMP 计算程序的脚本，采用这种形式提交也是可以的，只是在 OpenMP 计算程序之外的其它计算和处理部分，会只在一个 core 上执行，部分浪费了资源。

3.3.1.2.3. MPI 并行作业

目前在魔方系统上推荐使用 MVAPICH 1.0 版本，编译好后，计算任务通过下面的脚本提交。

APP_NAME=snode	指定计算任务进入 snode 调度队列
NP=128	任务需 128 个 core，在 snode 中必须为 16 整数倍
NP_PER_NODE=16	每节点要求有 16 个 core 可用
RUN=”/path/to/vasp”	执行计算命令，如果不在系统查找目录里则给出全路径名 系统会自动调用 mpirun 并给出适当的-n 和-machinefile 参数 不需要在此显式地给出

采用这种方式提交，可执行文件**必须是**一个二进制的 MPI 程序，**不能为**含有 mpirun 命令的脚本或者其它脚本。

3.3.1.2.4. OpenMP+MPI 混合并行作业

由于前面所述，这种并行方式，通常情况下可以采用每节点上的 16 个核心，分为四组，每组由一个 MPI 进程，产生并管理四个 OpenMP 线程的配置。这也是推荐的并行作业负载分配。

4 个 MPI 进程×每进程 4 个 OpenMP 线程
8 个 MPI 进程×每进程 2 个 OpenMP 线程
2 个 MPI 进程×每进程 8 个 OpenMP 线程
1 个 MPI 进程×每进程 16 个 OpenMP 线程

以上是四种并行负载分配方案，可以通过典型算例做性能测试后，决定采用哪一种。下面用 ***m*** 代表 MPI 进程数，***k*** 代表每 MPI 进程产生和管理的 OpenMP 线程数，为了避免其它计算作业的相互影响，一般这类作业**只提交到 snode 队列，并注意设置 *m*×*k*=16。**

#!/bin/sh	
APP_NAME=snode	指定 snode 队列
NP=128	总计 128core
NP_PER_NODE=16	每节点 16core，独占计算节点
RUN="RAW"	
M=m **K=k**	*m* 和 *k* 见上
rm -f hosts.list **for i in `echo $LSB_HOSTS`; do** **echo $i >>hosts.list** **done**	产生 hostfile 文件

cat hosts.list | sort | uniq > hosts.uniq
rm -f hosts.mpi
for i in `cat hosts.uniq`; do
for j in `seq 1 $M`; do
echo $i >>hosts.mpi
done
done
	注意
	每个节点 MPI 进程数

export OMP_NUM_THREADS=$K	设定环境变量
N=`cat hosts.mpi | wc -l | awk '{ print $1 }'`	计算 MPI 进程总数
**mpirun_rsh -np $N -hostfile **	执行 mpirun

hosts.mpi /path/to/executable -cmd_line_options	必要的参数 可执行程序和参数

3.3.1.2.5. 需要大内存的计算

在魔方系统上，每个 core 平均有 4G 内存，应对普通计算一般没有问题，但对某些特定的应用，如果内存使用量巨大的话，需要设法加以解决。

在 snode 队列中独占一个计算节点，64G 的内存中会保留 4G 给操作系统和其它基本软件组件，剩余 60G 可供用户程序使用，若该节点只有一个 core 运行计算，则其可以使用所有的 60G 可用内存，若两个 core 则各自可用 30G，以此类推。

普通串行脚本如下

#!/bin/sh	
APP_NAME=snode	指定计算任务进入 snode 调度队列
NP=16	任务需 16 个 core，在 snode 中必须为 16 整数倍
NP_PER_NODE=16	**每节点要求有 16 个 core 可用，独占该计算节点**
RUN=”RAW”	
/path/to/big_mem_code	执行大内存需要的串行计算

