中国科学技术大学超级计算中心

刀片及胖节点超级计算系统使用指南

李会民

2016年5月27日

目录

1 前言

本用户使用指南主要将对在中国科学技术大学超级计算中心刀片及胖节点超级计算系统上进行编译以及运行作业做一基本介绍，详细信息请参看相应的指南。

为了便于查看，主要排版约定如下：

- 命令：*command parameters*
- 文件：*/path/file*
- 脚本文件：

```
export OPENMPI=/opt/openmpi-1.4.3_intel-11.1.073
export PATH=$OPENMPI/bin:$PATH
export MANPATH=$MANPATH:$OPENMPI/share/man
```

- 命令输出：

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
serial	50	Open:Active	-	120	-	-	0	0	0	0
fat48	43	Open:Active	-	264	-	-	0	0	0	0
fat64	43	Open:Active	-	128	-	-	0	0	0	0
long	40	Open:Active	-	828	-	-	144	0	144	0
normal	30	Open:Active	-	828	-	-	873	216	657	0

由于受水平和时间所限，错误和不妥之处在所难免，欢迎指出错误和改进意见，本人将尽力完善。请从中国科大超算中心主页（http://scc.ustc.edu.cn）上下载更新后的手册。

2 刀片及胖节点超算系统概述

中国科大超级计算中心刀片及胖节点超级计算系统，主要由曙光CB60-G刀片、CB60-G2刀片、联想B710刀片、曙光A950服务器、曙光I620r-G服务器、曙光I420r-G服务器、浪潮TS850服务器、联想万全R630G7服务器、HP DL580 G7服务器构成，共计105个计算节点1528个CPU核心，总双精度峰值计算能力为每秒15.46万亿次。具体参数为：

- 管理和用户登录节点：
  - 节点名为sugon。
  - 用户主登录节点，可以进行编译与通过作业调度系统提交作业。禁止直接在此节点上运行作业。
  - 曙光I620机架式服务器一台，2颗主频2.40GHz的Intel Xeon E5620 x86_64 4核CPU（共8核），8GB DDR3 1333MHz内存。
- 计算节点：
  - 刀片或机架式计算节点：用户都可使用。

节点名	型号	数量	CPU	内存(GB)
node1-node40	曙光CB60-G	40	Intel Xeon X5650，共12核，主频2.66GHz	24
node41-node60	曙光CB60-G2	20	Intel Xeon X5650，共12核，主频2.66GHz	24
node61-node70	联想B710	10	Intel Xeon X5650，共12核，主频2.66GHz	24
node71-node80	曙光CB60-G2	10	Intel Xeon X5650，共12核，主频2.66GHz	48
node93-node103	曙光CB60-G或I620r-G	11	Intel Xeon E5620，共8核，主频2.4GHz	16
node104-node105	曙光I420r-G	2	Intel Xeon E5620，共4核，主频2.4GHz	8

  - 胖计算节点：更多CPU核大共享内存节点，数量有限，需要特殊申请使用。

节点名	型号	数量	CPU	内存(GB)
node81-node83	浪潮TS850	3	Intel Xeon E7540，共48核，主频2.0GHz	128
node84-node85	浪潮TS850	2	Intel Xeon E7-8837，共64核，主频2.66GHz	256
node86-node87	曙光A950	2	AMD Opteron 8431，共48核，主频2.40GHz	128
node88	曙光A620r-G	1	AMD Opteron 6168，共24核，主频1.90GHz	32
node89-node90	联想R630G7	2	Intel Xeon E7430，共16核，主频2.13GHz	64
node91-node92	HP DL580G7	2	Intel Xeon E7-4807，共24核，主频1.87GHz	96

- 存储：13TB曙光DS600-F20 SAS磁盘阵列，用户主目录*/home*所用存储。
- 磁盘配额：用户默认50GB，如需更大空间，请提出申请。
- 计算网络：千兆以太网。
- 操作系统：x86_64架构的64位CentOS 6.5 Linux操作系统。
- 编译器：Intel、PGI、GNU等C/C++ Fortran编译器[1]。
- 数值函数库：Intel MKL。
- 并行环境：Open MPI，支持MPI并行程序；各节点内的CPU为共享内存，节点内既支持分布式内存的MPI并行方式，也支持共享内存的OpenMP并行方式；同时支持在节点内部共享内存，节点间分布式内存的混合并行模式。
- 资源管理和作业调度：Platform LSF。
- 常用公用软件安装目录：*/opt*，请自己查看有什么软件，有些软件需要在自己*~/.bashrc*等配置文件中设置后才可以使用。

图 1: 刀片及胖节点超算系统

[1]Intel、PGI编译器为试用版。

3 用户登录与文件传输

本系统的操作系统为x86_64的CentOS 6.2，不支持TELNET方式登录，用户需以SSH方式（在MS Windows下可利用putty、SSH Secure Shell Client等支持SSH协议的客户端软件[2]）登录到用户登录节点（sugon，IP为211.86.151.102）后进行编译、提交作业等操作，用户数据可以利用FTP和SFTP协议进行数据传输。为了安全，短时间内5次密码错误登录时的IP将被自动封锁10分钟，可以换个IP登录，或联系我们解封。

用户可以在登录节点上运行*passwd*命令修改密码（利用*passwd*命令在登录节点之外的节点上修改密码无效）。请不要设置简单密码和向无关人员泄漏密码，以免造成您的损失。

用户登录进来的默认语言环境为中文zh_CN.UTF-8，以方便查看登录后的中文提示。如果希望使用英文或GBK中文，可以在*~/.bashrc*中添加export LC_ALL=C或export LC_ALL=zh_CN.GBK。

针对zh_CN.UTF-8的putty客户端配置修改如下[3]：

- 打开putty主程序，选择SESSION登入到服务器
- 点击左上角change setting....，打开设置面板
- 选择window －＞ Appearance －＞ Font settings －＞ Change...，选择Fixedsys字体，字符集选择CHINESE_GB2312
- 在window －＞ Appearance －＞ Translation中，Received data assumed to be in which character set中，把Use font encoding改为UTF-8
- 切换到session选项，选中常用的那个，点击SAVE，把这些设置保存在session里面（否则下次打开又不支持中文）。

账户开设时，默认50GB磁盘存储空间，请及时清除不需要的文件，以便释放空间。运行*quota*命令可以查看当前自己的磁盘空间使用情况，*du␣-hs 目录*可以查看目录占用的空间。如需要更大存储空间，请与管理人员联系。超算中心无法提供数据备份服务，数据一但丢失将无法恢复，请及时下载保存自己的数据。

[2]putty下载：

- http://scc.ustc.edu.cn/yhsq/dlrjxz/200910/t20091014_13029.html
- http://www.chiark.greenend.org.uk/sgtatham/putty/download.html

[3]putty客户端中文设置请参考：

- http://scc.ustc.edu.cn/yhsq/dlrjxz/200910/t20091014_13029.html

超算系统可从校内IP登录，校外一般无法直接访问。如果需要从校外等登录，可以使用学校的VPN（教师网络通带有此功能，学生的不带），也可联系我们放开此校外IP（如不清楚自己在校外当前使用的IP，可利用http://scc.ustc.edu.cn/myip.php获取当前使用的IP），或者申请超算中心VPN（http://scc.ustc.edu.cn/vpn/），此VPN设置了只能访问超算服务器，无法访问校内外其它资源。

CentOS(Community ENTerprise Operating System)是Linux主流发行版之一，它来自于Red Hat Enterprise Linux依照开放源代码规定释出的源代码所编译而成。由于出自同样的源代码，因此有些要求高度稳定性的服务器以CentOS替代商业版的Red Hat Enterprise Linux使用。两者的不同在于CentOS并不包含封闭源代码软件。一般来说可以用*man␣命令*或者命令加-h或–help等选项来查看该命令的详细用法，详细信息可参考CentOS、Red Hat Enterprise Linux AS手册或通用Linux手册。

4 串行及OpenMP程序编译

在本系统上可运行C/C++、Fortran的串行程序，以及与OpenMP和MPI结合的并行程序。用户只需在登录节点（sugon）上以相应的编译命令和选项进行编译即可（用户不应到其余节点上进行编译，以免影响系统效率，其它节点一般只设置了运行作业所需要的库路径等，未必设置了编译环境）。当前安装的编译环境主要为：

- C/C++、Fortran编译器：Intel、PGI和GNU编译器，支持OpenMP并行。
- MPI并行环境：Open MPI并行环境

用户可以单独设置自己所需的串行编译环境（如在*~/.bashrc*中设置），也可直接运行
mpi-selector-menu[4]命令按照提示选择自己使用的MPI和串行编译环境，注意数字后需要加u，设置完成后最好重新登录以便设置生效：

```
Current system default : openmpi-1.4.5_intel -12.1.3.293
Current user default : openmpi-1.4.5

    ”u” and ”s” modifiers can be added to numeric and ”U”
    commands to specify ”user” or ”system-wide”.

1. openmpi-1.4.3_intel -11.1.073
2. openmpi-1.4.4_intel -12.1.3.293
3. openmpi-1.4.5_intel -12.1.3.293
4. openmpi-1.4.5_pgi-10.6
U. Unset default
Q. Quit

Selection (1-4[us], U[us], Q):
```

