数学函数库的使用

李会民
hmli@ustc.edu.cn

中国科学技术大学 超级运算中心

2011年10月

Intel MKL

科大超算系统Intel和AMD CPU系统上有Intel MKL：MKL安装后的路径类似下面，请登录具体系统后查看，选择使用：

- */opt/ intel /mkl/10.0.4.023*
- */opt/ intel /Compiler/11.1/073/mkl*
- */opt/ intel /composerxe−2011.3.174/mkl*
- */opt/ intel −10.1.008/mkl/10.2.2.025*

环境设置

设置针对em64t的MKL所需的INCLUDE、LD_LIBRARY_PATH和MANPATH等环境变量:

- bash下可在*~/.bashrc*之类文件中添加类似以下代码之一（与具体版本对应）

```
. /opt/intel/mkl/10.0.4.023/tools/environment/mklvarsem64t.sh
. /opt/intel/Compiler/11.1/073/mkl/tools/environment/mklvarsem64t.sh
. /opt/intel/composerxe/mkl/bin/intel64/mklvars_intel64.sh
```

- csh下可在*~/.login*之类文件中添加以下代码之一（与具体版本对应）

```
. /opt/intel/mkl/10.0.4.023/tools/environment/mklvarsem64t.csh
. /opt/intel/Compiler/11.1/073/mkl/tools/environment/mklvarsem64t.csh
. /opt/intel/composerxe/mkl/bin/intel64/mklvars_intel64.csh
```

注意：.与/之间有个空格

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)

MKL目录内容

目录	内容
<mkl_dir>	MKL主目录，比如/opt/intel/Compiler/11.1/064/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/32	包含IA32架构的静态库和共享目标文件
<mkl_dir>/lib/em64t	旧系列版本MKL路径，包含EM64T架构的静态库和共享目标文件
<mkl_dir>/lib/intel64	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>/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

一般程序编译时，用户主要需要设定的是<mkl_dir>/include、<mkl_dir>/lib/em64t之类，比如-L<mkl_dir>/lib/em64t和-I<mkl_dir>/include。

链接MKL

可以采用两种方式链接：

- 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/11.1/064/mkl/lib/em64t/libmkl_solver.a \
/opt/intel/Compiler/11.1/064/mkl/lib/em64t/libmkl_intel.a \
/opt/intel/Compiler/11.1/064/mkl/lib/em64t/libmkl_intel_thread.a \
/opt/intel/Compiler/11.1/064/mkl/lib/em64t/libmkl_core.a \
/opt/intel/Compiler/11.1/064/lib/intel64/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）

```
< files   to   link>
−L<MKL path> −I<MKL include>
[−I<MKL include>/{32|em64t|{ilp64|lp64}|64/{ 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_lapack]  −lmkl_core

{−liomp5|−lguide} [− lpthread ]  [−lm]
```

- *[]*内的表示可选，|表示其中之一、*{}*表示含有
- 具体链接参数与MKL的具体版本有关，请务必查看对应版本的官方手册，一般在*/opt/ intel /Compiler/11.1/073/Documentation/en_US/mkl*类似目录下

IBM数学函数库

IBM JS22上目前安装的数学函数库主要有：

- 数学加速子系统：Mathematics Acceleration Subsystem(MASS)
- 基本线性代数子系统库：Basic Linear Algebra Subprograms (BLAS)
- 工程与科学子函数库：Engineering and Scientific Subroutine Library(ESSL)
- 并行工程与科学子函数库：Parallel Engineering and Scientific Subroutine Library(PESSL)

上面数学函数库集合了常用的BLAS、LAPACK、ScaLapack、FFT等数学函数库，用户可以直接调用，以提高性能、加快开发。

数学加速子系统：MASS

数学加速子系统（Mathematics Acceleration Subsystem-MASS），是优化的基本数学内部函数，包括sin、exp等函数，具有32和64位版本。编译时，在某些优化级别时会自动调用，或者也可设定编译参数以进行显式调用。

