青岛生物能源与过程研究所超算系统使用指南

中国科学院青岛生物能源与过程研究所 超级计算中心 李会民

2010 年 11 月

目录

1 前言

本用户使用指南主要将对在中国科学院青岛生物能源与过程研究所超级计算平台的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，超级计算中心工作人员也将为用户需要提供必要的技术支持。

由于受水平和时间所限，错误和不妥之处在所难免，欢迎用户指出错误和改进意见，我们将尽力完善。

2 青能所超算系统概述

青能所超级计算平台于 2009 年建成，由超级计算中心负责建设与维护等，主要由曙光天潮 TC2600 刀片服务器和天阔 A950r-F 机架服务器构成（简称曙光集群），总峰值计算能力为每秒 3.6 万亿次，其具体参数为：

- TC2600 刀片：38 台，每刀片两颗主频 2.1GHz 的 AMD Opteron 2352 x86_64 四核处理器（共八核），16GB 内存及一块 146GB SAS 硬盘，38 台刀片浮点计算能力为每秒 2.55 万亿次。
- A950r-F 服务器：两台，每台八颗主频 2.2GHz 的 AMD Opteron 8354 x86_64 四核处理器（共 32 核），64GB 内存及两块 146GB SAS 硬盘，每台计算能力为每秒 0.28 万亿次，两台合计每秒 0.56 万亿次。
- IO 兼计算节点：六台 A620r-FX 机架服务器，每台两颗主频 2.1GHz 的 AMD Opteron 2352 x86_64 四核处理器（共八核），8GB 内存及一块 146GB SAS 硬盘，六台合计总计算能力每秒 0.4 万亿次。
- 管理及用户登录节点：各一台 A620r-FX 机架服务器，每台两颗主频 2.1GHz 的 AMD Opteron 2352 x86_64 四核处理器（共八核），16GB 内存及一块 146GB SAS 硬盘，两台合计总计算能力每秒 0.13 万亿次。
- 计算网络：1、InfiniBand，单向 10Gb/s；2、千兆以太网。
- 管理网络：千兆以太网。
- 存储与备份：各一台 5TB（八块 750GB SATA 硬盘）的 HDS WMS 100 存储。

- 操作系统：CentOS 5.4。
- 编译器：Intel、PGI、GNU 等 C/C++ Fortran 编译器。
- 数学函数库：ACML、MKL 及 BLAS、Goto BLAS、ATLAS、LAPACK、ScaLAPACK 等。
- 并行环境：Open MPI、MVAPICH、MVAPICH2，支持 MPI 并行程序；各刀片节点内的八核和各 A950r-F 节点内的 32 核为共享内存，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。
- 作业管理：Platform LSF 6.0。

3 用户登录与文件传输

曙光集群的操作系统为 CentOS 5.4，不支持 telnet 方式登录，用户需以 ssh[1]方式登录后进行编译、提交作业等操作，用户数据可以利用 ftp和 sftp 协议进行传输（ftp 客户端也许要关闭被动(passive)模式，建议在客户端设置使用安全的 sftp 协议）。

用户登录进来后默认的 shell 为 bash，用户可以利用 ypchsh 命令修改为所需的 shell；用户可以在登录节点上运行 yppasswd 命令修改密码（注意：利用 passwd 命令在登录节点上修改密码无效）。

用户登录进来的默认语言环境为英语，如果希望使用中文，可以在 ~/.bashrc 中 export␣LC_ALL=zh_CN.GBK 或 export␣LC_ALL=zh_CN.UTF8 分别设置为使用 GBK 或 UTF8 中文（注意 ␣ 表示空格）。

CentOS(Community ENTerprise Operating System)是 Linux 发行版之一，它是来自于 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。

[1]在 MS Windows 下可利用 putty 等支持 ssh 协议的客户端软件进行登录，putty 下载：http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

4 串行及 OpenMP 程序编译

在曙光集群上可运行 C/C++、Fortran 的串行程序，以及与 OpenMP 和 MPI 结合的并行程序。用户只需要在登录节点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）。当前安装的编译环境主要为：

