
魔方超级计算机应用环境

上海超级计算中心

应用技术部

2010年3月

目 录

1. 基本环境.....	3
1.1. 系统软件环境.....	3
1.1.1. 操作系统	3
1.1.2. 作业调度系统	4
1.1.3. 编译器和并行实现	4
1.1.4. 数学库	4
2. 应用部署.....	5
2.1. 源代码软件.....	5
2.2. 工程计算商业软件.....	5
2.3. 计算队列分布.....	6
2.3.1. 源代码计算队列.....	6
2.3.2. 工程计算商业软件队列.....	7
3. 上机操作.....	8
3.1. 登录和传输文件.....	9
3.1.1. VPN 验证.....	9
3.1.2. 终端登录.....	11
3.1.3. 数据传输.....	12
3.2. 编译.....	13
3.3. 作业提交、管理和软件使用.....	14
3.3.1. 作业提交	14
3.3.2. 作业管理	15
3.3.3. 源代码软件使用	16
3.3.3.1. espresso	16
3.3.3.2. gamess.....	17
3.3.3.3. WIEN2K	17
3.3.4. 工程计算商业软件使用	18

3.3.4.1. Ansys 软件	18
3.3.4.2. Cfx 软件	20
3.3.4.3. Fluent 软件	21
3.3.4.4. Dyna 软件	23
3.3.4.5. Abaqus 软件	26
3.3.4.6. Sctetra 和 stream 软件	26
3.3.4.7. StarCD 软件	27
3.3.4.8. Feko 软件	27
3.3.4.9. Nastran 软件	28
3.3.4.10. Marc 软件	29
3.3.4.11. Dytran 软件	30
3.3.4.12. Pamcrash 软件	31
3.3.4.13. Hyperworks 软件	32
3.4. 用户管理	33
3.4.4. 主机用户密码修改	33
3.4.5. 主机用户权限调整	33
3.4.6. VPN 用户密码修改	33
附录 A: 魔方机器硬件环境	34
附录 B: 简单 LINUX 命令	36
附录 C: 魔方超级计算机上部署的编译库	40
附录 D: 其他类型作业脚本	42
1. 普通串行计算	42
2. 共享内存并行作业	42
3. MPI 并行作业	43
4. OpenMP+MPI 混合同行作业	43
5. 需要大内存的计算	44
6. 串行和并行混合的计算脚本	46
7. 大量相同类型计算	47

魔方超级计算机应用环境

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。本文主要介绍在魔方超级计算机上部署的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。A 区主要部署各类工程计算商业软件，由 82 个机架式服务器构成，每个服务器包含 8 颗 AMD Barcelona 1.9G Hz 四核处理器，128G 共享内存。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\$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 **FTW** 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应用软件， 下面将分别对源代码软件和工程计算商业软件进行介绍。

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、 lammmps、 smeagol、 namd、 DL_POLY、 meep 等， 其中部分软件由用户协助测试完成。

2.2. 工程计算商业软件

工程计算商业软件的测试和部署主要是在机架式服务器的 A 区完成的， 表一列出了目前在魔方超级计算机上测试过的工程计算商业软件

表一、魔方超级计算机上测试过的工程计算商业软件

软件名称	软件版本	使用队列
Fluent	6.3.26	fluent
Abaqus	6.8-1	abacus
	6.9-1	
Dyna	970_6763	dyna
	971_7600.2.1224	
	971_R3.1	
	971_R3.2.1	
	971_R4.2	
	971_R4.2.1	

Ansys	11.0	ansys
Cfx	11.0sp1	cfx
Sc/tetra	V8	cradle
Stream	V8	cradle
Nastran	2008r1	nastran
	md2007r1	
Marc	2008r1	marc
Dytran	2008r1	dytran
Starcd	4.08c	starcd
Feko	5.5	feko
	5.0	
Pamcrash	2008.0	pamcrash
Radioss Optistruct	10.0	hyperworks

2.3. 计算队列分布

2.3.1. 源代码计算队列

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。

2.3.2. 工程计算商业软件队列

工程计算商业软件队列主要分布在 A 区。A 区的节点编号为 fnode01-fnode82，按数字顺序增加。表二和表三分别给出了 A 区的节点分组情况和队列划分情况。

表二、魔方超级计算机 A 区节点区域划分

Hosts	Node Number	Cores	Hosts Group Name
fnode01-fnode14	14	448	cf1HG
fnode15-fnode21	7	224	cf2HG
fnode22-fnode35	14	448	cf3HG
fnode36-fnode45	10	320	caeHG
fnode47	1	32	lockHG
fnode48-fnode54	7	224	fea3HG
fnode55-fnode61	7	224	largediskHG
fnode62-fnode68	7	224	fea2HG
fnode69-fnode81	13	416	fea1HG
注： ◆ 节点组 largediskHG 所有节点的临时文件存放目录/tmp 空间为 200G，其余节点的/tmp 目录空间为 100G。 ◆ Fnode46 为编译节点，用于程序调试和编译。			

表三、魔方超级计算机 A 区队列划分

软件名称	使用队列名称	可调度节点组	优先级
Ansys	ansys	largediskHG+1、fea2HG	40
Abaqus	abaqus	largediskHG+1、fea3HG	40
dyna	dyna	fea1HG+1、fea2HG	40
nastran	nastran	largediskHG+1、fea2HG	40
marc	marc	largediskHG+1、fea2HG	40
dytran	dytran	largediskHG+1、fea2HG	40
cfx	cfx	cf1HG+2、cf2HG+1、cf3HG	40
fluent	fluent	cf2HG+1、cf1HG	40
starcad	starcad	cf3HG	40
Sctetra/stream	cradle	cf3HG	40

feko	feko	fea3HG+1、 fea2HG	40
pamcrash	pamcrash	fea3HG+1 fea2HG	40
Radioss Optistruct	hyperworks	fea3HG+1 fea2HG	40
自编程序	esource	caeHG+1、 cfd3HG、 fea2HG、 fea3HG	40

A 区队列的作业排队策略都采用 FAIRSHARE 的模式。

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**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。无论是 A 区、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，运行小规模测试。