本节主要介绍串行程序和OpenMP并行程序的编译，MPI并行程序的编译将在后面介绍。

4.1 编译器简介

4.1.1 Intel C/C++ Fortran编译器简介

Intel C/C++ Fortran编译器是一种主要针对Inetl平台的高性能编译器，可用于开发复杂且要进行大量计算的程序。

[4]此命令虽然是针对MPI设计的，但设置后会更改串行的编译环境。

系统已经安装并设置默认使用64位的12.1.3.293版本的Intel编译器，目录为*/opt/intel/composer_xe_2011_sp1.9.293*，用户直接使用即可，无需自己设置。

系统还安装有64位（未装32位的）11.1.073版本的编译器，安装的目录为*/opt/intel/Compiler/11.1/073*，用户如想使用，可以在自己的*~/.bashrc*之类环境设置文件中添加代码（␣表示空格）：

```
./opt/intel/Compiler/11.1/073/bin/ifortvars.sh␣intel64
```

Intel编译器编译C和C++源程序的编译命令分别为*icc*和*icpc*；编译Fortran源程序的命令为*ifort*。*icpc*命令使用与*icc*命令相同的编译器选项，利用*icpc*编译时将后缀为.c和.i的文件看作为C++文件，而利用*icc*编译时将后缀为.c和.i的文件则看作为C文件。用*icpc*编译时，总会链接C++库，而用*icc*编译时，只有在编译命令行中包含C++源文件时才链接C++库。

官方手册目录：

- 12.1.3.293版本：*/opt/intel/composer_xe_2011_sp1.9.293/Documentation*
- 11.1.073版本：*/opt/intel/Compiler/11.1/073/Documentation*

4.1.2 PGI C/C++ Fortran编译器简介

PGI C/C++ Fortran编译器是一种针对多种CPU与操作系统的高性能编译器，可用于开发复杂且要进行大量计算的程序。当前安装的版本为2010 v10.6，安装在*/opt/pgi/linux86-64/10.6*。

PGI编译器编译C、C++、Fortran 77源程序的命令分别为*pgcc*、*pgCC*和*pgf77*，编译Fortran 90（为了描述方便，本文中将Fortran 90、95、2003、2008标准统称为Fortran 90）的源程序的命令有*pgf90*、*pgf901*、*pgf902*、*pgf90_ex*、*pgf95*和*pgfortran*。

官方手册目录：*/opt/pgi/linux86-64/10.6/doc*。

4.1.3 GNU C/C++ Fortran编译器简介

GNU C/C++ Fortran（GCC）编译器为系统自带的编译器，当前安装的版本为4.4.6和3.4.6，默认为4.4.6版本，用户无需特殊设置即可使用。GNU编译器编译C、C++源程序的命令分别为4.4.6版本的*gcc*和*g++*及3.4.6版本的*gcc34*和*g++34*；4.4.6版本的*gfortran*可以直接编译Fortran 77、90源程序。3.4.6版本的*g77*只能编译F77程序，不可编译Fortran 90源程序。

4.2 C/C++程序的编译

本节主要介绍C/C++源程序的常用编译方式。建议采用性能较好的Intel编译器，用户也可以选择适合自己程序的编译器，以获取更好的性能。

4.2.1 输入输出文件后缀与类型的关系

编译器默认将按照输入文件的后缀判断文件类型，输入文件的后缀与类型的关系见表1。

表 1: 输入文件后缀与类型的关系

文件名	解释	动作
filename.c	C源文件	传给编译器
filename.C filename.CC filename.cc filename.cpp filename.cxx	C++源文件	传给编译器
filename.a filename.so	库文件	传递给链接器
filename.i	已预处理的文件	传递给标准输出
filename.o	目标文件	传递给链接器
filename.s	汇编文件	传递给汇编器

编译器默认将输出按照文件类型与后缀相对应，输出文件的后缀与类型的关系见表2。

表 2: 输出文件后缀与文件类型的关系

文件名	解释
filename.i	已预处理的文件，由使用-p选项生成
filename.o	目标文件，由添加-c选项生成
filename.s	汇编文件，由添加-s选项生成
a.out	默认生成的可执行文件

4.2.2 Intel C/C++编译器重要编译选项

- -Bdynamic：在运行时动态链接所需要的库。
- -Bstatic：静态链接用户生成的库。
- -c：仅编译成目标文件（.o文件）。
- -fast：最大化整个程序的速度。这里是所谓的最大化，还是需要结合程序本身使用合适的选项，默认不使用此选项。
- -g：包含调试信息。
- -ip：在单个文件中进行过程间优化(Interprocedural Optimizations-IPO)。
- -ipo[n]：在多文件中进行过程间优化，非负整数n为可生成的目标文件数。
- -I<头文件目录>：指明头文件的搜索路径。
- -L<库目录>：指明库的搜索路径。
- -l<库文件>：指明所需链接的库名，如库名为libxyz.a，则可用-lxyz指定。
- -o file：指定生成的文件名。
- -openmp：编译OpenMP程序。注意：在本系统上只能在同一个节点内的CPU上运行OpenMP程序，提交作业时请结合相应选项，以保证在同一个节点运行。
- -O<级别>：设定优化级别，默认为O2。O与O2相同，推荐使用；O3为在O2基础之上增加更激进的优化，比如包含循环和内存读取转换和预取等，但在有些情况下速度反而慢，建议在具有大量浮点计算和大数据处理的循环时的程序使用。
- -p：进行概要导向优化(Profile Guided Optimization-PGO)。
- -shared: 生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项。
- -static：静态链接所有库。
- -std=<标准>：标准可以为c89、c99、gnu89、gnu++98或c++0x，分别对应相应标准。
- -w：编译时不显示任何警告，只显示错误。
- -wall：编译时显示所有警告。

- -x␣<类型>：类型可以为c、c++、c-header、cpp-output、c++-cpp-output、assembler、assembler-with-cpp或none，分别表示c源文件等，以使所有源文件都被认为是此类型的。

建议仔细看看编译器手册中关于程序优化的部分，特别是IPO、PGO和HLO部分，多加测试，选择适合自己程序的编译选项以提高性能。

4.2.3 PGI C/C++编译器重要编译选项

PGI编译器选项非常多，下面仅仅是列出一些本人认为常用的关于编译C程序的*pgcc*命令的重要选项。编译C++程序的*pgCC*命令有稍微不同，建议仔细查看PGI相关资料。

- 一般选项：
  - -#：显示编译器、汇编器、链接器的调用信息。
  - -c：仅编译成目标文件（.o文件）。
  - -defaultoptions和-nodefaultoptions：是否使用默认选项，默认为使用。
  - -flags：显示所有可用的编译选项。
  - -help[=option]：显示帮助信息，option可以为groups、asm、debug、language、linker、opt、other、overall、phase、phase、prepro、suffix、switch、target和variable。
  - -Minform=level：控制编译时错误信息的显示级别。level可以为fatal、file、severe、warn、inform，默认为-Minform=warn。
  - -noswitcherror：显示警告信息后，忽略未知命令行参数并继续进行编译。默认显示错误信息并且终止编译。
  - -o file：指定生成的文件名。
  - -show：显示现有pgcc命令的配置信息。
  - -silent：不显示警告信息，与-Minform=severe等同。
  - -v：详细模式，在每个命令执行前显示其命令行。
  - -V：显示编译器版本信息。
  - -w：编译时不显示任何警告，只显示错误。
- 优化选项：
  - -fast：编译时选择针对目标平台的普通优化选项。用*pgcc␣-fast␣-help*可以查看等价的开关。优化级别至少为O2，参看-O选项。

- -fastsse：对支持SSE和SSE2指令的CPU（如Xeon）编译时选择针对目标平台的优化选项。用*pgcc -fastsse -help*可以查看等价的开关，优化级别至少为O2，参看-O选项。
- -fpic或-fPIC：编译器生成位置无关代码，以便可用于生成共享目标文件（动态链接库）。
- -Kpic或-KPIC：与-fpic或-fPIC相同，为了与其余编译器兼容。
- -Minfo[=option[,option,…]]：显示有用信息到标准错误输出，选项可为all、autoinline、inline、ipa、loop或opt、mp、time或stat。
- -Mipa[=option[,option,…]]和-Mnoipa：启用指定选项的过程间分析优化，默认为-Mnoipa。
- -Mneginfo=option[,option…]：使编译器显示为什么特定优化没有实现的信息。选项包括concur、loop和all。
- -Mnoopenmp：当使用-mp选项时，忽略OpenMP并行指令。
- -Mnosgimp：当使用-mp选项时，忽略SGI并行指令。
- -Mpfi：生成概要导向工具，此时将会包含特殊代码收集运行时的统计信息以用于子序列编译。-Mpfi必须在链接时也得使用。当程序运行时，会生成概要导向文件pgfi.out。
- -Mpfo：启用概要导向优化，此时必须在当前目录下有概要文件pgfi.out。
- -Mprof[=option[,option,…]]：设置性能功能概要选项。此选项可使得结果执行生成性能概要，以便PGPROF性能概要器分析。
- -mp[=option]：打开对源程序中的OpenMP并行指令的支持。
- -O[level]：设置优化级别。level可设为0、1、2、3、4，其中4与3相同。
- -pg：使用gprof风格的基于抽样的概要刨析。

- 调试选项：
  - -g：包含调试信息。
- 预处理选项：
  - -C：预处理时保留C源文件中的注释。
  - -D<name[=def]>：预处理时定义宏name为def。
  - -dD：打印源文件中已定义的宏及其值到标准输出。
  - -dI：打印预处理中包含的所有文件信息，含文件名和定义时的行号。
  - -dM：打印预处理时源文件已定义的宏及其值，含定义时的文件名和行号。
  - -dN：与-dD类似，但只打印源文件已定义的宏，而不打印宏值。

