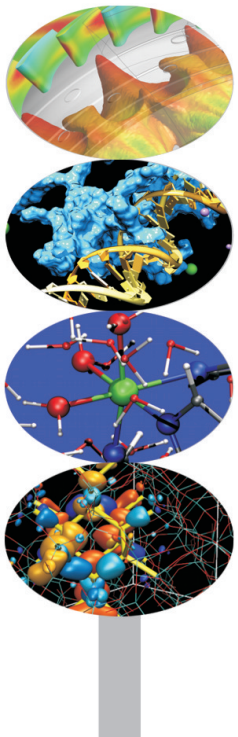
CINECA SCAI
SuperComputing Applications and Innovation

SAPIENZA
UNIVERSITÀ DI ROMA

Programmazione Avanzata

Francesco Salvadore

CINECA Roma - SCAI Department



Roma, 2015

CINECA

Outline

Compilatori e ottimizzazione

Librerie scientifiche

- Introduzione
- Algebra lineare
- Discrete Fourier Transform

Outline

Librerie scientifiche

- Cosa sono?
  - collezioni di procedure
  - finalizzate alla risoluzione di un problema specifico o ad eseguire una particolare operazione
  - forniscono all'utente pezzi di codici da riutilizzare
- Vengono usate spesso (anche inconsapevolmente)
  - In C quando fate **`#include <stdio.h>`**
  - In Matlab quando fate un prodotto tra matrici **`A*B`**

Librerie: vantaggi

- Semplificano il lavoro
- Migliorano la modularità
- Riducono il tempo di codifica
- In genere i sottoprogrammi sono:
  - corretti
  - robusti
  - efficienti
- Consentono di coniugare portabilità ed efficienza
- Sono pronte all'uso
- Sono realizzate da esperti del settore

Librerie: svantaggi

- Non eliminano il problema della scelta dell'algoritmo
- Bisogna sempre verificare che la libreria sia:
  - corretta
  - robusta
  - efficiente
- Essendo alcune librerie "chiuse" possono non essere modificabili
- Possono presentare comunque "bachi" di programmazione
- Leggere sempre attentamente la documentazione
- Fate attenzione: la soluzione di un problema generale non è in genere ottimale per un problema particolare

Librerie statiche e dinamiche

- Una libreria può essere statica o dinamica
  - entrambe sono richieste in fase di compilazione aggiungendo **`-L<directory_libreria> -l<nome_libreria>`**
- Libreria statica:
  - estensione **`.a`**
  - tutti i simboli oggetti della libreria vengono inclusi nell'eseguibile al momento del linking
  - se si crea una libreria che si appoggia ad un'altra non include i suoi simboli: l'eseguibile dovrà linkare tutte le librerie in cascata
  - eseguibile più efficiente
- Libreria dinamica:
  - estensione **`.so`**
  - richiede di specificare la directory in cui cercare la librerie in fase di esecuzione (per esempio settando la variabile di ambiente **`LD_LIBRARY_PATH`**)
  - **`ldd <nome_eseguibile>`** dice le librerie dinamiche richieste dall'eseguibile
  - eseguibile più snello

Panoramica

- Difficile avere una panoramica completa
  - tante tipologie
  - e uno scenario in continua evoluzione
  - anche per le nuove architetture (GPU, MIC, ARM)

- Tipologie di uso comune
  - algebra lineare
  - fft
  - input-output
  - calcolo parallelo
  - mesh decomposition
  - suite

Outline

Algebra lineare

- Per applicazioni massive è cruciale il tipo di parallelizzazione
  - alcune sono già fornite multi-threaded o anche parallele a memoria distribuita
- Memoria condivisa
  - BLAS
  - GOTOBLAS
  - LAPACK/CLAPACK/LAPACK++
  - ATLAS
  - PLASMA
  - SuiteSparse
- Memoria distribuita
  - Blacs (solo suddivisione)
  - ScaLAPACK
  - PSBLAS
  - Elemental

BLAS ...

- **B**asic **L**inear **A**lgebra **S**olver
- Libreria per operazioni di algebra lineare di base (es. prodotto di matrici e vettori)
- Le BLAS sono divivise in tre livelli:
  - Livello 1:(1978) operazioni tra vettori tipo: $y \leftarrow \alpha x + y$, ma anche prodotti scalari e norme ($O(n)$)
  - Livello 2: (1985) operazioni tra vettori e matrici tipo: $y \leftarrow \alpha Ax + \beta y$, ma anche soluzione di sistemi triangolari ($O(n^2)$)
  - Livello 3: (1987) operazioni tra matrici tipo: $C \leftarrow \alpha AB + \beta C$ ($O(n^3)$)
- Scritta in Fortran
- Standard de-facto per operazioni tra matrici

BLAS / 2

- Le subroutine BLAS si applicano a dati reali e complessi, in semplice o doppia precisione
- Operazioni scalare-vettore (O(n))
  - SWAP scambio vettori
  - COPY copia vettori
  - SCAL cambio fattore di scala
  - NRM2 norma L2
  - AXPY somma: Y + A*X
- Operazioni vettore-matrice ($O(n^2)$)
  - GEMV prodotto vettore/matrice generica
  - HEMV prodotto vettore/matrice hermitiana
  - SYMV prodotto vettore/matrice simmetrica
- Operazioni matrice-matrice ($O(n^3)$)
  - GEMM prodotto matrice/matrice generica
  - HEMM prodotto matrice/matrice hermitiana
  - SYMM prodotto matrice/matrice simmetrica

BLAS level

S Real, D Double Precision, C Complex, Z Double Complex

- Level 1
  - xAXPY: $y \leftarrow \alpha x + y$
  - xDOT: $dot \leftarrow x^T \cdot y$
  - xNRM2: $nrm2 \leftarrow \|x\|_2$
  - xASUM: $asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$
  - IxAMAX: $amax \leftarrow 1^{st} k \mid |re(x_k)| + |im(x_k)| = \max(|re(x_k)| + |im(x_k)|)$
  - ...
- Level 2
  - xGEMV: $y \leftarrow \alpha Ax + \beta y$
  - xTRSV: $y \leftarrow A^{-1}x$
  - xGER: $y \leftarrow \alpha x \cdot y^T + A$
  - ...
- Level 3
  - xGEMM: $y \leftarrow \alpha op(A)op(B) + \beta C$
  - xTRSM: $B \leftarrow \alpha op(A^{-1})B$
  - ...