在 A 区的编译节点为 fnode41，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。

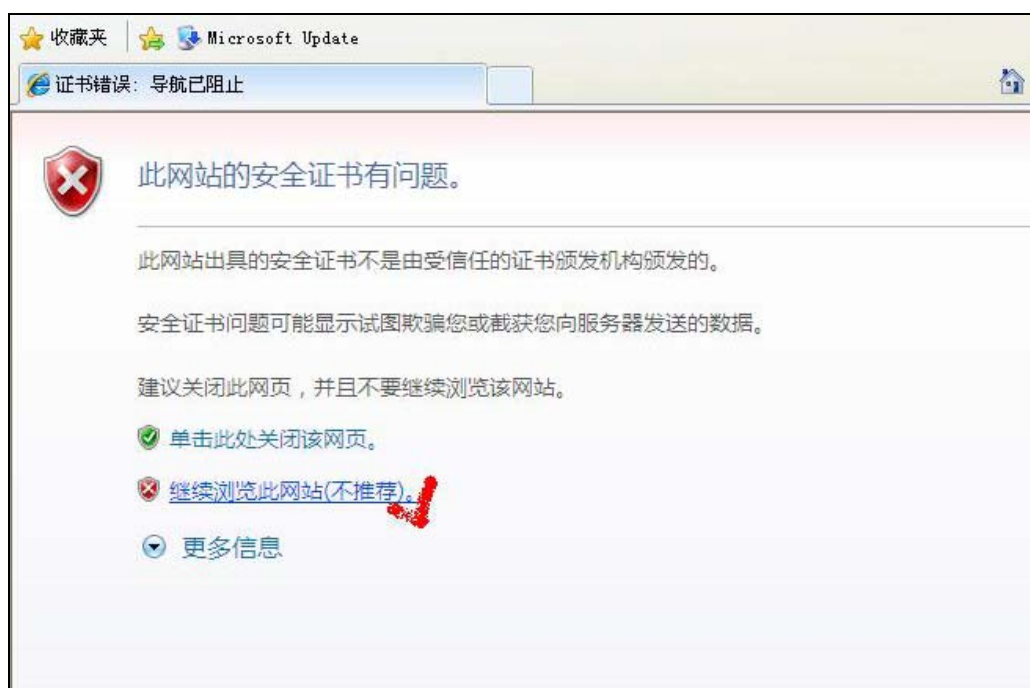
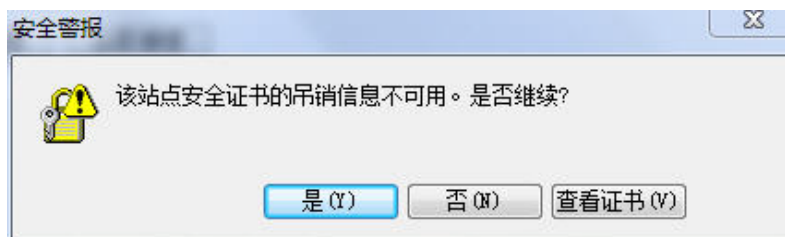
下面分别介绍上机操作的相关内容。

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，此时两次选择“是”，可以进入登录界面。



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使用说明，并根据使用说明配置 VPN 客户端。其中，客户端软件的安装使用说明可参照《SSL VPN 用户使用说明(Win7)》的使用说明。



上海超级计算中心欢迎您

WEB应用 | 主机应用 | 注销 | 帮助 | 修改密码

欢迎访问上海超级计算中心

wshi, 请选择你的应用

链接:

- SSL VPN用户使用说明 (XP)
- SSL VPN用户使用说明 (Linux)
- SSL VPN用户使用说明 (Vista)
- SSL VPN客户端软件 (Windows)
- SSL VPN用户使用说明 (Win7)
- SSL VPN客户端软件 (Linux)

对于 WindowsXP、Windows2000 和 Vista 客户，可以选择“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”的控件后即可顺利登录中心网络。也可以选择下载“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 (Win7)》的使用说明。按照提示安装该软件后，从“Profile”菜单选择 New，按照下图格式输入网址、用户名和密码即可。

Create a Profile

This Profile For All Users Save Password

SecurID Cert-Anonymous

Profile Name: **超算VPN地址**

SPX Host or IP:

SPX Host User Name: **用户名**

SPX Host Password: **口令**

Re-enter Password:

当通过网络或者客户端链接成功后，客户可以使用内部网址访问模仿超级计算机的 A、B、C 三个分区，在没有切断链接前，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。

需要注意的是，对于 Windows7 系统，目前只能支持客户端登录，不能够通过 Https 的方式通过网页登录主机。在 Windows7 下，如果已经在网页上安装了“Array Networks Clients Software”控件，则需要从“添加/删除程序”删除之前安装的 VPN 客户端。此外，部分系统还需要从 IE 浏览器的“工具” | “Internet 选项” | “浏览历史记录”中，选择“设置” | “查看对象”，将“Array Networks Clients Software”的加载项删除，之后重新安装 VPN 客户端即可。

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 A、B、C 区域。

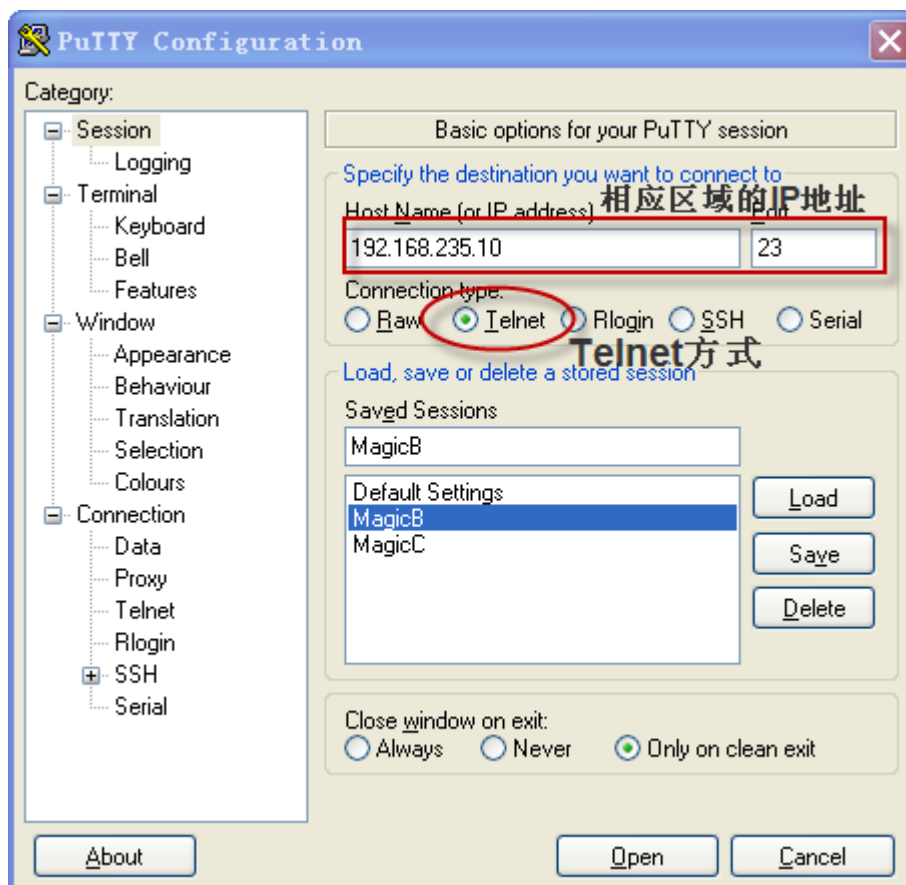
魔方超级计算机的内部 IP 地址为：

主机	服务	内网地址
魔方 A 区	telnet	192.168.235.110
	ftp	192.168.235.120
魔方 B 区	telnet	192.168.235.10
	telnet	192.168.235.20
	ftp	192.168.235.50
	telnet	192.168.235.30
魔方 C 区	telnet	192.168.235.40
	ftp	192.168.235.70