OpenMP 计算脚本如下，假定可以执行四个 OpenMP 线程，每个 15G 内存。

#!/bin/sh	
APP_NAME=snode	指定计算任务进入 snode 调度队列
NP=16	任务需 16 个 core，在 snode 中必须为 16 整数倍
NP_PER_NODE=16	**每节点要求有 16 个 core 可用，独占该计算节点**
RUN=”RAW”	
export OMP_NUM_THREADS=4	指定每个节点启动 4 个线程
/path/to/big_mem_code	执行大内存需要的串行计算

MPI 计算脚本如下，假定每节点执行 *m*=4 个 MPI 进程，各自可用 15G 内存。

#!/bin/sh	
APP_NAME=snode	指定 snode 队列
NP=128	总计 128core

NP_PER_NODE=16	**每节点 16core** **独占计算节点**
RUN=”RAW”	
M=m	*m* 见上

rm -f hosts.list
for i in `echo $LSB_HOSTS`; do
echo $i >>hosts.list
done
cat hosts.list | sort | uniq > hosts.uniq
rm -f hosts.mpi
for i in `cat hosts.uniq`; do
for j in `seq 1 $M`; do
echo $i >>hosts.mpi
done
done
	产生 hostfile 文件
	注意
	每个节点 MPI 进程数

N=`cat hosts.mpi | wc -l | awk '{ print $1 }'`	计算 MPI 进程总数
mpirun_rsh -np $N -hostfile hosts.mpi ** **/path/to/executable -cmd_line_options	执行 mpirun 必要的参数 可执行程序和参数

3.3.1.2.6. 串行和并行混合的计算脚本

很多情况下，计算是由预处理、并行计算和后处理分析，或者串行计算和 MPI 并行计算混杂的方式进行。也有一些软件的计算脚本，既有串行计算或者串行处理命令，又有并行计算命令。此时可以按照下面的例子改写脚本。

#!/bin/sh	
MY_MPI_TYPE=mvapich **MY_MPI_HOME=/home/compiler/mpi/mvapich/1.0/gcc.pgf90**	设定 MPI 参数
APP_NAME=”snode”	计算任务进入 snode 队列
NP=16	任务需 16 个 core

NP_PER_NODE=16	每节点有 16 个 core 可用
RUN=”RAW”	
…	
/path/to/prepare -cmd_line_options	串行处理或者计算
…	
/home/lsf/7.0/linux2.6-glibc2.3-x86_64/bin/mpirun.lsf	并行计算，注意 mpirun 后
/path/to/mpi_calc -cmd_line_options	无需 np 和 machinefile 参数
…	
/path/to/postprocessing	后处理
…	可以继续其它串行或者并
…	行计算

3.3.1.2.7. 大量相同类型计算

可以使用 bash 或者其它脚本来通过循环方式批量提交作业。但需要控制好每个计算的规模，以及全部计算作业的数量。

3.3.2. 源代码软件使用

此处只说明一些常用源代码的特殊提交脚本，其他更多源代码的特殊提交脚本可以询问您的技术支持工程师。

3.3.3.1. espresso

expresso 作业的特殊之处在于它的输入文件需要用-in 参数传递。一个 espresso 文件的脚本例子如下，另外这个脚本也是一个很好的串并行混合执行的参考例子。

```
#!/bin/sh
APP_NAME="score"
NP=8
RUN="RAW"
CURDIR=$PWD
#start creating .nodelist
rm -rf $CURDIR/.nodelist >& /dev/null
```

```
for i in `echo $LSB_HOSTS`
   do
   echo $i >> $CURDIR/.nodelist
done
#nodelist done
EXEDIR=/home/users/twang/software/intel9.1_compiler/espresso-4.0.4/bin
mpirun -np $NP -machinefile $CURDIR/.nodelist $EXEDIR/pw.x -in ren.scf.fit.in > ren.scf.fit.out
mpirun -np $NP -machinefile $CURDIR/.nodelist $EXEDIR/pw.x -in ren.scf.in > ren.scf.out
mpirun -np $NP -machinefile $CURDIR/.nodelist $EXEDIR/ph.x -in ren.ph.in > ren.ph.out
mpirun -np 1 -machinefile $CURDIR/.nodelist $EXEDIR/q2r.x < ren.q2r.in > ren.q2r.out
mpirun -np 1 -machinefile $CURDIR/.nodelist $EXEDIR/matdyn.x < ren.dyn.in > ren.dyn.out
mpirun -np 1 -machinefile $CURDIR/.nodelist $EXEDIR/matdyn.x < ren.ep.in > ren.ep.out
```