BLAS / 3

- Operazioni matrice-matrice ($O(n^3)$)
  - GEMM prodotto matrice/matrice generica
  - HEMM prodotto matrice/matrice hermitiana
  - SYMM prodotto matrice/matrice simmetrica

- GOTOBLAS
  - Kazushige Goto, un ricercatore del Texas Advanced Computing Center (University of Texas at Austin), ha ottimizzato manualmente in assembler le subroutine BLAS per diversi supercomputer

DGEMM


```
SUBROUTINE DGEMM(TRANSA,TRANSB,M,N,K,ALPHA,A,LDA,B,LDB,BETA,C,LDC)
*    .. Scalar Arguments ..
     DOUBLE PRECISION ALPHA,BETA
     INTEGER K,LDA,LDB,LDC,M,N
     CHARACTER TRANSA,TRANSB
*    .. Array Arguments ..
     DOUBLE PRECISION A(LDA,*),B(LDB,*),C(LDC,*)
*
* Purpose
*
* DGEMM  performs one of the matrix-matrix operations
*        C := alpha*op( A )*op( B ) + beta*C,
*        where  op( X ) is one of op( X ) = X   or   op( X ) = X'
...
```

DGEMM arguments


```
*  Arguments
*
*  TRANSA - CHARACTER*1.
*           On entry, TRANSA specifies the form of op( A )
*           to be used in the matrix multiplication as follows:
*
*              TRANSA = 'N' or 'n',  op( A ) = A.
*
*              TRANSA = 'T' or 't',  op( A ) = A'.
*
*              TRANSA = 'C' or 'c',  op( A ) = A'.
*  M  - INTEGER.
*       On entry,  M  specifies  the number  of rows  of the
*       matrix op( A )  and of the  matrix  C.  M  must  be at
*       least  zero.
*  N  - INTEGER.
*       On entry,  N  specifies the number  of columns of the
*       matrix op( B ) and the number of columns of the matrix C.
*       N must be at least zero.
*  K  - INTEGER.
*       On entry,  K  specifies  the number of columns of the
*       matrix op( A ) and the number of rows of the matrix
*       op( B ). K must be at least  zero.
...
```

DGEMM errori tipici

- Tipi non coerenti

```
CALL DGEMM('N','N',n,n,n,1.0,A,n,B,n,0.0,C,n)

CALL SGEMM('N','N',n,n,n,1.0,A,n,B,n,0.0,C,n)
CALL DGEMM('N','N',n,n,n,1.d0,A,n,B,n,0.d0,C,n)
```

- Aree di memoria:

```
Parameter (pad=9)
Real*8 A(n+pad,n),B(n+pad,n), C(n+pad,n)
CALL DGEMM('N,'N',n,n,n,1.d0,A,n,B,n,0.d0,C,n)

CALL DGEMM('N,'N',n,n,n,1.d0,A,n+pad,B,n+pad,0.d0,C,n+pad)
```

BLAS e non solo. distribuzioni

- **ACML** AMD Core Math Library
  Include tra l'altro BLAS, LAPACK, FFT, Random Generators
- **ATLAS** Automatically Tuned Linear Algebra Software
- **CUDA SDK** NVIDIA CUDA include BLAS on GPU
- **ESSL** Engineering Scientific Subroutine Library
  - BLAS, LAPACK, ScaLAPACK, solutori sparsi, FFT e altro, anche MPI.
- **Goto BLAS** Kazushige Goto's implementation of BLAS
- **Intel MKL** The Intel Math Kernel Library
  - include Linear Algebra, FFT, Vector Math and Statistics, anche MPI
- **uBLAS** A generic C++ template class library providing BLAS functionality (part of the Boost library)
- **GSL** The GNU Scientific Library
  - wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total.
- ...

BLAS from FORTRAN

- La BLAS sono scritte in Fortran 77
- Basta chiamarle

```
call DGEMM('N','N',n,n,n,1.d0,a,n+pad,b,n+pad,0.d0,d,n+pad)
```

- E linkarle (**`-LBLAS_PATH -lblas_name`**)
- Distribuzioni specifiche per l'architettura saranno in generale più performanti di quelle compilate a mano.

BLAS from C/C++

- Attenzione, chiamando direttamente `dgemm_` otteremo il prodotto delle trasposte
- meglio usare l'interfaccia `C` alle **blas**, **cblas**

```
...
double a[nn][nn+npad];
double b[nn][nn+npad];
double c[nn][nn+npad];
...
 cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
      n, n, n, one, (double*)a, np, (double*)b, np,
      zero, (double*)c, np);
...
```

- In `C++` usiamo **ublas**, o se usiamo **cblas**, facciamo attenzione a includerla

```
extern "C" { #include "cblas.h"}
```

ATLAS

- ATLAS: Automatic Tuned Linear Algebra Solver
- BLAS e alcune funzioni di LAPACK
  - LU decomposition
  - Cholesky decomposition
- Open source: `http://math-atlas.sourceforge.net/`
- Libreria orientata alle prestazioni
  - adattività
  - cache oblivious
  - AEOS: Automated Empirical Optimization of software

Installare ATLAS

Una volta scaricata:

```
tar -jxvf atlas3.8.3.tar.bz2
cd ATLAS
mkdir atlas3.8.3
cd atlas3.8.3/

../configure -b 32
--prefix=/home/marco/ATLAS/atlas3.8.3
make build
make time
make install
```

se tutto va bene

```
gfortran myprogram.f
-L/home/marco/ATLAS/atlas3.8.3/lib -lf77blas
-latlas
```