在 windows 系统下，出于使用方便的考虑，我们建议用户使用 telnet 工具进行登录。推荐使用免费 telnet 软件 Putty，下载的官方地址为：

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>，也可以通过华军下载等国内知名网站获得。

下面以 putty 为例演示魔方超级计算机的登录过程。登录 B 区的示例如下：



点击“Open”或回车后会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`

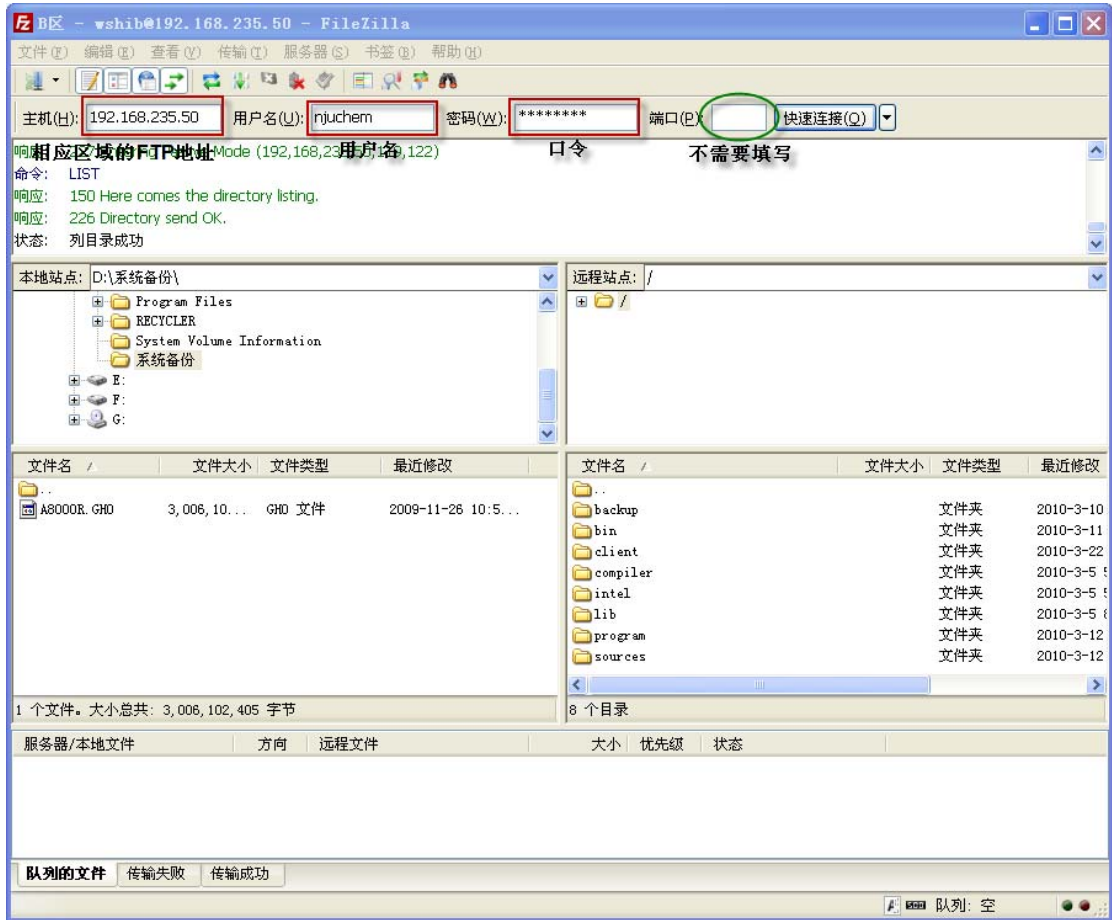
3.1.3. 数据传输

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，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。A 区的编译节点为 `fnode46`；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 `rsh` 或者 `ssh`，例如用户可以使用命令 `rsh a110` 或 `ssh a110` 登录编译节点 `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、管理和软件使用

3.3.1. 作业提交

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。

其他类型作业脚本请参看附录 D。

3.3.2. 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。

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 <code>bjobs</code> 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 <code>bstop</code> 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  RSU
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 <code>bstop</code> 命令挂起
DONE	作业正常结束， <code>exit</code> 代码为 0
EXIT	作业退出， <code>exit</code> 代码不为 0

3.3.3. 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

`espresso` 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`，并使用类似于下列的这个脚本提交。

```

#!/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=${STRUCTEDIT_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
```

3.3.4. 工程计算商业软件使用

此处只说明使用各个工程计算商业软件的作业脚本的具体差异和使用注意点。

3.3.4.1. Ansys 软件

命令流格式文件计算

如果 ansys 计算文件是命令流格式文件（如 input.dat），可参照以下作业脚本（ansys.lsf）：

作业脚本范例	参数说明
APP_NAME=ansys	指定计算任务所要进入的调度队列ansys
NP=32	计算任务所需的CPU总数
NP_PER_NODE=32	计算时每个节点使用的CPU数，为可选参数
RUN= "ansys -b < input.dat "	具体执行的ansys命令

DB 格式文件计算

如果 ansys 计算文件是 db 格式（如 jobname.db），那么除了作业脚本（ansys.lsf）外还需要额外建立一个命令流文件（如 input.dat），在文件中读入 db 文件并设置相应的求解参数（具体请参阅 ansys 帮助手册）。

示例如下：

```
Resume, 'jobname', 'db'    !! 读入数据库文件jobname.db
/Solu
Eqslv,dpcg                !! 指定ANSYS并行计算求解器
Dsproc,32                 !! 指定所需CPU总数
Dsopt,script              !! 指定采用域分解
Solve
Save,all
Finish
```

此外，在设置模型参数时请确保模型中设置的 proc 数目和作业脚本中的 NP 数目一致，以免造成资源浪费和不必要的错误。

作业脚本 RUN 行具体参数

```
RUN="ansys [-ver=xxx] -b [-p xxx] -i xxx -o xxx [-smp|-mpp] [-mpi=xxx] [-mpi_opts=xxx]
[other ansys parameter]"
```

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
[][]	表示参数为互斥的可选参数	
-ver=xxx	指定 ansys 软件版本	11.0
-b	设置 ansys 以后台运行方式运行	
-p xxx	指定计算采用的 ansys 模块名称	-p ane3fl
-i xxx	指定 ansys 计算的输入文件	
-o xxx	指定 ansys 计算的输出文件	