XL C/C++中调用MASS标量库

MASS标量库libmass.a安装在/usr/lib下，包含一些优化过的常用数学内部函数，这些函数在用户使用如下参数编译程序时将自动调用：

- -qhot -qignerrno -qnostrict
- -qhot -O3
- -qsmp -O3
- -O4
- -O5

XL C/C++编译器自动对大多数数学函数调用更快的MASS函数，实际上，编译器首先尝试调用等价的MASS向量库进行向量调用，如果失败，那么将调用标量函数。当编译器实现自动代替数学库函数时，它调用的库包含在系统库libxlopt.a中。调用时，用户无需在源码中添加任何特殊的调用，或者特别指定链接到libxlopt库。

明确指明使用MASS标量函数

如果用户未使用上述的任何优化选项，想明确指定调用MASS标量函数，可以采用下述方法:

- 在源码中包含math.h以提供函数原型（anint、cosisin、dnint、sincos除外）
- 在源码中包含mass.h以提供anint、cosisin、dnint、sincos函数的原型
- 在链接时指定libmass.a

XL C/C++中调用MASS向量库

在用户使用如下参数编译程序时将自动调用MASS向量库:

- -qhot -qignerrno -qnostrict
- -qhot -O3
- -O4
- -O5

XL C/C++编译器会自动尝试通过调用等价的MASS向量函数来向量调用除vdnint、vdint、vsincos、vssincos、vcosisin、vscosisin、vqdrt、vsqdrt、vrqdrt、vsrqdrt、vpopcnt4、vpopcnt8之外的系统数学函数。编译器自动代替数学库函数时，它调用的库包含在系统库libxlopt.a中，用户无需在源码中添加任何特殊的调用，或者特别指定链接到libxlopt库。

明确指明使用MASS向量函数

如果用户没使用上述的优化选项，想显式指定调用的库，可在源文件中包含mass.h以调用下面的库：

- libmassv.a：通用向量库
- libmassvp3.a：某些函数针对POWER3优化，其余等价于libmassv.a中的
- libmassvp4.a：某些函数针对POWER4优化，其余等价于libmassv.a中的
- libmassvp5.a：某些函数针对POWER5优化，其余等价于libmassv.a中的
- libmassvp6.a：某些函数针对POWER6优化，其余等价于libmassv.a中的，建议在JS22上使用

XL C/C++中调用MASS库的编译与链接

为了使程序链接时调用MASS库，请在链接参数中添加mass和massv（或massvp3、massvp4、massvp5、massvp6，针对JS22建议massvp6），比如用如下方式进行编译：

xlc progc.c −o progc −lmass −lmassv

如果用户想对某些函数调用libmass.a标量库，而对其余的函数调用通常的数学库libm.a中的，可按下面流程进行编译链接：

- 生成一包含想调用函数的列表（可为一纯文本文件）。例如在程序sample.c中只想从libmass.a调用快速的正切函数，那么可生成一个文件fasttan.exp，内部只需要有一行：tan。
- 利用*ld*命令链接libmass.a库，生成共享目标文件：
  ld −bexport:fasttan.exp −o fasttan.o −bnoentry −lmass −bmodtype:SR
- 利用*ar*命令打包此共享库：
  ar −q libfasttan.a fasttan.o
- 利用*xlc*生成最终的可执行文件时，在标准数学库libm.a前指明包含MASS函数的目标文件。这将只链接在此目标文件中的函数（在上述例子中为tan函数），其余的将调用标准数学库中的函数：
  xlc sample.c −o sample −Ldir_containing_libfasttan −lfasttan −lm

XL Fortran中调用MASS标量库

MASS标量库libmass.a安装在/usr/lib下，包含一些优化过的常用数学内部函数，这些函数在用户使用如下参数编译程序时将自动调用：

- -qhot -qnostrict
- -qhot -O3
- -qsmp -O3
- -O4
- -O5

XL Fortran编译器自动对大多数数学函数调用更快的MASS函数，实际上，编译器首先尝试调用等价的MASS向量库进行向量调用，如果失败，那么将调用标量函数。当编译器自动代替数学库函数时，它调用的库包含在系统库libxlopt.a中。用户无需在源码中添加任何特殊的调用，或者特别指定链接到libxlopt库。