3.3.3.2. gamess

gamess 程序的特别之处在于它的执行文件用脚本封装在 rungms 中，所以需要改写 rungms，并使用类似于下面的这个脚本提交。

```
#!/bin/sh
APP_NAME="debug"
NP=4
RUN="RAW"

INP=exam01.inp
VER=00
export SCR=$PWD/$LSB_JOBID
/bin/mkdir -p $PWD/$LSB_JOBID
$HOME/gamess/rungms $INP $VER $NP > log
```

3.3.3.3. WIEN2K

WIEN2K 是一个 mpi 和 openmp 混合编程的程序。下面这个脚本提供了提交 WIEN2K 作业

的例子

```
 #!/bin/sh
APP_NAME=score
NP=4
RUN="RAW"
export SCRATCH=$HOME/scratch/wien2k
export WIENROOT=$HOME/WIEN2k_08
export STRUCTEDIT_PATH=$WIENROOT/SRC_structeditor/bin
export PATH=$PATH:$WIENROOT:$STRUCTEDIT_PATH:.
export OCTAVE_EXEC_PATH=${PATH}::
export OCTAVE_PATH=${STRCUTEDIT_PATH}::
export PATH=.:$WIENROOT:$PATH
#start creating .machines
echo 'granularity:1' >.machines
echo "lapw0:"`echo $LSB_HOSTS |cut -d" " -f1` >> .machines
for i in `echo $LSB_HOSTS`
do
    echo "1:"$i >> .machines
done
echo 'extrafine:1' >>.machines

echo "------------------------------------------------"
# Run the parallel executable "WIEN2K"
time run_lapw -p -i 10 -ec 0.000001 >out
```

3.4. 作业管理

下面列出常用的作业管理命令，如果需要更详细的资料可以参考作业调度系统相关手册。

命令	用途
bsub –J job_name ./task.lsf	提交 LSF 任务，成功后会给出此任务的 JOBID -J 后给出作业名称，可以省略-J 参数，test.lsf 应有执行权限
bjobs	查看自己的所有运行任务情况；说明：输入 bjobs 后，会列出当前用户正在运行的所有作业，最左边一列数字是每个作业的 JOBID，一些其他命令使用的时候需要调用这个 JOBID。
bjobs -l	查看所有运行任务的详细情况
bjobs –l JOBID	查看 JOBID 这个任务的详细情况
bpeek JOBID	查看某任务屏幕输出
bpeek -f JOBID	跟踪查看某任务屏幕输出
bkill JOBID	终止某任务运行
bkill JOBID1 JOBID2 JOBID3	终止多个任务运行
busers	查看用户账号计算资源权限
bqueues	查看所有任务队列的状态
bstop JOBID	临时挂起某个计算作业，为其它计算腾出资源
bresume JOBID	恢复由 bstop 挂起的作业

执行 busers 命令的屏幕输出如下

```
bqwang@a111:~/test/d.dppc> busers
USER/GROUP          JL/P    MAX  NJOBS   PEND    RUN  SSUSP  USUSP    RSV
bqwang                 -    512    128      0    128      0      0      0
```