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

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，在 AMD Opteron 平台上性能也不错，可用于开发复杂且要进行大量计算的程序。

10 系列版本的 Intel C/C++ 编译器安装在 /opt/intel/cce/x，Fortran 编译器在 /opt/intel/fce/x（x 为版本号，当前安装的版本号为 10.0.023），11 系列的编译器安装在 /opt/intel/Compiler/y/z（当前安装的为 11.1.064，对应的 y 为 11.1，z 为 064），用户可以查看类似目录下还安装有哪些版本。

系统已经设置默认使用 11.1 系列，如果想自己指定使用的版本，可以在 ~/.bashrc 之类的文件中添加类似下面的代码：

- 10.0.023 版本：
  - C/C++：

```
. /opt/intel/cce/10.0.023/bin/iccvars.sh
```

    注意：␣ 表示空格；前面有个 .，以后类似。

  - Fortran：

```
. /opt/intel/fce/10.0.023/bin/ifortvars.sh
```

- 11.1.064 版本：
  - C/C++：

```
. /opt/intel/Compiler/11.1/064/bin/iccvars.sh intel64
```

  - Fortran：

```
. /opt/intel/Compiler/11.1/064/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。

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。

PGI 编译器安装在 /opt/pgi/linux86-64/x（x 为版本号，当前安装的版本为 7.1），用户可以查看类似目录下还安装有哪些版本。系统已经设置默认使用 7.1 版本，用户如果想使用其余版本的编译器，需要自己指定使用 x 版本，可以在 ~/.bashrc 之类的文件中添加：

```
PGI=/opt/pgi/linux86-64/x
export PATH=$PATH:$PGI/bin
export MANPATH=$MANPATH:$PGI/man
```

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 9x 源程序的命令有 pgf90、pgf901、pgf902、pgf90_ex 和 pgf95。

4.1.3 GNU C/C++ Fortran 编译器简介

GNU C/C++ Fortran 4.1.2 编译器为系统自带的默认编译器，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 gcc 和 g++；编译 Fortran 77 和 9x 源程序的命令分别为 g77 和 gfortran。gfortran 可以直接编译 Fortran 77 源程序，但 g77 不可编译 Fortran 9x 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。我们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：在曙光集群上只能在同一个节点内的 CPU 上跑 OpenMP 程序，提交作业时请结合 -a openmp 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。
- -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。另外 Intel 手册是针对 Intel 处理器写的，曙光集群采用的是 AMD Opteron Barcelona 处理器，因此参数未必合适，建议参考 Intel XEON 处理器的编译参数，并仔细选择，特别是保证结果的正确性。

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我们认为常用的关于编译 C 程序的 pgcc 的重要选项。编译 C++ 程序的 pgCC 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 指令的处理器（如 Opteron）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学库。默认为 -Knoieee ，将使用更快的但精确性低的方式。
  - -Ktrap=[option,[option]...]：控制异常发生时处理器的操作。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 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，多加测试，选择适合自己程序的编译选项以提高性能。曙光集群采用的是 AMD Opteron Barcelona 处理器，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 4.1.2 和 3.4.6，选项非常多，下面仅仅是列出一些针对 4.1.2 我们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、k8、opteron 等。
  - -march=cpu-type：设置指令针对的 CPU 类型，可为：generic、k8、opteron 等。
  - -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。曙光集群采用的是 AMD Opteron Barcelona 处理器，需要仔细选择，首先是保证结果的正确性。

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。

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

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。

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

- -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。
- -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：在曙光集群上只能在同一个节点内 CPU 上跑 OpenMP 程序，提交作业时请结合 -a openmp -R “span[hosts=1]” 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 Intel 处理器写的，曙光集群采用的是 AMD Opteron Barcelona 处理器，因此参数未必合适，建议参考 Intel XEON 处理器的编译参数，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我们认为常用的编译 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 指令的处理器（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学库，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，处理器的操作。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。曙光集群采用的是 AMD Opteron Barcelona 处理器，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 4.1.2，支持大部分 gcc 选项，下面仅仅是列出一些针对 4.1.2 的 gfortran 我们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、k8、opteron 等。
  - -march=cpu-type：设置指令针对的 CPU 类型，可为：generic、k8、opteron 等。
  - -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。曙光集群采用的是 AMD Opteron Barcelona 处理器，需仔细选择，首先要保证结果的正确性。

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 起的标准，但 gfortran 支持 Fortran 77 标准。

4.4 OpenMP 程序的编译与运行

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

- Intel 编译器：
  - **icc -o yourprog-omp -openmp yourprog.c**
    将 OpenMP 的 C 程序 yourprog-omp.c 编译为可执行文件 yourprog-omp。
  - **ifort -o yourprog-omp -openmp 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 -o yourprog-omp -mp yourprog.c**
    将 OpenMP 的 C 程序 yourprog-omp.c 编译为可执行文件 yourprog-omp。
  - **pgf90 -o yourprog-omp -mp 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 -o yourprog-omp -fopenmp yourprog.c**
    将 OpenMP 的 C 程序 yourprog-omp.c 编译为可执行文件 yourprog-omp。
  - **gfortran -o yourprog-omp -fopenmp 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=8 设置八个进程运行。注意，曙光集群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，在提交作业时需要添加 -a openmp -R “span[hosts=1]” 选项以保证在同一个节点运行。

5 MPI 并行程序编译

曙光集群的通信网络为 InfiniBand（建议和默认）和千兆以太网（仅建议在不支持 InfiniBand 网络的情况下使用），系统上安装的 MPI 并行环境为针对 Infiniband 网络的 Open MPI 和 MVAPICH（1 和 2 分支），并且都有分别对应 Intel、PGI、GNU 编译器的版本。系统已经设置默认使用 /usr/mpi/intel/openmpi-1.4.1，用户可以运行 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：