明确指明使用MASS标量函数

如果用户未使用上述的任何优化选项，想明确指定调用MASS标量函数，可采用下述方法：

- 在用户的应用中调用MASS标量库进行链接
- 所有MASS标量程序，除了那些在下面列出的函数被XL Fortran重组作为内部函数外，无需明确指定接口块。如果需要调用如下函数，需明确指定接口块，并且要在源码中引用mass.include：
  acosf、acosh、acoshf、asinf、asinh、asinhf、atan2f、atan、atanf、atanh、atanhf、cbrt、cbrtf、copysign、copysignf、cosf、coshf、cosisin、erff、erfcf、expf、expm1f、hypot、hypotf、lgammaf、logf、log10f、log1pf、rsqrt、sinf、sincos、sinhf、tanf、tanhf、x**y

XL Fortran中调用MASS向量库

在用户使用如下参数编译程序时将自动调用MASS向量库：

- -qhot -qnostrict
- -qhot -O3
- -O4
- -O5

编译器自动尝试通过调用等价的MASS向量函数来向量调用除vatan2、vsatan2、vdnint、vdint、vsincos、vssincos、vcosisin、vscosisin、vqdrt、vsqdrt、vrqdrt、vsrqdrt、vpopcnt4、vpopcnt8之外的系统数学函数。编译器自动代替数学库函数时，它使用的库包含在系统库libxlopt.a中，用户无需在代码中添加任何特殊的调用，或者特别指明链接到libxlopt库。向量库包含32位和64位的下列库：

- libmassv.a：通用向量库
- libmassvp3.a：某些函数针对POWER3优化，其余等价于libmassv.a中的
- libmassvp4.a：某些函数针对POWER4优化，其余等价于libmassv.a中的
- libmassvp5.a：某些函数针对POWER5优化，其余等价于libmassv.a中的
- libmassvp6.a：某些函数针对POWER6优化，其余等价于libmassv.a中的，建议在JS22上使用

XL Fortran中MASS库的编译与链接

为了使程序调用MASS库，请在链接参数中添加mass和massv（或massvp3、massvp4、massvp5、massvp6，建议针对JS22平台使用massvp6），可用如下方式进行编译：

xlf progf.f −o progf −lmass −lmassv

明确指明使用MASS标量函数

如果用户想对某些函数使用libmass.a变量库，而对其余的使用通常的数学库libm.a，可按下面流程进行编译链接：

- 生成一包含想调用函数的列表（可为一纯文本文件）。例如对sample.f只想从libmass.a调用快速的正切函数，那么可以生成一个文件fasttan.exp，内部只需要有一行：tan
- 利用*ld*命令链接libmass.a库，生成一个共享目标文件：
  ld −bexport:fasttan.exp −o fasttan.o −bnoentry −lmass −bmodtype:SR
- 利用*ar*命令打包此共享库：
  ar −q libfasttan.a fasttan.o
- 利用*xlf*生成最终的可执行文件时，在标准数学库libm.a前指明包含MASS函数的目标文件。这样将只链接在此目标文件中的函数（在此例子中为tan函数），其余的将使用标准数学库中的函数：
  xlf sample.f −o sample −Ldir_containing_libfasttan −lfasttan −lm

基本线性代数库：BLAS

基本线性代数库（Basic Linear Algebra Subprograms-BLAS）由libxlopt库提供，主要包含下面的内容：

- sgemv（单精度）和dgemv（双精度）：计算普通矩阵或其转矩阵的矩阵-向量乘
- sgemm（单精度）和dgemm（双精度）：计算普通矩阵或其转矩阵的乘与加

由于BLAS子程序是用Fortran写的，因此所有参数传递采用的是引用（reference）方式，所有数组是以列优先（column-major）方式存储。
注意：libxlopt中一些出错处理代码已被移除，在调用这些函数的时候不会有出错信息给出。