MAX: 用户可用 core 数上限

NJOBS: 已提交作业所需要的全部 core 数

PEND: 因种种原因正在队列中等待执行的作业所需全部 core 数

RUN: 正在运行的作业所使用的全部 core 数

SSUSP: 系统挂起的用户作业所使用 core 数

USUSP: 用户自行挂起的作业所使用 core 数

RSV: 系统为你预约保留的 core 数

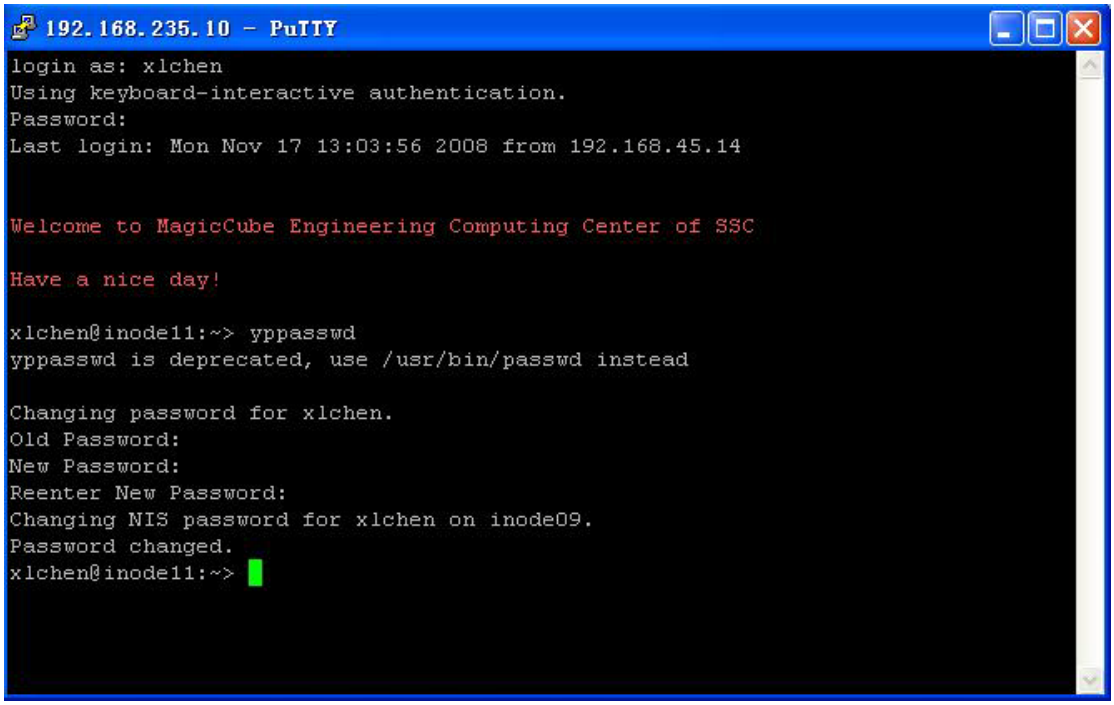
一个作业提交到队列后，将有可能为以下的几种状态之一。

PEND	任务在队列中排队等待
RUN	任务正在执行
PSUSP	任务在队列中排队等待时被用户挂起
SSUSP	任务被系统挂起
USUSP	任务被用户自行使用 bstop 命令挂起
DONE	作业正常结束，exit 代码为 0
EXIT	作业退出，exit 代码不为 0