```
Current system default: openmpi_intel-1.4.1
Current user default:    <none>

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

1. mvapich2_gcc-1.4.1
2. mvapich2_intel-1.4.1
3. mvapich2_pgi-1.4.1
4. mvapich_gcc-1.2.0
5. mvapich_intel-1.2.0
6. mvapich_pgi-1.2.0
7. openmpi_gcc-1.4.1
8. openmpi_intel-1.4.1
9. openmpi_pgi-1.4.1
U. Unset default
Q. Quit

Selection (1-9[us], U[us], Q):
```

5.1 MPI 并行程序的编译

Open MPI 和 MVAPICH/MVAPICH2 的编译命令主要为：mpicc、mpic++、mpicxx、mpiCC、mpif77 和 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，请参看编译器手册。

5.2 MPI 并行程序的运行

在曙光集群上，MPI 并行程序需结合作业调度系统 LSF 的作业提交命令 bsub 来运行，基本用法为 **bsub -q normal -o log -e err_file -n 8 mpiibjob ./yourprog-mpi**，注意：必须利用 mpiibjob 或 mpip4job 分别启动使用 InfiniBand 和千兆以太网的 MPI 程序，详细用法将在作业调度部分说明。

5.3 MPI 并行程序调试

集群缺乏专业的并行程序调试工具，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。

6 数学函数库

曙光集群上安装的数学函数库主要有 Intel Math Kernel Library(MKL)、AMD Core Math Library (ACML)以及我们编译的数学函数库（在 /opt/lib 下，注意名称，比如 libgoto_barcelonap-r1.26-intel.a 表示用 Intel 编译器编译的 goto 库），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 11.1.064，安装在 /opt/intel/Compiler/11.1/064/mkl/lib，具有 i386 和 AMD64(em64t) 两个版本，分别对应的目录为 32 和 em64t，以下将以 AMD64 系统为例介绍。在 bash 下可以通过在 ~/.bashrc 之类的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：