- -E：预处理每个.c文件，将结果发送给标准输出，但不进行编译、汇编或链接等操作。
- -I<头文件目录>：指明头文件的搜索路径。
- -M：打印make的依赖关系到标准输出。
- -MD：打印make的依赖关系到文件file.d，其中file是编译文件的根名字。
- -MM：打印make的依赖关系到标准输出，但忽略系统头文件。
- -MMD：打印make的依赖关系到文件file.d，其中file是编译的文件的根名字，但忽略系统头文件。
- -P：预处理每个文件，并保留每个file.c文件预处理后的结果到file.i。
- -U<name>：去除预处理中的任何name的初始定义。

• 链接选项：

- -Bdynamic：在运行时动态链接所需的库。
- -Bstatic：静态链接所需的库。
- -Bstatic_pgi：动态链接系统库时静态链接PGI库。
- -g77libs：允许链接GNU g77或gcc生成的库。
- -l<库文件>：指明所需链接的库名。如库为libxyz.a，则可用-lxyz指定。
- -L<库目录>：指明库的搜索路径。
- -m：显示链接拓扑。
- -Mrpath和-Mnorpath：默认为-rpath，以给出包含PGI共享目标的路径。用-Mnorpath可以去除此路径。
- -pgf77libs：链接时添加pgf77运行库，以允许混合编程。
- -r：生成可以重新链接的目标文件。
- -R<directory>：对共享目标文件总搜索directory目录。
- -pgf90libs：链接时添加pgf90运行库，以允许混合编程。
- -shared：生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项。
- -soname<name>：生成共享目标时，用内在的DT_SONAME代替指定的name。
- -u<name>：传递给链接器，以生成未定义的引用。

• 语言选项：

- -B：源文件中允许C++风格的注释，指的是以//开始到行尾内容为注释。除非指定-C选项，否则这些注释被去除。

  - -c8x或-c89：对C源文件采用C89标准。
  - -c9x或-c99：对C源文件采用C99标准。

- 平台相关选项：
  - -Kieee和-Knoieee：浮点操作是否严格按照IEEE 754标准。使用-Kieee时一些优化处理将被禁止，并且使用更精确的数值库。默认为-Knoieee，将使用更快的但精确性低的方式。
  - -Ktrap=[option,[option]…]：控制异常发生时CPU的操作。选项可为divz、fp、align、denorm、inexact、inv、none、ovf、unf，默认为none。
  - -Msecond_underscore和-Mnosecond_underscore：是否对已有_的Fortran名字添加第二个_。为与g77兼容时使用，因g77默认符号后添加第二个_。
  - -mcmodel=small|medium：使内存模型是否限制目标小于2GB(small)或允许数据块大于2GB(medium)。medium时暗含-Mlarge_arrays选项。
  - -tp target：target可以为nehalem-64等，默认与编译时的平台一致。

建议仔细看看编译器手册中关于程序优化的部分，多加测试，选择适合自己程序的编译选项以提高性能。本系统采用的是Intel Xeon X5650（Westmere核心架构）CPU，需要仔细选择，特别是保证结果的正确性。

4.2.4 GNU C/C++编译器GCC重要编译选项

GNU编译器GCC是Linux系统自带的编译器，系统安装的版本为4.4.6和3.4.6，选项非常多，下面仅仅是列出一些针对4.4.6本人认为常用的重要选项，建议仔细看GCC相关资料。

- 控制文件类型的选项：
  - -x language：明确指定而非让编译器判断输入文件的类型。language可为：
    * c、c-header、c-cpp-output
    * c++、c++-header、c++-cpp-output
    * objective-c、objective-c-header、objective-c-cpp-output
    * objective-c++ objective-c++-header objective-c++-cpp-output
    * assembler、assembler-with-cpp
    * ada
    * f95、f95-cpp-input
    * java

* treelang

当language为none时，禁止任何明确指定的类型，其类型由文件名后缀决定。

– -c：仅编译成目标文件（.o文件），并不进行链接。
– -o file：指定生成的文件名。
– -v：详细模式，显示在每个命令执行前显示其命令行。
– -###：显示编译器、汇编器、链接器的调用信息但并不进行实际编译，在脚本中可以用于捕获驱动器生成的命令行。
– –help：显示帮助信息。
– –target-help：显示目标平台的帮助信息。
– –version：显示编译器版本信息。

• C语言选项：

– -ansi：C模式时，支持所有ISO C90指令。在C++模式时，去除与ISO C++冲突的GNU扩展。
– -std=：控制语言标准，可以为c89、iso9899:1990、iso9899:199409、c99、c9x、iso9899:1999、iso9899:199x、gnu89、gnu99、gnu9x、c++98、gnu++98。
– -B：在源文件中允许C++风格的注释，指的是以//开始到行尾内容为注释。除非指定-C选项，否则这些注释被去除。
– -c8x或-c89：对C源文件采用C89标准。
– -c9x或-c99：对C源文件采用C99标准。

• 警告选项：

– -fsyntax-only：仅仅检查代码的语法错误，并不进行其它操作。
– -w：编译时不显示任何警告，只显示错误。
– -Wfatal-errors：遇到第一个错误就停止，而不尝试继续运行显示更多错误信息。

• 调试选项：

– -g：包含调试信息。
– -ggdb：包含利用gbd调试时所需要的信息。

- 优化选项：
  - -O[level]：设置优化级别。优化级别level可以设置为0、1、2、3、s。
- 预处理选项：
  - -C：预处理时保留C源文件中的注释。
  - -D name：预处理时定义宏name的值为1。
  - -D name=def：预处理时定义name为def。
  - -U name：预处理时去除的任何name初始定义。
  - -undef：不预定义系统或GCC声明的宏，但标准预定义的宏仍旧被定义。
  - -dD：显示源文件中定义的宏及其值到标准输出。
  - -dI：显示预处理中包含的所有文件，包括文件名和定义时的行号信息。
  - -dM：显示预处理时源文件中定义的宏及其值，包括定义时文件名和行号。
  - -dN：与-dD类似，但只显示源文件中定义的宏，而不显示宏值。
  - -E：预处理各.c文件，将结果发给标准输出，不进行编译、汇编或链接。
  - -I<头文件目录>：指明头文件的搜索路径。
  - -M：打印make的依赖关系到标准输出。
  - -MD：打印make的依赖关系到文件file.d，其中file是编译文件的根名字。
  - -MM：打印make的依赖关系到标准输出，但忽略系统头文件。
  - -MMD：打印make的依赖关系到文件file.d，其中file是编译的文件的根名字，但忽略系统头文件。
  - -P：预处理每个文件，并保留每个file.c文件预处理后的结果到file.i。
- 链接选项：
  - -pie：在支持的目标上生成位置无关的可执行文件。
  - -s：从可执行文件中去除所有符号表。
  - -rdynamic：添加所有符号表到动态符号表中。
  - -static：静态链接所需的库。
  - -shared：生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项。
  - -shared-libgcc：使用共享libgcc库。
  - -static-libgcc：使用静态libgcc库。
  - -u <symbol>：确保符号symbol未定义，强制链接一个库模块来定义它。

  - -I<头文件目录>：指明头文件的搜索路径。
  - -l<库文件>：指明所需链接的库名，如库为libxyz.a，则可用-lxyz指定。
  - -L<库目录>：指明库的搜索路径。
  - -B<路径>：设置寻找可执行文件、库、头文件、数据文件等路径。

- Intel 386和AMD x86-64平台相关选项：
  - -mtune=cpu-type：设置优化针对的CPU类型，可为：generic、core2、k8、opteron等，core2为针对本系统至强Xeon CPU的。
  - -march=cpu-type：设置指令针对的CPU类型，CPU类型与上行中一样。
  - -mieee-fp和-mno-ieee-fp：浮点操作是否严格按照IEEE标准。

- 约定成俗的选项：
  - -fpic：生成位置无关的代码以用于共享库。
  - -fPIC：如果目标机器支持，将生成位置无关的代码。
  - -fopenmp：编译OpenMP并行程序。
  - -fpie和-fPIE：与-fpic和-fPIC类似，但生成的位置无关代码，只能链接到可执行文件中。

建议仔细看看编译器手册中关于程序优化的部分，多加测试，选择适合自己程序的编译选项以提高性能。

4.2.5 C/C++程序编译举例

- Intel C/C++编译器编译举例：
  - *icc -o yourprog yourprog.c*
    将C程序yourprog.c编译为可执行文件yourprog。
  - *icpc -o yourprog yourprog.cpp*
    将C++程序yourprog.cpp编译为可执行文件yourprog。
  - *icc -o yourprog-omp -openmp yourprog.c*
    将OpenMP指令并行的C程序yourprog-omp.c编译为可执行文件yourprog-omp。

- PGI C/C++编译器编译举例：
  - *pgcc -o yourprog yourprog.c*
    将C程序yourprog.c编译为可执行文件yourprog。

- *pgCC -o yourprog yourprog.cpp*
  将C++程序yourprog.cpp编译为可执行文件yourprog。
- *pgcc -o yourprog-omp -mp yourprog.c*
  将OpenMP指令并行的C程序yourprog-omp.c编译为可执行文件yourprog-omp。

- GNU C/C++编译器编译举例：
  - *gcc -o yourprog yourprog.c*
    将C程序yourprog.c编译为可执行文件yourprog。
  - *g++ -o yourprog yourprog.cpp*
    将C++程序yourprog.cpp编译为可执行文件yourprog。
  - *gcc -o yourprog-omp -fopenmp yourprog.c*
    将OpenMP指令并行的C程序yourprog-omp.c编译为可执行文件yourprog-omp。