3.5. 用户管理

3.5.1. 主机用户密码修改

用户通过 telnet 登陆主机后，可以通过 yppasswd 命令来修改主机密码。



修改后重新登陆时便要求输入新的主机密码。

3.5.2. 主机用户权限调整

用户需要调整使用队列、最大 CPU 数等权限时，请联系您的项目经理。

3.5.3. 远程用户密码修改

用户需要修改远程密码时，请联系您的项目经理。

3.5.4. 远程用户权限调整

用户需要变更或调整远程 IP 地址、接入方式、接入时间等权限时，请联系您的项目经理。

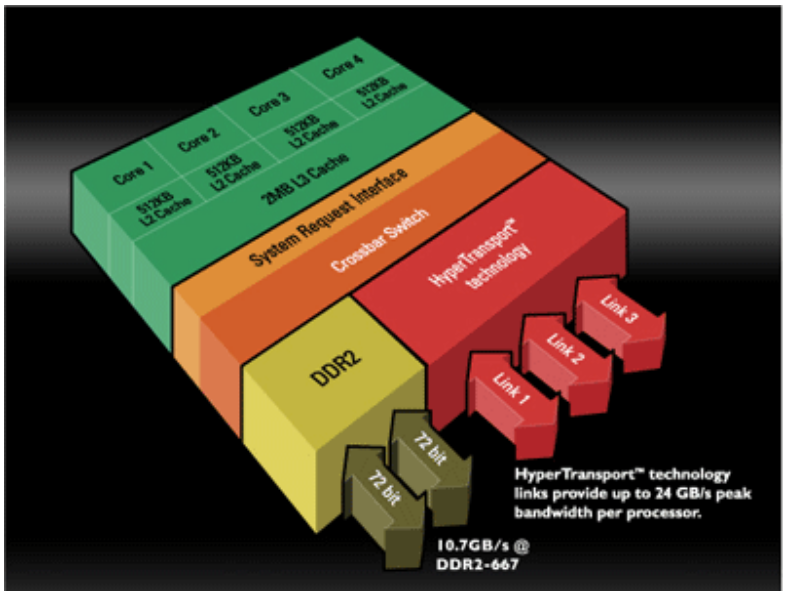
附录 A：魔方机器硬件环境

魔方超级计算机系统被分成三个部分即 A、B、C 三个区域，A 区已于 2014 年 9 月关闭，只有 B 区和 C 区提供计算服务。B 区由 650 个刀片式服务器组成，每个服务器含 4 个 AMD Barcelona 1.9GHz 低功耗型 Opteron 8347HE 四核处理器，64GB 共享内存，InfiniBand 光纤网络互联，主要用于各种源代码类计算；C 区由 800 个刀片式服务器组成，每个服务器含 4 个 AMD Barcelona 1.9GHz 低功耗型 Opteron 8347HE 四核处理器，InfiniBand 光纤网络互联，其中 300 个刀片含 32GB 共享内存，500 个刀片含 64GB 共享内存，主要用于各种源代码类计算。下面分别介绍魔方超级计算机的基本硬件配置。

以 B 区为例，一个计算节点的典型硬件配置如下：

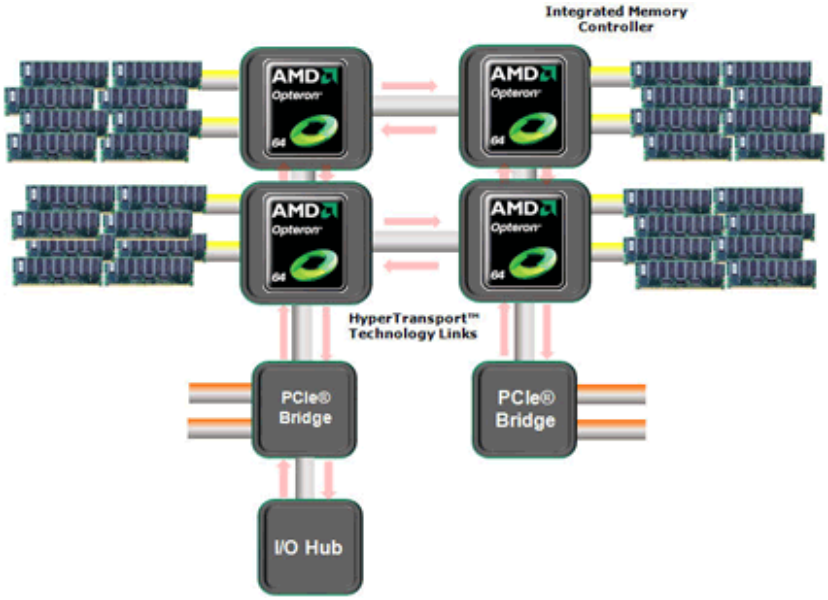
- 四路四核 AMD Barcelona 1.9GHz 低功耗型 Opteron 8347HE 处理器（每节点合计 16 核）
- 每处理器内建内存控制器用于访问本身的 16GB 内存，每节点合计 64GB 内存（四个内存控制器，平均每个处理器核心 4GB 内存）
- Voltaire InfiniBand 4x DDR (Double Data Rate)光纤网络，理论带宽 20Gbits 每秒

