Presentation

Mathematical libraries at the CHPC

Martin Cuma
Center for High Performance Computing
University of Utah
mcuma@chpc.utah.edu

October 27, 2004

http://www.chpc.utah.edu
Overview

• What and what for?
• Basic linear algebra libraries.
• More advanced libraries.
• Commercial library packages.
• Summary.
• scientific codes frequently use basic mathematical operations
• analytic functions (sin, exp,...)
• linear algebra operations (vectors, matrices)
• transforms (Fourier, Laplace,...)
• no need to code this again
• often, user code is not too efficient
Basic problem

- take e.g. matrix multiply

\[
\begin{align*}
\text{DO } & \text{ I=1,M} \\
& \text{DO } \text{ J=1,R} \\
& \quad \text{DO } \text{ K=1,N} \\
& \quad \quad \text{C(I,J) = A(I,K) * B(K,J)} \\
& \quad \text{ENDDO} \\
& \text{ENDDO} \\
& \text{ENDDO}
\end{align*}
\]

- it will most likely not approach the peak performance of the computer you are running on

\[C = AB\]
Why to optimize

- Various ways to speed-up calculation
- memory caching
- prefetching logic
- register reuse
- parallel vector instructions (MMX, SSE, 3DNow)
• Basic Linear Algebra Subroutines
• Level 1 – vector-vector operations
  ▪ CAXPY(N,A,X,XMAX,Y,YMAX)
• Level 2 – matrix-vector operations
  ▪ ZGEMV(‘N’,M,1.,A,MMAX,X,XMAX,0.,Y,YMAX)
• Level 3 – matrix-matrix operations
  ▪ DGEMM(‘N’,‘N’,M,R,N,1.,A,MMAX,B,MMAX,0.,C,MMAX)
• More information
• http://netlib2.cs.utk.edu/blas/
How to decode BLAS names

• First letter – precision (S,D,C,Z)
• Level 1 – next letters – operation
e.g. AXPY - constant times vector plus vector
• Level 2, 3
  ▪ 2\textsuperscript{nd} and 3\textsuperscript{rd} letters – type of vector
e.g. GE - general matrix
  ▪ following letters – operation
e.g. MV – matrix-vector product
BLAS at the CHPC

• Compaq ES40 - Sierra
  ▪ part of CXML library

(http://h18000.www1.hp.com/math/documentation/cxml/dxml.3dxml.html)

`f90 -lcxml source_name.f -o executable_name`
• **ACML (AMD Core Math Library)**
  
  **separate builds for major compilers**

  - pgf90  
    
    ```bash
    -L/uufs/$CLUSTER/sys/pkg/acml/std/sgi32/lib -lacml
    source_name.f -o executable_name
    ```

  - g77  
    
    ```bash
    -L/uufs/$CLUSTER/sys/pkg/acml/std-gnu32/lib -lacml
    source_name.f -o executable_name
    ```

  - pathf90  
    
    ```bash
    -L/uufs/$CLUSTER/sys/pkg/acml/std_pscale/lib -lacml
    source_name.f -o executable_name
    ```

• **ATLAS**

  **only GNU build**

  ```bash
  g77  
  
  ```bash
  ```bash
    -L/uufs/$CLUSTER/sys/pkg/atlas/lib -lf77blas -latlas
    source_name.f -o executable_name
  ```
BLAS at the CHPC – Arches

• **ATLAS**

  g77 -L/uufs/$CLUSTER/sys/pkg/atlas/lib -lf77blas -latlas source_name.f -o executable_name

  pathf90 -L/uufs/$CLUSTER/sys/pkg/atlas/lib -lpathf90blas -latlas source_name.f -o executable_name

  pgf90 -L/uufs/$CLUSTER/sys/pkg/atlas/lib -lpgf90blas -latlas source_name.f -o executable_name

• **ACML (AMD Core Math Library)**

  pgf90 -L/uufs/$CLUSTER/sys/pkg/acml/lib -lacml source_name.f -o executable_name

• **GOTO**

  ▪ **optimized GEMM**

  g77 source_name.f -o executable_name -Wl,-rpath=/uufs/$CLUSTER/sys/pkg/goto/std/lib /uufs/$CLUSTER/sys/pkg/goto/std/lib/libgoto.so

• **PGI BLAS - unoptimized**

  pgf90 source_name.f -o executable_name -lblas
PARAMETER ( MMAX = 10, NMAX = 10, KMAX = 10)
REAL A(MMAX,NMAX)
REAL B(NMAX,KMAX)
REAL C(MMAX,KMAX)
REAL X(MMAX), Y(NMAX)
INTEGER M,N,K