4.3 Fortran程序的编译

4.3.1 输入输出文件后缀与类型的关系

编译器默认将按照输入文件的后缀判断文件类型，输入文件的后缀与类型的关系见表3。

编译器默认将输出按照文件类型与后缀相对应，输出文件的后缀与类型的关系见表4。

4.3.2 Intel Fortran编译器重要编译选项

- -Bdynamic：运行时动态链接所需要的库。
- -Bstatic：静态链接用户生成的库。
- -c：仅编译成目标文件（.o文件）。
- -convert [关键字]：转换无格式数据的类型，比如关键字为big_endian和little_endian时，分别表示无格式的输入输出为big_endian和little_endian格式，更多格式类型，请看编译器手册。
- -cpp：对源代码进行预处理，等价于-fpp。
- -extend-source[size]：指明固定格式的Fortran源代码宽度，选项size可为72、80和132。也可直接用-72、-80和-132指定，默认为72字符。

表 3: 输入文件后缀与文件类型的关系

文件名	解释	动作
filename.a	目标库文件	传给编译器
filename.f filename.for filename.ftn filename.i	固定格式的Fortran源文件	被Fortran编译器编译
filename.fpp filename.FPP filename.F filename.FOR filename.FTN	固定格式的Fortran源文件	自动被Fortran编译器预处理后再被编译
filename.f90 filename.i90	自由格式的Fortran源文件	被Fortran编译器编译
filename.F90	自由格式的Fortran源文件	自动被Fortran编译器预处理后再被编译
filename.s	汇编文件	传递给汇编器
filename.so	库文件	传递给链接器
filename.o	目标文件	传递给链接器

表 4: 输出文件后缀与类型的关系

文件名	解释	生成方式
filename.o	目标文件	编译时添加-c选项生成
filename.so	共享库文件	编译时指定为共享型，如添加-shared，并不含-c
filename.mod	模块文件	编译含有MODULE声明时的源文件生成
filename.s	汇编文件	编译时添加-S选项生成
a.out	默认生成的可执行文件	编译时没有指定-c时生成

- -fast：最大化整个程序的速度。这里是所谓的最大化，还是需要结合程序本身使用合适的选项。
- -fixed：指明Fortran源代码为固定格式，默认由文件后缀决定格式类别。
- -fpic：生成位置无关代码，当编译成共享目标文件时必须使用此选项，等价于-fPIC，默认为-fno-pic。
- -free：指明Fortran源程序为自由格式，默认由文件后缀决定格式类别。
- -g：包含调试信息。
- -ip：在单个文件中进行过程间优化(Interprocedural Optimizations-IPO)。
- -ipo[n]：在多文件中进行过程间优化，非负整数n为可生成的目标文件数。
- -I<头文件目录>：指明头文件的搜索路径。
- -implicitnone：指明默认变量名为未定义，建议在写程序时添加implicit none语句，以避免出现由于默认类型造成的错误。
- -L<库目录>：指明库的搜索路径。
- -l<库文件>：指明所需链接的库名，如库文件为libxyz.a，则可用-lxyz指定。
- -nofree：指明Fortran源程序为固定格式。
- -openmp：编译OpenMP指令并行程序，注意：在本系统上只能在同一个节点内CPU上运行OpenMP程序，提交作业时请结合相应选项，以保证在同一个节点上运行。
- -O<级别>：设定优化级别。默认为O2，O与O2相同，推荐使用。O3为在O2基础之上增加更激进的优化，比如包含循环和内存读取转换和预取等，但在有些情况下速度反而慢，建议在具有大量浮点计算和大数据处理的循环时的程序使用。
- -p：进行概要导向优化(Profile Guided Optimization-PGO)。
- -shared: 生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项。
- -stand <标准>：以指定Fortran标准进行编译，编译时显示源文件中不符合此标准的信息。标准可为f03、f90、f95和none，分别对应显示不符合Fortran 2003、90、95的代码信息和不显示任何非标准的代码信息，也可写为-std<标准>，此时标准不带f，可为03、90、95。
- -static：静态链接所有库。

- -unroll[n]：循环最大可展开的层数，与性能相关。
- -us：编译时给外部用户定义的函数名添加一个下划线，等价于-assume underscore，如果编译时显示_函数找不到时也许添加此选项即可解决。
- -w：编译时不显示任何警告，只显示错误。
- -wall：编译时显示所有警告。
- -X：编译时不用默认的头文件搜索目录，与-I结合可使用指定的头文件目录。

建议仔细看看编译器手册中关于程序优化的部分，特别是IPO、PGO和HLO部分，多加测试，选择适合自己程序的编译选项以提高性能。另外Intel手册是针对IntelCPU写的，本系统采用的是AMD Opteron BarcelonaCPU，因此参数未必合适，建议参考Intel XeonCPU的编译参数，并仔细选择，首先是保证结果的正确性。

4.3.3 PGI Fortran编译器重要编译选项

PGI编译器选项非常多，下面仅仅是列出一些本人认为常用的编译Fortran 9x程序的pgf90重要选项，编译Fortran 77程序的pgf77等编译命令也许有部分不同，建议仔细看PGI相关资料。

- 一般选项：
  - -#：显示编译器、汇编器、链接器的调用。
  - -c：仅编译成目标文件（.o文件）。
  - -defaultoptions和-nodefaultoptions：是否使用默认选项，默认为使用。
  - -flags：显示所有可用的编译参数。
  - -help[=option]：显示帮助信息，option可以为groups、asm、debug、language、linker、opt、other、overall、phase、phase、prepro、suffix、switch、target和variable。
  - -Minform=level：控制编译时错误信息的显示级别，level可以为fatal、file、severe、warn、inform，默认为-Minform=warn。
  - -noswitcherror：显示警告信息后，忽略未知命令行参数继续进行编译；默认为显示错误信息并终止编译。
  - -o file：指定生成的文件名。
  - -show：显示现有pgf90命令的配置信息。
  - -silent：不显示警告信息，与-Minform=severe等同。

  - -v：详细模式，显示在每个命令执行前显示其命令行。
  - -V：显示编译器版本信息。
  - -w：编译时不显示任何警告，只显示错误。

- 优化选项：
  - -fast：编译时选择针对目标平台的普通优化参数，用pgf90 -fast -help可以查看等价的开关。优化级别至少为O2，参看-O参数。
  - -fastsse：对支持SSE和SSE2指令的CPU（如Opteron）编译时选择针对目标平台的普通优化参数，用pgcc -fastsse -help可以查看等价的开关。优化级别至少为O2，参看-O参数。
  - -fpic或-fPIC：编译器生成位置无关代码，以便可以用于生成共享目标文件（动态链接库）。
  - -Kpic或-KPIC：与-fpic或-fPIC相同，为了与其余编译器兼容。
  - -Minfo[=option[,option,…]]：显示有用的信息到标准错误输出，选项可以为all、autoinline、inline、ipa、loop或opt、mp、time或stat。
  - -Mipa [=option[,option,…]]和-Mnoipa：对过程间分析启用和指定参数，默认为-Mnoipa。
  - -Mneginfo=option[,option…]：使编译器生成关于为什么特定优化没有实现的信息。选项包括concur、loop和all。
  - -Mnoopenmp：当使用-mp选项时，忽略OpenMP指令。
  - -Mnosgimp：当使用-mp选项时，忽略SGI并行指令。
  - -Mpfi：生成概要导向工具，此时将会包含特殊代码以收集运行时的统计信息以用于子序列的编译中。-Mpfi必须在链接时也得使用。当程序运行时，会生成概要导向文件pgfi.out。
  - -Mpfo：启动概要导向优化，此时必须在当前目录下存在概要文件pgfi.out。
  - -Mprof[=option[,option,…]]：设置性能功能概要选项。用此选项可使结果执行生成性能概要，以便PGPROF性能概要器可以分析。
  - -mp[=option]：打开对源程序中的OpenMP并行指令的支持。
  - -O[level]：设置优化级别。level可设为0、1、2、3、4，其中4与3相同。
  - -pg：使用gprof风格的基于抽样的概要刨析。

- 调试选项：
  - -g：包含调试信息。

- 预处理选项：

- -C：预处理时保留C源文件中的注释。
- -D<name[=def]>：预处理时定义name为def。
- -dD：显示源文件中定义的宏及其值到标准输出。
- -dI：显示预处理中包含的所有文件，包括文件名和定义时的行号信息。
- -dM：显示预处理时源文件中定义的宏及其值，包括定义时文件名和行号。
- -dN：与-dD类似，但只显示源文件中已定义的宏，而不显示宏值。
- -E：预处理各.c文件，将结果发给标准输出，不进行编译、汇编或链接。
- -I<头文件目录>：指明头文件的搜索路径。
- -M：显示make的依赖关系到标准输出。
- -MD：显示make的依赖关系到文件file.d，其中file是编译文件的根名字。
- -MM：显示make的依赖关系到标准输出，但忽略系统头文件。
- -MMD：显示make的依赖关系到文件file.d，其中file是编译的文件的根名字，但忽略系统头文件。
- -P：预处理每个文件，并保留每个file.c文件预处理后的结果到file.i。
- -U<name>：预处理去除时name的初始定义。