每个计算节点内部，全部 64GB 内存由 64 位操作系统统一编址，对所有 core 都可访问，但属于该 core 所在处理器的内存，由内建内存控制器直接存取，访问速度较快，由其他处理器控制的内存，需要通过 AMD 特有的 HyperTransport 总线来访问，相对来说存取性能稍差。下图是 AMD Barcelona 四核处理器的架构示意图，每个核心独立使用 512KB 的 L2 缓存，四个核心共享 2MB 的 L3 缓存、HyperTransport 总线和 DDR2 内存控制器。



图一、AMD Barcelona 四核处理器的架构

下图是四路 AMD Barcelona 处理器的计算节点示意图。



图二、四路 AMD Barcelona 处理器的计算节点

附录 B：简单 LINUX 命令

名称：cd

语法：cd [dirName]

说明：更改工作目录至 dirName。其中 dirName 可表示为绝对路径或相对路径。

范例： 跳到 /usr/bin/：

cd /usr/bin

名称：pwd

语法：pwd [--help][--version]

说明：显示工作目录。

名称 : ls

语法： ls [-alrtAFR] [name...]

说明： 显示指定工作目录下之内容（列出目前工作目录下的文件及子目录)。

范例：列出目前工作目录下所有名称是 s 开头的文件：

ls s*

名称： mkdir

语法：mkdir [-p] dirName

说明：建立名称为 dirName 的子目录。

范例：在工作目录下，建立一个名为 AAA 的子目录：

mkdir AAA

名称：cp

语法：cp [options] source... directory

说明：将一个文件拷贝为另一文件，或将数个文件拷贝至另一目录。

范例：将文件 aaa 复制(已存在)，并命名为 bbb :

cp aaa bbb

名称：mv

语法：mv [options] source... directory

说明：重新命名文件，或将数个文件移至另一目录。

范例：将文件 aaa 更名为 bbb :

mv aaa bbb

名称： tar

说明：备份文件。可用来建立备份文件，或还原备份文件。

语法：如需备份 test 目录下的文件，并命名为 test.tar，可执行命令：

tar –cvf test.tar test/

如果需要在备份时，将 test 目录进行压缩，并保存为 test.gz，则需要执行命令：

tar –zcvf test.gz test/

如需解压缩相关的 test.tar 文件，可执行命令：

tar –xvf test.tar

如果需要解压缩相对应的 gz 后缀名文件，可执行命令：

tar –zxvf test.gz

名称：rm

语法：rm [options] name...

说明：删除文件及目录。

范例：删除后缀名为.c 的文件：

rm *.c

名称：rmdir

语法： rmdir [-p] dirName

说明： 删除空的目录。

范例：将工作目录下，名为 AAA 的子目录删除 :