链接libxlopt库中的BLAS

默认的情况下，当用XL C/C++编译时，libxlopt库自动被链接到应用程序中，但是如果用户使用第三方的BLAS库，并且想利用libxlopt提供的BLAS函数，那么用户必须在其余BLAS库之前指明libxlopt库。例如，如第三方库的名字为libblas，用户可以利用下面命令编译：

xlc app.c –lxlopt –lblas

Fortran调用的时候，需要按照如下方式调用：

xlf app.f –lxlopt –lblas

此时编译器将从libxlopt库中调用sgemv、dgemv、sgemm、dgemm函数，而其余的BLAS函数则从libblas中调用。

工程与科学子函数库：ESSL

工程与科学子函数库（Engineering and Scientific Subroutine Library-ESSL）是一个优秀的子程序集合，包括广大针对不同科学和工程应用方面所需的数学函数，主要特征为高性能、函数兼容性及易用性。
ESSL主要针对以下计算领域对性能进行了优化：

- 线性代数子函数
- 矩阵运算
- 线性方程
- 本征系统分析
- 傅立叶变换、卷积和相关性、相关计算
- 排序与搜索
- 插值
- 数学积分
- 随机数产生

使用ESSL时需要注意

- ESSL的安装目录为/usr/lib，编译的调用参数为-lessl
- 对于SMP形式的ESSL库，可用XL Fortran的XLSMPOPTS或OMP_NUM_THREADS环境变量来影响SMP下的执行
- 如用户需用64位的ESSL，需要在编译的时候添加-q64参数
- ESSL支持XL Fortran的编译时选项-qextname，以在函数后添加 _，避免此类函数找不到
- 利用XL Fortran编译时，-qessl编译参数允许在Fortran 90内部过程中调用ESSL函数
- AIX系统中，ESSL只能采用动态方式链接，不能采用静态方式

C程序调用ESSL的基本编译方式

- C/C++程序的ESSL的头文件为essl.h，安装在/usr/include下
- 除非用户想指明自己定义的复杂数据，否则用户一般无需对现有的C编译过程进行修改
- 如果用户想定义自己的短精度和长精度复数数据，需要在编译参数中分别添加-D_CMPLX或-D_DCMPLX，否则将自动使用ESSL头文件中定义的短精度和长精度复数数据

一般编译方式

ESSL库名		命令
SMP	32-bit	*cc_r −O xyz.c −lesslsmp* *cc_r −O −D_CMPLX −D_DCMPLX xyz.c −lesslsmp*
	64-bit	*cc_r −O −q64 xyz.c −lesslsmp* *cc_r −O −D_CMPLX −D_DCMPLX −q64 xyz.c −lesslsmp*
串行	32-bit	*cc_r −O xyz.c −lessl* *cc_r −O −D_CMPLX −D_DCMPLX xyz.c −lessl*
	64-bit	*cc_r −O −q64 xyz.c −lessl* *cc_r −O −D_CMPLX −D_DCMPLX −q64 xyz.c −lessl*
串行	32-bit	*cc_r −O xyz.c −lessl* *cc_r −O −D_CMPLX −D_DCMPLX xyz.c −lessl*
	64-bit	*cc −O −q64 xyz.c −lessl* *cc −O −D_CMPLX −D_DCMPLX −q64 xyz.c −lessl*

C++程序调用ESSL的基本编译方式

- C/C++程序的PESSL的头文件为essl.h，安装在/usr/include目录下。与C程序不同，C++程序调用ESSL进行编译时必需指定-qnocinc=/usr/include/essl。
- 如果用户使用IBM Open Class Complex Mathematics库，将自动使用ESSL头文件中指定的短精度和长精度复数数据。如果用户想定义自己的短精度和长精度复数数据，需要在编译链接参数中分别添加-D_CMPLX，否则ESSL将用IBM Open Class Complex Mathematics库或标准数学库。
- 如用户想明确指定对复数计算使用标准数学库，需添加编译参数-D_ESV_COMPLEX_。
- EESSL头文件支持两种描述标量输出参数，默认的参数被声明为类型引用（type reference）。如果用户想声明为指针引用，请在编译参数中添加-D_ESVCPTR。