ATLAS output


```
cat bin/INSTALL_LOG/SUMMARY.LOG |more
...
IN STAGE 1 INSTALL:  SYSTEM PROBE/AUX COMPILE
  Level 1 cache size calculated as 32KB.

  dFPU: Separate multiply and add instructions with 4 cycle pipeline.
        Apparent number of registers : 9
        Register-register performance=3390.52MFLOPS
  sFPU: Separate multiply and add instructions with 4 cycle pipeline.
        Apparent number of registers : 9
        Register-register performance=4284.25MFLOPS
...
```

ATLAS output


```
IN STAGE 2 INSTALL:  TYPE-DEPENDENT TUNING
STAGE 2-1: TUNING PREC='d' (precision 1 of 4)
STAGE 2-1-1 : BUILDING BLOCK MATMUL TUNE
 The best matmul kernel was ATL_dmm2x2x128_sse2.c, NB=52,
 written by R. Clint Whaley
 Performance: 7222.74MFLOPS (360.96 percent of of detected clock rate)
  (Gen case got 3115.69MFLOPS)
mmNN    : ma=1, lat=4, nb=24, mu=6, nu=1 ku=4, ff=0, if=6, nf=1
          Performance = 2724.34 (37.72 of copy matmul, 136.15 of clock)
mmNT    : ma=1, lat=8, nb=24, mu=6, nu=1 ku=24, ff=0, if=6, nf=1
          Performance = 2605.24 (36.07 of copy matmul, 130.20 of clock)
mmTN    : ma=1, lat=2, nb=24, mu=6, nu=1 ku=24, ff=0, if=6, nf=1
          Performance = 2693.56 (37.29 of copy matmul, 134.61 of clock)
```

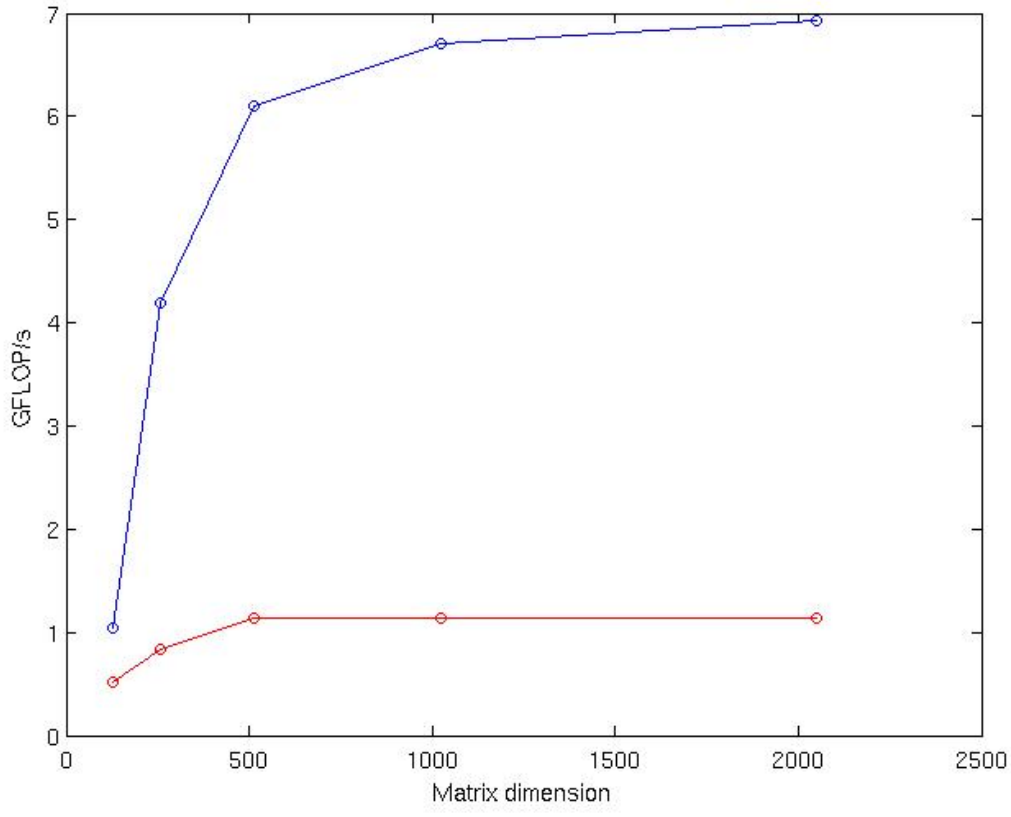
ATLAS output


```
...
Reference clock rate=2394Mhz, new rate=2001Mhz
 Refrenc : % of clock rate achieved by reference install
 Present : % of clock rate achieved by present ATLAS install
```

	single precision				double precision			
	real		complex		real		complex	
Benchmark	Refrenc	Present	Refrenc	Present	Refrenc	Present	Refrenc	Present
kSelMM	567.0	602.0	527.5	568.0	334.0	361.0	321.0	340.3
kGenMM	162.8	138.1	164.5	176.0	155.5	155.7	154.4	164.3
kMM_NT	135.8	147.6	132.7	151.2	104.6	130.2	115.7	136.9
kMM_TN	159.6	172.0	123.4	125.9	123.3	134.6	127.1	140.2

ATLAS DGEMM Performance on Core 2 Duo @ 2GHz


```
gfortran -fomit-frame-pointer -mfpmath=sse -msse3
-O2 -m32 mmatlas.f -o dgemmatlas
-L/home/marco/ATLAS/atlas3.8.3/lib -lf77blas
-latlas
```