• 链接选项：

- -Bdynamic：运行时动态链接所需的库。
- -Bstatic：静态链接所需的库。
- -Bstatic_pgi：对动态链接系统库时静态链接PGI库。
- -g77libs：允许链接GNU g77或gcc生成的库。
- -l<库文件>：指明所需链接的库名，如库为libxyz.a，则可用-lxyz指定。
- -L<库目录>：指明库的搜索路径。
- -m：显示链接拓扑。
- -Mrpath和-Mnorpath：默认为-rpath，以设置包含PGI共享目标的路径。用-Mnorpath可以去除此路径。
- -pgf77libs：链接时添加pgf77运行库，以允许混合编程。
- -r：生成可以重新链接的目标文件。
- -R<directory>：对共享目标文件总搜索directory目录。
- -pgf90libs：链接时添加pgf90运行库，以允许混合编程。
- -shared：生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项。

- -soname<name>: 生成共享目标时, 用内在的DT_SONAME代替指定的name。
- -u<name>：传递给链接器，以生成未定义的引用。

• 语言选项：

- -byteswapio或-Mbyteswapio：对无格式Fortran数据文件在输入输出时从大端（big-endian）到小端（little-endian）交换比特，或者相反。此选项可以用于读写Sun或SGI等系统中的无格式的Fortran数据文件。
- -i2：将INTEGER变量按照2比特处理。
- -i4：将INTEGER变量按照4比特处理。
- -i8：将默认的INTEGER和LOGICAL变量按照4比特处理。
- -i8storage：对INTEGER和LOGICAL变量分配8比特。
- -Mallocatable[=95|03]：按照Fortran 95或2003标准分配数组。
- -Mbackslash和-Mnobackslash：将反斜线(\)当作正常字符（非转义符）处理，默认为-Mnobackslash。-Mnobackslash导致标准的C反斜线转义序列在引号包含的字串中重新解析。-Mbackslash则导致反斜线被认为和其它字符一样。
- -Mextend：设置源代码的行宽为132列。
- -Mfixed、-Mnofree和-Mnofreeform：强制对源文件按照固定格式进行语法分析，默认.f或.F文件被认为固定格式。
- -Mfree和-Mfreeform：强制对源文件按照自由格式进行语法分析，默认.f90、.F90、.f95或.F95文件被认为自由格式。
- -Mi4和-Mnoi4:将INTEGER看作INTEGER*4。-Mnoi4将INTEGER看作INTEGER*2。
- -Mnomain：当链接时，不包含调用Fortran主程序的目标文件。
- -Mr8和-Mnor8：将REAL看作DOUBLE PRECISION，将实(REAL)常数看作双精度(DOUBLE PRECISION)常数。默认为否。
- -Mr8intrinsics [=float]和-Mnor8intrinsics: 将CMPLX看作DCMPLX, 将REAL看作DBLE。添加float选项时，将FLOAT看作DBLE。
- -Msave和-Mnosave：是否将所有局部变量添加SAVE声明，默认为否。
- -Mupcase和-Mnoupcase：是否保留名字的大小写。-Mnoupcase导致所有名字转换成小写。注意，如果使用-Mupcase，那么变量名X与变量名x不同，并且关键字必须为小写。
- -Mcray=pointer：支持Cray指针扩展。
- -module directory：指定编译时保存生成的模块文件的目录。
- -r4：将DOUBLE PRECISION变量看作REAL。

  - -r8：将REAL变量看作DOUBLE PRECISION。

- 平台相关选项：

  - -Kieee和-Knoieee：浮点操作是否严格按照IEEE 754标准，默认为不。使用-Kieee时一些优化处理被禁止，并且使用更加精确的数值库，默认为-Knoieee，将使用更快的但精确性低的方式。
  - -Ktrap=[option,[option]…]：控制异常发生时，CPU的操作。选项可以为divz、fp、align、denorm、inexact、inv、none、ovf、unf，默认为none。
  - -Mlarge_arrays和-Mnolarge_arrays：是否允许数组大于2GB，默认不允许。当使用-mcmodel=medium时暗含-Mlarge_arrays选项。
  - -mcmodel=small|medium：使用内存模型是否限制目标小于2GB(small)或允许数据块大于2GB(medium)，medium时暗含-Mlarge_arrays选项。
  - -Msecond_underscore和-Mnosecond_underscore：是否对已经有_的Fortran名字添加第二个_。主要在与g77兼容时使用，g77默认给符号添加第二个_。
  - -Mvarargs和-Mnovarargs：是否生成从Fortran调用C程序时用变量参数调用序列，默认为否。
  - -tp target：target可以为amd64、amd64e、barcelona、barcelona-64、k8-32、k8-64、k8-64e、x64等，默认与编译时的平台一致。

建议仔细看看编译器手册中关于程序优化的部分，特别是IPA、PGA等部分，多加测试，选择适合自己程序的编译选项以提高性能。本系统采用的是AMD Opteron BarcelonaCPU，需仔细选择，首先要保证结果的正确性。

4.3.4　GNU Fortran编译器重要编译选项

GNU Fortran编译器是Linux系统自带的Fortran编译器，系统安装的版本为4.4.6，支持大部分*gcc*选项，下面仅仅是列出一些针对4.4.6的*gfortran*本人认为常用的重要选项，建议仔细看GNU Fortran和gcc的相关资料。

- 控制Fortran语言类型的选项：

  - -ffree-form和-ffixed-form：声明源文件是自由格式还是固定格式，默认从Fortran 90起的源文件为自由格式，之前的Fortran 77等的源文件为固定格式。
  - -fdefault-double-8：设置DOUBLE PRECISION类型为8比特。
  - -fdefault-integer-8：设置INTEGER和LOGICAL类型为8比特。
  - -fdefault-real-8：设置REAL类型为8比特。

  - -fno-backslash：将反斜线(\)当作正常字符（非转义符）处理。
  - -ffixed-line-length-<n>：设置固定格式源代码的行宽为n。
  - -ffree-line-length-<n>：设置自由格式源代码的行宽为n。
  - -fmax-identifier-length=<n>：设置名称的最大字符长度为n，Fortran 95和200x的长度分别为31和65。
  - -fimplicit-none：禁止变量的隐式声明，所有变量都需要显式声明。
  - -fcray-pointer：支持Cray指针扩展。
  - -fopenmp：编译OpenMP并行程序。
  - -std=<std>：指明Fortran标准，std可以为f95、f2003、legacy。
  - -M<dir>和-J<dir>：指定编译时保存生成的模块文件目录。
  - -fconvert=<conversion>：指定对无格式Fortran数据文件表示方式，其值可以为：native，默认值；swap，在输入输出时从大端（big-endian）到小端（little-endian）交换比特，或者相反；big-endian，用大端方式读写；little-endian，用小端方式读写。

- 一般选项：
  - -c：仅编译成目标文件（.o文件），并不进行链接。
  - -o file：指定生成的文件名。
  - -v：详细模式，显示在每个命令执行前显示其命令行。
  - -###：显示编译器、汇编器、链接器的调用信息但并不进行实际编译，在脚本中可以用于捕获驱动器生成的命令行。
  - –help：显示帮助信息。
  - –target-help：显示目标平台的帮助信息。
  - –version：显示编译器版本信息。

- 警告选项：
  - -fsyntax-only：仅仅检查代码的语法错误，并不进行其余操作。
  - -w：编译时不显示任何警告，只显示错误。
  - -Wfatal-errors：遇到第一个错误就停止，而不尝试继续运行。

- 调试选项：
  - -g：包含调试信息。
  - -ggdb：包含利用gbd调试时所需要的信息。

- 优化选项：
  - -O[level]：设置优化级别。优化级别level可以设置为0、1、2、3、s。
- 预处理选项：
  - -C：保留预处理的C源文件中的注释。
  - -D name：在预处理中定义宏name的值为1。
  - -D name=def：在预处理中定义name为def。
  - -U name：去除预处理中的任何name初始定义。
  - -undef：不预定义系统或GCC声明的宏，但标准预定义的宏仍旧被定义。
  - -dD：显示源文件中定义的宏及其值到标准输出。
  - -dI：显示预处理中包含的所有文件，包括文件名和定义时的行号。
  - -dM：显示预处理时源文件中定义的宏及值，含定义时文件名和行号。
  - -dN：与-dD类似，但只显示源文件中定义的宏，而不显示宏值。
  - -E：预处理各文件，将结果发给标准输出，不进行编译、汇编或链接。
  - -I<头文件目录>：指明头文件的搜索路径。
  - -M：打印make的依赖关系到标准输出。
  - -MD：打印make的依赖关系到文件file.d，其中file是编译文件的根名字。
  - -MM：打印make的依赖关系到标准输出，但忽略系统包含。
  - -MMD：打印make的依赖关系到文件file.d，其中file是编译的文件的根名字，但忽略系统头文件。
  - -P：预处理每个文件，并保留每个file.c文件预处理后的结果到file.i。
- 链接选项：
  - -pie：在支持的目标上生成位置无关的可执行文件。
  - -s：从可执行文件中去除所有符号表。
  - -rdynamic：添加所有符号表到动态符号表中。
  - -static：静态链接所需的库。
  - -shared：生成共享目标而不是可执行文件，必须在编译每个目标文件时使用-fpic选项。
  - -shared-libgcc：使用共享libgcc库。
  - -static-libgcc：使用静态libgcc库。
  - -u <symbol>：确保符号symbol未定义，强制连接一个库模块来定义它。

  - -I<头文件目录>：指明头文件的搜索路径。
  - -l<库文件>：指明所需链接的库名，如库为libxyz.a，则可用-lxyz指定。
  - -L<库目录>：指明库的搜索路径。
  - -B<路径>：设置寻找可执行文件、库、头文件、数据文件等路径。