-smp -mpp	设置采用共享内存的计算模式还是分布式计算模式	mpp
-mpi=xxx	设置 ansys 并行计算采用的 mpi 类型,	hpmpi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	-ibv -prot
注释: Ansys 原有参数-mp、-machines、-dis、-mpi 会被忽略, 用户不用指定这些参数。		

3.3.4.2. Cfx 软件

作业脚本文件示例

```
APP_NAME=cfx
```

```
NP=2
```

```
NP_PER_NODE=1
```

```
RUN="cfx -def StaticMixer.def -mpi=hp"
```

计算精度、内存及并行方式选择

在作业脚本文件中 RUN 行可以设置 cfx 计算的精度、内存、初始条件等, 可以分别通过 cfx 软件参数-single 或-double、-s <factor>、-ini <file>等来指定。具体可以参考 cfx 帮助手册。

并行方式的选择可以通过-mpi=xxx 方式来指定, 可以设置为 hpmpi、mpich 或 pvm 三种。

作业中止及保存

在计算过程用户可以用 bpeek 命令监控计算进展情况, 也可以中止作业并保存结果退出。具体操作如下:

```
cd
```

```
echo 'export PATH=/home/software/ansys_inc/v110/CFX/bin:$PATH' >> .bashrc (只有在第一次执行时需要)
```

```
cd 工作目录
```

```
cfx5stop -dir *.dir
```

作业脚本 RUN 行具体参数

```
RUN="cfx [-ver=xxx] -def xxx [other cfx parameters] [-mpi=xxx] [-mpi_opts=xxx]"
```

参数	参数说明	参数默认值
[]	表示参数为可选参数, 用户可以根据需要设置	
[][]	表示参数为互斥的可选参数	
-ver=xxx	指定 cfx 版本	11.0
-def xxx	指定 cfx 的输入文件	

-mpi=xxx	设置 abaqus 并行计算采用的 mpi 类型,包括 hpmpi、mpich、pvm 三种	hpmpi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	-ibv -prot
注释: ① 有 cfx 软件的-par-dist、-start-method 参数会被忽略,用户不用设置这些参数。		

3.3.4.3. Fluent 软件

Fluent 作业提交时采用后台运行和命令流的计算方式,而不是大家常用的图形操作方式。

提交作业前用户需要准备 cas/dat 文件,同时还要准备一个作业脚本文件和一个 journal 文件。在 journal 文件中描述了 fluent 的所有操作命令,采用 fluent 软件的 TUI 方式记录(具体可参考 Fluent User Guide)。

作业脚本文件示例

作业脚本范例	参数说明
APP_NAME=fluent	指定计算任务所要进入的调度队列fluent
NP=64	计算任务所需的CPU总数
NP_PER_NODE=32	计算时每个节点使用的CPU数,为可选参数
RUN= "fluent 3d -g -i journal "	具体执行的fluent命令

稳态计算 journal 文件示例

命令流内容	说明
file/read-case-data /home/user/xlchen/testcase/test.gz /solve/iterate 500 /file/write-case-data /home/user/xlchen/testcase/gasok exit y	导入计算所需的模型文件 (cas文件和dat文件) 设置迭代步数并进行迭代 保存计算结果 (cas文件和dat文件) 退出fluent计算程序

非稳态计算 journal 文件示例

命令流内容	说明
-------	----

file/read-case-data	导入计算所需的模型文件（cas文件和dat文件）
/home/user/xlchen/testcase/test.gz	
/solve/set/time-step	设置时间步长
0.0001	
/solve/dual-time-iterate	设置总的运行时间步数
500	
20	设置每个时间步内迭代次数
/file/write-case-data	保存计算结果（cas文件和dat文件）
/home/user/xlchen/testcase/gasok	
exit	退出fluent计算程序
y	

计算结果自动保存

用户在设置 fluent 的 cas 文件时，需要设置自动保存（autosave），以便保留中间结果。或者用户在提交作业时指定 checkpoint 时间间隔，也可以起到自动保存的功效。具体说明如下：

```
bsub -k "`pwd` 120" fluent.lsf
```

其中，-k参数可选项具体说明如下：

```
-k "checkpoint_dir [init=initial_checkpoint_period] [checkpoint_period] [method=method_name]"
```

checkpoint_dir为checkpoint文件存放目录

init=initial_checkpoint_period 第一次checkpoint时间周期，单位为分钟

checkpoint_period 正常checkpoint时间周期，单位为分钟，建议最少60分钟

作业脚本 RUN 行具体参数

```
RUN="fluent 2d|3d|2ddp|3ddp -g -i journal [-rxxx] [-pxxx] [-mpi=xxx] [-mpi_opts=xxx] [other fluent parameters]"
```

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
	表示参数为互斥的参数	
2d 3d 2ddp 3ddp	指定 fluent 计算模型为二维或三维以及计算精度	
-g	指定 fluent 以后台运行方式进行计算	
-i journal	指定 fluent 的输入脚本	
-rxxx	指定 fluent 的版本	6.3.26
-pxxx	指定 fluent 并行计算采用的通讯方式，包括 default、net、ib.ofed	

	ethernet、ib.ofed、ib.udapl 几种	
-mpi=xxx	指定 fluent 并行计算采用的 mpi 类型,包括 hpmpi、mpich 两种	hpmpi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项,视 mpi 类型不同而有所差别	
注释:		
① Fluent 原有参数-t、-cnf、-lsf、-ssh、-machines 会被忽略,用户不用指定这些参数。		

3.3.4.4. Dyna 软件

Dyna 计算是比较典型的并行计算程序,可以达到很高的并行度和加速效率。

作业脚本文件示例

作业脚本范例	参数说明
APP_NAME=dyna	指定计算任务所要求的调度队列dyna
NP=64	计算任务所需的CPU总数
NP_PER_NODE=32	计算时每个节点使用的CPU数,为可选参数
RUN="dyna i=jobname.key"	具体执行的dyna命令,可以根据要求设置相关lsdyna参数

Dyna 参数说明

p=pfile

pfile是一个文本文件,文件名即为“pfile”,放置在与*.key文件相同的目录下,用来指定结果文件和临时文件写入的路径。例如,在pfile文件中写入如下一行:

```
dir { global result local tmp }
```

表示结果文件写入当前目录的result目录下,临时文件放入当前目录的tmp目录下,如果当前目录下没有这两个目录系统会自动创建。

另外利用pfile还可以实现一些高级功能,用户可自行查阅LSDYNA关键字手册Appendix L: LS-DYNA MPP User Guide: PFILE。

memory=400M memory2=50M

memory选项用来设置内存,这行参数表示主进程指定使用400,000,000word大小的内存(注意单位,1word=4byte),即1.6G内存;而其它进程指定使用50,000,000word大小的内存,即200M内存。随着cpu数量的增加,memory2的大小可以近似线性的减小。例如用4个cpu计算一个题目,需要设置memory=80M和memory2=20M,也就是说该题目需要内存数量为80M+3×20M=140M(word);现在若改用16个cpu来计算,那么memory=80M仍然保持不变,而memory2可以设置为6M,即总内存数量为80M+15×6M=170M(word)。