Intel MKL

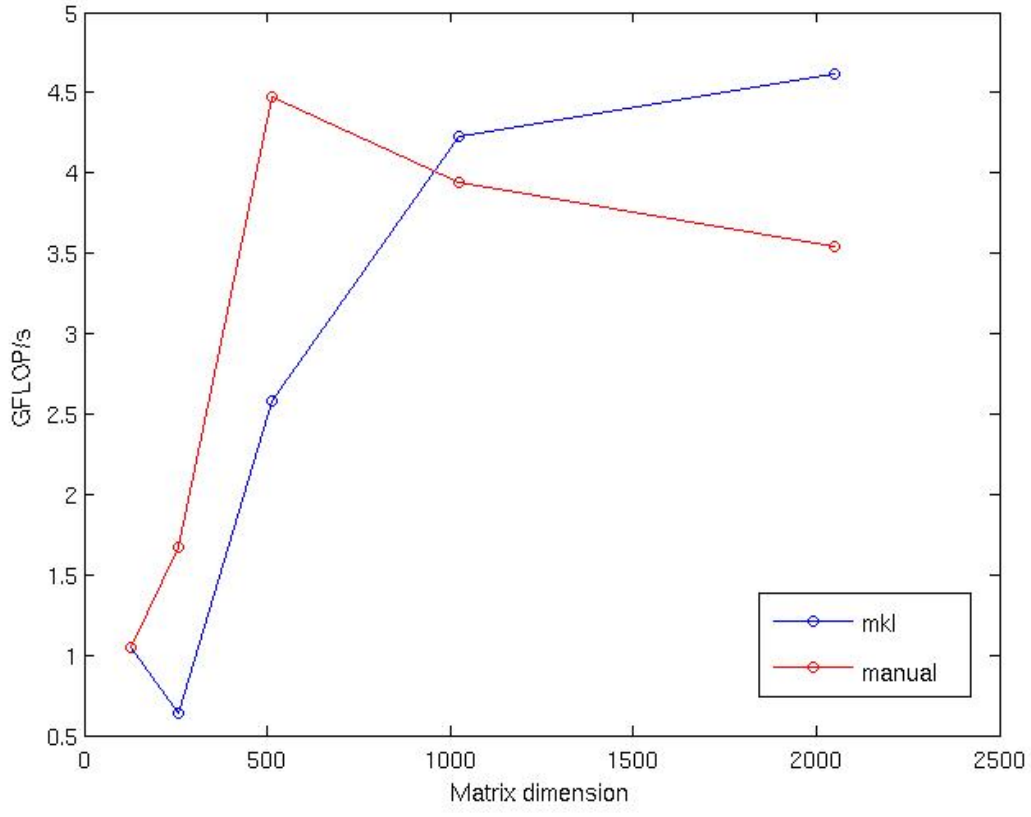
- MKL: Math Kernel Library
- BLAS e LAPACK e
  - scalaPACK
  - Sparse Solvers
  - Fast Fourier Transform
  - Vector Math Library
  - Vector Random Generator
  - Linpack Benchmark
  - ...

MKL layered model

- Interface layer
  ad es. `libmkl_intel` per intel e `libmkl_gf` per GNU
- Threading layer
  ad es. `libmkl_intel_thread`, `libmkl_gnu_thread`, `libmkl_sequential`
- Computational layer
  comprende la parte computazionale che dipende solo dal tipo di architettura, (32/64 bit): `libmkl_core`
- Compiler Support RTL layer: `libiomp5`

MKL DGEMM Performance on Core 2 Duo @ 2 GHz


```
ifort -fast mmmkl.f $MKL/libmkl_intel.a
$MKL/libmkl_intel_thread.a $MKL/libmkl_core.a -L$MKL -liomp5
-lpthread
```



Linpack

- Libreria per la soluzione di sistemi di algebra lineare
- Una delle prime librerie per il calcolo scientifico
- Scritta da J. Dongarra, C. Moler e altri
- Nata per i primi supercalcolatori
- Basata su BLAS per le operazioni tra vettori e matrici
- Oramai sostituita da LAPACK, ma si trova ancora su alcuni codici
- Usata per stilare la top500 (`www.top500.org`)
- `www.netlib.org/linpack`

LAPACK & Co.

- LAPACK: Linear Algebra PACKage
  - evoluzione di LINPACK e EISPACK
  - soluzione di problemi di algebra lineare, tra cui sistemi di equazioni lineari, problemi di minimi quadrati, autovalori
- ATLAS: Automatically Tuned Linear Algebra Software
  - implementazione BLAS e di alcune routine di LAPACK efficiente grazie alla procedura di autotuning che avviene durante l'installazione
- PLASMA: Parallel Linear Algebra Software for Multi-core Architectures
  - soluzione di sistemi lineari, progettate per essere efficienti su processori multi-core, funzionalità simili a LAPACK ma più limitate
- SuiteSparse
  - collezione di pacchetti per matrici sparse

Ancora algebra lineare

- Calcolo di autovalori/autovettori
  - EISPACK: calcolo di autovalori e autovettori, con versioni specializzate per matrici di diversi tipi, reali e complesse, hermitiane, simmetriche, tridiagonali
  - ARPACK: problemi agli autovalori di grandi dimensioni. La versione parallela è un'estensione della libreria classica e usa le librerie BLACS e MPI
- Algebra lineare a memoria distribuita
  - BLACS: linear algebra oriented message passing interface
  - ScaLAPACK: Scalable Linear Algebra PACKage
  - Elemental: framework per algebra lineare densa
  - PSBLAS: Parallel Sparse Basic Linear Algebra Subroutines
  - SLEPc: problemi agli autovalori

LAPACK

- LAPACK: Linear Algebra PACKage
- Algebra di matrici dense ed a banda
- Consente di risolvere:
  - Sistemi lineari (con termini noti multipli)
  - Problemi agli autovalori
  - Singular Value Decomposition
  - Altre fattorizzazioni (LU, Cholesky, QR, SVD, Schur, generalized Schur)
- Le routine che costituiscono LAPACK sono basate a loro volta sulle routine della libreria BLAS
- `http://www.netlib.org/lapack/`
- `http://www.cs.colorado.edu/~jessup/lapack/`

Installare LAPACK

- Scaricare
  `http://www.netlib.org/lapack/lapack.tgz`
- Scompattare l'archivio `tar zxvf lapack.tgz cd lapack3.2`
- Scegliere il `Make.inc` più vicino alla vostra architettura, ad es.
  - `cp INSTALL/make.inc.gfortran make.inc`
- Editate il file `make.inc` ad es.

```
OPTS = -mfpmath=sse -msse3 -O2 -m32
BLASLIB = /opt/lib/atlas3.8.3/lib/libf77blas.a
          /opt/lib/atlas3.8.3/lib/libatlas.a
LAPACKLIB = liblapack-atlas-gfortran.a
```

- Compilare la libreria

```
cd SRC
make
```

LAPACK

Tutte le routine hanno nomi del tipo **XYYZZZ**, dove

- **X** indica il tipo, **S** per REAL, **D** per Double, **C** per Complex e **Z** per Double Complex
- La seconda e terza lettera indicano il tipo di matrice
  - **GE** General
  - **GT** General Tridiagonal
  - **HE** Hermitiana
  - **OR** Ortogonale
  - **SR** Simmetrica a banda
  - **TR** Triangolare
  - ...