一般编译方式

ESSL库名		命令
SMP	32-bit	*xlC_r −O xyz.C −lesslsmp −qnocinc=/usr/include/essl* *xlC_r −O −D_CMPLX xyz.C −lesslsmp −qnocinc=/usr/include/essl* *xlC_r −O −D_ESV_COMPLEX_ xyz.C −lesslsmp −qnocinc=/usr/include/essl* *xlC_r −O −D_ESVCPTR xyz.C −lesslsmp −qnocinc=/usr/include/essl*
	64-bit	*xlC_r −O −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl* *xlC_r −O −D_CMPLX −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl* *xlC_r −O −D_ESV_COMPLEX_ −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl* *xlC_r −O −D_ESVCPTR −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl*
串行	32-bit	*xlC_r −O xyz.C −lessl −qnocinc=/usr/include/essl* *xlC_r −O −D_CMPLX xyz.C −lessl −qnocinc=/usr/include/essl* *xlC_r −O −D_ESV_COMPLEX_ xyz.C −lessl −qnocinc=/usr/include/essl* *xlC_r −O −D_ESVCPTR xyz.C −lessl −qnocinc=/usr/include/essl*
	64-bit	*xlC_r −O −q64 xyz.C −lessl −qnocinc=/usr/include/essl* *xlC_r −O −D_CMPLX −q64 xyz.C −lessl −qnocinc=/usr/include/essl* *xlC_r −O −D_ESV_COMPLEX_ −q64 xyz.C −lessl −qnocinc=/usr/include/essl* *xlC_r −O −D_ESVCPTR −q64 xyz.C −lessl −qnocinc=/usr/include/essl*
串行	32-bit	*xlC −O xyz.C −lessl −qnocinc=/usr/include/essl* *xlC −O −D_CMPLX xyz.C −lessl −qnocinc=/usr/include/essl* *xlC −O −D_ESV_COMPLEX_ xyz.C −lessl −qnocinc=/usr/include/essl* *xlC −O −D_ESVCPTR xyz.C −lessl −qnocinc=/usr/include/essl*
	64-bit	*xlC −O −q64 xyz.C −lessl −qnocinc=/usr/include/essl* *xlC −O −D_CMPLX −q64 xyz.C −lessl −qnocinc=/usr/include/essl* *xlC −O −D_ESV_COMPLEX_ −q64 xyz.C −lessl −qnocinc=/usr/include/essl* *xlC −O −D_ESVCPTR −q64 xyz.C −lessl −qnocinc=/usr/include/essl*

Fortran程序调用ESSL的基本编译方式

对于Fortran程序，用户无需修改现有的编译方式，只需要添加必要的参数即可

ESSL库名		命令
SMP	32-bit	*xlf_r −O −qnosave xyz.f −lesslsmp*
	64-bit	*xlf_r −O −qnosave −q64 xyz.f −lesslsmp*
串行	32-bit	*xlf_r −O −qnosave xyz.f −lessl*
	64-bit	*xlf_r −O −qnosave −q64 xyz.f −lessl*
串行	32-bit	*xlf_r −O xyz.f −lessl*
	64-bit	*xlf_r −O −q64 xyz.f −lessl*

并行工程与科学子函数库：PESSL

- 并行工程与科学子函数库（Parallel Engineering and Scientific Subroutine Library-PESSL）是可缩放的数学子函数库，支持利用高性能交换机连接的集群上的处理器进行并行处理的应用。PESSL支持在单程序多数据（SPMD）编程模型中采用MPI库。PESSL提供下面方面的通信：
  - 2和3级别的并行基本线性代数子程序（PBLAS）
  - 线性代数方程
  - 本征系统分析和奇异值分析
  - 傅立叶变换
  - 随机数产生
- 对于通信，PESSL包含利用MPI的基本线性代数通信子函数（BLACS），而计算则采用ESSL库，PESSL支持32位和64位的Fortran、C/C++程序的调用。
- 注意PESSL SMP库，是用于在提供并行环境（PE）中的MPI线程库，不能同时在多线程中调用。

使用PESSL时需要注意以下几点

- PESSL的安装目录为/usr/lib，编译的调用参数为-lpessl
- 对于SMP形式的PESSL库，可用环境变量XLSMPOPTS或OMP_NUM_THREADS来影响SMP下的执行
- 如用户需要用64位的PESSL，需要在编译的时候添加-q64参数
- PESSL支持XL Fortran的编译时选项-qextname，以在函数后添加 _，避免此类函数找不到
- ESSL和PESSL是共享库，必须联合使用。具有相同名字的等价子程序（比如libblas.a），尽管在编译参数中代替ESSL库的位置，也将不被使用
- AIX系统中，PESSL只能采用动态方式链接，不能静态链接