版本、精度选择和重启动

在作业脚本RUN行中可以通过-ver=xxx参数来选择用户需要的软件版本,-s|-d参数来选择单精度/双精度,具体可参考“[RUN行具体参数](#)”。

Dyna计算重启动分析的方法与单机版dyna相同。对于简单重启动,只要在脚本文件命令行

中把*i=jobname.key*改为*r=d3dumpnn*，如：

```
APP_NAME=dyna
```

```
NP=64
```

```
RUN="dyna r=d3dump01 p=pfile memory=50m memory2=20m"
```

如果是完全重启动，还要有一个数据输入文件，如：

```
APP_NAME=dyna
```

```
NP=32
```

```
RUN="dyna i=fullrestart.k r=d3dump01 p=pfile memory=50m memory2=20m"
```

Dyna Switch 功能实现

在数据文件目录下新建一个D3KIL文件，并在文件中输入sw1、sw2等开关选项即可进行监控或写重启动文件等操作。例如命令echo sw2 > D3KIL操作后，dyna程序会自动检测这个d3kil文件，并重新估计计算时间，用bpeek命令可以查看到详细信息。之后dyna程序会自动删除这个d3kil文件。注意如果d3kil文件内容是空的，则默认其内容为sw1。

计算结果处理

计算结束后，rcforc、nodout、secforc等ASCII文件均存储在dbout.*中，LS-Dyna提供了一个dumpbdb命令可以把这些结果文件提取出来，但是这个命令不能在接入节点上执行，需要登陆到计算节点上去。具体操作如下：

```
cd
```

```
echo 'export PATH=/home/software/dyna/post/:$PATH' >> .bashrc （只有在第一次执行时需要）
```

```
bhosts | grep ok
```

这时屏幕上会显示所有可用的节点，登陆到其中一个空闲节点上，用cat命令把dbout.*合成一个文件，再用dumpbdb命令提取即可。

```
ssh fnode14
```

```
cat dbout.* > dbout
```

```
dumpbdb dbout
```

操作过程见下图：

```

192.168.235.10 - PuTTY
xlchen@inode11:~> cd
xlchen@inode11:~> echo 'export PATH=/home/software/dyna/post/:$PATH' >> .bashrc
xlchen@inode11:~> bhosts | grep ok | head
fnode01      ok      -      32      8      8      0      0      0
fnode02      ok      -      32     16     16      0      0      0
fnode03      ok      -      32      8      8      0      0      0
fnode05      ok      -      32     24     24      0      0      0
fnode06      ok      -      32     16     16      0      0      0
fnode08      ok      -      32     24     24      0      0      0
fnode09      ok      -      32     24     24      0      0      0
fnode10      ok      -      32      0      0      0      0      0
fnode11      ok      -      32      0      0      0      0      0
fnode12      ok      -      32     16     16      0      0      0
xlchen@inode11:~> ssh fnode11
Last login: Tue Aug  4 15:09:19 2009 from inode11
xlchen@fnode11:~> cd softtest/dynatest/
xlchen@fnode11:~/softtest/dynatest> cat dbout000* > dbout
xlchen@fnode11:~/softtest/dynatest> dumptdb dbout

```

用户也可以将时间历程文件设置输出为二进制格式，这样在计算任务完成后，程序会生成 binoutnnnn 文件，其中 binout0000 存放在 global 目录中，而其它 binoutnnnn 存放在 local 目录中（目录在 pfile 文件中指定，dir { global result local tmp }）。有关输出 binout 文件的具体信息可在 output.* 文件中查看。

作业脚本 RUN 行具体参数

RUN="dyna [-ver=xxx] [-s|-d] [-smp|-mpp] [-mpi=xxx] [-mpi_opts=xxx] i=x.k [other dyna parameters]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
[][]	表示参数为互斥的可选参数	
-ver=xxx	用于指定 dyna 的版本，魔方上现安装有 970_6763、971_7600.2.1224、971_R3.1、971_R3.2.1、971_R4.2 五个版本。	971_R3.2.1
-s -d	用于指定求解器精度为单精度或者双精度	-s
-smp -mpp	用于指定求解器并行方式为 smp 或者 mpp	-mpp
-mpi=xxx	用于指定求解器并行时采用的 mpi 类型，可选择的 mpi 类型包括 hp、scali、openmpi 三种	-mpi=hpmpi
-mpi_opts=xxx	用于指定求解器并行时的 mpi 参数	-mpi_opts=-ibv -prot
注： ① smp 并行计算时 ncpu 参数不用设置，lsf 自动会将申请的资源数分配给 dyna，最大不超过 32。 ② dyna 程序的所有其他参数用户可以根据需要指定，用户自行对参数合法性进行检查。 ③ dyna 软件 970 版本只有采用 hpmpi 的 970_6763 并行版本		

3.3.4.5. Abaqus 软件

RUN="abaqus [-ver=xxx] [job=xxx] [analysis|datacheck|...] input=xxx [other abaqus parameters] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
[][]	表示参数为互斥的可选参数	
-ver=xxx	指定 abaqus 版本，包括 6.8 和 6.9 版本	6.8
job=xxx	Abaqus 作业名称	
Analysis datacheck ...	设置进行分析计算或输入文件检查或其他操作	
input=xxx	指定 abaqus 输入文件	
-mpi=xxx	设置 abaqus 并行计算采用的 mpi 类型，	hpmpt
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	-ibv -prot
注释： ①原有 abaqus 的 cpus=xxx 参数被忽略，用户不必设置。		

3.3.4.6. Sctetra 和 stream 软件

作业脚本文件示例

作业脚本范例	参数说明
APP_NAME=cradle NP=64 NP_PER_NODE=32 RUN="cradle tutorial.s"	指定计算任务所要求的调度队列cradle 计算任务所需的CPU总数 计算时每个节点使用的CPU数，为可选参数 具体执行的sctetra或stream命令，可以根据要求设置相关参数

RUN 行参数说明

RUN="sctetra|stream [-ver=xxx] [-<start operation>[,<end operation>]] [-alone]

[-single|-double] case.s [other cradle software parameters] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
[][]	表示参数为互斥的可选参数	
-ver=xxx	指定 sctetra 或 stream 软件版本	8
-<start operation> [,<end operation>]	执行的具体操作模块，如求解、前处理等	全部
Single double	设置计算的精度	Single

Case.s	指定 sctetra 或 stream 输入文件	
-mpi=xxx	设置 sctetra 并行计算采用的 mpi 类型,	hpmpi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	-ibv -prot
注释: ①原有 sctetra 软件的-std、-hpc、<degree of parallelism>参数会被忽略, 用户不用设置这些参数。 ②方括号[]内的参数用户可以不设置, 系统会有默认设置。 ③采用 stream 软件求解时, RUN 行的第一个参数必须为 stream.		