rmdir AAA

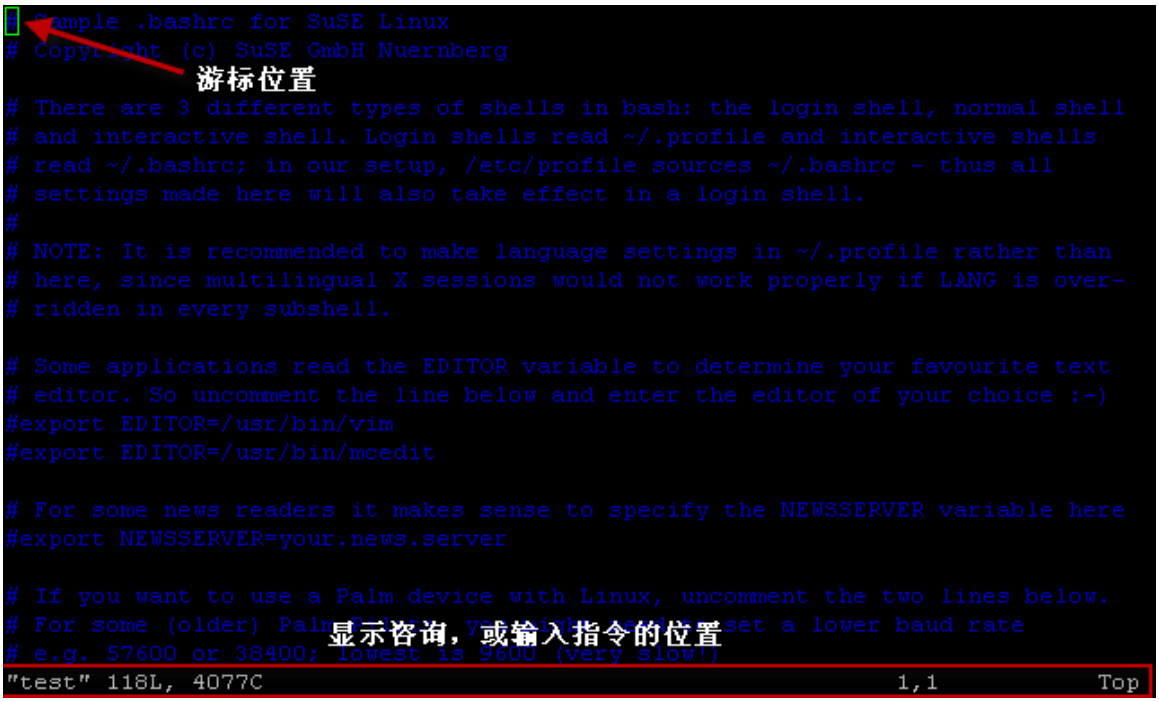
名称：vi

语法： vi filename

说明： Linux 下常用文本编辑器，可以对文档的内容进行输入、修改和删除。

范例：修改 test 文件。

Vi test



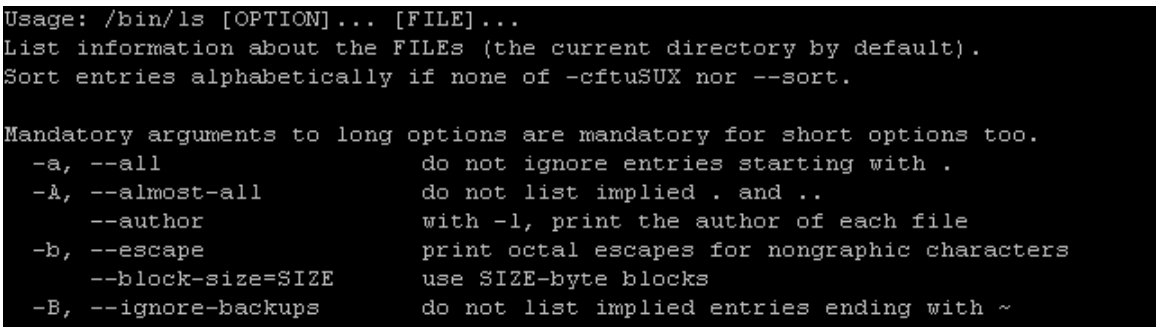
参数：

一般模式下按下“i”键 进入编辑模式，开始编辑文字。

按下“ESC” 键回到一般模式。

在一般模式中按下“:wq ”储存并退出 vi。

注：在 Linux 下，每一项命令都有多个参数可供选择，在命令后直接加“--help”可以显示该命令的格式以及所有命令参数。例如对于 ls 命令，可执行：ls --help 显示该命令的所有参数。