M = 2; N = 4; K = 3
CALL SGEMM('N', 'N', M, N, K, 1., A, MMAX, B, NMAX, 0., C, MMAX)
CALL SGEMV('N',M,N,1.,A,MMAX,X,MMAX,0.,Y,NMAX)

\[ C = AB \]
\[ \bar{y} = A\bar{x}^T \]

In C, must use CBLAS – C bindings for BLAS
For gcc, also must link fortran to C transform library:
gcc matmul_l.c -L/home/mcuma/linal/CBLAS/lib/LINUX -lcblas -lblas -L/usr/lib/gcc-lib/i386-redhat-linux/2.96 -lg2c -o matmul_l
Two components

- **MPI-BLACS – communication**
  (http://netlib2.cs.utk.edu/blacs/)

- **PBLAS – parallel BLAS**
  (http://netlib2.cs.utk.edu/scalapack/html/pblas_qref.html)

- **part of ScaLAPACK distribution**

- **Procedures to use PBLAS**
  - Initialize BLACS and set up processor mapping
  - Build local vector/matrix descriptions
  - Call PBLAS routine
• Compaq ES40 - Sierra
  ▪ CXML threaded library

  f90 -lcxmlp source_name.f -o executable_name
PARAMETER ( MMAX = 4)
PARAMETER ( LMAX = 2)
PARAMETER ( MBLOC = 2)
DOUBLE PRECISION A_loc(LMAX,LMAX)
DOUBLE PRECISION B_loc(LMAX,LMAX)
DOUBLE PRECISION C_loc(LMAX,LMAX)
C Matrix descriptors
INTEGER*4 desc_a(DESC_DIM), desc_b(DESC_DIM), desc_c(DESC_DIM)

CALL BLACS_PINFO( iam, nnodes )
C Get the default system context
CALL BLACS_GET(0,0,icontext)
C Take available processes and map them on the BLACS grid
CALL BLACS_GRIDINIT( icontext, 'R', 2, 2)
C Get the local process coordinates
C context, total number of rows, columns, this process's row, col.
CALL BLACS_GRIDINFO( icontext, npr, npc, myrow, mycol)

C global matrix dimensions
M = MMAX; N = MMAX; R = MMAX
C blocs used to distribute the matrix onto the nodes
   M_bloc = MBLOC;   N_bloc = MBLOC;   R_bloc = MBLOC
C local matrix dimensions
   M_loc = \text{NUMROC}(M,M_bloc,myrow,0,npr)
   N_loc = \text{NUMROC}(N,N_bloc,mycol,0,npc)
   R_loc = \text{NUMROC}(R,R_bloc,mycol,0,npc)
C Build the description vectors for the matrices
   desc_a(1) = 1                           ! Descriptor type
   desc_a(2) = icontext                    ! BLACS context
   desc_a(3) = M                           ! Global first dimension
   desc_a(4) = N                           ! Global second dimension
   desc_a(5) = M_bloc                      ! Blocking factor 1st dimension
   desc_a(6) = N_bloc                      ! Blocking factor 2nd dimension
   desc_a(7) = 0                           ! Process of first matrix row
   desc_a(8) = 0                           ! Process of first matrix column
   desc_a(9) = \text{NUMROC}(M,M_bloc,myrow,0,npr)  ! Leading dimension of local matrix
C The same for matrix B and C
...
C Call the parallel matrix multiply
   CALL PDGEMM('N', 'N', M, R, N, 1.d0, A_loc, ia, ja, desc_a,
*          B_loc, ib, jb, desc_b, 0.d0, C_loc, ic, jc, desc_c)
Higher level libraries

- LAPACK/ScaLAPACK
  - linear algebra problem solvers
- PETSc
  - PDE and ODE solvers
- NAG
  - commercial package, contains almost everything
- FFTW
  - FFT freeware library (Icebox)
• systems of linear equations, eigenvalue problems, ...

http://www.netlib.org/lapack/

Compilation on Arches e.g.

pgf77 -L/uufs/$CLUSTER/sys/pkg/lapack/std/lib
   -llapack -L/uufs/$CLUSTER/sys/pkg/atlas/std/lib
   -lf77blas -latlas source_name.f -o executable_name

• installed on Icebox, Arches

• ATLAS also contains some LAPACK routines
  use routines from ATLAS if possible

• CXML(Sierra) contains most of LAPACK routines
How to decode LAPACK names

• XYYZZZ name format
• X – precision (S,D,C,Z)
• YY – type of matrix
e.g. ge – general, he – Hermitian, ...
• ZZZ – computation to be performed
e.g. trf – LU factorization with partial pivoting
tri – inverse of matrix
sv – system of linear equations AX=B
ev – eigenvalues and eigenvectors
• parallel distributed version of LAPACK
  ▪ MPI_BLACS for communication
  ▪ PBLAS for BLAS

http://www.netlib.org/scalapack/

▪ installed on Icebox, Arches
▪ linking must include paths to all libraries – better to use Makefile based on the testing routines from the distribution
• **Icebox, Arches – AMDScalapack**