```
. /opt/intel/Compiler/11.1/064/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)
- 向量数学库函数(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 Helmholtz 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 的主要目录内容如下见表 5。

表 5: MKL 目录内容

目录	内容
<mkl_dir>	MKL 主目录，比如 /opt/intel/Compiler/11.0/081
<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/32	包含 IA32 架构的静态库和共享目标文件
<mkl_dir>/lib/em64t	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

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：

- 在链接行中，列举含有相对或绝对路径的库名，比如：
  **<ld> myprog.o /opt/intel/Compiler/11.0/081/em64t/libmkl_solver.a **
  **/opt/intel/Compiler/11.0/081/em64t/libmkl_intel.a **
  **/opt/intel/Compiler/11.0/081/em64t/libmkl_intel_thread.a **
  **/opt/intel/Compiler/11.0/081/em64t/libmkl_core.a **
  /opt/intel/Compiler/11.0/081/em64t/libguide.so -lpthread
  其中 <ld> 为链接命令，比如 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 mklvarsem64t.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。

链接 MKL 库时指明库的路径和库名如下：

−L<MKL␣path>␣−I<MKL␣path>
[−lmkl_LAPACK95]␣[−lmkl_blas95]
[cluster␣components]
[{−lmkl_{intel,␣intel_lp64,␣intel_ilp64,␣intel_sp2dp,␣gf,␣gf_lp64,␣gf_ilp64}]
[−lmkl_{intel_thread,␣sequential}]
[{−lmkl_solver,␣−lmkl_solver_lp64,␣−lmkl_solver_ilp64}]
{{[−lmkl_LAPACK]␣−lmkl_{ia32,␣em64t,␣ipf}},
−lmkl_core}}
[{−lguide,␣−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。

6.2 ACML

6.2.1 ACML 简介

AMD Core Math Library (ACML) 是一些数值函数的组合，并且特别针对 AMD64 平台处理器（如 Opteron）等做了优化。ACML 函数库具有 FORTRAN 77 和 C 语言接口，主要包含：

- BLAS - 基本线性代数子系统库，包含级别一的离散基本线性代数(Sparse BLAS)
- LAPACK - 线性代数库
- FFT - 傅立叶变换程序
- RNG - 随机数发生器和统计分布函数。

当前 ACML 安装在 /opt/acml4.4.0，按照编译器等分为几个目录，见下表 6：

表 6: ACML 目录内容

目录	内容
ifort32	针对 64 位 Intel 的编译器，不启用 OpenMP 并行指令。
ifort32_mp	针对多核下 32 位 Intel 的编译器，启用 OpenMP 并行指令。
ifort64	针对 64 位 Intel 的编译器，不启用 OpenMP 并行指令。
ifort64_mp	针对多核下 64 位 Intel 的编译器，启用 OpenMP 并行指令。
pgi32	针对 64 位 PGI 的编译器，不启用 OpenMP 并行指令。
pgi32_mp	针对多核下 32 位 PGI 的编译器，启用 OpenMP 并行指令。
pgi64	针对 64 位 PGI 的编译器，不启用 OpenMP 并行指令。
pgi64_mp	针对多核下 64 位 PGI 的编译器，启用 OpenMP 并行指令。
gfortran32	针对 64 位 gfortran 的编译器，不启用 OpenMP 并行指令。
gfortran32_mp	针对多核下 32 位 gfortran 的编译器，启用 OpenMP 并行指令。
gfortran64	针对 64 位 gfortran 的编译器，不启用 OpenMP 并行指令。
gfortran64_mp	针对多核下 64 位 gfortran 的编译器，启用 OpenMP 并行指令。

6.2.2 编译举例：

- Intel 编译器编译命令：

- ifort driver.f -L/opt/acml4.2.0/ifort64/lib -lacml
- gcc -c -I/opt/acml4.2.0/ifort64/include driver.c
  ifort -nofor-main driver.o -L/opt/acml4.2.0/ifort64/lib -lacml
- ifort -openmp driver.f -L/opt/acml4.2.0/ifort64_mp/lib -lacml_mp
- ifort -openmp driver.f -L/opt/acml4.2.0/ifort32_mp/lib -lacml_mp