附录 C：魔方超级计算机上部署的编译库

表一、在魔方超级计算机系统上部署的编译器

编译器	安装目录	版本	相关命令
GNU C	/usr/bin	4.1.2	gcc myprog.c
GNU C++	/usr/bin	4.1.2	g++ myprog.cpp
GNU Fortran	/usr/bin	4.1.2	gfortran myprog.f
PGI C	/home/compiler/pgi	7.0	pgcc myprog.c
PGI C++	/home/compiler/pgi	7.0	pgCC myprog.cpp
PGI Fortran77	/home/compiler/pgi	7.0	pgf77 myprog.f
PGI Fortran90	/home/compiler/pgi	7.0	pgf90 myprog.f90
PGI Fortran95	/home/compiler/pgi	7.0	pgf95 myprog.f95
Intel C/C++	/home/compiler/intel/9.1	9.1	icc myprog.c
	/home/compiler/intel/11.1	11.1	
Intel Fortran	/home/compiler/intel/9.1	9.1	ifort myprog.f
	/home/compiler/intel/11.1	11.1	
MPI C	详见以下 MPI 实现的具体版本		mpicc myprog.c
MPI C++			mpiCC myprog.cpp
MPI Fortran77			mpif77 myprog.f
MPI Fortran90			mpif90 myprog.f90

*目前魔方超级计算机上部署的 intel 编译器仅为测试版。

表二、魔方超级计算机系统上部署的 MPI 实现

MPI				
MPI 实现	后端编译器		版本	安装目录
mvapich	gcc	gfortran	1.0	/home/compiler/mpi/mvapich/1.0/gcc.gfortran
	gcc	.pgf90		/home/compiler/mpi/mvapich/1.0/gcc.pgf90
	icc	ifort-11.1		/home/compiler/mpi/mvapich/1.0/icc.ifort-11.1
	icc	ifort-9.1		/home/compiler/mpi/mvapich/1.0/icc.ifort-9.1

	pgcc	pgf90		/home/compiler/mpi/mvapich/1.0/pgcc.pgf90
openmpi	pgcc	pgf90	1.2.8	/home/compiler/mpi/openmpi/1.2.8/pgcc.pgf90
	gcc	gfortran	1.3.1	/home/compiler/mpi/openmpi/1.3.1/gcc.gfortran
	icc	ifort-11.1		/home/compiler/mpi/openmpi/1.3.1/icc.ifort-11.1
	icc	ifort-9.1		/home/compiler/mpi/openmpi/1.3.1/icc.ifort-9.1
	pgcc	pgf90		/home/compiler/mpi/openmpi/1.3.1/pgcc.pgf90
mpich	gcc	gfortran	1.2.7	/home/compiler/mpi/mpich-1.2.7/gcc.gfortran
	gcc	pgf90		/home/compiler/mpi/mpich-1.2.7/gcc.pgf90
	icc	ifort-11.1		/home/compiler/mpi/mpich-1.2.7/icc.ifort-11.1
	icc	ifort-9.1		/home/compiler/mpi/mpich-1.2.7/icc.ifort-9.1
	pgcc	pgf90		/home/compiler/mpi/mpich-1.2.7/pgcc.pgf90

查找编译命令所在的路径可以使用 which 命令，例如”which mpicc”将返回 mpicc 命令所在的具体路径。确认编译器的版本请在编译命令后使用-v 或者-V 参数，例如”gcc -v”、”pgf90 -V”，MPI 编译器的详细命令行调用则可以用”mpicc -show”获得。