- Intel 386和AMD x86-64平台相关选项：
  - -mtune=cpu-type：设置优化针对的CPU类型，可为：generic、core2、k8、opteron等，core2为针对本系统至强Xeon CPU的。
  - -march=cpu-type：设置指令针对的CPU类型，CPU类型与上行中一样。
  - -mieee-fp和-mno-ieee-fp：浮点操作是否严格按照IEEE标准。

- 约定成俗的选项：
  - -fno-automatic：将每个程序单元的本地变量和数组声明具有SAVE属性。
  - -ff2c：与g77和f2c生成的代码兼容。
  - -fno-underscoring：不在名字后添加_。注意：gfortran默认行为与g77和f2c不兼容，为了兼容需要加-ff2c选项。除非使用者了解与现有系统环境的集成，否则不建议使用-fno-underscoring选项。
  - -funderscoring：对外部名字没有_的加_，以与一些Fortran编译器兼容。
  - -fsecond-underscore：默认gfortran对外部名称添加一个_，如果使用此选项，那么将添加两个_。此选项当使用-fno-underscoring选项时无效。此选项当使用-ff2c时默认启用。
  - -fpic：生成位置无关的代码以用于共享库。
  - -fPIC：如果目标机器支持，将生成位置无关的代码。
  - -fpie和-fPIE：与-fpic和-fPIC类似，但生成的位置无关代码只能链接到可执行文件中。

建议仔细看看编译器手册中关于程序优化的部分，多加测试，选择适合自己程序的编译选项以提高性能。

4.3.5 Fortran程序编译举例

- Intel Fortran编译器编译举例：
  - *ifort -o yourprog yourprog.for*
    将Fortran 77程序yourprog.for编译为可执行文件yourprog。

- *ifort -o yourprog -static yourprog.f90*
  将Fortran 90程序yourprog.f90静态编译为可执行文件yourprog。
- *ifort -o yourprog-omp -openmp yourprog.f90*
  将OpenMP指令并行的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp。

• PGI Fortran编译器编译举例：

- *pgf77 -o yourprog yourprog.for*
  将Fortran 77程序yourprog.for编译为可执行文件yourprog。
- *pgf90 -o yourprog -static yourprog.f90*
  将Fortran 90程序yourprog.f90静态编译为可执行文件yourprog。
- *pgf90 -o yourprog-omp -mp yourprog.f90*
  将OpenMP指令并行的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp。

• GNU Fortran编译器编译举例：

- *g77 -o yourprog yourprog.for*
  将Fortran 77程序yourprog.for编译为可执行文件yourprog。
- *gfortran -o yourprog -static yourprog.f90*
  将Fortran 90程序yourprog.f90静态编译为可执行文件yourprog。
- *gfortran -o yourprog-omp -fopenmp yourprog.f90*
  将OpenMP指令并行的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp。

注意：g77不支持OpenMP，也不支持Fortran 90标准。

4.4 OpenMP程序的编译与运行

Intel、PGI、GNU编译器支持OpenMP并行，只需利用编译命令结合必要的OpenMP编译选项编译即可，对应此三种编译器的OpenMP选项分别为-openmp、-mp、-fopenmp。

• Intel编译器：

- *icc -openmp -o yourprog-omp yourprog.c*
  将OpenMP的C程序yourprog-omp.c编译为可执行文件yourprog-omp。
- *ifort -openmp -o yourprog-omp yourprog.f90*
  将OpenMP的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp。

- PGI编译器：
  - *pgcc -mp -o yourprog-omp yourprog.c*
    将OpenMP的C程序yourprog-omp.c编译为可执行文件yourprog-omp。
  - *pgf90 -mp -o yourprog-omp yourprog.f90*
    将OpenMP的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp。
- GNU编译器：
  - *gcc -fopenmp -o yourprog-omp yourprog.c*
    将OpenMP的C程序yourprog-omp.c编译为可执行文件yourprog-omp。
  - *gfortran -fopenmp -o yourprog-omp yourprog.f90*
    将OpenMP的Fortran 90程序yourprog-omp.f90编译为可执行文件yourprog-omp。

OpenMP的运行一般是通过在运行前设置环境变量OMP_NUM_THREADS来控制进程数，比如在bash中利用*export␣OMP_NUM_THREADS=12*设置十二个进程运行。注意，本系统为节点内共享内存节点间分布式内存的架构，因此只能在一个节点上的CPU之间运行OpenMP程序，在提交作业时需要使用相应选项以保证在同一个节点运行）。

5 MPI并行程序编译

本系统的通信网络千兆以太网，系统默认设置使用的Intel 12系列的编译器与Open MPI的组合，其安装目录为*/opt/openmpi-1.4.5*，建议使用。

用户也可以运行*mpi-selector-menu*命令按照提示选择自己使用的MPI环境，注意数字后需要加u，设置完成后最好重新登录以便设置生效：

```
Current system default : openmpi-1.4.5_intel -12.1.3.293
Current user default : openmpi-1.4.5

    "u" and "s" modifiers can be added to numeric and "U"
    commands to specify "user" or "system-wide".

1. openmpi-1.4.3_intel -11.1.073
2. openmpi-1.4.4_intel -12.1.3.293
3. openmpi-1.4.5_intel -12.1.3.293
4. openmpi-1.4.5_pgi-10.6
U. Unset default
Q. Quit

Selection (1-4[us], U[us], Q):
```

上述选项的格式：MPI实现名-MPI实现版本_编译器名-编译器版本，以上述为例，如果想使用Open MPI与PGI的配合，那么就输入4u。

5.1 MPI并行程序的编译

Open MPI的编译命令主要为：

- C程序：*mpicc*
- C++程序：*mpic++*、*mpicxx*、*mpiCC*
- Fortran 77程序：*mpif77*、*mpif90*
- Fortran 90程序：*mpif90*

对于并行程序，对应不同类型源文件的编译命令如下：

- *mpicc -o yourprog-mpi yourprog-mpi.c*
  将C语言的MPI并行程序yourprog-mpi.c编译为可执行文件yourprog-mpi。

- *mpicxx -o yourprog-mpi yourprog-mpi.cpp*
  将C++语言的MPI并行程序yourprog-mpi.cpp编译为可执行文件yourprog-mpi，也可换为*mpic++*或*mpiCC*。

- *mpif77 -o yourprog-mpi yourprog-mpi.f*
  将Fortran 77语言的MPI并行程序yourprog-mpi.f编译为可执行文件yourprog-mpi。

- *mpif90 -o yourprog-mpi yourprog-mpi.f90*
  将Fortran 90语言的MPI并行程序yourprog-mpi.f90编译为可执行文件yourprog-mpi。

MPI编译环境的编译命令实际上是调用Intel、PGI、GCC编译器进行编译，具体优化选项等，请参看Open MPI以及Intel、PGI和GCC编译器手册。

5.2 MPI并行程序的运行

在本系统上，MPI并行程序需结合作业调度系统TORQUE和Maui的作业提交命令*qsub*来调用作业脚本运行，请参看后面的作业调度系统介绍。

5.3 MPI并行程序调试

并行程序调试比比较困难，并行程序的调试一般来说只能利用显示语句来逐步定位错误，建议利用尽量少的进程数来调试以方便进行追踪。Intel编译器带有调试器：命令行的*idbc*和图形界面的*idb*，并且可以调试OpenMP和MPI并行程序，进入后运行help命令可以查看具体命令。

6 数值函数库

本系统上安装的数值函数库主要有Intel Math Kernel Library(MKL)，用户可以直接调用，以提高性能、加快开发。

6.1 Intel MKL

当前安装的MKL版本为10.2.4.073和10.3.9.293，分别安装在*/opt/intel/Compiler/11.1/073/mkl*和*/opt/intel/composer_xe_2011_sp1.9.293*。11.1.073版本只安装有em64t(AMD64)版本（未安装针对i386的32位版本），对应的目录为MKL主目录下的*lib/em64t*子目录，10.3.9.293版本的，具有em64t和i386版本（因为11.1.073版本的Intel编译器没有安装32位的，建议使用Intel编译器编译时不要使用i386版本，10.3.9.293版本的编译器安装有em64t和i386版本，编译器版本和MKL版本最好别混用，以免出现错误）。以下将以10.3.9.293针对em64t系统的版本为例介绍。在bash下可以通过在*~/.bashrc*之类的环境变量设置文件中添加下面代码设置MKL所需的环境变量INCLUDE、LD_LIBRARY_PATH和MANPATH等：

- 10.3.9.293版本的编译器和MKL：

```
../opt/intel/composer_xe_2011_sp1.9.293/bin/compilervars.sh␣intel64
```

- 10.2.4.073版本：

```
../opt/intel/Compiler/11.1/073/mkl/tools/environment/mklvarsem64t.sh
```

6.1.1 MKL主要内容

MKL主要包含如下内容：

- 基本线性代数子系统库(BLAS)
- 离散基本线性代数库(Sparse BLAS)
- 线性代数库(LAPACK)
- 可扩展性线性代数库(ScaLAPACK)
- 离散求解程序(Sparse Solver routines)
- 向量数值库函数(Vector Mathematical Library functions)
- 向量统计库函数(Vector Statistical Library functions)