LAPACK

Le ultime due o tre lettere indicano il tipo di calcolo, tanto per fare qualche esempio tra le oltre 200 subroutine

- Equazioni lineari
  - **SV** assegna a B la soluzione del sistema $AX = B$
- Minimi quadrati
  - **LS**
- Problemi agli autovalori e decomposizione in valori singolari
  - **EV** $A = Z\Lambda Z'$ con A Hermitiana
  - **ES** $A = ZTZ'$ con T triangolare superiore
  - **SVD** $A = U\Sigma U'$ con Σ diagonale

DGESV


```
      SUBROUTINE DGESV( N, NRHS, A, LDA, IPIV, B, LDB, INFO )
*
*  -- LAPACK driver routine (version 3.2) --
*     Univ. of Tennessee, Univ. of California Berkeley and
*     NAG Ltd..
*     November 2006
*
*     .. Scalar Arguments ..
      INTEGER            INFO, LDA, LDB, N, NRHS
*     ..
*     .. Array Arguments ..
      INTEGER            IPIV( * )
      DOUBLE PRECISION   A( LDA, * ), B( LDB, * )
      ...
```

DGESV


```
*  Purpose
*  =======
*
*  DGESV computes the solution to a real system of linear
*   equations A * X = B, where A is an N-by-N matrix
*   and X and B are N-by-NRHS matrices.
*
*  The LU decomposition with partial pivoting and
*  row interchanges is used to factor A as
*     A = P * L * U,
*  where P is a permutation matrix, L is unit lower triangular,
*  and U is upper triangular.  The factored form of A is then
*  used to solve the system of equations A * X = B.
```

DGESV


```
  Arguments
*  =========
*  N       (input) INTEGER
*          The number of linear equations, i.e., the order of the
*          matrix A.  N >= 0.
*  NRHS    (input) INTEGER
*          The number of right hand sides, i.e., the number of
*          columns of the matrix B.  NRHS >= 0.
*  A       (input/output) DOUBLE PRECISION array, dimension (LDA,N)
*          On entry, the N-by-N coefficient matrix A.
*          On exit, the factors L and U from the factorization
*          A = P*L*U; the unit diagonal elements of L are not
           stored.
```

DGESV


```
*  LDA     (input) INTEGER
*          The leading dimension of the array A.
           LDA >= max(1,N).
*  IPIV    (output) INTEGER array, dimension (N)
*          The pivot indices that define the permutation matrix P;
*          row i of the matrix was interchanged with row IPIV(i).
*  B       (input/output) DOUBLE PRECISION array, dimension
           (LDB,NRHS)
*          On entry, the N-by-NRHS matrix of right hand side
           matrix B.
*          On exit, if INFO = 0, the N-by-NRHS solution matrix X.
*  LDB     (input) INTEGER
*          The leading dimension of the array B.  LDB >= max(1,N).
*  INFO    (output) INTEGER
*          = 0:  successful exit
*          < 0:  if INFO = -i, the i-th argument had an illegal
                 value
*          > 0:  if INFO = i, U(i,i) is exactly zero.
*                The factorization  has been completed, but the
*                factor U is exactly  singular, so the solution
                 could not be computed.
```

Linking LAPACK


```
gfortran  sols.f -o sols
-L/opt/lib/atlas3.8.3/lib  -L/opt/lib/lapack-3.2/
-llapack-atlas-gfortran -lf77blas -latlas
```

```
gcc sols.c -c
gfortran  sols.o -o sols
-L/opt/lib/atlas3.8.3/lib  -L/opt/lib/lapack-3.2/
-llapack-atlas-gfortran -lf77blas -latlas
```

```
gcc sols.c -c -I/opt/lib/lapack-3.2/lapacke/include
gfortran  sols.o -o sols
-L/opt/lib/atlas3.8.3/lib  -L/opt/lib/lapack-3.2/
-llapcke -llapack-atlas-gfortran
-lf77blas -lcblas -latlas
```

Attenzione: l'ordine è importante.

LAPACK

- È una libreria estremamente corretta
- Evoluzione di librerie preesistenti come:
  - LINPACK
  - EISPACK
- È estremamente efficiente se esistono BLAS ottimizzate su cui appogiarsi
- Fornisce diverse subroutine per risolvere lo stesso problema
- Tocca al programmatore segliere quella più efficiente
  - È meglio fattorizzare una matrice con il metodo LU?
  - O è meglio usare Cholesky?

ScaLAPACK

- ScaLAPACK: Scalable Linear Algebra PACKage
- Libreria open-source
- Versione parallela (usando MPI o PVM) della LAPACK seriale
- Le routine che costituiscono ScaLAPACK sono basate a loro volta sulle routine delle librerie:
  - PBLAS
  - BLAS
  - BLACS: Basic Linear Algebra Communications Subprogram. È usata per il trasferimento di dati tra processi e consente la creazione ed il controllo di griglie di processi.
- Se è disponibile un'implementazione di BLAS specifica per la macchina in uso, le routine di ScalaPACK consentono di ottenere alte prestazioni
- Purché la rete di interconnessione sia sufficientemente veloce

Outline

FFT

- Salvo rarissime eccezioni, non conviene riscrivere una libreria FFT
- Ogni vendor fornisce FFT ottimizzate per la propria CPU
  - sono le meno portabili
  - non sempe implementano tutte le possibili trasformate
  - i formati dati nel campo complesso possono essere molto diversi
- Esistono inoltre librerie FFT multipiattaforma
  - Open source
    - FFTPACK
    - FFTW
  - Commerciali
    - NAG