C程序调用PESSL的基本编译方式

- C和C++程序的PESSL的头文件为essl.h和Cblacs.h，安装在/usr/include目录下
- 用户一般无需对现有的C编译过程进行修改，除非用户想指明自己定义的复杂数据
- 如果用户想定义自己的短精度和长精度复数数据，需要在编译链接参数中分别添加-D_CMPLX或-D_DCMPLX，否则将自动使用ESSL头文件中定义的短精度和长精度复数数据
- 当链接和运行C程序时，必须设置合适的参数，一般可采用下表中的编译方式

应用模式	命令
32-bit	*mpcc_r −O xyz.c −lesslsmp −lpesslsmp −lblacssmp* *mpcc_r −O −D_CMPLX −D_DCMPLX xyz.c −lesslsmp −lpesslsmp −lblacssmp*
64-bit	*mpcc_r −O −q64 xyz.c −lesslsmp −lpesslsmp −lblacssmp* *mpcc_r −O −q64 −D_CMPLX −D_DCMPLX xyz.c −lesslsmp −lpesslsmp −lblacssmp*

C++程序调用PESSL的基本编译方式

- C和C++程序的PESSL的头文件为essl.h和Cblacs.h，安装在/usr/include目录下
- 与C程序不同，C++程序调用PESSL进行编译时必需要指定-qnocinc=/usr/include/essl参数
- 如果用户使用IBM Open Class Complex Mathematics库，将自动使用ESSL头文件中指定的短精度和长精度复数数据
- 如果用户想定义自己的短精度和长精度复数数据，需要在编译链接参数中分别添加-D_CMPLX，否则ESSL将用IBM Open Class Complex Mathematics库或标准数学库
- 如果想明确指定想对复数计算使用标准数学库，请添加编译参数-D_ESV_COMPLEX_
- EESSL头文件支持两种描述标量输出参数，默认的参数被声明为类型引用（type reference）。如果用户想声明为指针引用，请在编译参数中添加-D_ESVCPTR

编译方式

模式	命令
32-bit	*mpCC_r −O xyz.C $CFLAGS* *mpCC_r −O −D_CMPLX xyz.C $CFLAGS* *mpCC_r −O −D_ESV_COMPLEX_ xyz.C $CFLAGS* *mpCC_r −O −D_ESVCPTR xyz.C $CFLAGS*
64-bit	*mpCC_r −O −q64 xyz.C $CFLAGS* *mpCC_r −O −D_CMPLX −q64 xyz.C $CFLAGS* *mpCC_r −O −D_ESV_COMPLEX_ −q64 xyz.C $CFLAGS* *mpCC_r −O −D_ESVCPTR −q64 xyz.C $CFLAGS*

$CFLAGS=−lesslsmp −lpesslsmp −lblacssmp −qnocinc=/usr/include/pessl

Fortran程序调用PESSL的基本编译方式

- 对于Fortran程序，用户一般无需修改现有的编译方式，但使用64位的Fortran 90离散线性代数方程子程序时，需在编译参数中加：-I/usr/lpp/pessl.rte.common/include/64
- Fortran程序调用PESSL一般采用下表的方式进行编译

应用模式	命令
32-bit	*mpxlf_r −O xyz.f −lesslsmp −lpesslsmp −lblacssmp*
64-bit	*mpxlf_r −O −q64 xyz.f −lesslsmp −lpesslsmp −lblacssm* *mpxlf_r −O −q64 xyz.f −lesslsmp −lpesslsmp * *−lblacssmp −I/usr/lpp/pessl . rte . common/include/64*

联系信息

李会民:

- 办公室：科大东区新科研楼A座网络信息中心二楼205室
- 办公电话：0551-3602248
- 电子信箱：hmli@ustc.edu.cn
- 个人主页：http://hmli.ustc.edu.cn
- 中国科大超算平台：http://scc.ustc.edu.cn