- PGI 编译器编译命令：
  - pgf77 -tp=k8-64 -Mcache_align driver.f -L/opt/acml4.2.0/pgi64/lib -lacml
  - pgf77 -tp=k8-32 -Mcache_align driver.f -L/opt/acml4.2.0/pgi32/lib -lacml
  - pgf77 -tp=k8-64 -mp -Mcache_align driver.f -L/opt/acml4.2.0/pgi64_mp/lib -lacml_mp
  - pgf77 -tp=k8-32 -mp -Mcache_align driver.f -L/opt/acml4.2.0/pgi32_mp/lib -lacml_mp
  - pgcc -c -tp=k8-64 -mp -Mcache_align -I/opt/acml4.2.0/pgi64_mp/include driver.c
    pgcc -tp=k8-64 -mp -Mcache_align driver.o -L/opt/acml4.2.0/pgi64_mp/lib -lacml_mp -lpgftnrtl -lm

- GNU 编译器编译命令：
  - gfortran -m64 driver.f -L/opt/acml4.2.0/gfortran64/lib -lacml
  - gfortran -m64 driver.f -L/opt/acml4.2.0/gfortran64/lib -static -lacml
  - gfortran -m64 driver.f /opt/acml4.2.0/gfortran64/lib/libacml.a
  - gfortran -m32 driver.f -L/opt/acml4.2.0/gfortran32/lib -lacml
  - gfortran -fopenmp -m64 driver.f -L/opt/acml4.2.0/gfortran64_mp/lib -lacml_mp
  - gfortran -fopenmp -m32 driver.f -L/opt/acml4.2.0/gfortran32_mp/lib -lacml_mp
  - gcc -m64 -I/opt/acml4.2.0/gfortran64/include driver.c -L/opt/acml4.2.0/gfortran64/lib -lacml -lgfortran

注意：

- 32 和 64 分别表示编译成 32 和 64 位的可执行程序

- 含有 _mp 的表示编译成 OpenMP 并行的可执行程序
- 需要在自己的环境设置文件（如 ~/.bashrc）中设置以下环境变量，以保证运行时能找到此库，其中 COMPILER 为类似 ifort64 的目录：

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/acml4.2.0/COMPILER/lib
```

6.2.3 ACML_MV：快速数学和向量数学库

ACML_MV 是一些常用数学函数的快速和/或向量化的库，比如 sin、cos、exp 等。这些库充分发挥了 AMD64 架构性能优势，现在只有 64 位的 ACML，并且这些函数精度非常高。快速数学库的函数名字，一般以 fast 开头，比如对应 cos 函数的为 fastcos；向量数学函数库一般以 vrd2_、vrd4_、vrda_、vrs4_、vrs8_、vrsa_ 开始，其中 d 表示双精度，s 表示单精度，a 表示数组，2、4、8 分别表示输入参数的数目，例如 vrd4_log 表示具有 4 个双精度输入参数的向量化的 log 函数。具体使用方法，请查看 ACML 手册。

使用这些库的编译命令为类似前面所说的调用 ACML 的命令，如调用针对 Intel 64 位编译器的库，添加 -L/opt/acml4.2.0/ifort64/lib -lacml_mv，注意这里是 acml_mv，不是 acml。

7 作业管理系统

曙光集群利用 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（编译除外），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 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。

7.1.1 提交到特定队列：bsub -q

曙光集群共 38 个 TC2600 刀片节点（node10 - node47，每节点八核，16GB 内存）、五个 A620r-F 节点（io1 - io5，每节点八核，8GB 内存）和两个 A950r-F 节点（node48 和 node49，每节点 32 核，64GB 内存）。利用 -q 选项可以指定提交到哪个队列，现有的队列为[2]：

- normal：所需要的 CPU 核数不超过八个时，作业将在 io1 - io5，node10 - node47 上运行，此为默认队列。
- long：所需要的 CPU 核数超过八个时，作业将在 io1 - io5，node10 - node46 上运行。
- serial：所需要的 CPU 核数为一个时，作业将在 node47 上运行。
- fat：所需要的 CPU 核数超过八个，但不超过 32 个并需共享内存时，作业将只在 node48 上运行。如不是必需使用大于八个核的共享内存程序时，尽量不要使用此队列，以便将资源让给需要大于八个核的共享内存程序的作业。

[2]作业队列会针对运行情况进行修改，详细队列请参看登录后的提示或运行 bqueues -l 命令查看

比如想提交到 normal 队列运行串行程序 executable1，可以：

bsub -q normal 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 <normal>.
```