- 傅立叶变换程序(Fourier Transform functions (FFT))
- 集群版傅立叶变换程序(Cluster FFT)
- 区间求解程序(Interval Solver routines)
- 三角变换程序(Trigonometric Transform routines)
- 泊松、拉普拉斯和哈密顿求解程序(Poisson, Laplace, and HelMHzoltz Solver routines)
- 优化（信赖域）求解程序(Optimization (Trust-Region) Solver routines)

6.1.2 MKL目录内容

MKL的主要目录内容见表5。

6.1.3 链接MKL

为了在程序中链接MKL库中的*libyyy.a*或*libyyy.so*，可以采用两种方式：如以下环境变量已经设好：

- MKLROOT=/opt/intel/composer_xe_2011_sp1.9.293
- MKLPATH=$MKLROOT/lib/intel64
- MKLINCLUDE=$MKLROOT/include

那么可以：

- 在链接行中，列举含有相对或绝对路径的库名，比如：
  *<ld> myprog.o -L$MKLPATH -I$MKLINCLUDE -Wl,–start-group *
  *$MKLPATH/libmkl_intel_lp64.a $MKLPATH/libmkl_intel_thread.a *
  $MKLPATH/libmkl_core.a -Wl,–end-group -liomp5 -lpthread
  其中，<ld>为链接命令，如*ld*，*myprog.o*是用户的目标文件。
- 首先列举所需的MKL库，然后跟着系统库libpthread：
  在链接行中，利用-L<path>列举含有相对或绝对路径的库名（指明搜索库的路径），和-I<include>（指明搜索头文件的路径）。

如果已经利用前面所说的compilervars.sh设置好MKL环境变量，上面则可以简化为无需指定库所在的绝对路径，只需要利用-l<库名>指明所需要的库即可。

表 5: MKL目录内容

目录	内容
<mkl_dir>	MKL主目录，比如*/opt/intel/composer_xe_2011_sp1.9.293/mkl*
<mkl_dir>/benchmarks/linpack	包含OpenMP版的LINPACK的基准程序
<mkl_dir>/benchmarks/mp_linpack	包含MPI版的LINPACK的基准程序
<mkl_dir>/doc	MKL文档目录
<mkl_dir>/examples	一些例子，建议用户参考学习
<mkl_dir>/include	含有INCLUDE文件
<mkl_dir>/interfaces/blas95	包含BLAS的Fortran 90封装及用于编译成库的makefile
<mkl_dir>/interfaces/LAPACK95	包含LAPACK的Fortran 90封装及用于编译成库的makefile
<mkl_dir>/interfaces/fftw2xc	包含2.x版FFTW(C接口)封装及用于编译成库的makefile
<mkl_dir>/interfaces/fftw2xf	包含2.x版FFTW(Fortran接口)封装及用于编译成库的makefile
<mkl_dir>/interfaces/fftw3xc	包含3.x版FFTW(C接口)封装及用于编译成库的makefile
<mkl_dir>/interfaces/fftw3xf	包含3.x版FFTW(Fortran接口)封装及用于编译成库的makefile
<mkl_dir>/interfaces/fftw2x_cdft	包含2.x版MPI FFTW(集群FFT)封装及用于编译成库的makefile
<mkl_dir>/lib/intel64	较新系列版本MKL路径，包含EM64T架构的静态库和共享目标文件
<mkl_dir>/lib/em64t	较久系列版本MKL路径，包含EM64T架构的静态库和共享目标文件
<mkl_dir>/man/man3	MKL的man文档
<mkl_dir>/tests	一些测试文件
<mkl_dir>/tools/builder	包含用于生成定制动态可链接库的工具
<mkl_dir>/tools/environment	包含用于设置环境变量的shell脚本
<mkl_dir>/tools/support	包含使用Intel Premier支持时所需要的包ID和许可代码等信息

或者采用链接MKL库时指明库的路径和库名方式，如下：

-L<MKL␣path>␣-I<MKL␣include>
[-I<MKL␣include>/{ia32|intel64|{ilp64|lp64}}]
[-lmkl_blas{95|95_ilp64|95_lp64}]
[-lmkl_lapack{95|95_ilp64|95_lp64}]
[␣<cluster␣components>␣]
-lmkl_{intel|intel_ilp64|intel_lp64|intel_sp2dp|gf|gf_ilp64|gf_lp64}
-lmkl_{intel_thread|gnu_thread|pgi_thread|sequential}
-lmkl_core
-liomp5␣[-lpthread]␣[-lm]

注意：上面是动态链接的命令，如果想静态链接，需要将含有-l的库名用含有库文件的路径来代替，比如用$MKLPATH/libmkl_core.a代替-lmkl_core，其中$MKLPATH为用户定义的指向MKL库目录的环境变量。

注：*[]*内的表示可选，|表示其中之一、*{}*表示含有。

7 作业管理系统

本系统利用Platform公司的LSF进行资源和作业管理，所有需要运行的作业均必须通过作业提交命令*bsub*提交，提交后可利用相关命令查询作业状态等。为了利用*bsub*提交作业，需要在*bsub*中指定各选项和需要执行的程序。注意：

- 不要在登录节点直接运行作业（编译等日常操作除外），以免影响其余用户的正常使用
- 如果不通过作业调度系统直接在计算节点上运行将会被监护进程直接杀掉

7.1 查看队列情况：bqueues

利用*bqueues*可以查看现有队列信息，例如：

bqueues

```
QUEUE_NAME PRIO STATUS MAX JL/U JL/P JL/H NJOBS PEND RUN SUSP
```

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
mem64	59	Open:Active	-	32	-	-	16	0	16	0
mem96	58	Open:Active	-	48	-	-	24	0	24	0
mem48	55	Open:Active	-	120	-	-	0	0	0	0
serial	50	Open:Active	-	24	-	-	25	0	25	0
fat48	43	Open:Active	-	264	-	-	76	0	76	0
fat64	43	Open:Active	-	128	-	-	340	320	20	0
long	40	Open:Active	-	840	-	-	288	192	96	0
8cpu	36	Open:Active	-	88	-	-	0	0	0	0
normal	30	Open:Active	-	840	-	-	2316	1500	816	0

其中，主要列的含义为：

- QUEUE_NAME：队列名
- PRIO：优先级，数字越大优先级越高
- STATUS：状态。Open:Active表示已激活，可使用；Closed:Active表示已关闭，不可使用
- MAX：队列对应的最大CPU核数，-表示无限，以下类似
- JL/U：单个用户同时可以的CPU核数
- NJOBS：排队、运行和被挂起的总作业所占CPU核数

- PEND：排队中的作业所需CPU核数
- RUN：运行中的作业所占CPU核数
- SUSP：被挂起的作业所占CPU核数

其中：fat48、fat64、mem48、mem64、mem96队列对应大共享内存服务器，数量有限，仅供真正有需求的用户使用，需要单独申请才可以使用。

7.2 提交作业：bsub

用户需要利用*bsub*提交作业，其基本格式为*bsub␣[options]␣command␣[arguments]*。其中options设置队列、CPU核数等的选项，必需在command之前，否则将作为command的参数；arguments为设置作业的可执行程序本身所需要的参数，必须在command之后，否则将作为设置队列等的选项。下面将给出常用的几种提交方式。

7.2.1 提交到特定队列：bsub -q

利用-q选项可以指定提交到哪个队列，作业队列会针对运行情况进行修改，请注意参看登录后的提示或运行*bqueues␣-l*命令查看，现有的队列为：

- normal：所需要的CPU核数不超过12个时，作业将在node1 - node70上运行，此为默认队列。
- long：所需要的CPU核数超过12个时，作业将在node1 - node70上运行。
- serial：所需要的CPU核数为一个时，作业将在node88上运行。
- fat48：提交到此队列上的作业可运行于node81 - node83，node86 - node87节点上，大共享内存服务器，需要特殊申请使用权限。
- fat64：提交到此队列上的作业可运行于node84，node85节点上，大共享内存服务器，需要特殊申请使用权限。

比如想提交到serial队列运行串行程序executable1，可以：

*bsub -q serial executable1*或*bsub executable1*

如果提交成功，将显示类似下面的输出：

```
Job <79722> is submitted to default queue <serial >.
```

其中79722为此作业的作业号，以后可利用此作业号来进行查询及终止等操作。

7.2.2 运行串行作业：bsub -q serial

运行串行作业，请使用串行队列serial，比如：

bsub -q serial executable-serial

作业将在node88运行。

7.2.3 指明所需要的CPU核数：bsub -n

利用-n选项指定所需要的CPU核数（一般来说核数和进程数一致），比如下面指定利用12个CPU核（由-n 12指定）运行MPI（由*mpijob*指明为运行MPI程序）程序：

bsub -q normal -n 12 mpijob executable-mpi1

如需要的核数多于12个，需利用-q long或-q fat48、-q fat64指明使用long或fat48、fat64等队列。

7.2.4 运行MPI作业：bsub -n NUM mpijob

如果需要运行MPI作业，需要利用*mpijob*调用MPI可执行程序，并用-n选项指定所需的CPU核数，比如下面指定利用24颗CPU核运行MPI程序executable-mpi1：

bsub -q long -n 24 mpijob executable-mpi1

7.2.5 运行OpenMP共享内存作业：bsub -q

本系统只能在同一个节点内部运行OpenMP共享内存的作业，此时需要添加利用-q参数指定队列（normal、fat48、fat64队列会自动保证只在单个节点内运行，long队列跨节点运行，要除外）选项：

bsub -q normal -n 12 executable-omp1

7.2.6 运行MPI和OpenMP共享内存混合并行作业

需要针对需求利用LSF环境变量特殊处理，如果自己不清楚怎么处理，请联系管理人员。

7.2.7 运行排他性作业：bsub -x

如果需要独占节点运行，此时需要添加-x选项：

bsub -x -q normal -n 8 executable-omp1

注意：排他性作业在运行期间，不允许其余的作业提交到运行此作业的节点，并且只有在某节点没有任何其余的作业在运行时才会提交到此节点上运行。如果不需要采用排他性运行，请不要使用此选项，否则将导致作业必须等待完全空闲的节点才会运行，也许将增加等待时间。