FFT: pre-calcolo dei coeficienti

- Solitamente in un codice si ripetono FFT di sequenze della stessa lunghezza o di poche lunghezze fissate
- Sarebbe oneroso ricalcolare ogni volta i coefficienti
- Tutte le librerie consentono di precalcolare i coefficienti per tutte le FFT di sequenze della stessa lunghezza
- FFTW consente di precalcolare dei “plan” che codificano anche la strategia ottima di calcolo per la lunghezza e per il layout dei dati in memoria
- Preparare le aree di appoggio per ogni lunghezza utilizzata nel programma ed inizializzarle con i coefficienti, seguendo la documentazione della libreria in uso

FFT: accessi in memoria

Alcuni consigli:

- La FFT minimizza i calcoli
- Ma l'accesso in memoria è non locale (per via della "butterfly")
  - non è cache friendly
  - non è vettorizzabile né in software né in hardware
- Calcolando contemporaneamente più FFT della stessa lunghezza è possibile
  - riutilizzare i coefficienti (ridurre le load)
  - utilizzare meglio la cache (o vettorizzare)
- Tutte le librerie danno la possibilità di calcolare le FFT di più trasformate della stezza lunghezza con una sola chiamata
  - su macchine con cache è efficiente solo per lunghezze "corte"
  - su macchine vettoriali è sempre efficiente

FFT 2D e 3D

- FFT di un array bidimensionale
  - FFT di tutte le righe
  - FFT di tutte le colonne
- Uno dei due passaggi può essere molto efficiente:
  - in FORTRAN la trasformata delle righe
  - in C quella delle colonne
- preferire sempre la chiamata di libreria per FFT 2D invece di svolgere singolarmente i due passaggi in 1D
- Essendo tipicamente potenze di 2
  - il cache trashing è in agguato $\Longrightarrow$ ricorrere al padding
- Stesso discorso per le trasformate 3D ...

FFTW

- FFTW: Fastest Fourier Transform in the West
- Libreria FFT
  - Adattività
  - AEOS: Automated Empirical Optimization of Software (come ATLAS)
  - In fase di calcolo dei coefficienti, seguendo una strategia “dividi e conquista” cerca quale sia la migliore strategia di calcolo della FFT, vista l’architettura e la dimensione della FFT
  - È in continuo sviluppo
  - È sviluppata principalmente per architetture Intel, anche se l’ultima versione include il supporto per ARM
  - È parte integrante di software di comunità (CPMD, ...)
- `http://www.fftw.org`

Installare FFTW

Una volta scaricata

```
tar zxvf fftw-3.2.1.tar.gz
cd fftw-3.2.1/
mkdir lib
cd lib
 ../configure --prefix=/opt/lib/fftw-3.2.1
make
make install
```

Se tutto va bene:

```
gfortran myfft.f -L/opt/lib/fftw-3.2.1/lib -lfftw3
```

oppure

```
ifort myfft.f -L/opt/lib/fftw-3.2.1/lib -lfftw3
```

FFTW 1D


```
program fft
  implicit none
  ...
  integer, parameter :: n=5000000     ! size of the matrix
  integer, parameter :: npad= 0       ! padding

  complex(kind(1.d0)) ::    in(1:n+npad), out(1:n+npad)
  integer, parameter  :: fftw_forward=-1
  integer, parameter  :: fftw_estimate=64
  integer(selected_int_kind(18)) :: plan_forward
  ...
  call dfftw_plan_dft_1d ( plan_forward, n, in, out,&
       FFTW_FORWARD, FFTW_ESTIMATE )
  call dfftw_execute ( plan_forward,in ,out )
  ...
  call dfftw_destroy_plan ( plan_forward )
  ...
end program fft
```

FFT 1D


```
call dfftw_plan_dft_1d ( plan_forward, n, in, out,
     &  FFTW_FORWARD, FFTW_ESTIMATE )
```

- Il primo argometo, `n`, è la dimensione del vettore da trasformare
- Il secondo e il terzo sono il vettore di input e quello di output
- `FFTW_FORWARD` (-1) e `FFTW_BACKWARD` (+1) sono il segno dell'esponente nella trasformata, e quindi la direzione della trasformata
- L'ultimo argomento può essere `FFT_MEASURE` (0) che misura i tempi di diverse trasformate per trovare quella ottima di lunghezza `n`, oppure `FFTW_ESTIMATE` che senza calcolare nessuna trasformata ne seglie una, in generale sotto-ottimale. Se dobbiamo calcolare più trasformate della stessa lunghezza useremo `FFT_MEASURE`, altrimenti `FFTW_ESTIMATE`. Attenzione, `in` e `out` vengono riscritti se scegliamo `FFT_MEASURE`

FFT 1D

Una volta calcolato il “plan” può essere riusato quante volte vogliamo per trasformare `in` in `out` tramite

```
call dfftw_execute ( plan_forward , in , out)
```

Alla fine il “plan” può essere deallocato chiamando

```
call dfftw_destroy_plan ( plan_forward , in , out)
```

Attenzione, le trasformate non sono normalizzate, se quindi calcoliamo la trasformata FORWARD e poi quella BACKWARD, dobbiamo dividere per `n` il risultato se vogliamo ottenere il vettore iniziale.