/uufs/$CLUSTER/sys/pkg/amdscalapack/std

- **for Makefile, see**

/uufs/$CLUSTER/sys/pkg/amdscalapack/std/gnu32/scalapack_examples/GNUmakefile
• Portable, Extensible Toolkit for Scientific Computation

http://www-fp.mcs.anl.gov/petsc/

- mostly used for ODE and PDE problems
- parallelized using MPI
- extensive set of vector and matrix operations and data types
- written in C++ using OOP

• Installed on Icebox
PETSc – how to compile

**Linking process**

1. **set environmental variables**
   
   ```
   setenv PETSC_DIR /uufs/$CLUSTER/sys/pkg/std/petsc
   setenv PETSC_ARCH `$/PETSC_DIR/bin/petscarch`
   ```

2. **choose a suitable example from** `$PETSC_DIR/src`
   
   modify it and compile as
   
   ```
   make BOPT=g exnn
   ```

3. **if serial, just run the executable, if parallel, use** `mpirun`
   
   ```
   /uufs/$CLUSTER/sys/pkg/mpich/std/bin/mpirun -np 4 -machinefile $PBS_NODEFILE ./ex20
   ```

Parallel version a bit more complicated – must use mapping functions to distribute vectors/matrices across the nodes
• Numerical Algorithms Group library
  http://www.nag.com/
  ▪ separate libraries for C, Fortran
  ▪ serial and parallel (SMP and MPI) versions
• Installed on Icebox, Arches
• Full math library
  ▪ Complex arithmetic, Fourier transforms
  ▪ Statistics, function approximations
  ▪ Quadrature, ODEs
  ▪ Interpolation, curve fitting, maxima and minima of fcts.
  ▪ Linear algebra (LAPACK + more)
• **Linking process**

1. set licence key environmental variable
   
   ```
   setenv LM_LICENSE_FILE
   /uufs/$CLUSTER/sys/pkg/NAG/std/license.dat
   ```

2. choose library to link and run compiler/linker
   
   ```
   pgf77 -L/uufs/$CLUSTER/sys/pkg/NAG/std/fl/pgf77
       -fast source.f -lnag -o executable.exe
   ```

   - in C/C++, must include header file path `-l/....`
   - there are separate Fortran libraries for g77 and pg77
   - more info on the CHPC webpage
• Fastest Fourier Transform in the West
  (http://www.fftw.org)
  • one and multi-dimensional FFT routines
  • first – create plan – selects the fastest way to
do FFT for the particular problem (data size)
  • then perform FFT as many times as we like
  • finally – destroy the plan
• Installed on Icebox
• On Sierra, use FFT routines in CXML
• Compilation:
  
  pgcc/pathcc/gcc  -L/uufs/$CLUSTER/sys/pkg/fftw/std/lib/
  -lfftw source.c  -o executable
  
  pgf90/pathf90/g77  -L/uufs/$CLUSTER/sys/pkg/fftw/std/lib/
  -lfftw source.c  -o executable
C     .. Array Arguments ..
complex*16 f(np1,np2,*)
C     .. FFT grid size ..
integer n1, n2, n3
integer plan

if (mode .eq. 1) then
C     ..  forward transform
    call fftw3d_f77_create_plan_(plan,n1,n2,n3,FFTW_FORWARD,
                                    FFTW_ESTIMATE + FFTW_IN_PLACE)
    call fftwnd_f77_one_(plan, f, 0)
    call fftwnd_f77_destroy_plan_(plan)
else
C     ..  back transform
    call fftw3d_f77_create_plan_(plan,n1,n2,n3,FFTW_BACKWARD,
                                    FFTW_ESTIMATE + FFTW_IN_PLACE)
    call fftwnd_f77_one_(plan, f, 0)
    call fftwnd_f77_destroy_plan_(plan)
endif
Summary

- **BLAS** – Basic Linear Algebra Subroutines
  - 3 levels, vector-vector, vector-matrix, matrix-matrix
- **LAPACK/ScaLAPACK**
  - Linear algebra problem solver library
- **PETSc**
  - ODE and PDE solver library in C++
- **NAG**
  - Numerical Algorithms Group set of libraries
- **FFTW**
  - Freeware FFT library
Some useful resources

• CHPC Math Lib webpage
  http://www.chpc.utah.edu/software/docs/mat_lib.html
  http://www.chpc.utah.edu/short_courses/MatLib