7.2.8 指明输出、输出文件运行：bsub -i -o -e

作业的正常屏幕输入文件（指的是类似 命令 < 输入文件 方式的文件）、正常屏幕输出到的文件和错误屏幕输出的文件可以利用-i、-o和-e选项来分别指定，运行后可以通过查看指定的这些输出文件来查看运行状态，文件名可利用%J与作业号挂钩。比如指定executable1的输入、正常和错误屏幕输出文件分别为：executable1.input、executable-作业号.log和executable1-作业号.err：

bsub -i executable1.input -o executable1-%J.log -e executable1-%J.err executable1

建议打开-o和-e参数，以便查看作业为什么出问题等，如果需要管理人员协助解决，请告知这些输出，以及运行目录，怎么运行的等，以便管理人员能获取足够的信息及时处理。

7.2.9 交互式运行作业：bsub -I

如果需要运行交互式的作业（如在运行期间需要手动输入参数等进行交互），需要结合-I参数，建议只是在调试期间使用，平常作业还是尽量不要使用此选项，类似选项还有-Ip和-Is：

bsub -I executable1

7.2.10 使用fat48和fat64大共享内存队列

fat48、fat64队列对应大共享内存服务器，数量有限，仅供真正有需求的用户使用，需要单独申请经工作人员批准设定好权限后才可使用。

提交时需要利用*-R "rusage[mem=xxx]"*参数设定所需要的内存，其中xxx为单核所需要的内存（单位为MB），即此作业所需要内存总数除以所需要的核数。如提交一个32进程的大共享内存作业openmp-job到fat48队列运行，申请使用64GB内存，需指定*rusage[mem=2000]*，即总申请内存/申请进程数：64000/32=2000：

bsub -q fat48 -n 32 -o %J.log -e %J.err -R "rusage[mem=2000]" ./openmp-job

7.3 终止作业：bkill

利用*bkill*命令可以终止某个运行中或者排队中的作业，比如：

bkill 79722

运行成功后，将显示类似下面的输出：

```
Job <79722> is being terminated
```

7.4 挂起作业：bstop

利用*bstop*命令可临时挂起某个作业以让别的作业先运行，例如：

bstop 79727

运行成功后，将显示类似下面的输出：

```
Job <79727> is being stopped.
```

此命令可以将排在队列前面的作业临时挂起，以让后面的作业先运行。虽然也可以作用于运行中的作业，但并不会因为此作业被挂起而允许其余作业占用此作业所占用的CPU运行，实际资源不会释放，因此建议不要随便对运行中的作业进行挂起操作，如果运行中的作业不再想继续运行，请用*bkill*终止。

7.5 继续运行被挂起的作业：bresume

利用*bresume*命令可继续运行某个挂起某个作业，例如：

bresume 79727

运行成功后，将显示类似下面的输出：

```
Job <79727> is being resumed.
```

7.6 设置作业最先运行：btop

利用*btop*命令可最先运行排队中的某个作业，例如：

btop 79727

运行成功后，将显示类似下面的输出：

```
Job <79727> has been moved to position 1 from top.
```

7.7 设置作业最后运行：bbot

利用*bbot*命令可设定最后运行排队中的某个作业，例如：

bbot 79727

运行成功后，将显示类似下面的输出：

```
Job <79727> has been moved to position 1 from bottom.
```

7.8 修改排队中的作业选项：bmod

利用*bmod*命令可修改排队中的某个作业的选项，比如想将排队中的运行作业号为79727的作业的执行命令修改为executable2并且换到fat48队列，可以：

bmod -Z executable2 -q fat48 79727

```
Parameters of job <79727> are being changed.
```

7.9 查看作业的排队和运行情况：bjobs

利用*bjobs*可以查看作业的运行情况，比如有哪些作业在运行，哪些在排队，某个作业运行在哪个节点上，以及为什么没有运行等，例如：

bjobs

```
JOBID USER STAT QUEUE FROM_HOST EXEC_HOST JOB_NAME SUBMIT_TIME
79726 hmli RUN normal sugon 2*node31 *executab1 Mar 12 19:20
                            1*node18
                            1*node4
79727 hmli PEND long sugon *executab2 Mar 12 19:20
```

上面显示作业79726在运行，分别在node31、node18和node4上运行2、1、1个进程；而作业79727处于排队中尚未运行，查看未运行的原因可以利用：

bjobs -l 79727

```
Job Id <79727>, sugon <hmli>, Project <default >, Status <PEND>,
Queue <long> , Command <executab2>
Sun Mar 12 14:15:07: Submitted from host <sugon>,
CWD <$HOME>, Requested Resources <type==any && swp>35>;
PENDING REASONS:
SCHEDULING PARAMETERS:
            r15s r1m r15m ut pg io ls it tmp swp mem
loadSched -  0.7 1.0 - 4.0 - - - - - -
loadStop  -  1.5 2.5 - 8.0 - - - - - -
```

以下为另外几个常用参数：

- -u username：查看某用户的作业，如username为all，则查看所有用户的作业。
- -q queuename：查看某队列上的作业。
- -m hostname：查看某节点上的作业。

7.10 查看运行中作业的屏幕正常输出：bpeek

利用*bpeek*命令可查看运行中作业的屏幕正常输出，例如：

bpeek 79727

```
<< output from stdout >>
Radius(nm): 300.000
```

如果在运行中用-o和-e分别指定了正常和错误屏幕输出，也可以通过直接查看指定的文件的内容来查看屏幕输出。

如果想连续查看某个作业的输出，请添加-f参数。

7.11 查看各节点的运行情况：lsload

利用*lsload*命令可查看当前各节点的运行情况，例如：

lsload

```
HOST_NAME status  r15s  r1m  r15m  ut   pg ls    it    tmp    swp    mem
node10     ok     0.0   0.0  0.0    0%   3.5 0    2050 9032M 4000M 16G
node11     locku  0.0   0.0  0.0    0%   3.5 0    2050 9032M 4000M 16G
```

ut列表示利用率。status列中的locku表示在进行排他性运行。

7.12 查看各节点的空闲情况：bhosts

利用*bhosts*命令可查看当前各节点的空闲情况，例如：

bhosts

```
HOST_NAME STATUS JL/U MAX NJOBS RUN SSUSP USUSP RSV
node12    closed -    4   2     2   0     0     0
node10    ok     -    2   2     1   0     0     0
```

STATUS列中的ok表示可以接收新作业，closed表示已经被占满。

7.13 查看用户信息：busers

利用*busers*可以查看用户信息，例如：

busers hmli

USER/GROUP	JL/P	MAX	NJOBS	PEND	RUN	SSUSP	USUSP	RSV
hmli	-	22	40	32	8	0	0	0

8 运行Gaussian等大IO程序注意事项

因为Gaussian程序临时存盘文件非常大，对系统的IO（磁盘的输入输出）压力非常大，主存储无法满足需要，请务必将利用GAUSS_SCRDIR变量其临时存盘文件目录设置为*/tmp*或者*/tmp/用户名*，这样会使用计算节点的本地硬盘，不仅不会影响其他用户，而且自己的作业效率也高。

设置方式为在*~/.bashrc*中添加：

```
export GAUSS_SCRDIR=/tmp
```

或

```
export GAUSS_SCRDIR=/tmp/$USER
```

$USER为Linux系统下的环境变量，代表用户账户名。

如果在日志中只有下面类似信息显示，而无其它信息，一般是*/tmp*目录中无对应用户的目录，请按照上述方式设置。

```
Entering Gaussian System, Link 0=g09
Input=CF3-CH3-radical.com
Output=CF3-CH3-radical.log
```

```
PGFIO/stdio: No such file or directory
PGFIO-F-/OPEN/unit=11/error code returned by host stdio - 2.
 file name = /tmp/用户名/Gau-10502.inp
 in source file ml0.f, at line number 182
  --- traceback not available
```

一般是设置了GAUSS_SCRDIR=/tmp/用户名，但计算节点无对应用户的文件夹（计算节点重启后，此目录会被系统删除，需要重新建立）。解决办法：在出错节点建立对应文件夹，比如利用*ssh node10 mkdir /tmp/$USER*在node10上建立。为了保险最好检查一下所有计算节点，在没有的节点上建立此文件夹，或者运行*/opt/bin/mktmpdir $USER*在所有节点上建立。

关于Gaussian等专业软件本身的问题，更应该用户自己去学习Gaussian手册，管理人员能力和精力有限，也不是这个专业的，不可能对专业业软件了解很多。Gaussian官方手册：http://www.gaussian.com/g_tech/g_ur/g09help.htm。

9 联系方式

- 超算中心：
  - 电话：0551-63602248
  - 信箱：sccadmin@ustc.edu.cn
  - 主页：http://scc.ustc.edu.cn
  - 办公室：中国科大东区新图书馆一楼东侧超级计算中心126室
- 李会民：
  - 电话：0551-63600316
  - 信箱：hmli@ustc.edu.cn
  - 主页：http://hmli.ustc.edu.cn