FFTW 1D e 2D

Altre trasformate:

```
call dfftw_plan_dft_r2c_1d ( plan_forward, n, in, out,
      & FFTW_ESTIMATE )

call dfftw_plan_dft_c2r_1d ( plan_backward, n, in, out,
       & FFTW_ESTIMATE )

call dfftw_plan_dft_2d ( plan_forward, nx, ny, in, out,
     &  FFTW_FORWARD, FFTW_ESTIMATE )

call dfftw_plan_dft_r2c_2d ( plan_forward, nx, ny,
       & in, out, FFTW_ESTIMATE )

call dfftw_plan_dft_c2r_2d ( plan_backward, nx, ny,
      & out, in2, FFTW_ESTIMATE )
```

FFTW 3D


```
      double complex arr
      dimension arr(L,M,N)
      integer*8 plan

      call dfftw_plan_dft_3d(plan, L,M,N, arr,arr,
     &                       FFTW_FORWARD, FFTW_ESTIMATE)
      call dfftw_execute_dft(plan, arr, arr)
      call dfftw_destroy_plan(plan)
```

MKL FFT


```
program FFT
  use mkl_dfti
  complex(kind(1.d0)) :: in(1:n+npad), out(1:n+npad)
  type(dfti_descriptor), pointer :: Desc_Handle
  integer ::   status
  real(kind(1.d0)) ::   Scale
  ...
  Status=DftiCreateDescriptor(Desc_Handle,DFTI_DOUBLE,
           &DFTI_COMPLEX,1,n)
  Status=DftiSetValue(Desc_Handle,
              &DFTI_PLACEMENT,DFTI_NOT_INPLACE)
  Status=DftiCommitDescriptor(Desc_Handle)
  Status=DftiComputeForward(Desc_Handle,in,out)
  Scale = 1.0/real(n, KIND=8)
  Status = DftiSetValue(Desc_Handle, DFTI_BACKWARD_SCALE, Scale)
  Status = DftiCommitDescriptor( Desc_Handle )
  Status = DftiComputeBackward( Desc_Handle, out,in)
  ...
  Status = DftiFreeDescriptor(Desc_Handle)
end program fft
```

MKL FFT

- **`Status = DftiCreateDescriptor(Desc_Handle, & Precision, Forward_Domain, Dimension, Length)`**
  Alloca la memoria per la struttura dati del desrittore e lo inizializza con i valori di default
- **`Status = DftiCommitDescriptor( Desc_Handle )`**
  Calcola tutte le inizializzazione che facilitano il calcolo della FFT
- **`Status = DftiFreeDescriptor(Desc_Handle)`**
  Libera la memoria allocata per il descrittore
- **`Status = DftiComputeForward( Desc_Handle, in,out)`** e
  `Status = DftiComputeForward( Desc_Handle, in,out)` calcolano la trasformata FORWARD e BACKWARD rispettivamente

MKL FFT

- **`Status = DftiSetValue(Desc_Handle, Config_Param, Config_Value)`**
  Serve per modificare il valore di alcuni parametri, ad esempio: **`DFTI_NUMBER_OF_TRANSFORM`** (scalar integer), **`DFTI_FORWARD_SCALE`** (floating point scalar) fattore di scala per la trasformata FORWARD, **`DFTI_PLACEMENT`** (Named constant), **`DFTI_INPLACE`** oppure **`DFTI_NOT_INPLACE`** per sovrascrivere o no il risultato, ...

Ricordate inoltre il layred model delle MKL?
Per linkare, ad esempio:

```
ifort fftmkl.f90 $MKL_LIB/libmkl_intel.a
$MKL_LIB/libmkl_intel_thread.a $MKL_LIB/libmkl_core.a
-L$MKL_LIB -lomp5 -lpthread -I$MKL_MOD
```

Librerie FFT

- Open source
  - FFTPACK `http://www.netlib.org/fftpack/`
  - FFTW `http://www.fftw.org/`
- Commerciali multipiattaforma
  - NAG `www.nag.com`
- Commerciali propietare
  - IBM: essl
  - Compaq: cxml
  - Sun: Sunperf
  - Intel: mkl
  - GPU NVIDIA: CUDA SDK

Versioni - Overview

- MKL: Intel Math Kernel Library
  - Major functional categories include Linear Algebra, Fast Fourier Transforms (FFT), Vector Math and Statistics. Cluster-based versions of LAPACK and FFT are also included to support MPI-based distributed memory computing.
- ACML: AMD Core Math Library
  - Libreria di funzioni altamente ottimizzate per processori AMD. Include tra l'altro BLAS, LAPACK, FFT, Random Generators
- GSL: GNU Scientific Library
  - The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.
- ESSL (IBM): Engineering and Scientific Subroutine library
  - BLAS, LAPACK, ScaLAPACK, solutori sparsi, FFT e altro. La versione parallela utilizza MPI.

Input/Output

- Le librerie di I/O risultano particolarmente utili per
  - interoperabilitá: C/Fortran, Little Endian/Big Endian,...
  - visualizzazione
  - analisi di sub-set
  - metadati
  - I/O parallelo
- HDF5: “is a data model, library, and file format for storing and managing data”
- NetCDF: “NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data”
- VTK: “open-source, freely available software system for 3D computer graphics, image processing and visualization”

Calcolo parallelo e mesh

- MPI: Message Passing Interface
  - standard piú diffuso per la parallelizzazione in memoria distribuita
  - implementazioni piú importanti come librerie: MPICH e OpenMPI
- Decomposizione di mesh
  - METIS e ParMETIS: “can partition a graph, partition a finite element mesh, or reorder a sparse matrix”
  - Scotch e PT-Scotch: “sequential and parallel graph partitioning, static mapping and clustering, sequential mesh and hypergraph partitioning, and sequential and parallel sparse matrix block ordering”

Suite per il calcolo scientifico

- Trilinos
  - open source object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems.
  - The Trilinos 11.2 general release contains 54 packages: Amesos, Amesos2, Anasazi, AztecOO, Belos, CTrilinos, Didasko, Epetra, EpetraExt, FEI, ForTrilinos, Galeri, GlobiPack, Ifpack, Ifpack2, Intrepid, Isorropia, Kokkos, Komplex, LOCA, Mesquite, ML, Moertel, MOOCHO, NOX, Optika, OptiPack, Pamgen, Phalanx, Piro, Pliris, PyTrilinos, RTOp, Rythmos, Sacado, SEACAS, Shards, ShyLU, STK, Stokhos, Stratimikos, Sundance, Teko, Teuchos, ThreadPool, Thyra, Tpetra, TriKota, TrilinosCouplings, Trios, Triutils, Xpetra, Zoltan, and Zoltan2.
  - Trilinos website: `http://trilinos.sandia.gov`
- PETSc
  - suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.
  - It supports MPI, shared memory pthreads, and NVIDIA GPUs
  - PETSc website: `http://www.mcs.anl.gov/petsc/`