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。

7.1.2 运行串行队列：bsub -q serial

运行串行作业，请使用串行队列 serial，比如：

bsub -q serial executable-serial

7.1.3 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），比如下面指定利用八个核（由 -n 8 指定）运行 MPI（由 mpiibjob 或 mpip4job 指定）程序：

bsub -q normal -n 8 mpiibjob executable-mpi1

如需要的核数多于八个，需利用 -q long 或 -q fat 指明使用 long 或 fat 队列。

7.1.4 运行 MPI 作业：bsub -n NUM mpiibjob|mpip4job

如果需要运行 MPI 作业，需要利用 mpiibjob 或 mpip4job 分别调用 InfiniBand 或千兆以太网的可执行程序（具体需要使用哪个命令，不能任意决定，需由编译成的可执行文件使用的网络决定，系统配置的默认编译环境为使用 InfiniBand 网络，但二进制发布的非在本系统上编译的可执行程序也许使用千兆以太网），并用 -n 选项指定所需的 CPU 核数，比如下面指定利用 16 颗 CPU 核运行使用 InfiniBand 网络的 MPI 程序 executable-mpi1：

bsub -q long -n 16 mpiibjob executable-mpi1

7.1.5 运行 OpenMP 共享内存作业：bsub -a openmp -R “span[hosts=1]”

集群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 -a openmp -R “span[hosts=1]” 选项：

bsub -a openmp -R “span[hosts=1]” -q normal -n 8 executable-omp1

7.1.6 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：排他性运行在运行期间，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。另外使用排他性运行时，将按照一个节点内的所有 CPU 核数进行收费。

7.1.7 指明输出、输出文件运行：bsub -i -o -e

作业的输入文件、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

7.1.8 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 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
```

7.3 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。

7.4 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.
```

7.5 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.
```

7.6 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.
```

7.7 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 fat 队列，可以：

bmod -Z executable2 -q fat 79727

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

7.8 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  user      2*node31 *executab1 Mar 12 19:20
                                    1*node18
                                    1*node4
79727 hmli  PEND long    user                *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>, User <hmli>, Project <default>, Status <PEND>,
Queue <long> , Command <executab2>
Sun Mar 12 14:15:07: Submitted from host <user.qibebt.ac.cn>,
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。

7.9 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。

7.10 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。

7.11 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。

7.12 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
normal      30  Open:Active  -   8    -    -    22    2   20    0
```

long	30	Open:Active	–	304	–	–	52	12	40	0
fat	30	Open:Active	–	32	–	–	3	0	3	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

7.13 查看用户信息：buser

利用 buser 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 技术支持

青岛生物能源与过程研究所超级计算中心主页为：http://scc.qibebt.cas.cn，此主页包括系统介绍、运行监控以及相关文档等。超级计算中心工作人员也将为用户提供相应的技术支持，如有需要请与我们联系：

- 电话：0532-80662795
- 信箱：
  - 公共：scc@qibebt.ac.cn
  - 李会民：lihm@qibebt.ac.cn、hmli@ustc.edu.cn

由于受水平和时间所限，错误和不妥之处在所难免，欢迎用户指出错误和改进意见，我们将尽力完善。