3.3.4.7. StarCD 软件

RUN="star [-ver=xxx] -case=xxx [other star parameters] [-mvmesh] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
[]	表示参数为可选参数, 用户可以根据需要设置	
[][]	表示参数为互斥的可选参数	
-ver=xxx	指定 star 版本, 安装有 3.26, 4.08 及 4.08c 版本	4.0.8
-case=xxx	指定 star 的输入文件	
-mpi=xxx	设置 star 并行计算采用的 mpi 类型,包括 hpmpi、scampi 几种	hpmpi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	-ibv -prot
注释: ① 有 starcd 软件的-nodfile=xxx、-batch、-mpphosts 参数会被忽略, 用户不用设置这些参数。		

3.3.4.8. Feko 软件

Feko 软件具有多个模块, 包括有 runfeko、adaptfeko、optfeko、timefeko 等模块命令, 均可以通过使用 feko 队列来实现。

作业脚本文件示例

作业脚本范例	参数说明
APP_NAME=feko	指定计算任务所要求的调度队列feko
NP=2	计算任务所需的Core总数
NP_PER_NODE=32	计算时每个节点使用的Core数, 为可选参数
RUN="feko test.fek"	具体执行的feko命令, 可以根据要求设置相关参数

RUN 行参数说明

RUN="feko|adaptfeko|optfeko|timefeko [-ver=xxx] casename [other feko software parameters] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
	表示参数为互斥的必选参数，可以设置为 feko、runfeko、adaptfeko、optfeko、timefeko 等	feko
-ver=xxx	指定 feko 软件的版本，可设置为 5.5 或 5.0	5.5
casename	指定 feko 软件的输入文件	
-mpi=xxx	设置 feko 并行计算采用的 mpi 类型，	5.5 版本为 intelmpi，5.0 版本为 mpich
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	
注释： ①原有 feko 软件的-np、--machines-file、--adaptfeko-options、--runfeko-options 等参数会被忽略，用户不用设置这些参数。 ②方括号[]内的参数用户可以不设置，系统会有默认设置。		

3.3.4.9. Nastran 软件

作业脚本文件示例

作业脚本范例	参数说明
<pre>APP_NAME=nastran NP=1 NP_PER_NODE=32 RUN="nastran test.bdf"</pre>	指定计算任务所要求的调度队列nastran 计算任务所需的Core总数 计算时每个节点使用的Core数，为可选参数 具体执行的nastran命令，可以根据要求设置相关参数

RUN 行参数说明

RUN="nastran [-ver=xxx] [-smp|-mpp] jid=casename [other nastran software parameters] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
[-smp -mpp]	表示互斥的可选参数，设置并行模式为 smp 或 mpp	smp

-ver=xxx	指定 nastran 软件的版本，可设置为 2008 或 2007	2008
casename	指定 nastran 软件的输入文件名	
-mpi=xxx	设置 nastran 并行计算采用的 mpi 类型,	hpmmpi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	
<p>注释:</p> <p>① 原有 nastran 软件的 hosts、auth、parallel、dmparallel 等参数会被忽略，用户不用设置这些参数。</p> <p>② 方括号[]内的参数用户可以不设置，系统会有默认设置。</p> <p>③ 无论并行模式为 smp 或 mpp ， Nastran 软件目前最多只能使用一个节点（32cores）。</p> <p>④ Nastran 软件安装时设置 sdirectory 默认值为/tmp， memory 默认值为 ESTIMATE</p> <p>⑤ 用户可以根据需要设置相应的 nastran 计算参数（如临时文件存放位置、内存大小等）以优化计算速度，具体可以参考 nastran 用户手册。</p>		

3.3.4.10. Marc 软件

作业脚本文件示例（单文件计算）

作业脚本范例	参数说明
<pre>APP_NAME=marc NP=2 NP_PER_NODE=32 RUN="marc -jid casename"</pre>	<p>指定计算任务所要求的调度队列marc</p> <p>计算任务所需的Core总数</p> <p>计算时每个节点使用的Core数，为可选参数</p> <p>具体执行的marc命令,可以根据要求设置相关参数</p>

多文件计算示例

作业脚本范例	参数说明
<pre>APP_NAME=marc NP=2 NP_PER_NODE=32 RUN="marc -np \$NP -jid casename"</pre>	<p>指定计算任务所要求的调度队列marc</p> <p>计算任务所需的Core总数</p> <p>计算时每个节点使用的Core数，为可选参数</p> <p>具体执行的marc命令,可以根据要求设置相关参数</p>

RUN 行参数说明

RUN="marc [-ver=xxx] [-smp|-mpp] -jid casename [-np|-nps \$NP] [other marc software parameters] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
----	------	-------

[]	表示参数为可选参数，用户可以根据需要设置	
[-smp -mpp]	表示参数为互斥的可选参数，可以根据需要选择。	smp
[-ver=xxx]	指定 marc 软件的版本	2008r1
[-np -nps \$NP]	指定 marc 输入文件为多文件形式（-np \$NP）还是单文件（-nps \$NP）	-nps \$NP
-jid casename	指定 marc 软件的输入文件	
-mpi=xxx	设置 marc 并行计算采用的 mpi 类型，	Hpmapi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	
<p>注释：</p> <p>① 原有 marc 软件的-np、-nproc、-nps、-nthread、-hostfile 等参数会被忽略，用户不用设置这些参数。并行规模可以通过前述的 NP 参数来指定。</p> <p>② 用户可以根据输入文件形式选择多文件或单文件计算形式，通过-np \$NP 或-nps \$NP 参数来控制。</p> <p>③ 方括号[]内的参数用户可以不设置，系统会有默认设置。</p> <p>④ 用户可以根据需要设置相应的 marc 计算参数以优化计算速度，具体可以参考 marc 用户手册。</p>		

3.3.4.11. Dytran 软件

作业脚本文件示例

作业脚本范例	参数说明
<pre>APP_NAME=dytran NP=2 NP_PER_NODE=32 RUN="dytran jid=casename"</pre>	<p>指定计算任务所要求的调度队列dytran</p> <p>计算任务所需的Core总数</p> <p>计算时每个节点使用的Core数，为可选参数</p> <p>具体执行的dytran命令，可以根据要求设置相关参数</p>

RUN 行参数说明

RUN="dytran [-ver=xxx] jid=casename [other dytran software parameters] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
[-ver=xxx]	指定 dytran 软件的版本	2008r1
jid=casename	指定 dytran 软件的输入文件	
-mpi=xxx	设置 dytran 并行计算采用的 mpi 类型，	hpmapi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	

注释:

- ① 原有 dytran 软件的 ncpus=、bat=no 等参数会被忽略，用户不用设置这些参数。并行规模可以通过前述的 NP 参数来指定。
- ② 方括号[]内的参数用户可以不设置，系统会有默认设置。
- ③ Dytran 软件目前只支持 smp 并行模式，跨节点并行尚不支持。
- ④ 用户可以根据需要设置相应的 dytran 计算参数以优化计算速度，具体可以参考 dytran 用户手册。

3.3.4.12. Pamcrash 软件

作业脚本文件示例

作业脚本范例	参数说明
<pre>APP_NAME=pamcrash NP=32 NP_PER_NODE=32 RUN="pamcrash casename.pc"</pre>	<p>指定计算任务所要求的调度队列pamcrash</p> <p>计算任务所需的Core总数</p> <p>计算时每个节点使用的Core数，为可选参数</p> <p>具体执行的pamcrash命令，可以根据要求设置相关参数</p>

RUN 行参数说明

RUN="pamcrash [-ver=xxx] [-smp|-mpp] casename [other pamcrash software parameters] [-mpi=xxx] [-mpi_opts=xxx] [-wd=/home/aa]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
[-ver=xxx]	指定 pamcrash 软件的版本	2008.0
-smp -mpp	指定采用 pamcrash 软件并行计算方式为 SMP 或 MPP	mpp
casename	指定 pamcrash 软件的输入文件为 casename	
-mpi=xxx	设置 pamcrash 并行计算采用的 mpi 类型，	hpmapi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项	-ibv -prot
-wd=/home/aa	指定计算工作目录	当前目录

注释:

- ① 原有 pamcrash 软件的 -np、-arch、-cf 等参数会被忽略，用户不用设置这些参数。并行规模可以通过前述的 NP 参数来指定，并行方式可以通过-smp|-mpp 来指定。
- ② 方括号[]内的参数用户可以不设置，系统会有默认设置。
- ③ 用户可以根据需要设置相应的 pamcrash 计算参数以优化计算速度，具体可以参考 pamcrash 用户手册。

3.3.4.13. Hyperworks 软件

作业脚本文件示例

作业脚本范例	参数说明
<pre>APP_NAME=hyperworks NP=32 NP_PER_NODE=32 RUN="radioss casename"</pre>	<p>指定计算任务所要求的调度队列hyperworks</p> <p>计算任务所需的Core总数</p> <p>计算时每个节点使用的Core数，为可选参数</p> <p>具体执行的radioss或optistruct命令，可以根据要求设置相关参数</p>

RUN 行参数说明

RUN="radioss|optistruct casename [-ver=xxx] [-smp|-mpp] [other hyperworks solver parameters] [-mpi=xxx] [-mpi_opts=xxx]"

参数	参数说明	参数默认值
[]	表示参数为可选参数，用户可以根据需要设置	
Radios optistruct	指定求解器为 radioss 或 optistruct	
casename	指定 hyperworks 软件的输入文件为 casename	
[-ver=xxx]	指定 hyperworks 软件的版本 xxx	10
[-smp -mpp]	指定采用 hyperworks 软件并行计算方式为 SMP 或 MPP	-mpp
-mpi=xxx	设置 hyperworks 并行计算采用的 mpi 类型,xxx	hpmapi
-mpi_opts=xxx	设置并行计算时 mpi 的参数选项 xxx	-ibv -prot

注释:

- ① 使用时要求 RUN 行的第一个参数必须为 radioss 或 optistruct, 第二个参数必须为输入文件。
- ② 计算默认采用 -smp 并行模式, 如果需要采用 smp 并行模式必须增加参数 -mpp。
- ③ 计算默认采用双精度求解, 如果需要使用单精度求解可以使用 -sp 选项。
- ④ 采用 radioss solver 进行求解时可以使用 -starter、-engine、-both 选项, 默认为 -both 选项;
- ⑤ 原有 hyperworks 软件的 -nproc、-ncpu、-cpu、-proc 等参数会被忽略, 用户不用设置这些参数。并行规模可以通过前述的 NP 参数来指定。并行相关的参数 -mpi、-mpipath 也会被忽略, 用户可以通过 -mpi=xxx、-mpi_opts=xxx 参数来指定。
- ⑥ 方括号 [] 内的参数用户可以不设置, 系统会有默认设置。
- ⑦ 用户可以根据需要设置相应的 hyperworks 计算参数以优化计算速度, 具体可以参考 hyperworks solver 使用手册。

3.4. 用户管理

3.4.4. 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。



```
192.168.235.10 - PuTTY
login as: xlchen
Using keyboard-interactive authentication.
Password:
Last login: Mon Nov 17 13:03:56 2008 from 192.168.45.14

Welcome to MagicCube Engineering Computing Center of SSC
Have a nice day!

xlchen@inode11:~> yppasswd
yppasswd is deprecated, use /usr/bin/passwd instead

Changing password for xlchen.
Old Password:
New Password:
Reenter New Password:
Changing NIS password for xlchen on inode09.
Password changed.
xlchen@inode11:~>
```

修改后重新登陆时便要求输入新的主机密码。

3.4.5. 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。

3.4.6. VPN 用户密码修改

用户可访问 <https://vpn.ssc.net.cn> 自行修改。“修改密码”在页面的右上角。



附录 A：魔方机器硬件环境

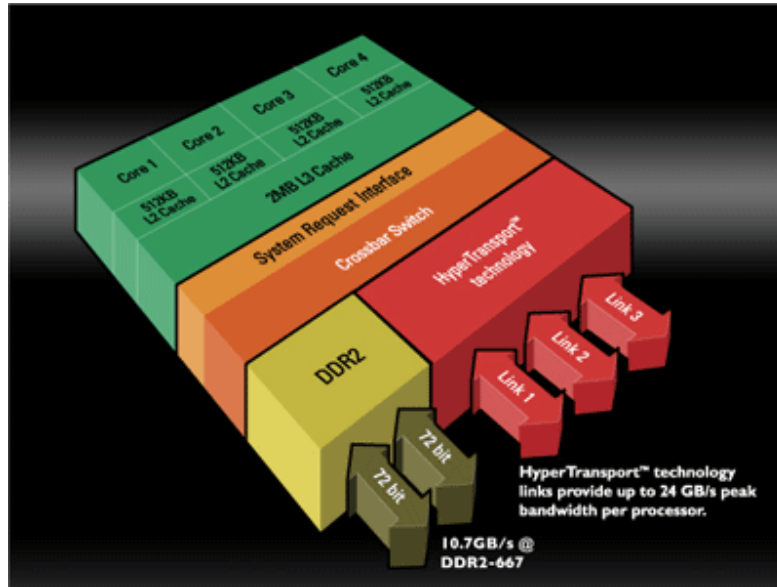
魔方超级计算机系统被分成三个部分即 A、B、C 三个区域，A 区由 82 个机架式服务器构成，每个服务器含 8 个 AMD Barcelona 1.9GHz 低功耗型 Opteron 8347HE 四核处理器，128GB 共享内存，InfiniBand 光纤网络互联，主要部署各类工程计算商业软件；B 区由 650 个刀片式服务器组成，每个服务器含 4 个 AMD Barcelona 1.9GHz 低功耗型 Opteron 8347HE 四核处理器，64GB 共享内存，InfiniBand 光纤网络互联，主要用于各种源代码类计算；C 区由 800 个刀片式服务器组成，每个服务器含 4 个 AMD 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。