Hands-on: BLAS

- Sostituire il codice matrice-matrice di matrixmul con una chiamata a `DGEMM`, la routine BLAS che esegue il prodotto matrice-matrice in doppia precisione
- `DGEMM` esegue (l'operatore `op` consente trasposizioni)

```
C := alpha*op( A )*op( B ) + beta*C,
```

- Argomenti della `DGEMM` : `http://www.netlib.org/blas/dgemm.f`
- Fortran: GNU, BLAS: consideriamo una versione base di `blas`:
  - esistono versioni piú efficienti a disposizione (e.g. ACML)

```
module load profile/advanced
module load gnu/4.6.3 blas/2011--gnu--4.6.3
gfortran -O3 matrixmulblas.F90 -L$BLAS_LIB -lblas
```

- Fortran: Intel, BLAS efficienti sono le proprietarie MKL
  - `sequential` (seriali)
  - `parallel` (multi-thread)

```
module load profile/advanced
module load intel/cs-xe-2013--binary
ifort -O3 -mkl=sequential matrixmulblas.F90
```

Hands-on: BLAS / 2

- C: compilatore Intel (MKL con cblas)
  - includere l'header file `#include<mkl.h>` nel sorgente
  - provare `-mkl=sequential` e `-mkl=parallel`

```
module load profile/advanced
module load intel/cs-xe-2013--binary
icc -O3 -mkl=sequential matrixmulblas.c
```

- C: compilatore GNU (GSL con cblas)
  - includere l'header file `#include <gsl/gsl_cblas.h>` nel sorgente

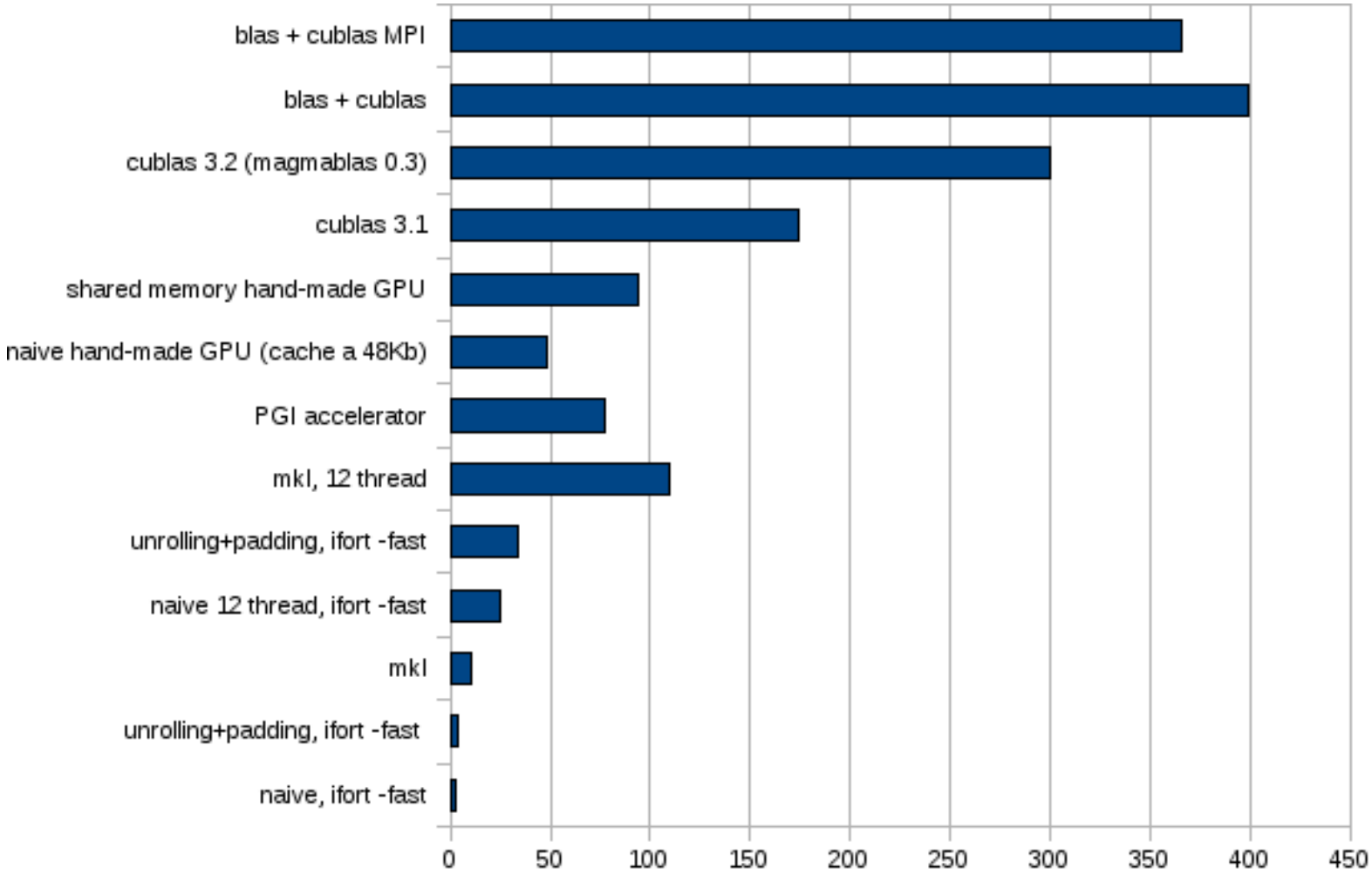
```
module load profile/advanced
module load gnu/4.8.0 gsl/1.15--gnu--4.8.0
gcc -O3 matrixmulblas.c -L$GSL_LIB -lgslcblas -I$GSL_INC
```

  - Confrontare le performance con quelle ottenibili con `-O3`/`-fast`
- Provare anche le versioni multithreaded: osservazioni?
- Riportare i **GFlop** e considerare matrici 4096x4096 (risultati piú stabili)

GNU -O3	Intel -fast	GNU-BLAS/GSL seq	Intel-MKL seq
—	Intel -fast -parallel	—	Intel-MKL par
—		—	

Si puó andare oltre?

- GPU...



Rights & Credits