魔方超级计算机硬件环境

以 B 区为例，一个计算节点的典型硬件配置如下：

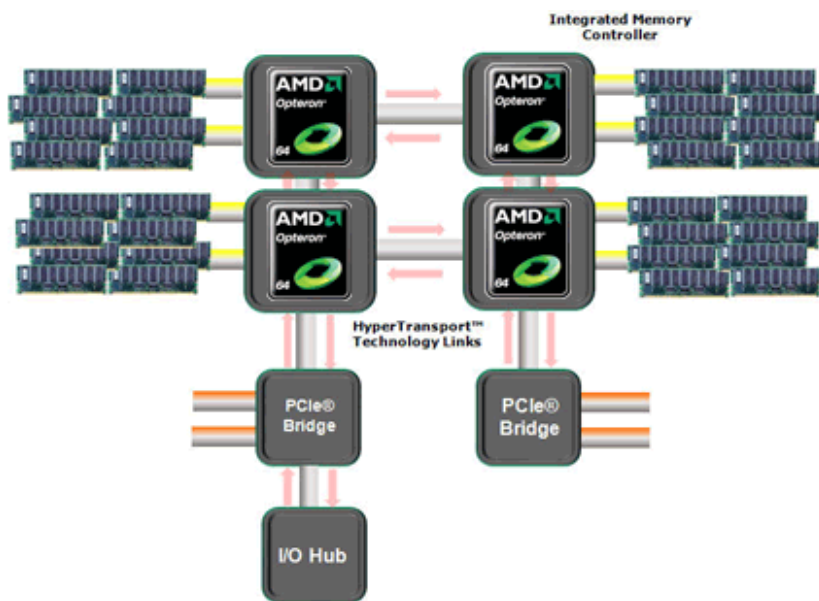
- 四路四核 AMD Barcelona 1.9GHz 低功耗型 Opteron 8347HE 处理器(每节点合计 16 核)
- 每处理器内建内存控制器用于访问本身的 16GB 内存，每节点合计 64GB 内存（四个内存控制器，平均每个处理器核心 4GB 内存）
- Voltaire InfiniBand 4x DDR (Double Data Rate)光纤网络，理论带宽 20Gbits 每秒

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
```

删除目录名称为 AAA 的非空目录或空目录

```
rm -f AAA
```

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

```
# Sample .bashrc for SuSE Linux
# Copyright (c) SuSE GmbH Nuernberg
# There are 3 different types of shells in bash: the login shell, normal shell
# and interactive shell. Login shells read ~/.profile and interactive shells
# read ~/.bashrc; in our setup, /etc/profile sources ~/.bashrc - thus all
# settings made here will also take effect in a login shell.
#
# NOTE: It is recommended to make language settings in ~/.profile rather than
# here, since multilingual X sessions would not work properly if LANG is over-
# ridden in every subshell.
#
# Some applications read the EDITOR variable to determine your favourite text
# editor. So uncomment the line below and enter the editor of your choice :-)
#export EDITOR=/usr/bin/vim
#export EDITOR=/usr/bin/mcedit
#
# For some news readers it makes sense to specify the NEWSSERVER variable here
#export NEWSSERVER=your.news.server
#
# If you want to use a Palm device with Linux, uncomment the two lines below.
# For some (older) Palm devices you may need to set a lower baud rate
# e.g. 57600 or 38400; lowest is 9600 (very slow!)
"test" 118L, 4077C 1,1 Top
```

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。

```
Usage: /bin/ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuSUX nor --sort.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                do not ignore entries starting with .
-A, --almost-all       do not list implied . and ..
    --author              with -l, print the author of each file
-b, --escape             print octal escapes for nongraphic characters
    --block-size=SIZE    use SIZE-byte blocks
-B, --ignore-backups    do not list implied entries ending with ~
```

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/mapi/mvapich/1.0/gcc.gfortran
	gcc	.pgf90		/home/compiler/mapi/mvapich/1.0/gcc.pgf90
	icc	ifort-11.1		/home/compiler/mapi/mvapich/1.0/icc.ifort-11.1
	icc	ifort-9.1		/home/compiler/mapi/mvapich/1.0/icc.ifort-9.1
	pgcc	pgf90		/home/compiler/mapi/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。

附录 D: 其他类型作业脚本

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。

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。

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. 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。

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**，所以注意 $m \times k = 16$ 。

#!/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	<i>m</i> 和 <i>k</i> 见上
K=k	
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	注意
for i in `cat hosts.uniq`; do	每个节点 MPI 进程数
for j in `seq 1 \$M`; do	
echo \$i >>hosts.mpi	
done	
done	
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 and 参数

5. 需要大内存的计算

在魔方系统上，每个 core 平均有 4G 内存，对于普通计算绰绰有余，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	产生 hostfile 文件
for i in `echo \$LSB_HOSTS`; do	
echo \$i >>hosts.list	
done	
cat hosts.list sort uniq > hosts.uniq	注意
rm -f hosts.mpi	每个节点 MPI 进程数

<pre>for i in `cat hosts.unique`; do for j in `seq 1 \$M`; do echo \$i >>hosts.mpi done done</pre>	
<pre>N=`cat hosts.mpi wc -l awk '{ print \$1 }`</pre>	计算 MPI 进程总数
<pre>mpirun_rsh -np \$N -hostfile hosts.mpi \ /path/to/executable -cmd_line_options</pre>	执行 mpirun 必要的参数 可执行程序 and 参数

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。

<pre>#!/bin/sh</pre>	
<pre>MY_MPI_TYPE=mvapich MY_MPI_HOME=/home/compiler/mpi/mvapich/1.0/gcc.pg90</pre>	设定 MPI 参数
<pre>APP_NAME="snode"</pre>	计算任务进入 snode 队列
<pre>NP=16</pre>	任务需 16 个 core
<pre>NP_PER_NODE=16</pre>	每节点有 16 个 core 可用
<pre>RUN="RAW"</pre>	
<pre>...</pre>	
<pre>/path/to/prepare -cmd_line_options</pre>	串行处理或者计算
<pre>...</pre>	
<pre>/home/lsf/7.0/linux2.6-glibc2.3-x86_64/bin/mpirun.lsf</pre>	并行计算，注意 mpirun 后
<pre>/path/to/mpi_calc -cmd_line_options</pre>	无需 np 和 machinefile 参数
<pre>...</pre>	
<pre>/path/to/postprocessing</pre>	后处理
<pre>...</pre>	可以继续其它串行或者并

...

行计算

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。